Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:

«Автоматизація прогнозування фінансових рядів на основі машинного навчання»

Виконав: студент IV курсу, групи 1АКІТ-21б
спеціальності 151 – Автоматизація та
комп'ютерно-інтегровані технології
(шифр і назва спеціальності)
Сироєжко О. О.
(прізвище та ініціали)

Керівник: к.т.н., доцент кафедри АІІТ
Маслій Р. В.
(прізвище та ініціали)
«10» 06 2025 р.

Рецензент: д.т.н., професор каф. КСУ
Юзьмчук М. С.
(прізвище та ініціали)
«16» 06 2025 р.

Допущено до захисту
завідувач кафедри АІІТ
«17» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Галузь знань 15 – Автоматизація та приладобудування
Спеціальність 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітньо-професійна програма Інтелектуальні комп'ютерні системи управління

ЗАТВЕРДЖУЮ
завідувач кафедри АІІТ
_____ д.т.н., проф. Бісікало О. В.
«___» _____ 2025 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Сироєжку Олексію Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Автоматизація прогнозування фінансових рядів на основі машинного навчання»
керівник роботи Маслій Роман Васильович, к.т.н, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «20» березня 2025 року №97
2. Термін подання студентом роботи 10.06.2025
3. Вихідні дані до роботи: автоматизована система прогнозування фінансових рядів на основі машинного навчання, звіт, згенерований звіт/графік. Апаратне забезпечення – комп'ютер/ноутбук ОЗП – 2 ГБ. Програмне забезпечення та середовище розробки – Windows, Visual Studio Code, Microsoft Office. Браузер – Google Chrome. Мова програмування – Python. Бібліотеки та фреймворки – Pandas, NumPy, Yfinance, Scikit-learn, Plotly.
4. Зміст текстової частини (перелік питань, які потрібно розробити) аналіз предметної області та конкурентних рішень, вибір архітектури та технологій для проектування системи, розробка UML діаграм, розробка та тестування програмного забезпечення
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень) UML діаграма компонентів, UML діаграма діяльності, UML діаграма послідовності, графічне зображення розробленого програмного забезпечення, презентація проекту

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	К.Т.Н., доцент кафедри АІІТ Маслій Р.В.	20.03.2025	20.03.25

7. Дата видачі завдання 20.03.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Термін виконання		Примітка
		початок	закінчення	
1	Вибір, узгодження та затвердження теми БКР	01.02.2025	11.02.2025	виконано
2	Аналіз предметної області та конкурентних рішень. Опрацювання літературних джерел.	12.02.2025	14.03.2025	виконано
3	Постановка задач для вирішення в БКР	17.03.2025	21.03.2025	виконано
4	Проектування архітектури системи, вибір технологій розробки	24.03.2025	04.04.2025	виконано
5	Розробка програмного забезпечення	07.04.2025	09.05.2025	виконано
6	Тестування програмного забезпечення	12.05.2025	16.05.2025	виконано
7	Оформлення пояснювальної записки та графічної частини	19.05.2025	23.05.2025	виконано
8	Попередній захист, доопрацювання, та рецензування БКР	26.05.2025	30.05.2025	виконано
9	Захист БКР	16.06.2025	20.06.2025	виконано

Студент _____ Сироєжко О. О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Маслій Р. В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 86 сторінок формату А4, в тому числі і додатків, на яких є 24 рисунки, 5 таблиць, список використаних джерел містить 36 найменувань.

Дана бакалаврська робота присвячена розробці автоматизованої веб-системи прогнозування фінансових часових рядів із застосуванням методів машинного навчання на мові програмування Python з використанням бібліотек scikit-learn, Prophet та інтерфейсного фреймворку Streamlit.

У роботі проведено аналіз предметної області та досліджено існуючі аналітичні системи у сфері фінансового прогнозування. На основі отриманих знань і аналізу було розроблено програмну архітектуру, яка включає модулі для автоматичного збору даних, їх обробки, побудови моделей машинного навчання та візуалізації прогнозів. Система реалізує принципи автоматизованого машинного прогнозування, включаючи підготовку даних, обчислення метрик точності та інтерактивний інтерфейс користувача.

Ключові слова: фінансові часові ряди, автоматизована система, машинне навчання, Python, Streamlit, прогнозування, аналіз даних.

ABSTRACT

The bachelor's thesis consists of 86 A4-format pages, including appendices, and contains 24 figures, 5 tables, and a list of 36 references.

This bachelor's thesis is devoted to the development of an automated web-based system for forecasting financial time series using machine learning methods implemented in the Python programming language with the help of libraries such as scikit-learn, Prophet, and the Streamlit interface framework.

The work includes an analysis of the subject area and a study of existing analytical systems in the field of financial forecasting. Based on the acquired knowledge and research, a software architecture was developed, comprising modules for automated data collection, preprocessing, machine learning model construction, and forecast visualization. The system implements principles of automated machine learning forecasting, including data preparation, calculation of accuracy metrics, and an interactive user interface.

Keywords: financial time series, automated system, machine learning, Python, Streamlit, forecasting, data analysis.

ЗМІСТ

ВСТУП

Актуальність. У сучасному цифровому світі фінансові ринки[1] генерують величезні обсяги даних щодня. Ці дані є цінним джерелом інформації для прийняття стратегічних та оперативних рішень у таких сферах, як інвестиційна діяльність, банківська справа, трейдинг, державне регулювання та управління ризиками. Прогнозування фінансових часових рядів — тобто передбачення майбутніх значень фінансових показників на основі їх попередніх значень — є ключовим завданням у багатьох аналітичних процесах.

Традиційні статистичні методи, що використовуються для прогнозування, мають ряд обмежень. Вони часто вимагають припущень щодо стаціонарності рядів, не враховують нелінійні залежності та погано справляються зі складною структурою даних. У реальних умовах фінансові ринки є динамічними, непередбачуваними і схильними до зовнішніх впливів, що робить класичні методи малоефективними для тривалих прогнозів.

З розвитком штучного інтелекту, а саме — машинного навчання, з’явилася можливість будувати більш гнучкі, адаптивні моделі прогнозування, які здатні виявляти приховані закономірності у великих і складних наборах даних без потреби уявного моделювання вручну. Використання таких методів, як нейронні мережі, ансамблеві моделі, методи кластеризації та регресії, дозволяє значно підвищити точність прогнозів.

Окрім цього, актуальність теми зумовлюється потребою в автоматизації аналітичних процесів. Сучасні компанії прагнуть мінімізувати вплив людського фактора, зменшити витрати часу на аналітику та отримувати оперативні прогнози в реальному часі. Це вимагає створення інтелектуальних систем, здатних самостійно обробляти дані, навчатися на новій інформації та генерувати точні й стабільні прогнози.

Таким чином, тема автоматизації прогнозування фінансових рядів за допомогою машинного навчання є надзвичайно актуальною з огляду на потреби

сучасного фінансового ринку, розвиток технологій обробки даних та загальну цифровізацію економічних процесів.

Метою дослідження є підвищення ефективності прогнозування фінансових рядів шляхом розробки автоматизованої системи прогнозування із використанням методів машинного навчання.

Для досягнення мети визначено такі основні завдання:

- здійснити аналіз предметної області та об’єкту;
- здійснити детальний аналіз конкурентних рішень;
- розробити постановку задачі;
- розробити необхідні UML-діаграми;
- визначити стек технологій для реалізації системи;
- розробити та протестувати програмне забезпечення системи.

Об’єктом дослідження є процес автоматизації аналізу та прогнозування фінансових рядів.

Предметом дослідження є алгоритми машинного навчання та програмні рішення для автоматизації прогнозування фінансових показників.

Практична цінність полягає у розробці програмного забезпечення, результати роботи якого можуть бути застосовані у фінансовій аналітиці, біржовій торгівлі, банківських системах, для побудови адаптивних сервісів прогнозування, що підвищують ефективність прийняття рішень.

Інноваційність розробки полягає в поєднанні сучасних методів машинного навчання з автоматизованим підходом до обробки даних та побудови прогнозних моделей, що дозволяє створити адаптивну й масштабовану систему прогнозування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА КОНКУРЕНТНИХ РІШЕНЬ ДЛЯ АВТОМАТИЗАЦІЇ ПРОГНОЗУВАННЯ ФІНАНСОВИХ РЯДІВ НА ОСНОВІ МАШИННОГО НАВЧАННЯ

1.1 Аналіз предметної області

Фінансові часові ряди[2] є однією з найважливіших форм представлення інформації у фінансово-економічному аналізі. Вони відображають динаміку зміни певного фінансового показника в часі, що дозволяє не лише спостерігати за поведінкою ринку, але й будувати обґрунтовані прогнози для майбутнього періоду. Саме через часові ряди здійснюється фіксація таких критичних даних, як ціни акцій, валютні курси, індекси фондових бірж, обсяги торгів, процентні ставки, а також інші ключові макро- та мікроекономічні індикатори.

Предметна область дослідження охоплює вивчення та аналіз структури фінансових часових рядів, виявлення характерних закономірностей, а також побудову методів, які дозволяють із високою точністю передбачити майбутню поведінку того чи іншого фінансового індикатора. У свою чергу, такий підхід є важливою складовою не лише в стратегічному плануванні та оцінці ризиків, але й у щоденних операційних рішеннях трейдерів, інвесторів та фінансових менеджерів.

Фінансові часові ряди являють собою впорядковані в часі послідовності числових значень, що відображають динаміку фінансових показників, таких як ціни акцій, обсяги торгів, валютні курси, процентні ставки, індекси тощо. Їх моделювання та прогнозування відіграє ключову роль у фінансовій аналітиці, стратегічному плануванні та автоматизованій торгівлі.

У порівнянні з іншими типами часових рядів, фінансові мають ряд специфічних властивостей[3]:

1. Стохастичність та високий рівень шуму: фінансові часові ряди схильні до впливу великої кількості випадкових факторів, серед яких — ринкові новини, політичні події, соціальні зміни, емоційна реакція інвесторів тощо. Як наслідок, у даних присутній суттєвий рівень шуму, що ускладнює побудову точних моделей.

2. Нестаціонарність: часто фінансові часові ряди не є стаціонарними: їхні середні значення, дисперсія та інші характеристики змінюються з часом. Для обробки таких рядів часто використовують диференціювання або інші методи приведення до стаціонарності.

3. Наявність автокореляцій: фінансові ряди можуть демонструвати короткострокові автокореляції, які використовуються в моделях типу AR (AutoRegressive) або ARIMA (AutoRegressive Integrated Moving Average). Проте ці залежності зазвичай є слабкими та нестабільними в довгостроковій перспективі.

4. Кластеризація волатильності: зміни у волатильності (мінливості) фінансових активів, як правило, мають тенденцію до згрупування у часі: періоди високої нестабільності змінюють періоди низької. Це явище є основою для моделей типу ARCH та GARCH.

5. Наявність важких хвостів (fat tails): розподіл прибутків часто відхиляється від нормального — у ньому значно більша ймовірність появи екстремальних значень. Така поведінка може призвести до серйозних помилок у прогнозах, якщо використовувати класичні статистичні припущення.

6. Нелінійність: фінансові часові ряди можуть містити складні, нелінійні взаємозв’язки між змінними. Це обмежує ефективність лінійних моделей і мотивує використання методів машинного навчання, здатних адаптуватися до складних структур даних.

7. Чутливість до зовнішніх факторів: фінансові ринки реагують на широкий спектр зовнішніх чинників: макроекономічну статистику, рішення центральних банків, геополітичну ситуацію тощо. Ці фактори можуть бути складними для формалізації та врахування в моделях, проте останні досягнення у сфері глибинного навчання та обробки природної мови (наприклад, використання новинних стрічок) дозволяють включати їх у процес прогнозування.

8. Ефективність ринку: гіпотеза ефективного ринку (Efficient Market Hypothesis, EMH) стверджує, що ціни активів повністю відображають наявну інформацію, а отже, майбутні ціни є непередбачуваними. Хоча ця гіпотеза

неодноразово ставилась під сумнів, вона має важливе значення для оцінки потенціалу прогнозних моделей.

Таким чином, особливості фінансових часових рядів вимагають обережного та комплексного підходу до їх моделювання. У зв'язку з цим все більшої популярності набувають алгоритми машинного навчання, які можуть адаптуватися до високої складності та невизначеності фінансових даних.

Для кращого розуміння суті фінансових рядів розглянемо типовий графік зміни ціни акції певної компанії упродовж року, зображений на рисунку 1.1:

Рисунок 1.1 - Приклад фінансового часового ряду з трьома ключовими складовими

На цьому графіку зображено типовий приклад фінансового часового ряду з трьома ключовими складовими:

- Синя крива — фактичний фінансовий ряд, який поєднує тренд, сезонність та шум;
- Червона пунктирна лінія — трендова складова, що демонструє довгострокове зростання;
- Зелена штрихова лінія — сезонна складова, яка відображає періодичні коливання;

- Шум, що не відображений окремо, але помітно впливає на зміну значень, моделює реальні коливання ринку.

Ця візуалізація ілюструє, наскільки складним є фінансовий ряд, та показує, чому традиційні підходи часто не справляються з його точним прогнозуванням — і чому потрібні сучасні, гнучкі методи машинного навчання.

Формально часовий ряд можна представити як послідовність вимірювань:

$$X = \{x_1, x_2, x_3 \dots, x_t\}, \qquad x_t \in \mathbb{R},$$

де x_t — значення фінансового показника в момент часу t, а t — дискретний часовий інтервал (день, година, хвилина).

Роль фінансових рядів в аналітиці

Фінансові часові ряди використовуються в аналітиці[4] для досягнення таких цілей:

1. Оцінка ризиків – виявлення нестабільності або волатильності активів.
2. Пошук оптимальних моментів для інвестування – аналіз трендів і точок зміни напрямку.
3. Побудова прогнозів – передбачення майбутніх цін або обсягів на основі історичних даних.
4. Автоматизація торгівлі – створення торгових алгоритмів (алготрейдинг).
5. Фінансова звітність та аудит – контроль змін ключових показників.

Фінансові установи, банки, біржі та інвестиційні фонди широко застосовують прогнозні моделі для оптимізації прибутків, підвищення стійкості до ринкових ризиків і побудови довгострокової стратегії розвитку.

1.2 Аналіз об’єкта автоматизації

Об’єктом автоматизації в межах даної роботи є процес прогнозування фінансових часових рядів, що включає етапи збору, обробки, аналізу та моделювання часових рядів з метою передбачення майбутніх значень фінансових показників (наприклад, цін акцій, індексів, валютних курсів тощо)[5]. Зазначений

процес є складним і багатокомпонентним, що обумовлює необхідність його структурованого аналізу.

У традиційному підході до аналізу фінансових рядів аналітик вручну здійснює підготовку даних, вибір і налаштування моделей, тестування якості прогнозів і візуалізацію результатів[6]. Такий підхід є трудомістким, вимагає значних знань з математики, програмування та фінансової аналітики, а також не дозволяє масштабувати обчислення на великі обсяги даних або забезпечити оперативне реагування на зміну ринкових умов.

У рамках автоматизації передбачається створення цілісної програмної системи, яка дозволяє значною мірою або повністю автоматизувати вищезазначені етапи. Така система має не лише зменшити навантаження на аналітика, але й забезпечити більш об’єктивний, формалізований і реплікований підхід до аналізу фінансових ринків[7].

З технічної точки зору, об’єкт автоматизації можна розглядати як інформаційно-аналітичний процес, в якому дані проходять через низку трансформацій — від сирих фінансових записів до прогнозних моделей і графічних звітів. Реалізація такої автоматизації потребує використання відповідного програмного забезпечення, алгоритмічних рішень, інтеграції з джерелами даних, а також налаштування взаємодії з кінцевим користувачем.

Особливої уваги заслуговує використання алгоритмів машинного навчання, які демонструють високу здатність до виявлення прихованих закономірностей у великих обсягах даних. Завдяки своїй адаптивності та здатності до самонавчання, моделі ML можуть забезпечувати більш точні прогнози порівняно з класичними статистичними підходами[8].

З урахуванням наведеного, автоматизована система прогнозування повинна не лише вміти "навчати" моделі, але й забезпечувати гнучку роботу з даними, проводити оцінку ефективності моделей, адаптуватися до нових даних та надавати зручні засоби представлення результатів. Такий підхід дозволяє підвищити ефективність аналітики, знизити ймовірність суб’єктивних помилок та скоротити час ухвалення рішень на фінансових ринках.

З огляду на це, автоматизація даного процесу передбачає створення програмної системи [9], яка:

- здійснює автоматизований збір фінансових даних з відкритих джерел або API;
- виконує попередню обробку та перетворення часових рядів (нормалізація, заповнення пропусків, видалення аномалій, побудова лагових ознак тощо);
- реалізує автоматичний вибір і навчання моделей машинного навчання для побудови прогнозів;
- забезпечує оцінку якості моделей за допомогою відповідних метрик (MAE, RMSE, R^2 тощо);
- надає візуалізацію результатів прогнозу та аналітичну звітність;
- дозволяє взаємодію користувача з системою через графічний або веб-інтерфейс.

Таким чином, об’єкт автоматизації включає в себе весь цикл роботи з фінансовим часовим рядом: від моменту завантаження даних до отримання прогнозного значення та його візуального подання користувачу.

1.3 Методи прогнозування фінансових рядів

Прогнозування фінансових часових рядів є складною задачею, що потребує врахування нелінійної динаміки, стохастичності, сезонності та потенційної наявності шумів у даних[10]. У контексті автоматизації цього процесу особливу увагу приділяють методам, які здатні адаптуватися до змін структури ринку, забезпечуючи високу точність передбачень. Усі методи прогнозування фінансових рядів умовно можна поділити на дві великі категорії: класичні статистичні методи та методи машинного навчання, зокрема глибокого навчання[11].

1.3.1 Класичні статистичні методи

Класичні статистичні методи[12] є основою прогнозування часових рядів. Вони відзначаються теоретичною обґрунтованістю, формальною інтерпретованістю параметрів і достатньою ефективністю в умовах стаціонарності даних. Незважаючи на певні обмеження щодо лінійності та необхідності попередньої обробки даних, ці методи залишаються актуальними в задачах фінансового прогнозування, зокрема для базового аналізу та побудови початкових моделей.

Авторегресивна модель (AR). Модель авторегресії першого порядку (AR(1)) описується рівнянням:

$$X_t = \phi_0 + \phi_1 X_{t-1} + \varepsilon_t,$$

де ϕ0 — вільний член, ϕ1 — коефіцієнт авторегресії, εt — випадкова компонента. Для вищих порядків (AR(p)) враховується більше попередніх значень. Ці моделі добре працюють у випадках, коли спостерігається автокореляція даних.

Модель ковзного середнього (MA). Вона моделює значення часового ряду як лінійну комбінацію поточних і попередніх випадкових збурень:

$$X_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q},$$

де μ — середнє значення ряду, θ — параметри MA-компонент. MA-моделі ефективні при моделюванні шумів і короткострокових випадкових змін у ряді.

Модель ARMA (Autoregressive Moving Average). Комбінуючи AR і MA, отримуємо модель ARMA(p, q), яка може враховувати як автокореляцію, так і випадкові флуктуації. Вона використовується для стаціонарних рядів і має вигляд:

$$X_t = \phi_0 + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t.$$

Модель ARIMA (Autoregressive Integrated Moving Average). У реальних фінансових рядах, як правило, наявні тренди або сезонні компоненти, що робить їх

нестабільними (нестатіонарними). ARIMA(p, d, q) включає інтегрування (d) — диференціювання даних для приведення ряду до стаціонарного вигляду:

$$\Delta^d X_t = \text{ARMA}(p, q),$$

де Δd — оператор d-разового диференціювання. ARIMA ефективно моделює фінансові індекси, валютні курси та інші економічні індикатори з лінійною тенденцією.

Сезонна модель SARIMA. Модель SARIMA (Seasonal ARIMA) розширює ARIMA, враховуючи сезонні цикли у даних. Вона позначається як SARIMA(p, d, q)(P, D, Q)_s, де:

- p, d, q — параметри звичайної ARIMA-частини,
- P, D, Q — параметри сезонної компоненти,
- s — довжина сезонного циклу (наприклад, 12 для щомісячних даних).

Ця модель широко використовується у фінансах для прогнозування цін активів з регулярними періодичними коливаннями (наприклад, товарні ринки або сезонні коливання попиту).

Експоненціальне згладжування. Методи експоненціального згладжування (Exponential Smoothing) базуються на тому, що останні спостереження мають більшу вагу для прогнозу, ніж старіші. Відомі варіанти:

- Simple Exponential Smoothing — для даних без тренду і сезонності.
- Holt’s Linear Trend Method — враховує тренд.
- Holt-Winters Method — враховує як тренд, так і сезонність.

Ці методи мають просту структуру та швидко обчислюються, що дозволяє використовувати їх у реальному часі. Вони ефективні у випадках, коли модель потрібно швидко оновлювати при надходженні нових даних.

Переваги та недоліки статистичних методів прогнозування:

Переваги:

- Висока інтерпретованість моделей;
- Порівняно просте налаштування;
- Широка підтримка в статистичних пакетах;

- Придатність для невеликих вибірок.

Недоліки:

- Непридатність для моделювання складних нелінійних процесів;
- Чутливість до стаціонарності;
- Складність адаптації до стрімких змін структури ринку.

Більш детальна порівняльна характеристика класичних статистичних методів прогнозування фінансових рядів описана в таблиці 1.1:

Таблиця 1.1 - Порівняльна характеристика класичних статичних методів

Метод	Враховує тренд	Враховує сезонність	Необхідна стаціонарність	Інтерпрето-ваність	Переваги	Недоліки
AR(p)	Частково	Ні	Так	Висока	Простота, ефективність при автокореляції	Погано працює з трендом і сезонністю
MA(q)	Ні	Ні	Так	Висока	Добре моделює випадкові флуктуації	Вимагає стаціонарності
ARMA(p,q)	Частково	Ні	Так	Висока	Гнучке поєднання AR і MA	Не придатне для нестатіонарних даних
ARIMA(p,d,q)	Так	Ні	Ні (інтегрується)	Середня	Підходить для трендових рядів	Не враховує сезонність
SARIMA	Так	Так	Ні (інтегрується)	Середня	Добре моделює як тренд, так і сезонність	Велика кількість параметрів
Експонен-ціальне згладжування	Так / Частково	Так (у Holt-Winters)	Ні	Висока	Простота, швидка адаптація.	Менш точне для складних патернів

1.3.2 Методи машинного навчання

Методи машинного навчання (ML)[15] стали потужною альтернативою класичним статистичним моделям у прогнозуванні фінансових часових рядів. Їх основна перевага полягає у здатності виявляти складні, нелінійні залежності, адаптуватися до великих обсягів даних і працювати з нестаціонарними рядами без попереднього диференціювання.

На відміну від класичних підходів, ML-моделі часто вимагають побудови ознак (feature engineering), включаючи лаги, ковзні середні, індикатори технічного аналізу тощо. Однак при належній підготовці ці моделі демонструють високу точність і стійкість до шуму. Розглянемо нижче основні моделі машинного навчання для прогнозування часових рядів.

Лінійна регресія (Linear Regression)

Хоч і належить до базових методів, лінійна регресія часто використовується як базова модель або в якості елементу ансамблю. У випадку великої кількості ознак може застосовуватись регуляризація:

- Ridge Regression (L2-регуляризація): зменшує перенавчання за рахунок штрафу на великі значення коефіцієнтів.

$$\mathcal{L}_{\text{ridge}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} w_j^2$$

- Lasso Regression (L1-регуляризація): також виконує відбір ознак (feature selection), зануляючи несуттєві коефіцієнти.

$$\mathcal{L}_{\text{lasso}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} |w_j|$$

Ці моделі добре інтерпретуються, але мають обмеження в умовах нелінійності даних. Нижче представлена формула моделі:

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \mathbf{w}^\top \mathbf{x}$$

Працює тільки за умови лінійних взаємозв’язків.

Метод опорних векторів (SVR – Support Vector Regression)

Це розширення SVM для регресійних задач. Метод опорних векторів для регресії використовує гіперплощину з допуском похибки ε\varepsilonε і шукає найплоскішу функцію, що потрапляє в допустимий інтервал. Завдяки використанню ядерних функцій (kernel trick), SVR дозволяє моделювати нелінійні залежності.

Переваги:

- Висока узагальнювальна здатність
- Ефективність при малих вибірках

Недоліки:

- Потребує масштабування ознак
- Висока складність для великих наборів даних

Може працювати з нелінійними залежностями за допомогою ядрових функцій (Radial Basis Function, Polynomial тощо).

Дерева рішень і ансамблеві методи

Decision Tree Regressor. Побудова дерева, де у вузлах приймаються рішення на основі значення ознак. Добре працює з нелінійностями, однак має схильність до переобучення.

Цей метод ітеративно розбиває простір ознак на області, в яких прогноз є сталою величиною. При кожному розбитті вибирається ознака xj та поріг s, що мінімізує середню квадратичну помилку:

$$\min_{j,s} \left[\frac{1}{|R_1|} \sum_{x_i \in R_1(j,s)} (y_i - \bar{y}_1)^2 + \frac{1}{|R_2|} \sum_{x_i \in R_2(j,s)} (y_i - \bar{y}_2)^2 \right],$$

де R1 та R2 — підмножини, отримані розбиттям.

Random Forest. Ансамбль багатьох дерев, що зменшує дисперсію за рахунок бутстрапінгу. Підвищує стабільність прогнозів та стійкість до шуму.

Ансамбль з B дерев, які тренуються на випадкових підмножинах даних і ознак. Прогноз — це середнє:

$$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} \hat{y}^{(b)}$$

Gradient Boosting Machines (GBM, XGBoost, LightGBM, CatBoost). Моделі бустингу працюють шляхом послідовного навчання слабких моделей (дерев), де кожна наступна мінімізує помилки попередньої. Особливо ефективні для складних структур даних з багатьма ознаками, включаючи лаги, коваріації, технічні індикатори тощо.

Будується послідовно, кожна нова модель ht(x) вчиться на залишках попередньої:

$$\hat{y}_t(x) = \hat{y}_{t-1}(x) + \eta h_t(x)\,,$$

де η — коефіцієнт навчання (learning rate), ht — слабкий учень (часто — дерево глибини 3–6). Тому фінальний прогноз має вигляд:

$$\hat{y}(x) = \sum_{t=1}^{T} \eta h_t(x)$$

Переваги:

- Підтримка нелінійних залежностей
- Автоматичне ранжування важливості ознак
- Висока точність

Недоліки:

- Повільне навчання
- Менш інтерпретовані, ніж простіші моделі

Метод k-найближчих сусідів (KNN — k-Nearest Neighbors)

Метод базується на припущенні, що майбутнє значення ряду можна передбачити на основі середніх значень найближчих спостережень у багатовимірному просторі ознак. Хоча метод простий, він може бути малоефективним на великих вибірках або за наявності шуму.

Штучні нейронні мережі (ANN, DNN)

Нейронні мережі[16] імітують принцип роботи біологічних нейронів і здатні апроксимувати будь-які неперервні функції. У контексті фінансового прогнозування найчастіше використовуються:

Multilayer Perceptron(MLP). Складаються з вхідного шару, одного або кількох прихованих шарів та вихідного шару. Можуть моделювати складні залежності, але не враховують часову послідовність без попереднього формування лагових ознак. Формула одного шару:

$$\mathbf{h}^{(l)} = \sigma\left(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right),$$

деW(l), b(l) — ваги та зсуви, σ - функція активації.

Artificial Neural Networks (ANN). Базова архітектура, що складається з одного або декількох прихованих шарів. Може моделювати складні функції завдяки нелінійним активаційним функціям (ReLU, Tanh, Sigmoid).

Deep Neural Networks (DNN). Глибокі мережі з багатьма шарами здатні вивчати складні шаблони та взаємозв'язки в даних, однак вимагають великої кількості навчальних прикладів.

Recurrent Neural Networks (RNN). Призначені для роботи з послідовними даними, оскільки мають зворотні зв'язки, які дозволяють зберігати інформацію про попередні кроки. RNN враховує залежності між часовими кроде ht— прихований стан в момент часу t, xt — вхід на кроці t, y^t — прогноз.

Long Short-Term Memory (LSTM)**.** Покращення RNN, що вирішує проблему зникнення градієнтів. Використовує спеціальні комірки пам'яті з воротами (input, forget, output gates), які дозволяють зберігати довготривалі залежності.

GRU (Gated Recurrent Unit). Спрощений варіант LSTM з меншою кількістю параметрів, але подібною продуктивністю.

Переваги:

- Орієнтація на часові залежності
- Актуальні для високочастотних фінансових даних

Недоліки:

- Потребують багато даних
- Складність налаштування гіперпараметрів

Залежно від задачі, характеру фінансового ряду (лінійний чи ні, стаціонарний чи сезонний, шумовий чи структурований), обирається відповідний тип моделі. Більш детальна порівняльна характеристика найпопулярніших методів машинного навчання прогнозування фінансових рядів описана в таблиці 1.2:

Таблиця 1.2 - Порівняльна характеристика методів машинного навчання

Метод	Лінійність	Врахування сезонності	Інтерпрето-ваність	Переваги	Недоліки
Linear Regression	Так	Через ознаки	Висока	Простота, швидкість	Погано працює з нелінійними залежностями
SVR	Ні	Через ядро/ознаки	Середня	Підходить для складних задач, стійкість до викидів	Чутлива до гіперпараметрів, повільна на великих даних
Random Forest	Ні	Через ознаки	Частково	Висока точність, стійкість до переобучення	Погано екстраполює за межі тренувальних даних
XGBoost	Ні	Через ознаки	Частково	Висока продуктивність, ефективне навчання	Складність налаштування, може переобучуватися
MLP (нейромережа)	Ні	Так	Низька	Гнучкість, висока точність	Потребує великого обсягу даних і часу на тренування

Продовження таблиці 1.2

LSTM	Ні	Так (вбудовано)	Низька	Враховує контекст і пам’ять	Складність реалізації, велика потреба в даних
KNN	Ні	Через шаблони	Висока	Простота, добре працює з повторюваними структурами	Не масштабується, чутлива до шуму

1.4 Метрики оцінки якості моделей прогнозування

Оцінка ефективності моделей прогнозування є ключовим етапом будь-якого дослідження в галузі фінансової аналітики[14]. Вибір адекватної метрики визначає не лише точність побудованих моделей, а й їх придатність до використання в реальних умовах. Різні метрики дозволяють оцінювати різні аспекти моделі: середню помилку, чутливість до викидів, узагальнювальну здатність, а також порівнювати моделі між собою.

Нижче наведено розгорнутий аналіз найбільш поширених метрик:

1. Середньоквадратична помилка (MSE — Mean Squared Error)

Формула:

$$\mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2$$

Сутність:

Вимірює середній квадрат різниці між фактичними значеннями yt і прогнозами y^t. MSE особливо чутлива до великих похибок, що робить її корисною у випадках, коли великі відхилення є критичними.

Переваги:

- Часто використовується як функція втрат у навчанні моделей.

- Підкреслює значущість сильних відхилень (викидів).

Недоліки:

- Неінтерпретована в одиницях виміру (вимірюється в квадраті ціни).
- Надмірна чутливість до одиничних великих похибок.

2. Корінь середньоквадратичної помилки (RMSE — Root Mean Squared Error)

Формула:

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}$$

Сутність:

Дає середню помилку прогнозу у тій самій одиниці виміру, що і самі значення ряду. Часто використовується як узагальнена оцінка точності моделі.

Переваги:

- Інтерпретована (наприклад, в гривнях чи доларах).
- Добре відображає середню «вартість» помилки.

Недоліки:

- Так само як MSE, схильна до впливу викидів.
- Може переоцінювати важливість окремих помилок.

3. Середня абсолютна помилка (MAE — Mean Absolute Error)

Формула:

$$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t|$$

Сутність:

Оцінює середнє абсолютне відхилення прогнозів від фактичних значень. На відміну від MSE/RMSE, однаково «штрафує» всі помилки незалежно від їхньої величини.

Переваги:

- Інтерпретована в реальних одиницях.

- Менш чутлива до викидів.
- Проста у використанні та поясненні.

Недоліки:

- Не надає більше «ваги» великим помилкам.
- Може не вловлювати різкі коливання (особливо в ринкових піках).

4. Коефіцієнт детермінації (R^2— Coefficient of Determination)

Формула:

$$R^2 = 1 - \frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{n}(y_t - \bar{y})^2}$$

Сутність:

Показує, яка частина дисперсії фактичних значень може бути пояснена побудованою моделлю. Значення R2R^2R2 варіюється від −∞-\infty−∞ до 1.

- R^2=1 — ідеальна модель.
- R^2=0 — модель нічого не пояснює.
- R^2<0 — модель гірша за наївне середнє.

Переваги:

- Зручна для порівняння моделей.
- Добре працює з лінійними моделями та стаціонарними рядами.

Недоліки:

- Може вводити в оману при прогнозуванні нестабільних часових рядів.
- Чутлива до викидів.
- Не є адекватною для задач з нелінійною структурою або зі зсувом в тренді.

5. MAPE (Mean Absolute Percentage Error)

Формула:

$$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right|$$

Сутність:

Показує середнє відхилення прогнозу у відсотках від фактичного значення. Особливо корисна для бізнес-інтерпретації.

Переваги:

- Відносна метрика (виражена у %).
- Зручна для порівняння моделей на різних наборах даних.

Недоліки:

- Неможливо обчислити при yt = 0.
- Схильна до великих спотворень при малих значеннях yt.

6. Symmetric MAPE (sMAPE)

Формула:

$$\text{sMAPE} = \frac{100\%}{n} \sum_{t=1}^{n} \frac{|\hat{y}_t - y_t|}{(|y_t| + |\hat{y}_t|)/2}$$

Переваги:

- Не має проблеми з діленням на нуль, на відміну від MAPE.
- Більш стабільна при роботі з нульовими або близькими до нуля значеннями.

Недоліки:

- Інтерпретація складніша, ніж у класичної MAPE.

7. Mean Directional Accuracy (MDA)

Формула:

$$\text{MDA} = \frac{1}{n} \sum_{t=2}^{n} \mathbb{I}[(y_t - y_{t-1})(\hat{y}_t - \hat{y}_{t-1}) > 0],$$

де I[] — індикатор істинності умови.

Сутність: ,

Оцінює, наскільки добре модель прогнозує напрямок зміни, а не конкретні значення. Це надзвичайно важливо для фінансових ринків, де правильне передбачення зростання чи падіння має вирішальне значення.

В залежності від конкретної задачі, вибирається своя пара метрик оцінки (див. таблицю 1.3):

Таблиця 1.3 – Метрики залежно від типу задачі

Тип задачі	Рекомендовані метрики
Абсолютна точність	RMSE, MAE
Відносна точність	MAPE, sMAPE
Порівняння моделей	R2R^2R2, RMSE
Аналіз напрямку тренду	MDA
Стійкість до викидів	MAE, sMAPE
Оптимізація моделі	MSE (як функція втрат)

Застосування декількох метрик дозволяє всебічно оцінити модель: з погляду точності, стабільності та практичного використання. У фінансовому прогнозуванні бажано поєднувати RMSE або MAE із R^2 або MDA — це дає змогу не лише зрозуміти точність, а й ефективність прийняття рішень на основі моделі.

1.5 Підходи до автоматизації прогнозування

Автоматизація прогнозування фінансових часових рядів передбачає створення інтегрованої системи, яка мінімізує необхідність участі людини в процесі побудови, оптимізації, оцінки та застосування прогнозних моделей[17]. Це дозволяє оперативно аналізувати великі обсяги даних, оперативно адаптувати моделі до нових умов та зменшити ризик помилок, пов’язаних з людським фактором.

Автоматизація охоплює кілька ключових етапів, кожен з яких може бути повністю або частково реалізований програмними засобами[18]:

Автоматизоване збирання та попередня обробка даних.

- Завантаження даних з API (наприклад, Yahoo Finance, Alpha Vantage, Binance API).
- Очистка даних (усунення пропущених значень, аномалій, шуму).
- Формування ознак (feature engineering):
  - обчислення технічних індикаторів (SMA, EMA, RSI, MACD тощо);
  - лагові змінні;
  - ковзні середні та ковзні вікна;
  - сезонність, часові мітки (день тижня, місяць тощо).

Автоматизований вибір і навчання моделей (AutoML).

AutoML (Automated Machine Learning) — підхід, що передбачає повну автоматизацію процесу побудови моделі, включаючи:

- Вибір алгоритму (лінійна регресія, дерева, бустинг, нейронні мережі тощо).
- Пошук найкращих гіперпараметрів (через Grid Search, Random Search, Bayesian Optimization).
- Крос-валідацію та перевірку стабільності результатів.
- Усунення проблем перенавчання.

Пайплайни обробки (ML Pipelines).

Для організації процесу навчання використовуються **пайплайни**, які поєднують усі етапи:

1. Підготовка ознак;
2. Масштабування/нормалізація;
3. Побудова моделі;
4. Оцінка результатів;
5. Збереження моделі.

Автоматизована оцінка якості прогнозу.

На основі попередньо визначених метрик якості (MAE, RMSE, R^2 тощо) система сама визначає, яка модель дає кращий результат, і зберігає її. Можлива побудова автоматизованих звітів із візуалізацією результатів, таких як:

- реальні vs прогнозовані значення;
- помилки прогнозу у часі;
- щільність розподілу залишків.

Збереження моделей та розгортання

Збереження найкращих моделей (joblib, pickle, ONNX, TensorFlow SavedModel).Використання REST API або локальних/хмарних серверів для інтеграції прогнозної системи в інформаційну інфраструктуру.

Також автоматизується контроль актуальності моделей (наприклад, регулярне перенавчання з новими даними).

Інтерфейси користувача (GUI / Web) для взаємодії з системою

Для зручності користувачів автоматизована система може містити:

- графічний інтерфейс (на Python з Tkinter, PyQt, Gradio);
- веб-інтерфейс (на Flask/Django/Streamlit);
- інтерактивні графіки (Plotly, Bokeh, Matplotlib);
- логіку введення параметрів: тикер, діапазон дат, обраний алгоритм тощо.

Автоматизація прогнозування фінансових рядів є важливим напрямком у сучасній прикладній аналітиці, що дозволяє підвищити точність, швидкість та масштабованість прогнозних систем. Інтеграція AutoML-підходів, модулів для підготовки даних, оптимізації моделей та інтерфейсів користувача забезпечує створення ефективних і адаптивних рішень для реального бізнес-середовища.

1.6 Аналіз конкурентних рішень та існуючих програмних систем

Сучасний ринок програмних рішень для фінансового аналізу та прогнозування є досить розвиненим і представлений як комерційними продуктами, так і відкритими програмними платформами. У більшості випадків такі системи орієнтовані на надання аналітичних інструментів для трейдерів, фінансових аналітиків, дослідників і компаній, що працюють у сфері фінансових технологій. У даному розділі розглянуто найбільш поширені програмні системи, що реалізують функції аналізу та прогнозування фінансових часових рядів, з акцентом на

застосування алгоритмів машинного навчання та автоматизацію аналітичного процесу.

1. MetaTrader 5 (MT5)

MetaTrader 5 [20] - це багатофункціональна платформа для фінансового аналізу, яка широко використовується трейдерами для технічного аналізу, автоматизованої торгівлі та тестування стратегій.

Основні функції:

- Робота з графіками в реальному часі.
- Понад 80 технічних індикаторів.
- Автоматизоване тестування торгових стратегій.
- Розробка власних алгоритмів на MQL5.
- Підключення до бірж через брокера.

Переваги:

- Потужна десктопна платформа з гнучкою аналітикою.
- Підтримка автоматизації (роботи-радники).
- Висока швидкість обробки даних.
- Зручний користувацький інтерфейс (див. рисунок 1.2)

Недоліки:

- Відсутність підтримки машинного навчання “з коробки”.
- Не підходить для інтеграції сучасних ML-фреймворків (TensorFlow, PyTorch).
- Платформа більше орієнтована на трейдинг, а не на дослідницьку аналітику.

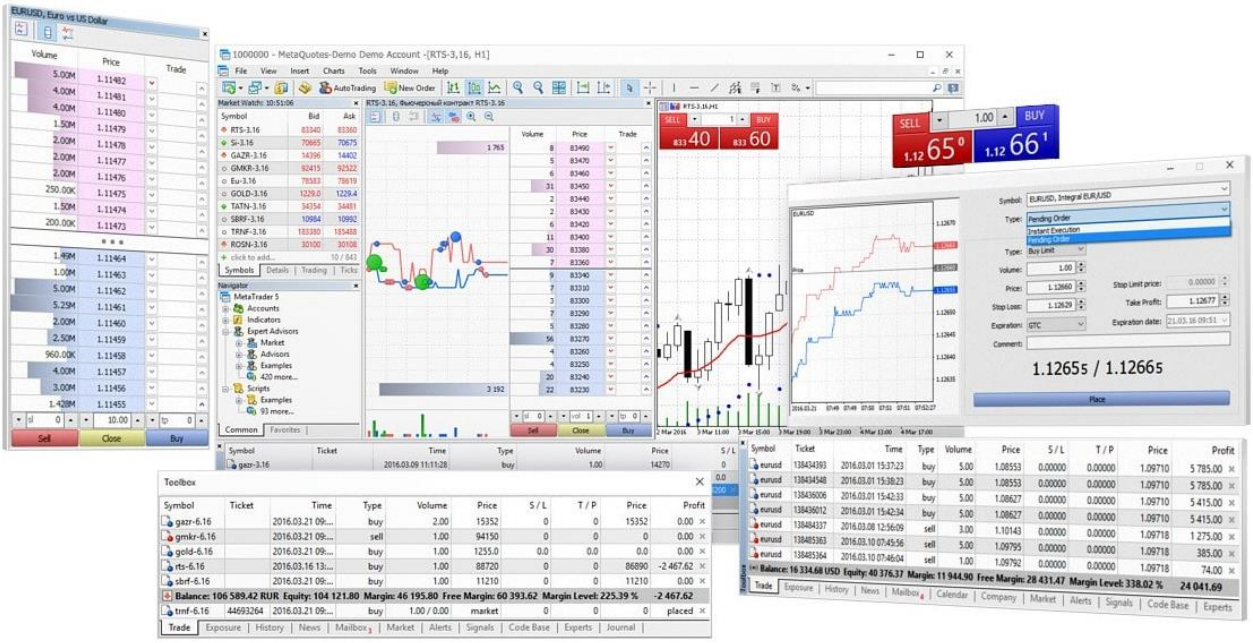


Рисунок 1.2 – Інтерфейс MetaTrader 5

2. TradingView

TradingView[21] — це одна з найпопулярніших онлайн-платформ для технічного аналізу, яка дозволяє користувачам створювати, переглядати та обмінюватися торговими ідеями, а також будувати складні графіки.

Основні функції:

- Веб-інтерфейс із зручними графіками.
- Понад 100 вбудованих індикаторів і можливість створення власних (Pine Script).
- Соціальна мережа трейдерів з прогнозами та аналітикою.
- Підтримка торгових ботів через API.

Переваги:

- Простота у використанні, доступність через браузер.
- Висока якість графічного відображення даних.
- Можливість отримання даних з багатьох ринків.
- Сучасні методи відображення – наприклад свічковий (див. рисунок 1.3)

Недоліки:

- Немає підтримки складних алгоритмів машинного навчання.
- Скрипти обмежені мовою Pine, яка не підтримує глибокі моделі.
- Обмежені можливості з автоматизації обробки великих обсягів історичних даних.

Рисунок 1.3 – Інтерфейс TradingView

3. Forecastr

Forecastr[22] — це веб-орієнтована платформа, призначена для створення фінансових прогнозів, головним чином у сфері планування прибутку, витрат та грошових потоків. Хоча вона не спеціалізується на машинному навчанні, вона пропонує повну автоматизацію процесу прогнозування для малого та середнього бізнесу.

Основні функції:

- Побудова фінансових моделей через інтерактивний веб-інтерфейс.
- Візуалізація прогнозів доходів, витрат та кеш-флоу (див. рисунок 1.4).

- Інтеграція з Excel, Google Sheets, QuickBooks тощо.
- Просте налаштування без програмування.

Переваги:

- Орієнтація на користувачів без технічної підготовки.
- Проста автоматизація типових сценаріїв прогнозування.
- Хмарна платформа з доступом через браузер.

Недоліки:

- Вузька бізнес-орієнтація (відсутність підтримки фінансових ринків).
- Немає гнучкої інтеграції з ML-фреймворками.
- Обмежена точність при роботі з волатильними часовими рядами (наприклад, акції, криптовалюта).

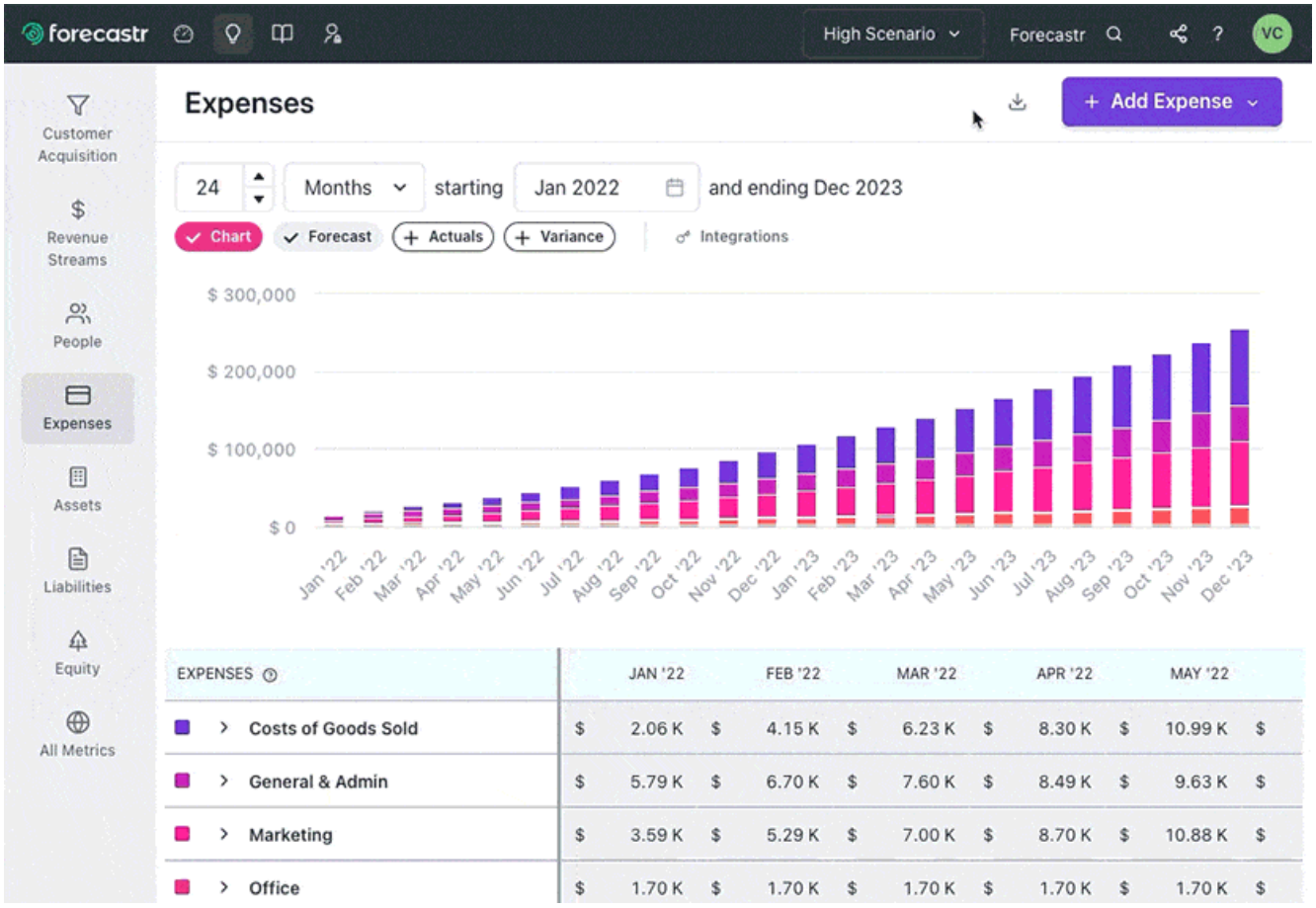


Рисунок 1.4 – Інтерфейс Forecastr

В таблиці 1.4 представлена більш інформативна порівняльна характеристика даних конкурентних рішень.

Таблиця 1.4 – Порівняльна характеристика конкурентних рішень

Платформа	Тип	ML-підтримка	Автоматизація	Призначення	Недоліки
MetaTrader 5	Десктоп	Немає	Часткова	Трейдинг, аналіз ринку	Відсутність ML
TradingView	Веб	Немає	Обмежена	Тех. аналіз, соціальна аналітика	Відсутність кастомних моделей
Forecastr	Веб	Обмежено	Так	Бізнес-аналітика	Не призначений для фін. ринків
Платформа	Тип	ML-підтримка	Автоматизація	Призначення	Недоліки

Аналіз конкурентних рішень демонструє, що на ринку домінують системи, які або орієнтовані на технічний аналіз без машинного навчання, або на просте автоматизоване прогнозування без гнучких моделей. Це підкреслює актуальність створення власної системи прогнозування фінансових рядів.

2 ВИБІР АРХІТЕКТУРИ ТА ТЕХНОЛОГІЙ ДЛЯ ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Постановка задачі та визначення основних функцій системи

У сучасних умовах високої волатильності фінансових ринків виникає потреба у системах, які дозволяють не лише аналізувати історичні дані, а й здійснювати прогнозування майбутніх цін фінансових інструментів з метою ухвалення обґрунтованих рішень. Одним із ефективних підходів до розв’язання цієї задачі є застосування методів машинного навчання, які здатні моделювати складні залежності між числовими характеристиками ринку.

Метою даної роботи є розробка веб-орієнтованої інформаційної системи для автоматизації процесу прогнозування фінансових часових рядів на основі історичних даних, отриманих із біржових джерел. Система повинна забезпечити інтуїтивно зрозумілий інтерфейс, дозволити користувачеві обрати компанію за її тикером, задати часовий проміжок для аналізу, обрати модель машинного навчання, виконати прогноз і візуалізувати результати.

На основі поставленої мети було сформульовано наступні основні функції системи:

1. Інтерактивне введення вхідних параметрів:
    - введення тикера компанії;
    - вибір початкової та кінцевої дати для збору біржових даних;
    - вибір моделі машинного навчання (Random Forest, Linear Regression, SVR, KNN);
    - вибір валюти для відображення результатів (USD, EURO, UAH);
    - завантаження власного CSV-файлу з фінансовими даними (за необхідності).
2. Автоматичне отримання історичних фінансових даних із сервісу Yahoo Finance або обробка даних із CSV-файлу.
3. Передобробка та підготовка даних:

- формування ознак із даних біржі;
- створення цільової змінної — прогнозованої ціни наступного дня;
- розділення вибірки на тренувальні та тестові дані.

4. Навчання та використання обраної моделі машинного навчання для побудови прогнозу.
5. Оцінка якості моделі за такими метриками:
    - MSE (Mean Squared Error),
    - RMSE (Root Mean Squared Error),
    - MAE (Mean Absolute Error),
    - R^2 (коефіцієнт детермінації).
6. Інтерактивна візуалізація результатів прогнозу у вигляді графіка:
    - реальні значення ціни;
    - прогнозовані значення;
    - середні значення;
    - довірчі межі для прогнозу.
7. Надання підказок щодо інтерпретації метрик оцінки моделі через контекстну довідку.

Таким чином, система реалізує повний цикл — від збору даних до візуального аналізу результатів прогнозу, забезпечуючи автоматизований підхід до аналізу фінансових часових рядів з можливістю гнучкої настройки моделі та інтерфейсу для користувача.

2.2 Аналіз та обґрунтування технологій розробки

Розробка системи прогнозування фінансових часових рядів потребує використання надійних та гнучких засобів програмування, аналізу даних, машинного навчання та візуалізації. У цьому підпункті розглянуто основні інструменти, що використовуються на різних етапах створення програмного забезпечення: від роботи з даними до побудови інтерфейсу користувача.

Мови програмування[24]:

Python — це високорівнева мова програмування, що має простий синтаксис, велику кількість бібліотек для обробки даних та розвинутий інструментарій для машинного навчання. Вона є стандартом де-факто в галузі Data Science. Python підтримує об'єктно-орієнтований, процедурний та функціональний підходи, що робить її універсальним інструментом для побудови як прототипів, так і повноцінних систем.

JavaScript — основна мова програмування для веб-фронтенду. Вона дозволяє створювати інтерактивні інтерфейси користувача, які можуть працювати з сервером через API. Також активно використовується в сучасних фреймворках (React, Vue, Svelte) для динамічного відображення графіків і таблиць.

SQL (Structured Query Language) — мова для управління реляційними базами даних. Забезпечує ефективне зберігання, вибірку та агрегацію великих обсягів історичних фінансових даних, що є важливим для навчання моделей.

Бібліотеки та фреймворки[25] для роботи з даними:

Pandas є ключовою бібліотекою Python для роботи з табличними даними. Вона забезпечує зручні структури (DataFrame) для обробки та аналізу фінансової інформації: очищення, агрегації, фільтрації та трансформації.

NumPy — бібліотека для наукових обчислень у Python, яка забезпечує ефективну роботу з великими масивами чисел, що часто використовується у попередній обробці даних перед передачею їх у модель.

Yfinance – бібліотека, що дозволяє отримувати історичні котирування акцій, облігацій, індексів та інших фінансових інструментів безпосередньо з Yahoo Finance. Підтримує завантаження даних у форматі pandas DataFrame.

Інструменти машинного навчання:

Scikit-learn — одна з найпопулярніших бібліотек Python для машинного навчання. Вона містить алгоритми регресії, класифікації, кластеризації, зниження розмірності, а також засоби для попередньої обробки та валідації моделей.

Statsmodels – це бібліотека надає широкий вибір статистичних моделей, включаючи ARIMA, SARIMA, VAR тощо, які активно застосовуються в аналізі часових рядів. Вона дозволяє будувати детальні звіти про ефективність моделі.

Prophet — це сучасний фреймворк прогнозування часових рядів, розроблений Meta (Facebook). Його основною перевагою є простота використання, автоматичне врахування сезонності, свят і трендів. Працює навіть на неповних або нерегулярних даних.

XGBoost / LightGBM - це потужні бібліотеки градієнтного бустингу, які демонструють високу точність у задачах регресії. Вони здатні враховувати взаємозв'язки між змінними та працювати з великою кількістю ознак, що може бути корисним для прогнозування фінансових серій.

Інструменти для візуалізації даних:

Matplotlib - класична бібліотека для побудови графіків у Python. Дає повний контроль над виглядом графіків, але менш інтерактивна.

Seaborn - побудована на основі matplotlib, бібліотека seaborn дозволяє швидко створювати естетичні статистичні графіки з меншою кількістю коду.

Plotly - інтерактивна бібліотека, яка дозволяє користувачеві взаємодіяти з графіками: масштабувати, наводити, вибирати тощо. Підтримується у вебінтерфейсах, таких як Streamlit.

Altair - декларативна бібліотека для побудови інтерактивних візуалізацій, яка добре інтегрується з pandas і дозволяє створювати чисті, аналітичні графіки.

Засоби створення користувацького інтерфейсу:

Streamlit — фреймворк для створення вебінтерфейсів на Python. Дозволяє створювати динамічні інтерфейси з мінімальним кодом, інтегруючи графіки, кнопки, форми та інші елементи взаємодії з користувачем.

Gradio - ще один Python-фреймворк для створення інтерфейсів, який популярний серед розробників моделей ШІ. Простий у використанні, проте менш гнучкий у порівнянні з Streamlit.

Flask / FastAPI – це фреймворки для створення серверної частини веб-застосунків. Flask — легкий і простий, підходить для невеликих проектів. FastAPI

— сучасний варіант з підтримкою асинхронності, OpenAPI та автоматичною генерацією документації.

Середовища розробки:

VS Code - легке, але функціональне середовище розробки з широкою підтримкою мов програмування, включаючи Python. Завдяки великій кількості розширень (наприклад, Python, Jupyter, Pylance) підходить для аналітики даних та машинного навчання. Має інтегрований термінал, підтримку Git, інтелектуальне автодоповнення, дебагер, а також можливість роботи з віртуальними середовищами. Підтримує роботу з Jupyter-ноутбуками прямо в редакторі.

PyCharm - потужне IDE для Python з підтримкою інтелектуального автодоповнення, зневадження, вбудованого тестування, інтеграції з системами керування версіями та середовищами віртуалізації. Зручно для розробки великих проєктів та роботи з фреймворками машинного навчання.

У результаті аналізу та порівняння засобів було обрано такі основні інструменти:

- Мова програмування: Python — через її простоту, універсальність, багатство бібліотек для машинного навчання, фінансового аналізу та інтеграції з інтерфейсами;
- Фреймворк інтерфейсу: Streamlit — через його здатність швидко створювати вебінтерфейси на основі Python без додаткових знань HTML/CSS;
- Бібліотеки для ML: Prophet, scikit-learn та statsmodels — як перевірені рішення для аналізу та прогнозування часових рядів;
- Інструменти візуалізації: plotly — як найзручніший варіант для інтерактивного графічного відображення прогнозу та реальних даних;
- Середовище розробки: VS Code – як універсальне середовище розробки.

Такий набір інструментів дозволяє створити ефективну, гнучку та зручну для користувача систему прогнозування фінансових часових рядів.

2.3 Проектування архітектури системи

Архітектура системи[26] прогнозування фінансових часових рядів повинна забезпечувати модульність, гнучкість, масштабованість і зручність у використанні. Вона має включати компоненти для збору даних, попередньої обробки, навчання моделей, візуалізації результатів і взаємодії з користувачем через графічний інтерфейс. Розглянемо загальну структуру системи за рівнями.

Загальна структура системи:

Система побудована за принципами багаторівневої (модульної) архітектури, яка складається з таких основних рівнів, та зображений на рисунку 2.1:

- Рівень збору та збереження даних (Data Layer)
- Рівень обробки та моделювання (Logic Layer)
- Рівень візуалізації та інтерфейсу (Presentation Layer)

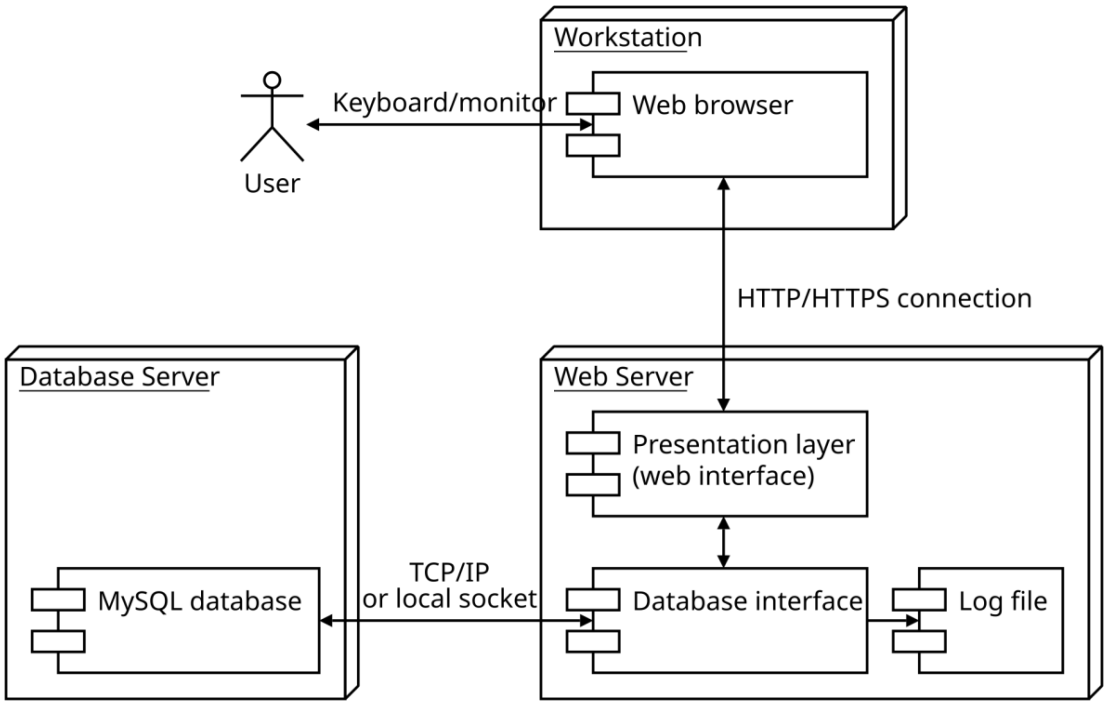


Рисунок 2.1 – UML діаграма компонентів

Опис компонентів архітектури:

Рівень збору та збереження даних:

Модуль збору фінансових даних. Використовує бібліотеку yfinance для отримання історичних котирувань акцій або індексів за вказаним тикером у певному діапазоні дат. Дані завантажуються у форматі pandas.DataFrame.

Сховище даних. Для локального зберігання даних використовується вбудована база даних SQLite або CSV-файли. Це дозволяє повторно використовувати дані без повторного запиту до зовнішніх API.

Рівень обробки та моделювання:

Модуль попередньої обробки. Включає очищення, фільтрацію, заповнення пропусків, нормалізацію та генерацію нових ознак (наприклад, рухомих середніх). Здійснюється за допомогою бібліотек pandas та numpy.

Модуль моделювання.В залежності від обраного методу, може використовуватись:

- Prophet — для прогнозування з урахуванням сезонності;
- statsmodels (ARIMA, SARIMA) — для класичних моделей часових рядів;
- scikit-learn (наприклад, LinearRegression, RandomForest) — для регресійних підходів.

Модуль забезпечує навчання моделі на тренувальній вибірці та побудову прогнозу.

Модуль оцінки якості моделі. Обчислює метрики точності прогнозу, такі як MAE (середня абсолютна помилка), RMSE (корінь з середньої квадратичної помилки), MAPE тощо.

Рівень візуалізації та інтерфейсу:

Модуль візуалізації результатів**.** Використовує бібліотеку plotly для побудови інтерактивних графіків, що відображають:

- історичні дані;
- прогнозні значення;
- зони невизначеності (інтервали довіри).

Графічний інтерфейс користувача (GUI). Побудований за допомогою фреймворку Streamlit. Дозволяє користувачу:

- обирати актив (тикер);
- вказувати період для аналізу;
- запускати прогнозування;
- переглядати результати у вигляді графіків та таблиць.

Архітектура реалізує чітке розділення обов'язків між модулями. Наприклад, компонент збору даних не залежить від обраної моделі прогнозу, а модель — від способу візуалізації. Це дозволяє легко замінити або оновити будь-який модуль без переробки всієї системи.

Переваги обраної архітектури:

- Гнучкість — можна підключати нові моделі, змінювати інтерфейс або джерела даних незалежно одне від одного.
- Масштабованість — з можливістю майбутнього розширення до веб-сервісу чи хмарної системи.
- Простота тестування — модулі ізольовані, тому тестуються окремо.
- Придатність для швидкої розробки MVP — за рахунок використання Streamlit та інтерактивної візуалізації.

2.4 Розробка UML діаграм

UML-діаграми (Unified Modeling Language) [27] — це стандартизовані графічні засоби для моделювання та опису структури й поведінки програмних систем. Вони забезпечують уніфікований спосіб візуалізації архітектури, логіки функціонування та взаємодії між компонентами застосунку. UML широко використовується на етапах проєктування програмного забезпечення, що дозволяє ще до початку розробки виявити логічні помилки, покращити комунікацію в команді та забезпечити узгоджене розуміння системи серед усіх зацікавлених сторін.

UML-діаграми застосовуються для:

- візуалізації структури й поведінки програмного забезпечення;
- специфікації функціональності системи та вимог до неї;
- документування архітектури проєкту;
- аналізу процесів та виявлення недоліків на ранніх етапах;
- конструювання програмного забезпечення на основі моделей.

Основні типи UML-діаграм:

- Діаграми класів — моделюють структуру системи, показуючи класи, атрибути, методи та зв’язки між ними;
- Діаграми послідовності — відображають взаємодію між об’єктами у часовій послідовності;
- Діаграми станів — описують життєвий цикл об’єкта та зміни його станів;
- Діаграми активностей — моделюють логіку виконання процесів, включаючи умови, цикли та дії;
- Діаграми компонентів — ілюструють фізичну структуру системи;
- Діаграми розгортання — демонструють розміщення компонентів на апаратному забезпеченні.

Переваги використання UML:

- Стандартизованість — UML є загальноприйнятим стандартом, що забезпечує зрозумілу та єдину нотацію;
- Уніфікація процесів розробки — дозволяє узгоджено працювати над системою команді розробників, аналітиків і тестувальників;
- Зниження ризиків — завдяки ранньому моделюванню можна виявити помилки в логіці роботи;
- Можливість повторного використання моделей — UML-діаграми можна використовувати при розробці нових версій програмного забезпечення або для інших проєктів.

Недоліки UML:

- Крива навчання — новачкам може бути складно швидко опанувати всі типи діаграм;
- Часозатратність — створення повного набору діаграм потребує часу і ресурсів;
- Відсутність автоматичної трансляції в код — діаграми не завжди однозначно відображають логіку реалізації.

Дані діаграми сприяють узгодженню вимог між розробниками та зацікавленими сторонами, дозволяють краще зрозуміти взаємодію компонентів системи та змоделювати їхню поведінку до початку реалізації.

2.4.1 Розробка UML діаграми діяльності

Для наочного представлення логіки роботи системи прогнозування фінансових рядів було побудовано UML діаграму діяльності. Вона описує основні дії користувача та обробку даних у процесі формування прогнозу з використанням моделей машинного навчання.

Діаграма починається з введення користувачем основних параметрів: тікера, діапазону дат, валюти та вибраної моделі. Далі система перевіряє, чи завантажено CSV-файл з історичними даними. Якщо файл завантажено, здійснюється його обробка. У випадку його відсутності система автоматично отримує необхідні дані з сервісу Yahoo Finance.

Наступним кроком є натискання користувачем кнопки "Прогнозувати". Після цього виконується перевірка валідності вхідних даних. Якщо дані виявляються некоректними, система відображає повідомлення про помилку. У протилежному випадку запускається процес навчання моделі машинного навчання, після чого відбувається оцінка її якості.

Завершальним етапом є візуалізація результатів прогнозування, а також відповідних метрик якості моделі.

UML діаграма діяльності зображена на рисунку 2.2:

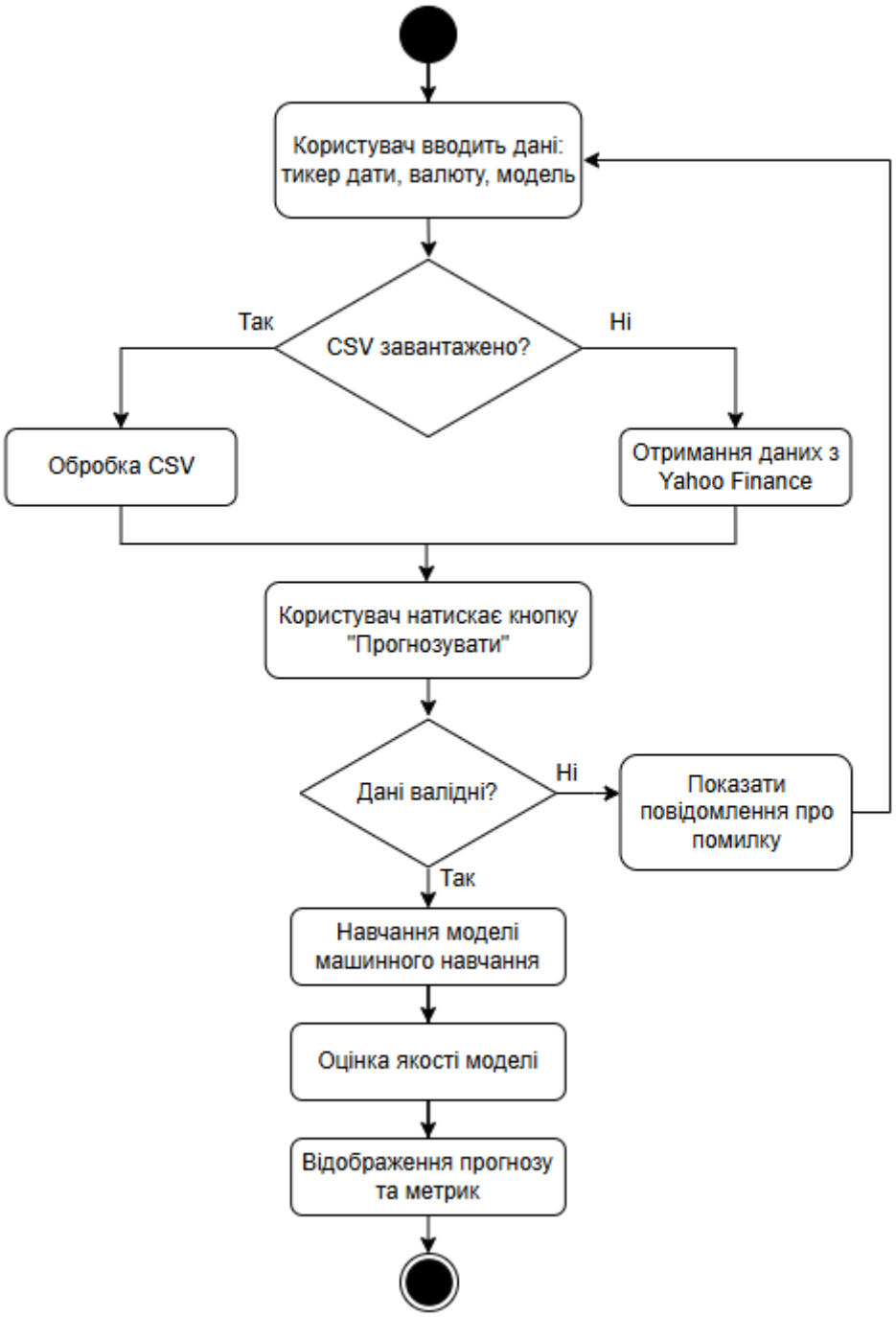


Рисунок 2.2 – UML діаграма діяльності

Дана UML діаграма діяльності дозволяє краще зрозуміти логіку взаємодії користувача з інтерфейсом системи, а також внутрішню послідовність обробки даних і виконання машинного навчання. Вона є важливим етапом під час проєктування та реалізації клієнт-серверної системи для автоматизації прогнозування фінансових рядів.

2.4.2 Розробка UML діаграми послідовності

UML-діаграма послідовності слугує для візуалізації того, як об'єкти або компоненти системи взаємодіють між собою у визначеній послідовності. Вона демонструє обмін повідомленнями між елементами протягом часу, дозволяючи краще зрозуміти порядок виконання дій, взаємозв’язки між частинами системи та вчасно виявити можливі логічні помилки або неточності.

Учасники діаграми:

- Користувач — ініціює процес прогнозування шляхом введення необхідних параметрів, зокрема тікера фінансового активу, інтервалу дат, моделі прогнозування та вибраної валюти.
- Інтерфейс — виступає посередником між користувачем та серверною частиною, забезпечуючи передачу введених даних та відображення результатів.
- Сервер — відповідає за обробку даних, перевірку їх коректності, виклики до зовнішніх сервісів і ML-модуля, а також за повернення результатів.
- ML Модуль — виконує безпосереднє навчання моделі машинного навчання, оцінку її якості та формування прогнозу.
- Yahoo Finance API — зовнішнє джерело фінансових даних, яке використовується у випадку відсутності локального файлу з даними.

Опис послідовності дій:

1. Ініціалізація: Користувач вводить параметри прогнозування через графічний інтерфейс. Ці параметри передаються до серверної частини для подальшої обробки.
2. Перевірка джерела даних: Сервер виконує перевірку наявності локального файлу (CSV). У разі його наявності дані обробляються без звернення до зовнішніх джерел. Якщо файл відсутній, сервер ініціює запит до ML-модуля на завантаження даних, який у свою чергу звертається до Yahoo Finance API.

3. Отримання та передача даних: Завантажені або оброблені дані повертаються через сервер до інтерфейсу, де користувач активує функцію прогнозування.
4. Валідація: Сервер проводить валідацію отриманих даних. У разі невідповідності вхідним вимогам система повертає повідомлення про помилку. У разі успішної перевірки дані передаються до ML-модуля.
5. Моделювання: ML-модуль проводить навчання вибраної моделі на основі переданих даних, здійснює її оцінку та формує прогноз разом із відповідними метриками точності.
6. Візуалізація результатів: Після завершення обробки результати передаються назад до інтерфейсу, де користувачу відображається графік прогнозу та оцінка моделі.

UML діаграма послідовності зображена на рисунку 2.3:

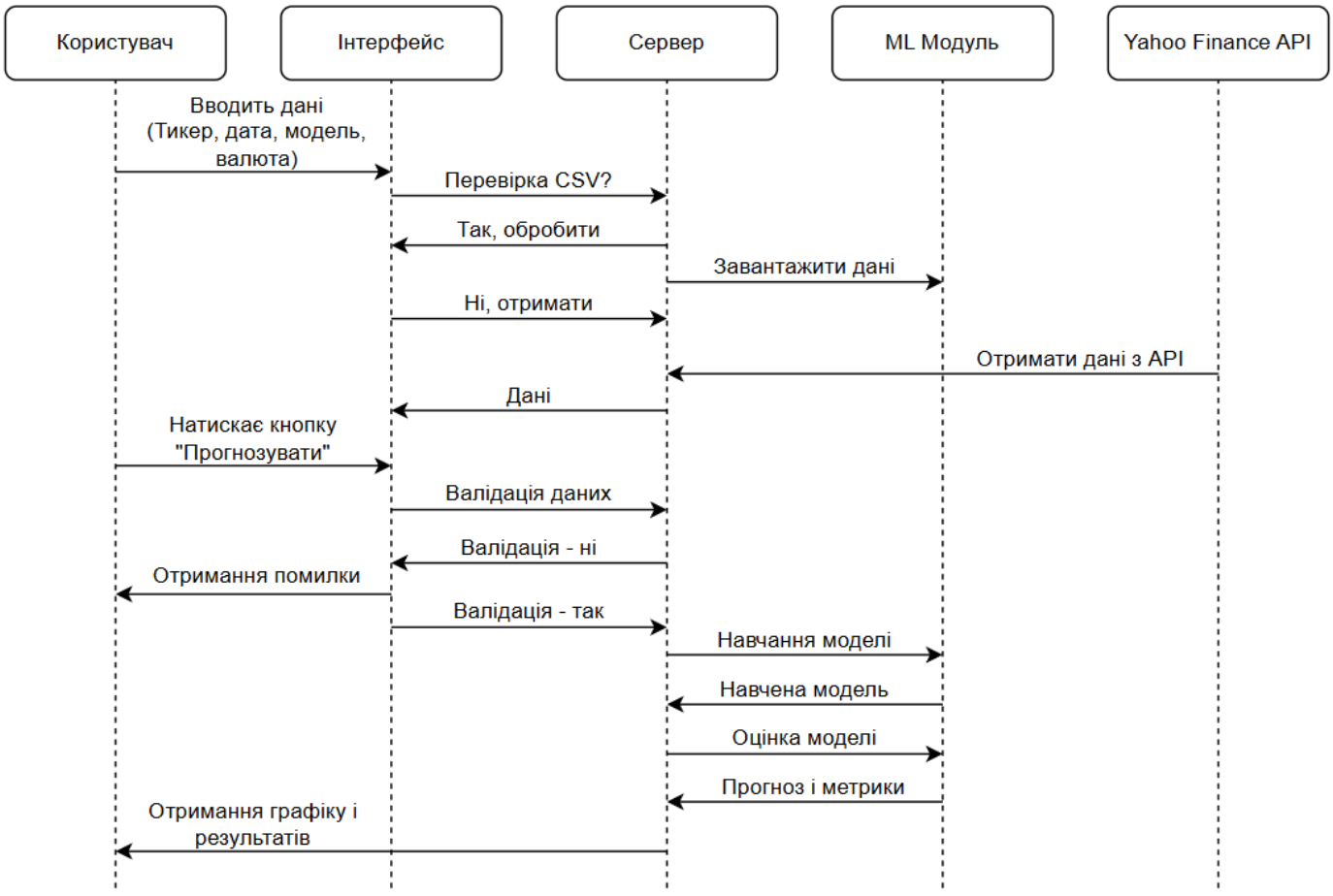


Рисунок 2.3 - UML діаграма послідовності

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Реалізація функціональних модулів

Програмне забезпечення реалізоване як веб-додаток з інтерактивним інтерфейсом для аналізу та прогнозування фінансових часових рядів. Уся логіка додатку структурована у вигляді функціональних модулів, кожен з яких виконує конкретну роль у загальному програмному циклі. Нижче подано опис основних модулів системи.

Модуль завантаження та попередньої обробки даних

Цей модуль відповідає за отримання вхідних даних у вигляді CSV-файлу або через API yfinance. Він також виконує базову обробку даних: перетворення дати, створення ознак і цільової змінної, видалення порожніх значень та підготовку до навчання моделі.

Основні функції:

- Завантаження CSV-файлу або отримання історичних даних акцій.
- Перетворення типів даних (наприклад, Date → datetime).
- Побудова нової цільової змінної Tomorrow — зсув значення Close на 1 день уперед.
- Розділення даних на ознаки X і цільову змінну y.
- Розбиття на тренувальну та тестову вибірки.

Фрагмент коду модуля:

```
if contents:
    df = pd.read_csv(io.StringIO(contents.split(',')[1]))
    df['Date'] = pd.to_datetime(df['Date'])
    df.set_index('Date', inplace=True)
elif selected_ticker:
    df = yf.download(selected_ticker, start=start_date, end=end_date)
    df.dropna(inplace=True)
```

```
df['Tomorrow'] = df['Close'].shift(-1)
df.dropna(inplace=True)
X = df[['Open', 'High', 'Low', 'Close', 'Volume']]
y = df['Tomorrow']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Коментар до реалізації:

- Завантаження реалізовано умовно: якщо файл є — використовуються локальні дані, інакше — звернення до API.
- Цільова змінна Tomorrow дозволяє створити модель для прогнозування наступного дня.
- Метод train_test_split гарантує узгоджене розділення для навчання та перевірки.

Модуль побудови та тренування моделей машинного навчання

Цей модуль відповідає за створення та навчання моделей. Користувач має змогу вибрати один із кількох алгоритмів регресії, які будуть натреновані на підготовлених даних.

Основні функції:

- Отримання вибору моделі з інтерфейсу.
- Ініціалізація відповідного регресора (RandomForest, LinearRegression, SVR, KNN).
- Навчання моделі на тренувальних даних.
- Побудова прогнозів на тестовій вибірці.

Фрагмент коду модуля:

```
if selected_model == 'Random Forest':
    model = RandomForestRegressor(n_estimators=100)
elif selected_model == 'Linear Regression':
    model = LinearRegression()
elif selected_model == 'SVR':
    model = SVR()
```

```
elif selected_model == 'K-Nearest Neighbors':
    model = KNeighborsRegressor()


model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

Коментар до реалізації:

- Підбір моделі здійснюється динамічно, на основі вибору користувача.
- Усі моделі взято з бібліотеки sklearn, що забезпечує модульність та розширюваність.
- fit() та predict() реалізують стандартний цикл машинного навчання.

Модуль валідації та оцінки якості моделей

Модуль оцінки якості відповідає за чисельне визначення точності прогнозу. Це дозволяє користувачеві зробити висновки щодо доцільності застосування тієї чи іншої моделі.

Основні функції:

- Обчислення метрик якості: MSE, RMSE, MAE, R^2.
- Формування текстового звіту для виведення у веб-інтерфейсі.

Фрагмент коду модуля:

```
mse = mean_squared_error(y_test, predictions)
rmse = mse ** 0.5
mae = mean_absolute_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
result_text = f"Оцінка моделі:\nMSE: {mse:.2f}, RMSE: {rmse:.2f}, MAE: {mae:.2f}, R2: {r2:.2f}"
```

Коментар до реалізації:

- Всі метрики є стандартом для задач регресії.
- Значення округлюються до двох знаків для зручності читання.
- Інтерфейс дозволяє швидко змінювати модель та порівнювати результати.

Модуль візуалізації прогнозів

Цей модуль будує інтерактивні графіки, які показують порівняння фактичних та прогнозованих значень. Також він може відображати середнє значення прогнозу та межі довіри.

Основні функції:

- Побудова графіків із використанням бібліотеки plotly.
- Відображення фактичних цін, прогнозу, середнього значення та стандартного відхилення.

Фрагмент коду модуля:

```
fig = go.Figure()
fig.add_trace(go.Scatter(y=y_test.values, name='Фактичне значення'))
fig.add_trace(go.Scatter(y=predictions, name='Прогноз'))
mean = predictions.mean()
std = predictions.std()
fig.add_trace(go.Scatter(y=[mean]*len(predictions), name='Середнє', line=dict(dash='dash')))
fig.add_trace(go.Scatter(y=[mean + std]*len(predictions), name='Верхня межа', line=dict(dash='dot')))
fig.add_trace(go.Scatter(y=[mean - std]*len(predictions), name='Нижня межа', line=dict(dash='dot')))
```

Коментар до реалізації:

- plotly забезпечує високий рівень інтерактивності (масштабування, наведення, експорт).
- Додаткові лінії (середнє, межі) покращують інтерпретацію результатів.
- Можлива подальша адаптація для порівняння кількох моделей одночасно.

Модуль автоматизації (пайплайни)

У даній реалізації автоматизація виконана частково через централізований callback. Один великий функціональний блок зв'язує усі етапи роботи моделі: від завантаження даних до побудови графіків.

Основні функції:

- Реагування на взаємодію користувача (вибір дати, моделі, файл, кнопка).
- Об’єднання всіх попередніх модулів в один обробник події.
- Забезпечення циклічного робочого процесу "дані → модель → прогноз → оцінка → графік".

Фрагмент коду:

```
@app.callback(
   [Output('forecast-graph', 'figure'), Output('result-output', 'children')],
   [Input('forecast-button', 'n_clicks')],
   [State('model-dropdown', 'value'), State('date-picker-range', 'start_date'),
    State('date-picker-range', 'end_date'), State('ticker-dropdown', 'value'),
    State('upload-data', 'contents')]
)
def update_forecast(n_clicks, selected_model, start_date, end_date, selected_ticker, contents):
   ...
   return fig, result_text
```

Коментар до реалізації:

- Callback об’єднує логіку взаємодії користувача з інтерфейсом та модульну обробку.
- Єдина точка входу в бізнес-логіку дозволяє легко модифікувати або розширювати функціонал.
- Для повної автоматизації можна винести етапи в окремі функції або скрипти (наприклад, у вигляді пайплайнів sklearn.pipeline або модулів mlflow).

Модуль обробки помилок

Модуль обробки помилок є необхідним для забезпечення стабільної роботи додатку, щоб обробляти всі непередбачувані ситуації, такі як проблеми з даними або несправності під час виконання.

Основні функції:

- Логування помилок для подальшого аналізу.
- Виведення повідомлень для користувача з поясненням проблеми.
- Обробка помилок на етапах завантаження даних, тренування моделі, прогнозування тощо.

Фрагмент коду модуля:

```
import logging
# Налаштування логування
logging.basicConfig(filename='error_log.txt', level=logging.ERROR)

# Обробка помилки завантаження даних
try:
    df = pd.read_csv(file_path)
except Exception as e:
    logging.error(f"Помилка при завантаженні даних: {str(e)}")
    print("Сталася помилка при завантаженні даних. Перевірте файл.")
# Обробка помилки тренування моделі
try:
    model.fit(X_train, y_train)
except Exception as e:
    logging.error(f"Помилка при тренуванні моделі: {str(e)}")
    print("Сталася помилка при тренуванні моделі. Перевірте вибір моделі та дані.")

# Загальна обробка для прогнозу
try:
    predictions = model.predict(X_test)
except Exception as e:
    logging.error(f"Помилка при прогнозуванні: {str(e)}")
    print("Сталася помилка при прогнозуванні. Перевірте модель та дані.")
```

Коментар до реалізації:

- Використано стандартну бібліотеку logging для запису помилок у файл. Це дозволяє розробникам та адміністраторам переглядати помилки після виконання програми.
- Для кожного етапу (завантаження даних, тренування, прогнозування) передбачена обробка можливих помилок.
- Повідомлення для користувача виводяться на екран, щоб він міг вчасно отримати інформацію про помилку та вжити відповідних заходів.

Усі функції, описані в попередніх модулях, організовані у зручному та інтуїтивно зрозумілому інтерфейсі, що дозволяє користувачеві без зусиль здійснювати прогнозування фінансових рядів, працювати з даними та переглядати результати. Далі ми детально розглянемо, як виглядає інтерфейс програми на різних етапах її використання.

Нижче наведено кілька скріншотів, які демонструють вигляд програми під час взаємодії з користувачем, включаючи введення даних, вибір моделі, результати прогнозування та візуалізацію графіків.

На рисунку 3.1 представлена головна сторінка розробленої системи:

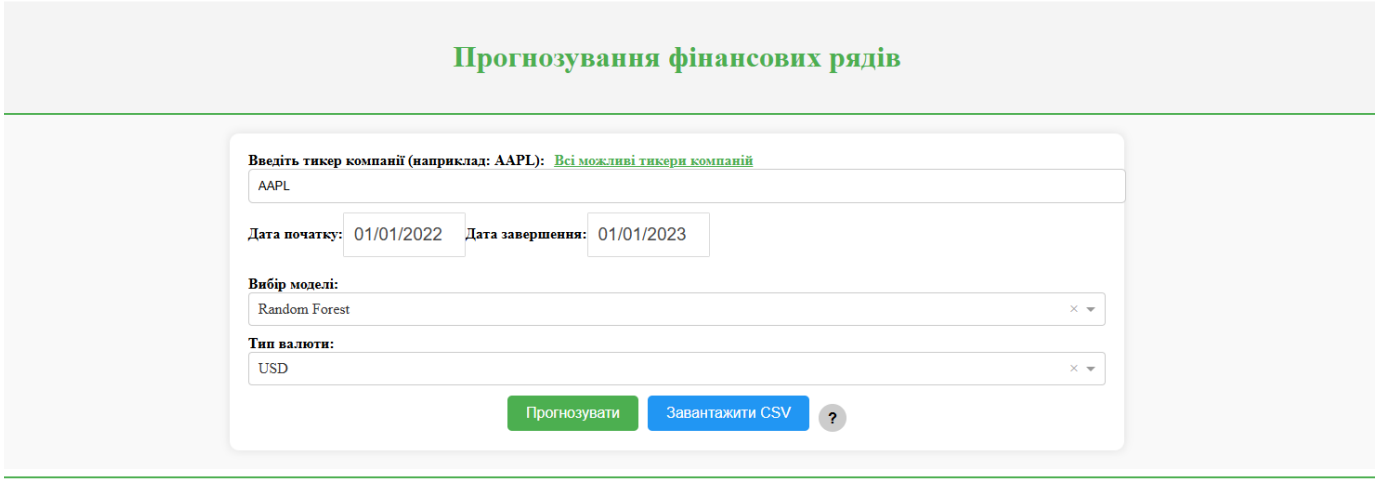


Рисунок 3.1 – Головна сторінка

На рисунку3.2 представлена сторінка прогнозування фінансових рядів для тикеру компанії Apple (AAPL):

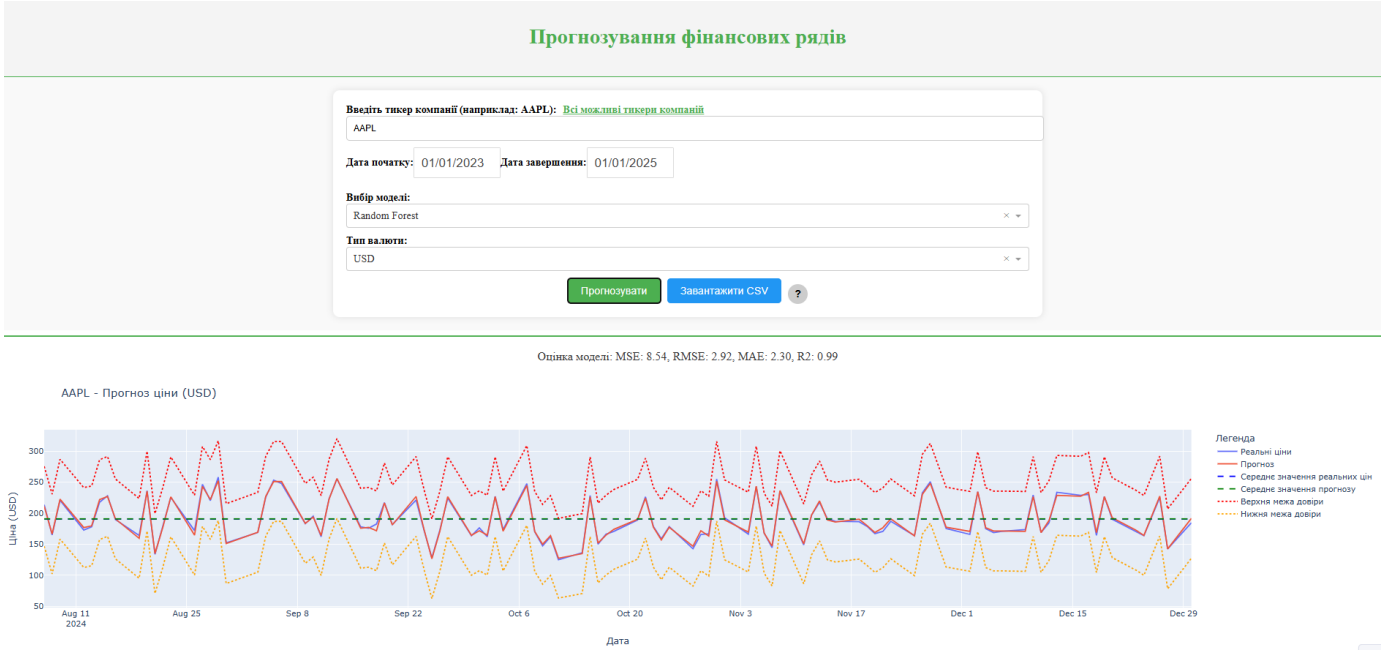


Рисунок 3.2 – Сторінка з активним прогнозуванням

3.2 Тестування програмного забезпечення

Тестування програмного забезпечення — це невід'ємний етап життєвого циклу розробки, що має на меті виявлення помилок, перевірку відповідності системи заданим вимогам і забезпечення надійності, коректності та зручності у використанні. У процесі реалізації системи прогнозування фінансових рядів було проведено функціональне, модульне та інтеграційне тестування з акцентом на точність обчислень та стабільність інтерфейсу.

Розглянемо наступні теоретичні основи тестування. Виділяють кілька основних видів тестування, що мають застосування до інформаційних систем:

- Модульне тестування (unit testing) — перевіряє окремі компоненти коду (функції, методи) на правильність роботи ізольовано від інших частин програми.
- Інтеграційне тестування (integration testing) — оцінює, як взаємодіють між собою компоненти системи після об'єднання.

- Системне тестування (system testing) — перевіряє всю систему в цілому, як єдине ціле, на відповідність початковим вимогам.
- Функціональне тестування — оцінює, чи виконує система заявлені функції, наприклад, коректність відображення графіків, обробку вхідних даних, роботу моделей.
- Регресійне тестування — застосовується після змін або оновлень системи для перевірки того, що нововведення не порушили існуючу функціональність.

Також існують два основні підходи до тестування:

- Чорний ящик (Black-box testing) — тестування без знання внутрішньої структури коду.
- Білий ящик (White-box testing) — тестування із доступом до внутрішньої логіки програми.

Система, розроблена для прогнозування фінансових рядів, має низку критичних компонентів, які потребують ретельної перевірки:

- Модуль завантаження даних — перевірявся на обробку коректного та некоректного вводу (неіснуючі тікери, порожні або пошкоджені CSV-файли).
- Модуль передобробки — тестувався на випадки відсутніх значень, пропусків у часовому ряді, обчислення скользьких середніх (rolling mean).
- Моделі машинного навчання (RandomForest, SVR, KNN, LinearRegression) — тестувалися на узгодженість виводу, стабільність прогнозу, коректність розділення на тренувальні та тестові вибірки.
- Візуалізація (Plotly/Dash) — перевірялась відповідність графіків значенням, точність масштабування, коректна побудова осей, підписів та стилів.
- Користувацький інтерфейс — перевірявся на зручність, читабельність, реакцію на помилки вводу.

У межах розробки системи прогнозування фінансових рядів було застосовано різні підходи до тестування: від модульного до функціонального та інтеграційного. Особливу увагу приділено перевірці коректної роботи моделей машинного навчання, точності обробки даних та стійкості інтерфейсу до некоректного вводу.

Після реалізації ключових модулів системи було сформовано набір тест-кейсів (див. таблицю 3.1), де кожен перевіряє окремий функціональний елемент або сценарій використання. Нижче, в таблиці 5, наведено приклади типових тестів із коротким описом вхідних умов, очікуваних результатів та статусу їх виконання.

Для кожного з них у ході практичного тестування було зроблено відповідні скріншоти (див. ничже), що підтверджують результат:

Таблиця 3.1 – Тестові випадки

Тест-кейс	Вхідні дані	Очікуваний результат	Статус
TC_01	Тікер “AAPL” + валідний діапазон дат	Дані успішно завантажені	Успішно Рис. 3.3
TC_02	Порожній файл CSV	Повідомлення про помилку завантаження	Успішно Рис. 3.4
TC_03	Модель RandomForest, 5 днів прогнозу, Euro	Побудований графік, дані у допустимому діапазоні	Успішно Рис. 3.5
TC_04	Некоректна модель (наприклад, “None”)	Повідомлення про необхідність вибору моделі	Успішно Рис. 3.6
TC_05	Некоректна вибрана дата	Повідомлення про необхідність вибору дати	Успішно Рис. 3.7
TC_06	Некоректна вибраний тип валюти	Повідомлення про необхідність вибору типу валюти	Успішно Рис. 3.8

TC_07	Некоректна вибраний тикера	Повідомлення про необхідність вибору тикеру	Успішно Рис. 3.9
TC_08	Натиск на кнопку «Всі можливі тикери компаній»	Перехід на сайт, зі списком всіх можливих тикерів компаній	Успішно Рис. 3.10
TC_09	Натиск на підказку «?»	Поява модульного меню з поясненням метрик оцінки	Успішно Рис. 3.11
TC_10	Вимкнення прогнозів на легенді графіку	Відсутність ліній на графіку	Успішно Рис. 3.12

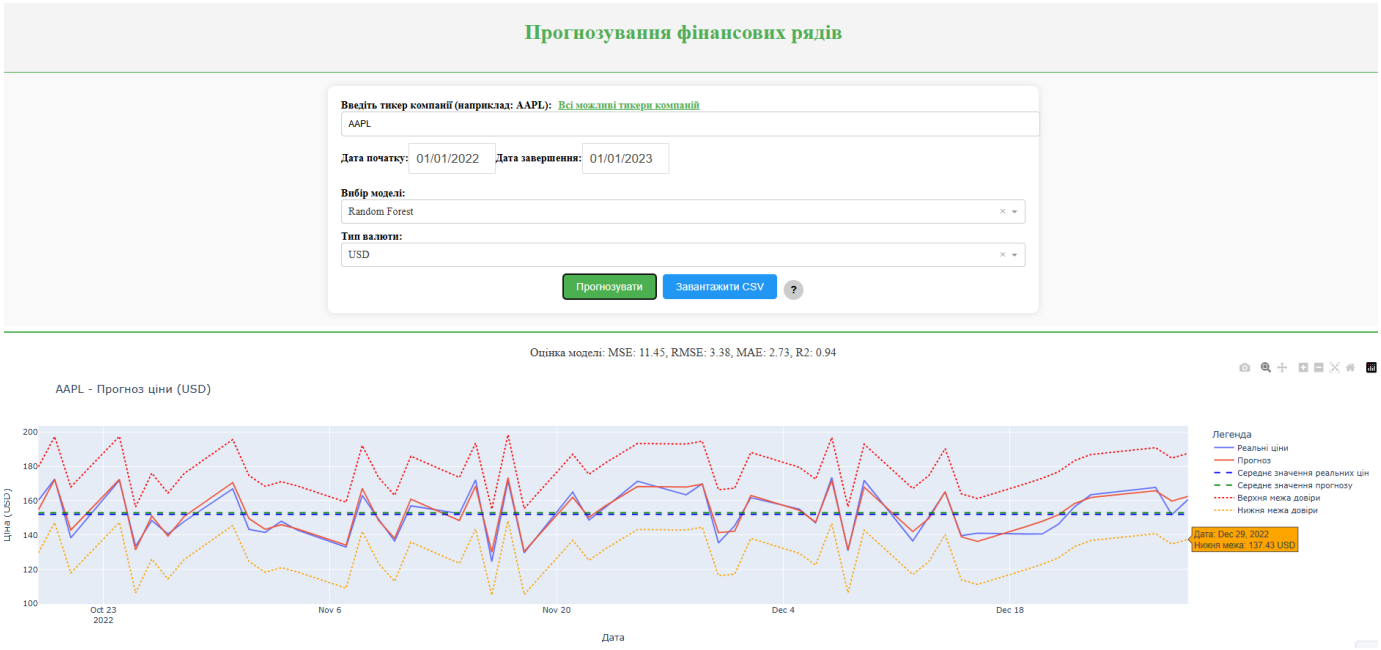


Рисунок 3.3 - TC_01

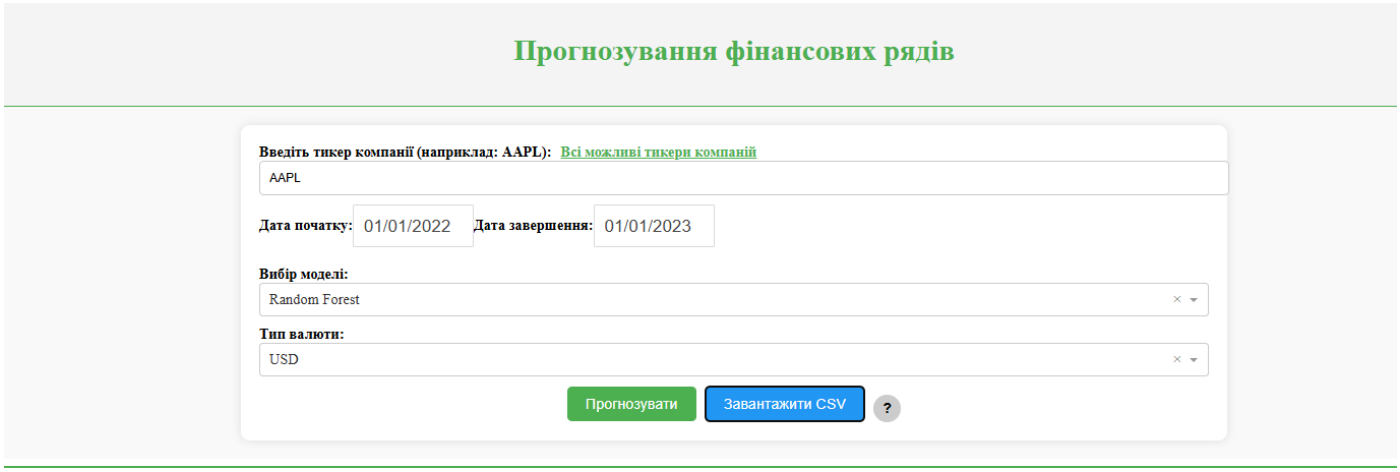


Рисунок 3.4 - TC_02

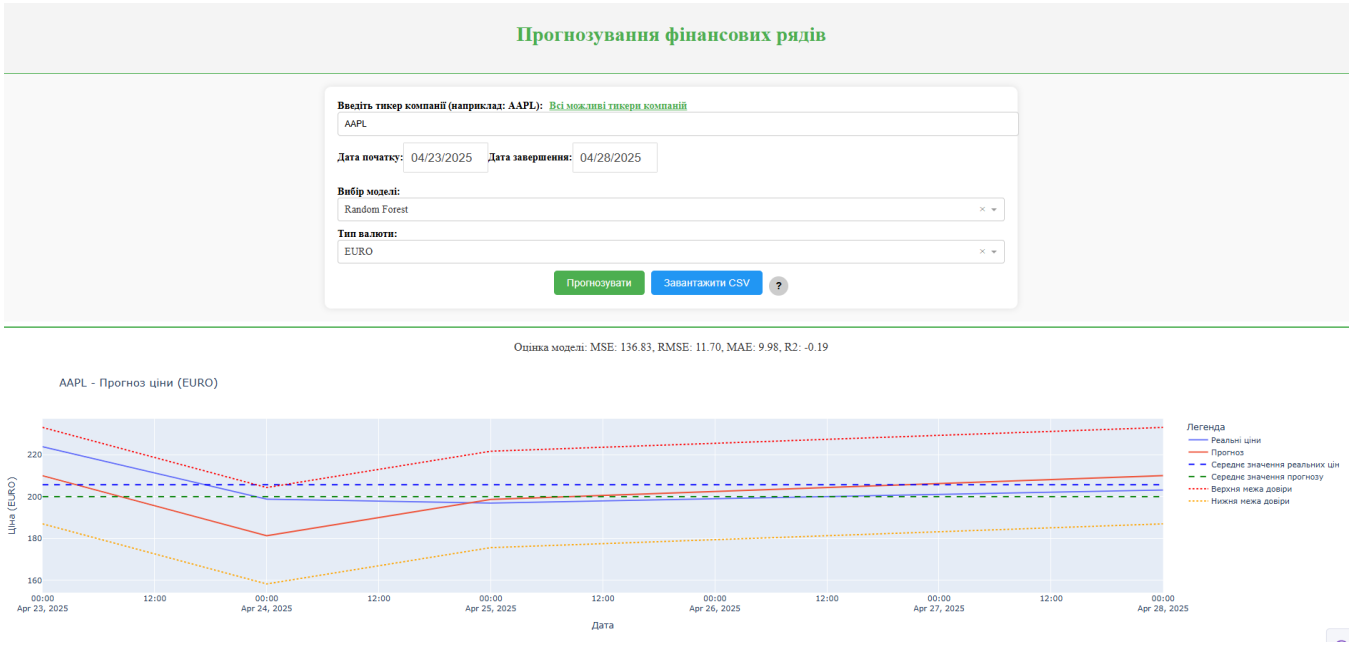


Рисунок 3.5 - TC_03

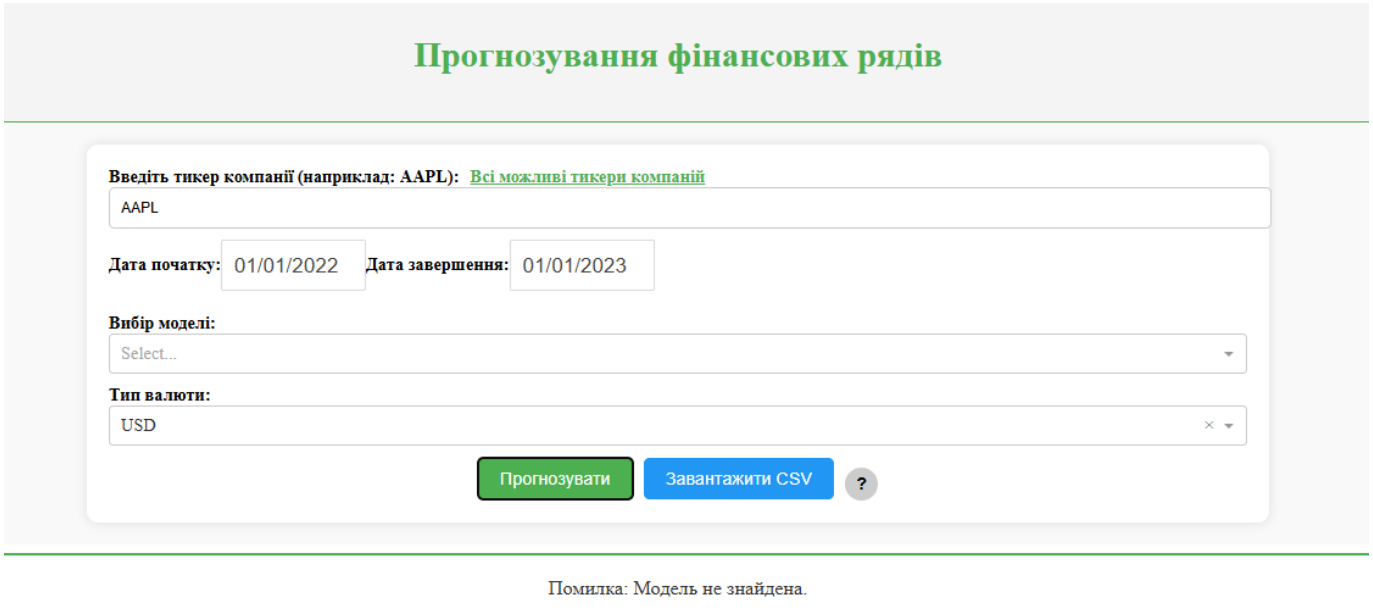


Рисунок 3.6 - TC_04

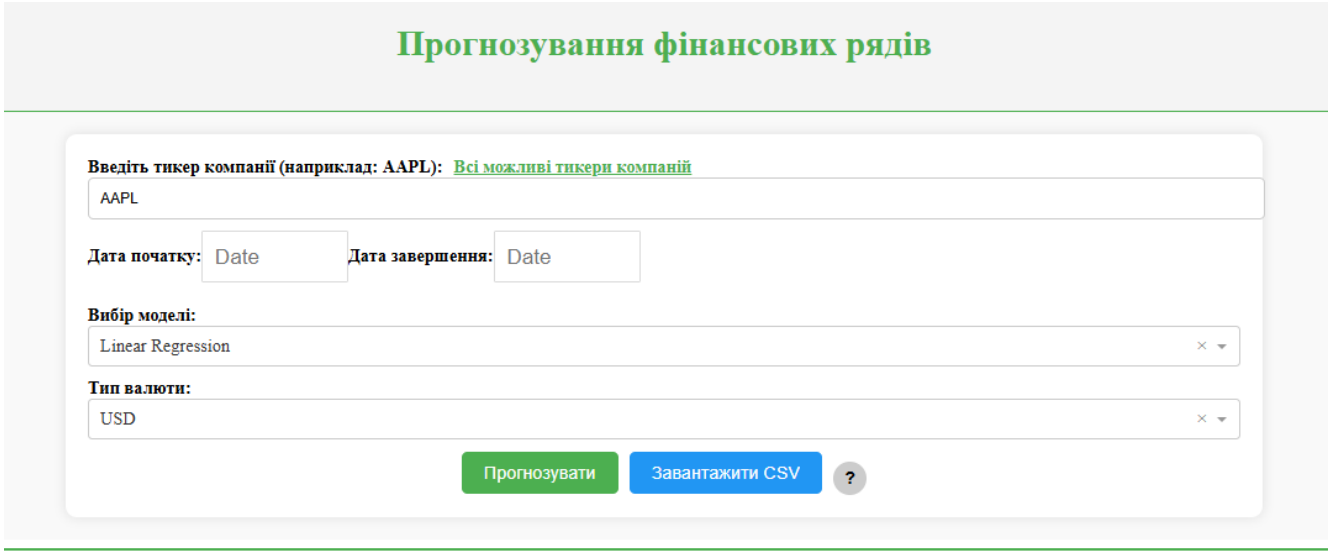


Рисунок 3.7 - TC_05

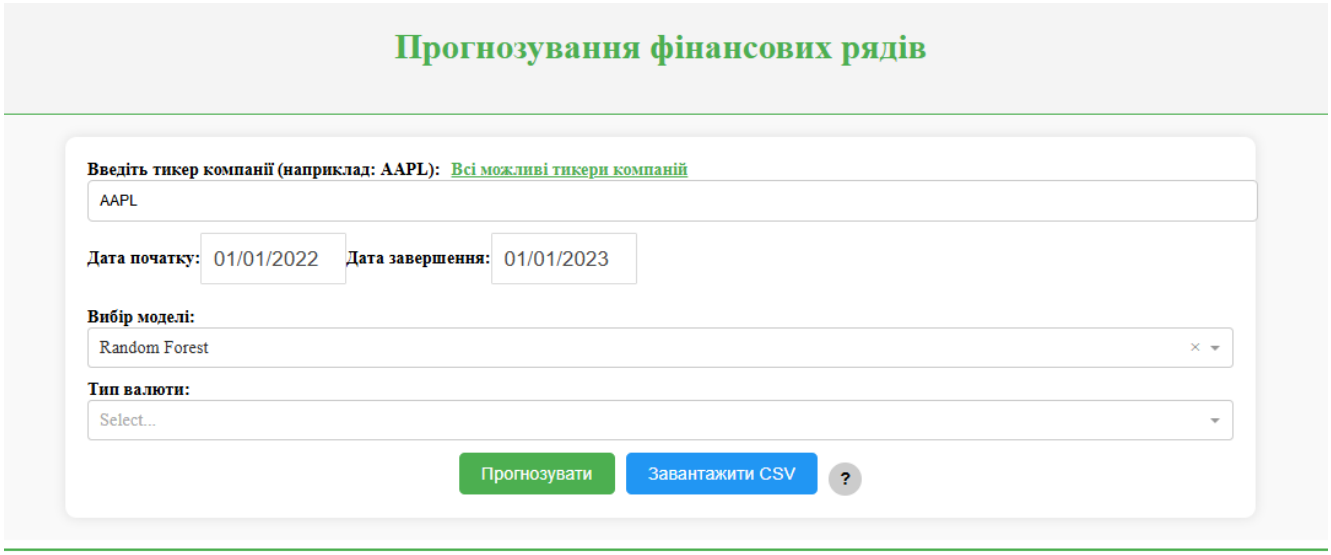


Рисунок 3.8 - TC_06

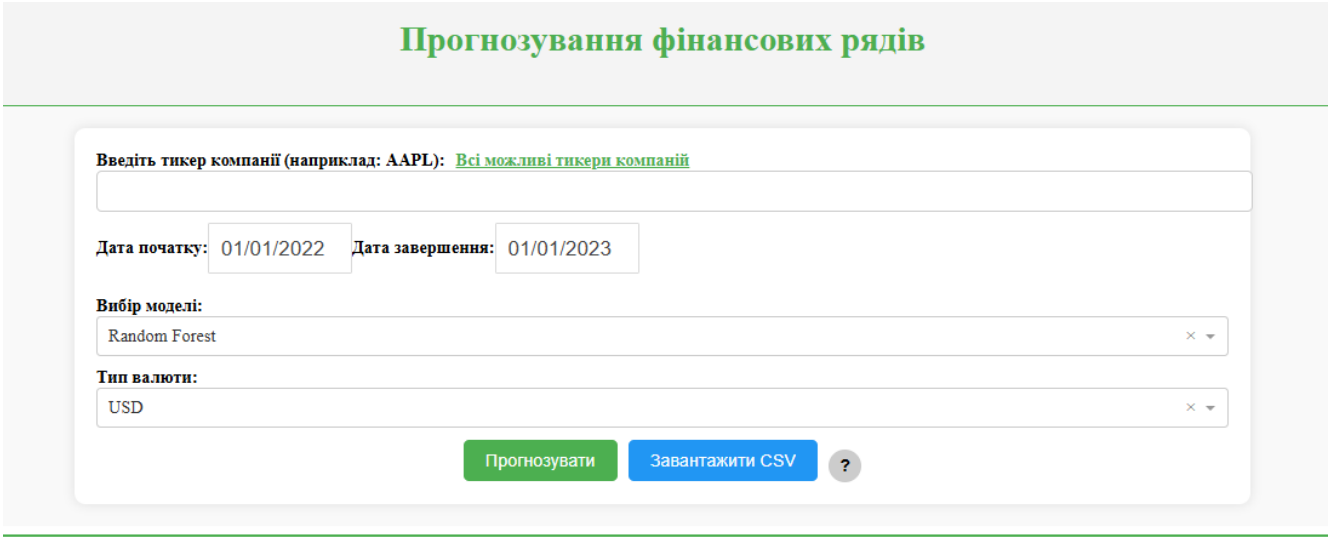


Рисунок 3.9 - TC_07

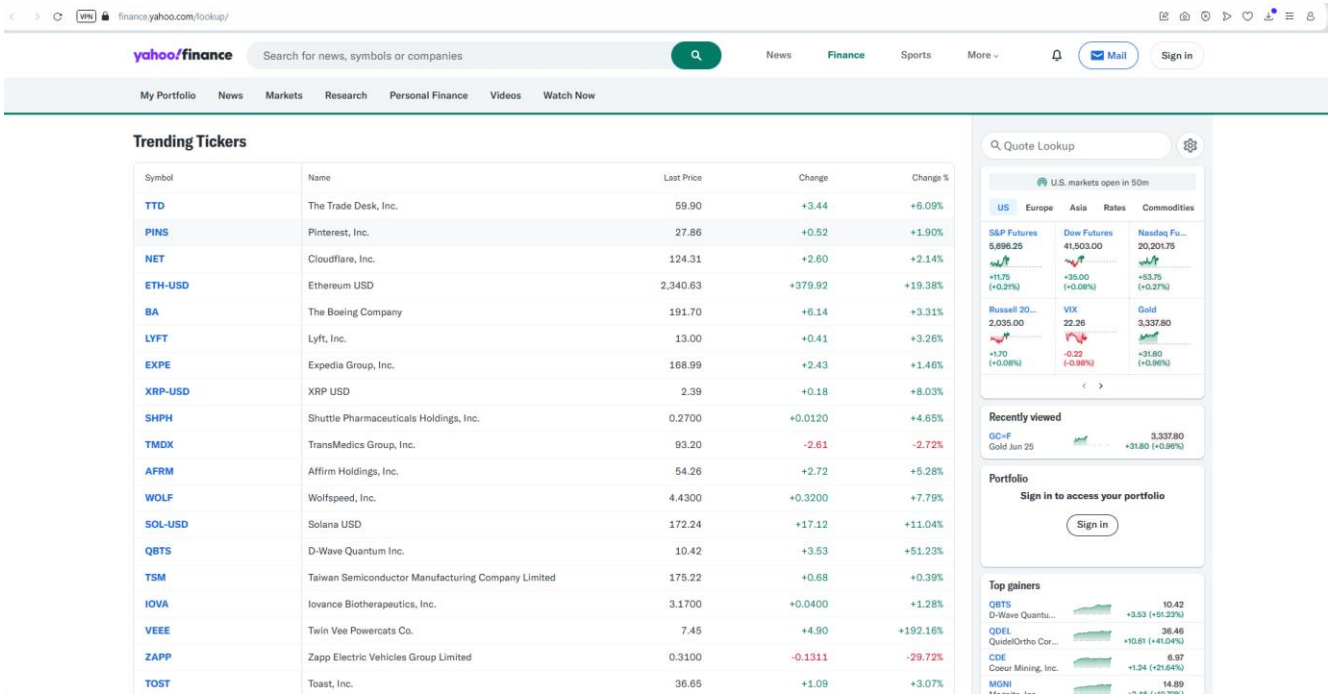


Рисунок 3.10 - TC_08

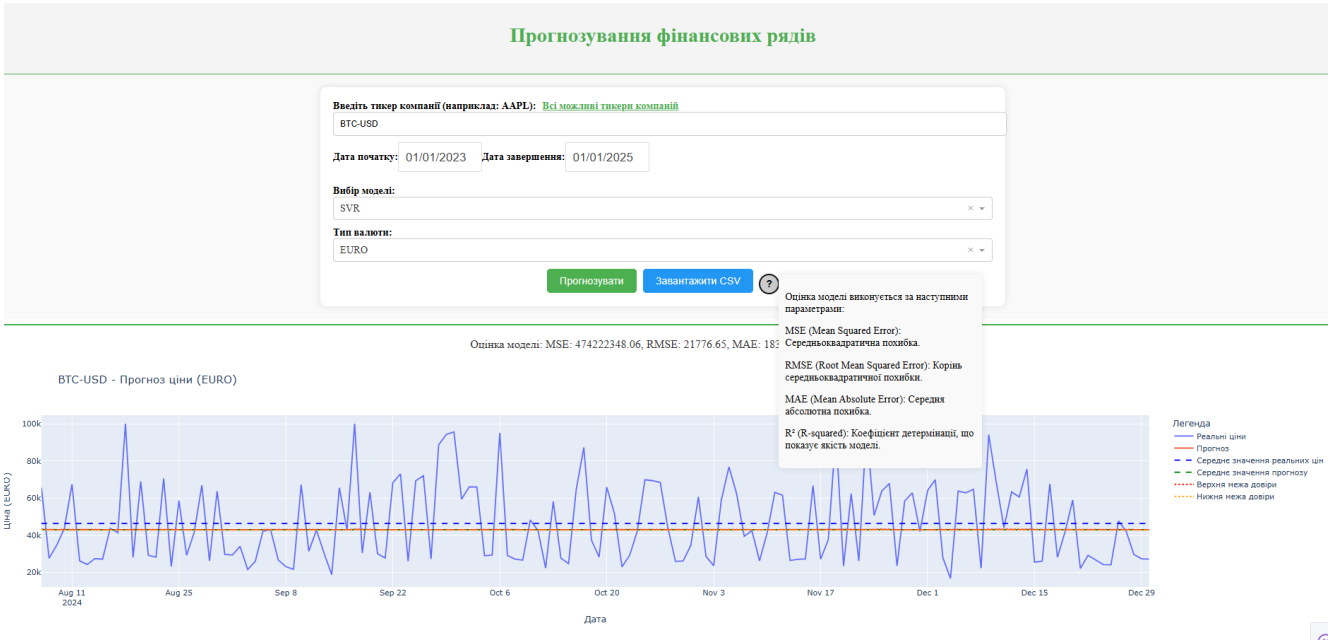


Рисунок 3.11 - TC_09

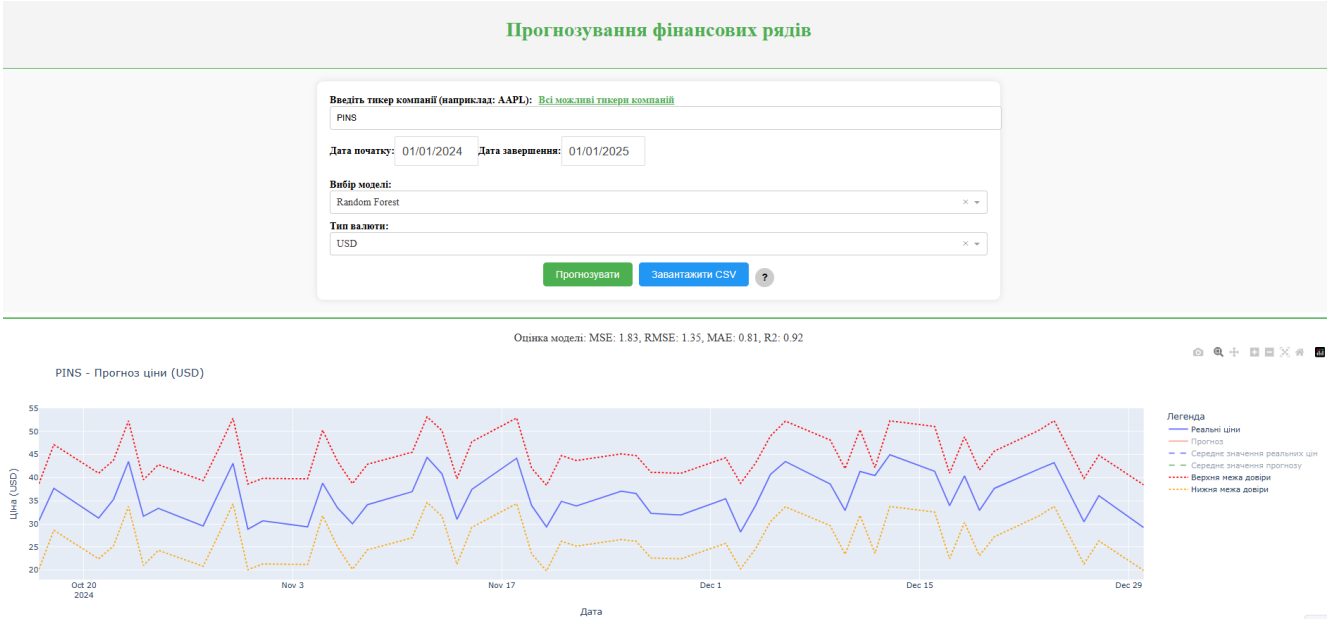


Рисунок 3.12 - TC_10

У ході тестування не було виявлено помилок, пов’язаних з обробкою відсутніх значень у даних та неочікуваними форматами дати. Система показала стабільність, коректну роботу з різними наборами даних та відповідність функціоналу заданим вимогам.

ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи було досліджено, обґрунтовано та реалізовано підхід до автоматизації прогнозування фінансових часових рядів з використанням методів машинного навчання. У ході дослідження автор здійснив комплексний аналіз предметної області, охарактеризував специфіку фінансових рядів як об’єкта автоматизації та виявив ключові виклики, пов’язані з їх моделюванням — стохастичність, нестаціонарність, волатильність, чутливість до зовнішніх чинників тощо.

На основі вивчення сучасних підходів до прогнозування було проведено порівняння класичних статистичних методів (ARIMA, SARIMA, експоненційне згладжування) із сучасними алгоритмами машинного навчання (Random Forest, SVR, XGBoost, LSTM тощо). Доведено, що ML-моделі здатні краще адаптуватися до складної природи фінансових даних та демонструють вищу точність прогнозування у динамічних ринкових умовах.

У межах технічної реалізації було сформульовано чітке завдання проєкту, обґрунтовано вибір архітектури, інструментів та стеку технологій. Основною мовою реалізації обрано Python, що дозволило ефективно поєднати інструменти обробки даних, моделювання та побудови інтерфейсу. Було розроблено інтерактивну веб-орієнтовану систему на основі Streamlit, яка реалізує повний цикл прогнозування: від збору даних з Yahoo Finance до візуалізації результатів і оцінки якості моделей.

Таким чином, у роботі реалізовано практично рішення, що автоматизує аналітичний процес і дозволяє користувачеві без глибоких технічних знань отримувати точні прогнози фінансових показників. Результати можуть бути застосовані в інвестиційній діяльності, трейдингу, банківській сфері та інших фінансово-аналітичних системах. Отриманий програмний продукт демонструє високий рівень адаптивності, гнучкості та потенціал до масштабування у майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Базилевич В. Д. Макроекономіка : підручник / В. Д. Базилевич. — Київ : Знання, 2020. — 678 с. — ISBN 978-617-07-0697-7.
2. Білик М. Д. Аналіз часових рядів : навч. посіб. / М. Д. Білик. — Київ : Центр учбової літератури, 2011. — 256 с. — ISBN 978-611-01-0104-3.
3. Гнилянська Л. М. Статистичні методи обробки економічної інформації / Л. М. Гнилянська. — Київ : Центр учбової літератури, 2015. — 304 с. — ISBN 978-617-673-418-3.
4. Стецюк П. І. Фінансова аналітика : навч. посіб. / П. І. Стецюк. — Львів : ЛНУ ім. І. Франка, 2016. — 220 с.
5. Яремчук О. Ю. Аналіз ринку цінних паперів : навч. посіб. / О. Ю. Яремчук. — Київ : КНЕУ, 2017. — 288 с.
6. Кравченко В. І. Інформаційні технології в економіці : навч. посіб. / В. І. Кравченко. — Київ : Центр учбової літератури, 2016. — 320 с.
7. Поліщук О. С. Інформаційні системи та технології на підприємстві / О. С. Поліщук. — Харків : ХНАДУ, 2018. — 200 с.
8. Романенко С. В. Інформаційні системи у фінансах / С. В. Романенко. — Київ : НАДУ, 2013. — 280 с.
9. Цапар В. С., Жученко О. А. Сучасні програмні засоби автоматизації технологічних процесів / В. С. Цапар, О. А. Жученко. — Київ : НТУУ «КПІ», 2012. — 128 с.
10. Герасименко І. В. Прогнозування соціально-економічних процесів : навч. посіб. / І. В. Герасименко. — Київ : Центр учбової літератури, 2014. — 312 с.
11. Дьяків Г. В. Теорія ймовірностей та математична статистика / Г. В. Дьяків. — Київ : Каравела, 2010. — 368 с. — ISBN 966-8019-48-6.
12. Качала Т. С. Методи статистичного аналізу : навч. посіб. / Т. С. Качала. — Тернопіль : ТНЕУ, 2014. — 290 с.
13. Назарчук О. П. Економетрика : навч. посіб. / О. П. Назарчук. — Львів : Видавництво ЛНУ, 2012. — 288 с.

14. Шевчук Р. В. Статистика : навч. посіб. / Р. В. Шевчук. — Львів : Новий Світ – 2000, 2013. — 384 с.
15. Дьяконов В. П. Методи машинного навчання / В. П. Дьяконов. — Київ : Наука і освіта, 2019. — 356 с.
16. Жуков Ю. А. Основи штучного інтелекту / Ю. А. Жуков. — Харків : ХНУРЕ, 2021. — 178 с.
17. Кириченко О. А. Машинне навчання в економіці : навч. посіб. / О. А. Кириченко. — Київ : КНЕУ, 2022. — 312 с.
18. Курган М. М. Основи Data Science : навч. посіб. / М. М. Курган. — Львів : ЛНУ, 2021. — 294 с.
19. Ящук О. І. Вступ до штучного інтелекту / О. І. Ящук. — Рівне : НУВГП, 2020. — 204 с.
20. MetaTrader5: A powerful platform for Forex and Exchange markets. URL: https://www.metatrader5.com
21. TraidingView: The best trades require research. URL: https://www.tradingview.com
22. Forecastr: Best Financial Forecasting Software & Expert CFO. URL: https://www.forecastr.co
23. Іванна Ткачук Яку мову програмування обрати початківцю. Поради досвідчених розробників. URL: https://dou.ua/lenta/articles/choosing-programming-language-for-junior/
24. Місюра С. В. Хмарні технології в сучасному програмуванні / С. В. Місюра. — Київ : Слово, 2022. — 146 с.
25. Носаль В. Фреймворки та середовища розробки вебзастосунків / В. Носаль. — Тернопіль : ТНТУ, 2020. — 198 с.
26. Троян М. Ю. Архітектура програмного забезпечення / М. Ю. Троян. — Львів : ЛНУ, 2020. — 270 с.
27. Юлія Каграманова Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти. URL: https://dou.ua/forums/topic/40575/
28. Scikit-learn: Machine Learning in Python. — URL: https://scikit-learn.org
29. Plotly Graphing Libraries for Python. — URL: https://plotly.com/python

30. Yahoo Finance API Documentation. — URL: https://www.yahoofinanceapi.com
31. Brownlee J. Deep Learning for Time Series Forecasting. — Machine Learning Mastery, 2018. — 573 p.
32. Zhang G., Eddy Patuwo B., Hu M. Forecasting with artificial neural networks: The state of the art // International Journal of Forecasting. — 1998. — №14(1). — С. 35–62.
33. Bishop C. M. Pattern Recognition and Machine Learning. — Springer, 2006. — 738 p.
34. Paszke A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library // Advances in Neural Information Processing Systems. — 2019. — №32.
35. Bandara, K., Bergmeir, C., & Smyl, S. (2020). Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. Expert Systems with Applications, 140.
36. Brownlee, J. (2020). Machine Learning Mastery with Python. Machine Learning Mastery.

ДОДАТКИ

ДОДАТОК А

(обов'язковий)

Технічне завдання

Вінницький національний технічний університет
Кафедра автоматизації та інтелектуальних інформаційних технологій

ЗАТВЕРДЖУЮ
завідувача кафедри АІІТ
д.т.н., професор Олег БІСІКАЛО

_____
(підпис)

«_23_» _березня_ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на бакалаврську кваліфікаційну роботу
«Автоматизація прогнозування фінансових рядів на основі машинного навчання»
08-31.БКР.010.02.000 ТЗ

Керівник бакалаврської кваліфікаційної роботи
к.т.н., доц. Роман МАСЛІЙ

_____
(підпис)

«_23_» _травня__ 2025 р.

Розробив студент гр. 1АКІТ-21б
_____Олексій СИРОЄЖКО

_____
(підпис)

«_23_» _травня__ 2025 р.

Вінниця ВНТУ – 2025 рік

1 Підстава для проведення робіт

Підставою для виконання бакалаврської кваліфікаційної роботи на тему: «Автоматизація прогнозування фінансових рядів на основі машинного навчання» є наказ № 97 від 20.03.2025 р.

Термін виконання робіт:

початок 03.09.2024 р.
кінець 10.06.2025 р.

2 Мета та вихідні дані для проведення робіт

Метою бакалаврської кваліфікаційної роботи є підвищення ефективності прогнозування фінансових рядів шляхом розробки автоматизованої системи прогнозування із використанням методів машинного навчання.

Вихідними даними для проведення робіт є індивідуальне завдання на бакалаврську кваліфікаційну роботу від 20.03.2025 р.

3 Етапи виконання робіт

Виконавцем всіх перерахованих в даному розділі етапів є: студент групи **1АКІТ-21б Сироєжко Олексій Олегович** факультету інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету, а замовником є кафедра автоматизації та інтелектуальних інформаційних технологій.

№ Етапу	Зміст етапу	Строки виконання
Е1	Вибір, узгодження та затвердження теми БКР	01.02 – 11.02
Е2	Аналіз предметної області та конкурентних рішень. Опрацювання літературних джерел.	12.02 – 14.03
Е3	Постановка задач для вирішення в БКР	17.03 – 21.03
Е4	Проектування архітектури системи, вибір технологій розробки	24.03 – 04.04
Е5	Розробка програмного забезпечення	07.04 – 09.05
Е6	Тестування програмного забезпечення	12.05 - 16.05
Е7	Оформлення пояснювальної записки та графічної частини	19.05 – 23.05
Е8	Попередній захист, доопрацювання, та рецензування БКР	26.05 – 30.05
Е9	Захист БКР	20.06 - 20.05

4 Призначення і галузь застосування

Основне призначення даної роботи полягає в підвищенні точності та ефективності аналізу фінансових даних, що дозволяє забезпечити прийняття обґрунтованих рішень у таких сферах, як фінанси, інвестування та управління ризиками. Застосування результатів цієї роботи можливе в банківському секторі для прогнозування валютних курсів, процентних ставок і динаміки кредитування; на фондовому ринку — для передбачення змін цін на акції, індекси та інші фінансові інструменти; у страхуванні — для оцінки ризиків та прогнозування страхових виплат; у сфері бюджетного планування — для моделювання майбутніх доходів і витрат на державному або корпоративному рівні; а також у фінансових аналітичних платформах, де необхідні інтелектуальні інструменти підтримки прийняття рішень. Застосування машинного навчання дозволяє значно підвищити якість аналітики та оптимізувати процеси прийняття рішень у сучасній фінансовій практиці.

5 Технічні дані

5.1 Апаратне забезпечення – комп’ютер/ноутбук.
5.2 Програмне забезпечення та середовище розробки – Windows, Visual Studio Code, Microsoft Office.
5.3 Мова програмування – Python.
5.4 Бібліотеки та фреймворки – Pandas, NumPy, Yfinance, Scikit-learn, Plotly.
5.5 Моделі машинного навчання – Random Forest, Linear Regression, SVR, KNN.

6 Джерела розробки

6.1 Білик М. Д. Аналіз часових рядів : навч. посіб. / М. Д. Білик. — Київ : Центр учбової літератури, 2011. — 256 с. — ISBN 978-611-01-0104-3.
6.2 Герасименко І. В. Прогнозування соціально-економічних процесів : навч. посіб. / І. В. Герасименко. — Київ : Центр учбової літератури, 2014. — 312 с.
6.3 Дьяконов В. П. Методи машинного навчання / В. П. Дьяконов. — Київ : Наука і освіта, 2019. — 356 с.
6.4 Носаль В. Фреймворки та середовища розробки вебзастосунків / В. Носаль. — Тернопіль : ТНТУ, 2020. — 198 с.

Здобувач ст. гр. 1АКІТ-21б _____ Олексій СИРОЄЖКО

ДОДАТОК Б

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

Автоматизація прогнозування фінансових рядів на основі машинного навчання

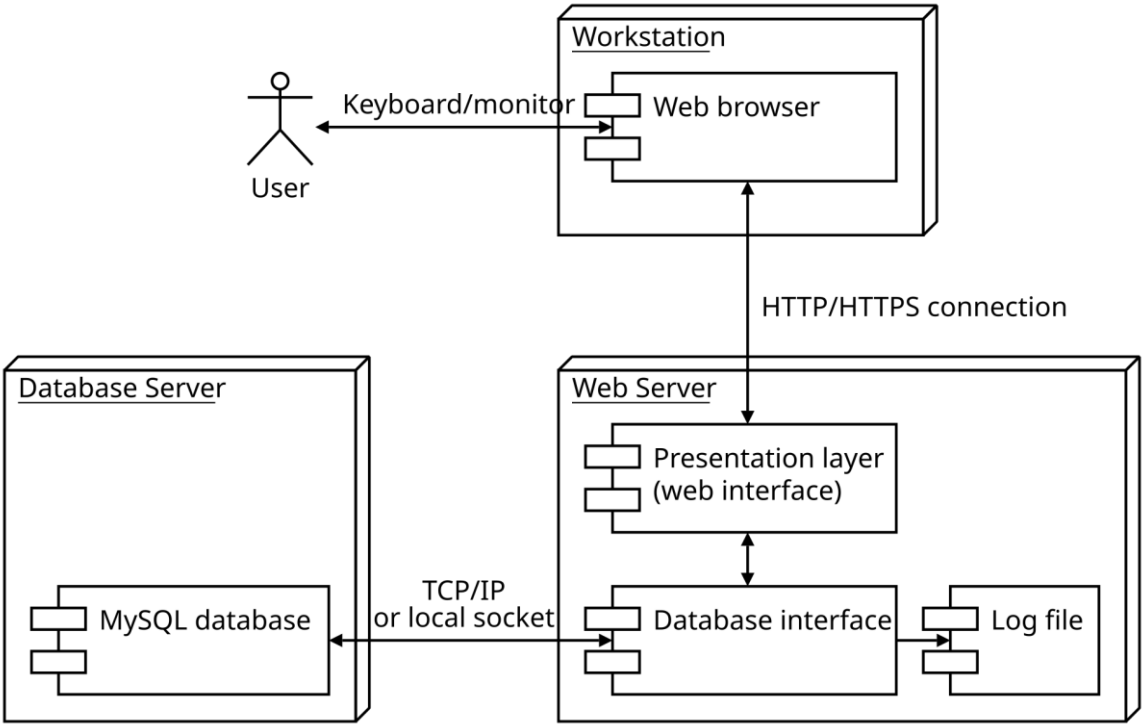


Рисунок Б.1 – UML діаграма компонентів

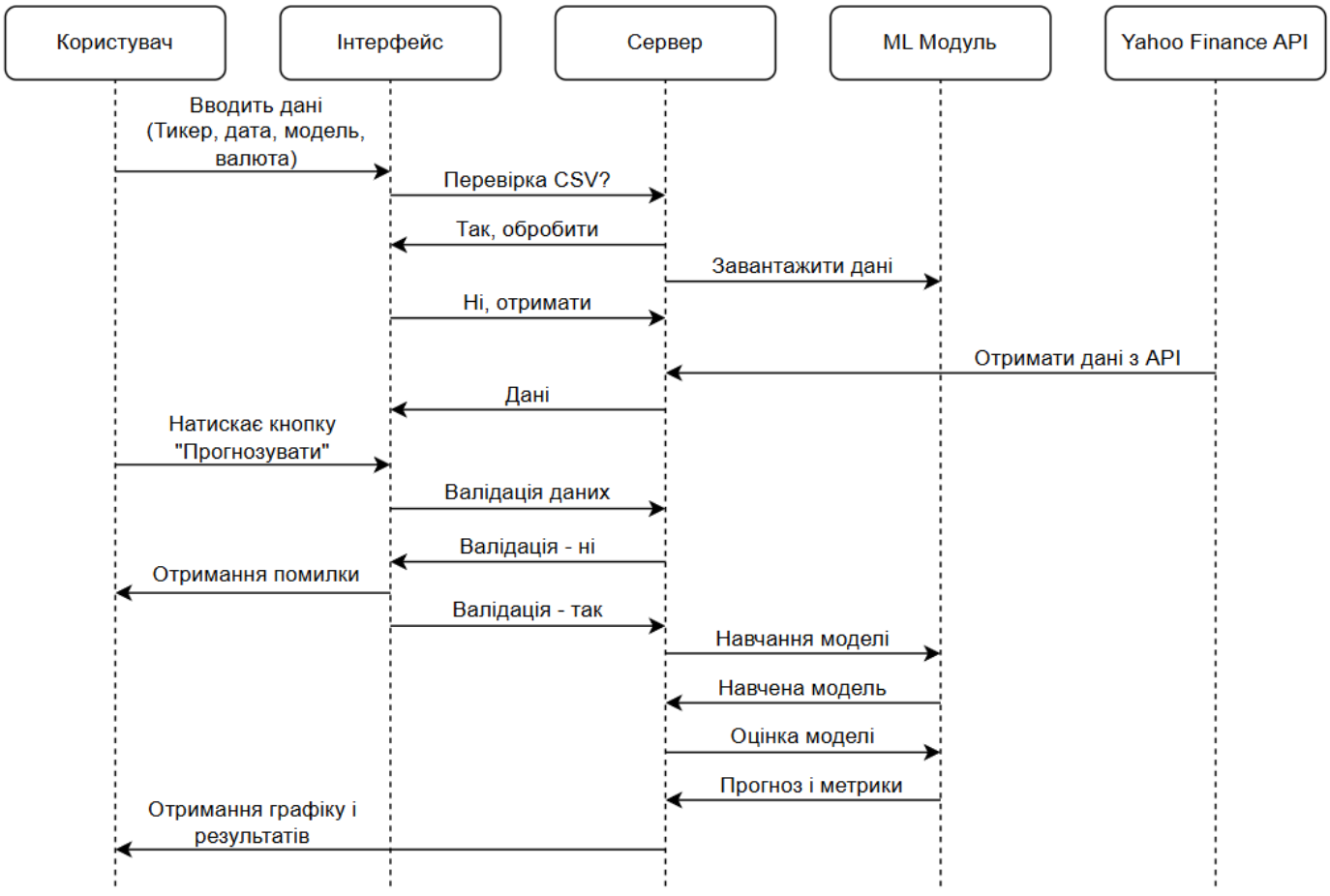


Рисунок Б.2 - UML діаграма послідовності

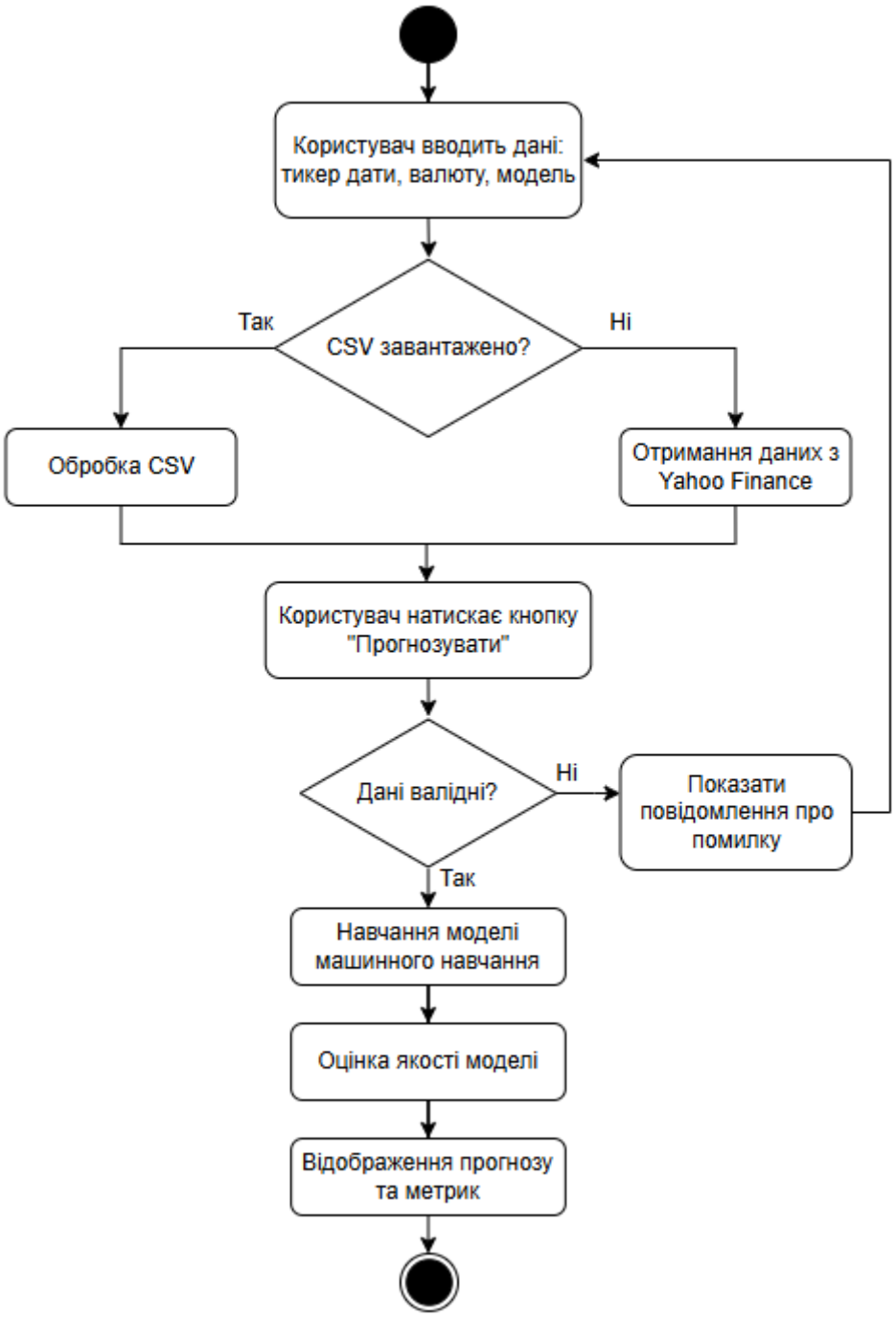


Рисунок Б.3 – UML діаграма діяльності

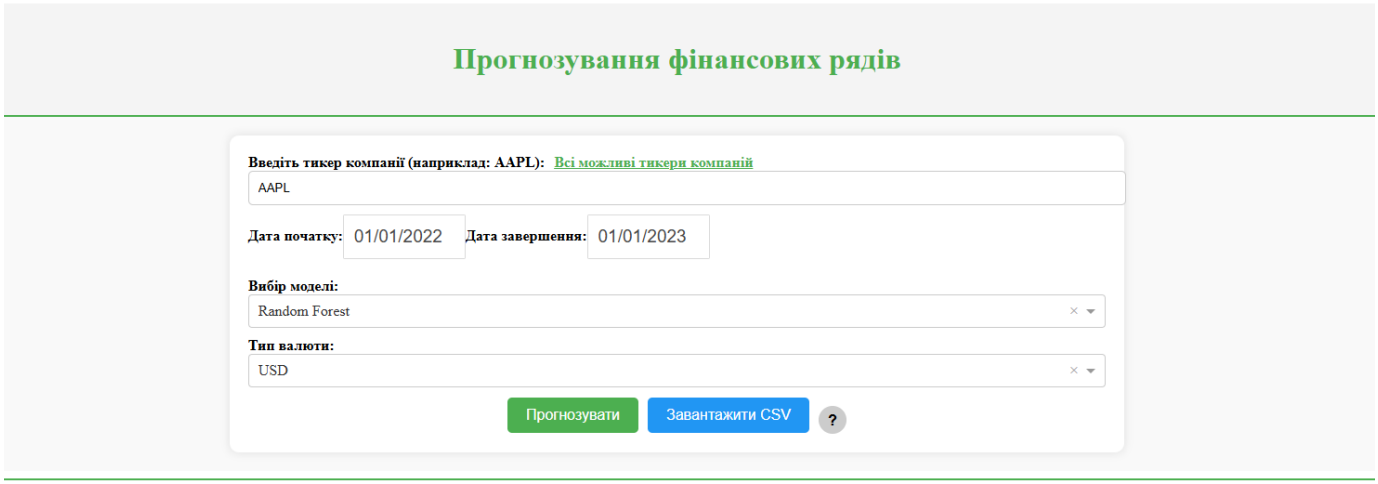


Рисунок Б.4 – Головна сторінка

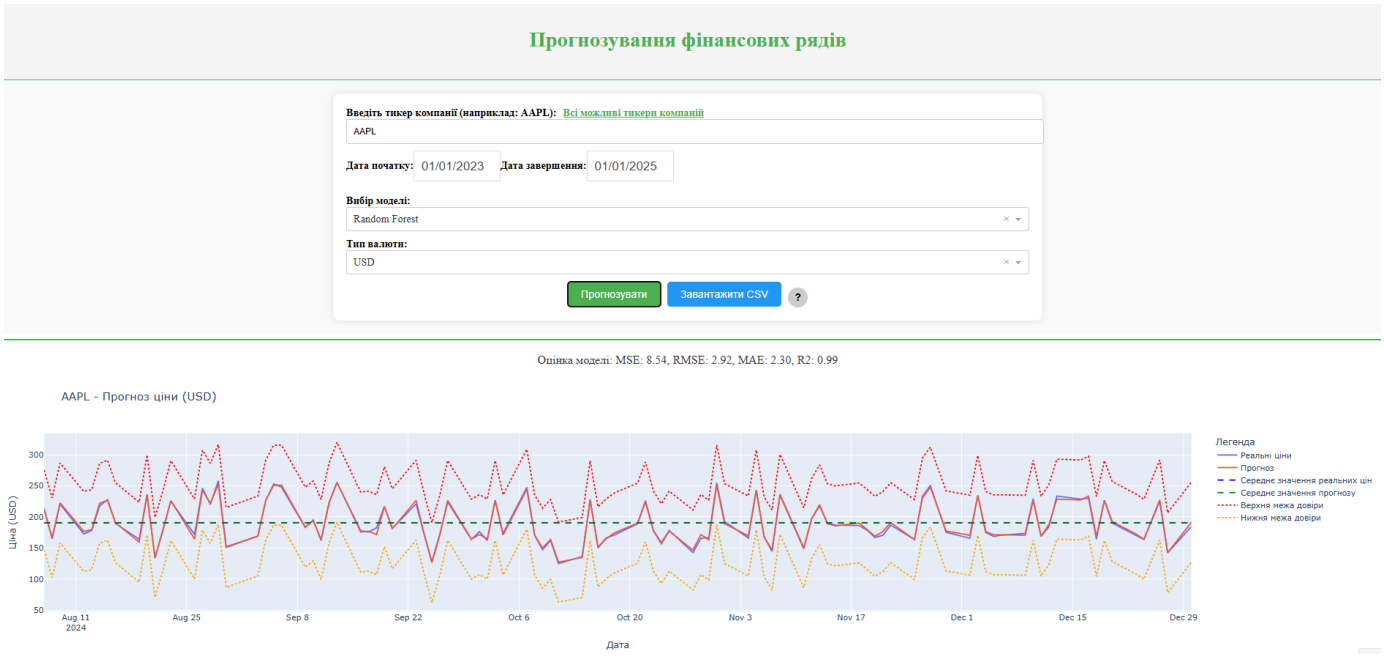


Рисунок Б.5 – Сторінка з активним прогнозуванням

ДОДАТОК В
(не обов'язковий)
Лістинг програми

```
import dash
from dash import dcc, html, Input, Output, State
import yfinance as yf
import pandas as pd
import plotly.graph_objs as go
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import io
import base64

# Ініціалізація додатку Dash
app = dash.Dash(__name__, suppress_callback_exceptions=True)
app.title = "Прогнозування фінансових рядів"

# Фіксовані курси валют
CURRENCY_RATES = {
    "USD": 1.0,
    "EURO": 0.91,
    "UAH": 36.8
}
```

```
# Додамо стилі через CSS
app.layout = html.Div([
    html.Div([
        html.H1("Прогнозування фінансових рядів", style={'textAlign': 'center', 'color':
         '#4CAF50'}),
    ], style={'backgroundColor': '#f4f4f4', 'padding': '20px', 'borderBottom': '2px solid #4CAF50'}),

    # Введення даних
    html.Div([
        html.Div([
            html.Label([
                "Введіть тикер компанії (наприклад: AAPL):",
                html.A("Всі можливі тикери компаній", href="https://finance.yahoo.com/lookup",
          target="_blank", style={'marginLeft': '10px', 'color': '#4CAF50'})
            ], style={'fontWeight': 'bold'}),
            dcc.Input(id='ticker-input', type='text', value='AAPL', style={'width': '100%', 'padding':
          '10px', 'marginBottom': '10px', 'border': '1px solid #ccc', 'borderRadius': '5px'}),

            html.Label("Дата початку:", style={'fontWeight': 'bold'}),
            dcc.DatePickerSingle(id='start-date', date='2022-01-01', style={'marginBottom': '10px'}),

            html.Label("Дата завершення:", style={'fontWeight': 'bold'}),
            dcc.DatePickerSingle(id='end-date', date='2023-01-01', style={'marginBottom': '10px'}),

            html.Label("Вибір моделі:", style={'fontWeight': 'bold', 'display': 'block', 'marginTop':
          '10px'}),
            dcc.Dropdown(
```

```
    id='model-dropdown',
    options=[
        {'label': 'Random Forest', 'value': 'Random Forest'},
        {'label': 'Linear Regression', 'value': 'Linear Regression'},
        {'label': 'SVR', 'value': 'SVR'},
        {'label': 'KNN', 'value': 'KNN'},
    ],
    value='Random Forest',
    style={'marginBottom': '10px'}
),

# Параметри для моделей
html.Div(id='model-parameters', style={'marginBottom': '10px'}),

html.Label("Тип валюти:", style={'fontWeight': 'bold'}),
dcc.Dropdown(
    id='currency-dropdown',
    options=[
        {'label': 'USD', 'value': 'USD'},
        {'label': 'EURO', 'value': 'EURO'},
        {'label': 'UAH', 'value': 'UAH'}
    ],
    value='USD',
    style={'marginBottom': '10px'}
),

# Кнопки дій
```

```
html.Div([
    html.Button('Прогнозувати', id='forecast-button', n_clicks=0, style={
        'backgroundColor': '#4CAF50', 'color': 'white', 'border': 'none', 'padding': '10px 20px',
        'textAlign': 'center', 'textDecoration': 'none', 'display': 'inline-block', 'fontSize': '16px',
        'marginRight': '10px', 'cursor': 'pointer', 'borderRadius': '5px'
    }),
    dcc.Upload(
        id='upload-data',
        children=html.Button('Завантажити CSV', style={
            'backgroundColor': '#2196F3', 'color': 'white', 'border': 'none', 'padding': '10px 20px',
            'textAlign': 'center', 'textDecoration': 'none', 'display': 'inline-block', 'fontSize': '16px',
            'cursor': 'pointer', 'borderRadius': '5px'
        }),
        multiple=False
    ),
    # Кнопка підказки та модальне вікно
    html.Div([
        html.Button(
            '?',  # Текст кнопки
            id='help-button',
            style={
                'backgroundColor': '#ccc',
                'color': 'black',
                'border': 'none',
                'borderRadius': '50%',
                'width': '30px',
                'height': '30px',
```

```
                'cursor': 'pointer',
                'textAlign': 'center',
                'fontWeight': 'bold',
                'fontSize': '16px',
                'lineHeight': '30px',
                'marginLeft': '10px'
            }
        ),
        html.Div(
            id='help-tooltip',
            children=[
                html.P("Оцінка моделі виконується за наступними параметрами:"),
                html.P("MSE (Mean Squared Error): Середньоквадратична похибка."),
                html.P("RMSE (Root Mean Squared Error): Корінь середньоквадратичної
похибки."),
                html.P("MAE (Mean Absolute Error): Середня абсолютна похибка."),
                html.P("R² (R-squared): Коефіцієнт детермінації, що показує якість моделі.")
            ],
            style={
                'display': 'none',  # Приховано за замовчуванням
                'backgroundColor': '#f9f9f9',
                'padding': '10px',
                'borderRadius': '5px',
                'boxShadow': '0px 0px 10px rgba(0, 0, 0, 0.1)',
                'position': 'absolute',
                'zIndex': 1000,
                'width': '300px',
```

```
                    'top': '0px',  # Вирівнювання по вертикалі
                    'left': '40px'  # Відступ праворуч від кнопки
                }
            )
        ], style={'position': 'relative', 'display': 'inline-block', 'marginTop': '10px'})
    ], style={'display': 'flex', 'justifyContent': 'center', 'alignItems': 'center', 'marginTop': '10px'})
], style={'width': '50%', 'margin': 'auto', 'padding': '20px', 'backgroundColor': '#fff',
 'boxShadow': '0px 0px 10px rgba(0, 0, 0, 0.1)', 'borderRadius': '10px'})
], style={'backgroundColor': '#f9f9f9', 'padding': '20px'}),

html.Hr(style={'border': '1px solid #4CAF50'}),

# Виведення результатів
html.Div(id='result-output', style={'textAlign': 'center', 'marginTop': '20px', 'fontSize': '18px',
    'color': '#333'}),

# Графік
html.Div([
    html.Div(
        dcc.Graph(id='forecast-graph', style={'marginTop': '20px', 'display': 'none'}),
        id='graph-container',
        style={'width': '100%'}
    )
], style={'position': 'relative', 'height': '400px'})  # Контейнер для графіка
])

# Обробка завантаження CSV
@app.callback(
```

```
    [Output('result-output', 'children'),
     Output('forecast-graph', 'figure'),
     Output('forecast-graph', 'style')],
    [Input('upload-data', 'contents'),
     Input('forecast-button', 'n_clicks')],
    [State('upload-data', 'filename'),
     State('ticker-input', 'value'),
     State('start-date', 'date'),
     State('end-date', 'date'),
     State('model-dropdown', 'value'),
     State('currency-dropdown', 'value')]
)
def handle_callbacks(contents, n_clicks, filename, ticker, start_date, end_date, model_type,
          currency):
    ctx = dash.callback_context

    if not ctx.triggered:
        return "", go.Figure(), {'display': 'none'}

    # Визначення, який елемент викликав callback
    triggered_id = ctx.triggered[0]['prop_id'].split('.')[0]

    if triggered_id == 'upload-data':
        # Обробка завантаження CSV
        if contents is None:
            return "Помилка: Файл не завантажено.", go.Figure(), {'display': 'none'}
```

```
    try:
        content_type, content_string = contents.split(',')
        decoded = base64.b64decode(content_string)
        df = pd.read_csv(io.StringIO(decoded.decode('utf-8')))

        if 'Date' not in df.columns or 'Close' not in df.columns:
            return "Помилка: Файл повинен містити стовпці 'Date' і 'Close'.", go.Figure(),
    {'display': 'none'}

        df['Date'] = pd.to_datetime(df['Date'])
        df.set_index('Date', inplace=True)

        fig = go.Figure()
        fig.add_trace(go.Scatter(x=df.index, y=df['Close'], mode='lines', name='Ціни'))
        fig.update_layout(title="Графік із завантаженого CSV",
                          xaxis_title="Дата",
                          yaxis_title="Ціна")

        return f"Файл {filename} успішно завантажено.", fig, {'display': 'block'}
    except Exception as e:
        return f"Помилка: {str(e)}", go.Figure(), {'display': 'none'}

elif triggered_id == 'forecast-button':
    # Обробка прогнозу
    if n_clicks == 0:
        return "", go.Figure(), {'display': 'none'}
```

```
errors = []
if not ticker:
    errors.append("Поле 'Тикер' не може бути порожнім.")
if not start_date:
    errors.append("Поле 'Дата початку' не може бути порожнім.")
if not end_date:
    errors.append("Поле 'Дата завершення' не може бути порожнім.")

if errors:
    return "Помилки:\n" + "\n".join(errors), go.Figure(), {'display': 'none'}

try:
    # Завантаження даних через yfinance
    df = yf.download(ticker, start=start_date, end=end_date)
    if df.empty:
        return "Помилка: Дані не знайдено. Перевірте тикер.", go.Figure(), {'display': 'none'}

    # Підготовка даних
    df["Tomorrow"] = df["Close"].shift(-1)
    df.dropna(inplace=True)

    X = df[["Open", "High", "Low", "Close", "Volume"]]
    y = df["Tomorrow"]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Вибір моделі
    if model_type == "Random Forest":
```

```
        model = RandomForestRegressor(n_estimators=100)
    elif model_type == "KNN":
        model = KNeighborsRegressor(n_neighbors=5)
    elif model_type == "SVR":
        model = SVR(kernel='rbf')
    elif model_type == "Linear Regression":
        model = LinearRegression()
    else:
        return "Помилка: Модель не знайдена.", go.Figure(), {'display': 'none'}

    # Навчання моделі
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)

    # Оцінка моделі
    mse = mean_squared_error(y_test, predictions)
    rmse = mse ** 0.5
    mae = mean_absolute_error(y_test, predictions)
    r2 = r2_score(y_test, predictions)

    result_text = f"Оцінка моделі:\nMSE: {mse:.2f}, RMSE: {rmse:.2f}, MAE: {mae:.2f}, R2:
{r2:.2f}"

    # Побудова графіка
    fig = go.Figure()

    # Лінія реальних цін
```

```
fig.add_trace(go.Scatter(
    x=df.index[-len(y_test):],
    y=y_test,
    mode='lines',
    name='Реальні ціни',
    hovertemplate='Дата: %{x}<br>Ціна: %{y:.2f} ' + currency + '<extra></extra>'
))

# Лінія прогнозу
fig.add_trace(go.Scatter(
    x=df.index[-len(y_test):],
    y=predictions,
    mode='lines',
    name='Прогноз',
    hovertemplate='Дата: %{x}<br>Прогноз: %{y:.2f} ' + currency + '<extra></extra>'
))

# Додаткові лінії: середнє значення реальних цін
mean_real_price = y_test.mean()
fig.add_trace(go.Scatter(
    x=df.index[-len(y_test):],
    y=[mean_real_price] * len(y_test),
    mode='lines',
    name='Середнє значення реальних цін',
    line=dict(dash='dash', color='blue'),
    hovertemplate=f'Середнє значення: {mean_real_price:.2f} {currency}<extra></extra>'
))
```

```
# Додаткові лінії: середнє значення прогнозу
mean_prediction = predictions.mean()
fig.add_trace(go.Scatter(
    x=df.index[-len(y_test):],
    y=[mean_prediction] * len(predictions),
    mode='lines',
    name='Середнє значення прогнозу',
    line=dict(dash='dash', color='green'),
    hovertemplate=f'Середнє значення: {mean_prediction:.2f} {currency}<extra></extra>'
))

# Розрахунок стандартного відхилення для прогнозів
std_dev = predictions.std()
upper_bound = predictions + (1.96 * std_dev)  # Верхня межа (95% довіра)
lower_bound = predictions - (1.96 * std_dev)  # Нижня межа (95% довіра)

# Додати верхню межу довіри
fig.add_trace(go.Scatter(
    x=df.index[-len(y_test):],
    y=upper_bound,
    mode='lines',
    name='Верхня межа довіри',
    line=dict(dash='dot', color='red'),
    hovertemplate='Дата: %{x}<br>Верхня межа: %{y:.2f} ' + currency + '<extra></extra>'
))
```

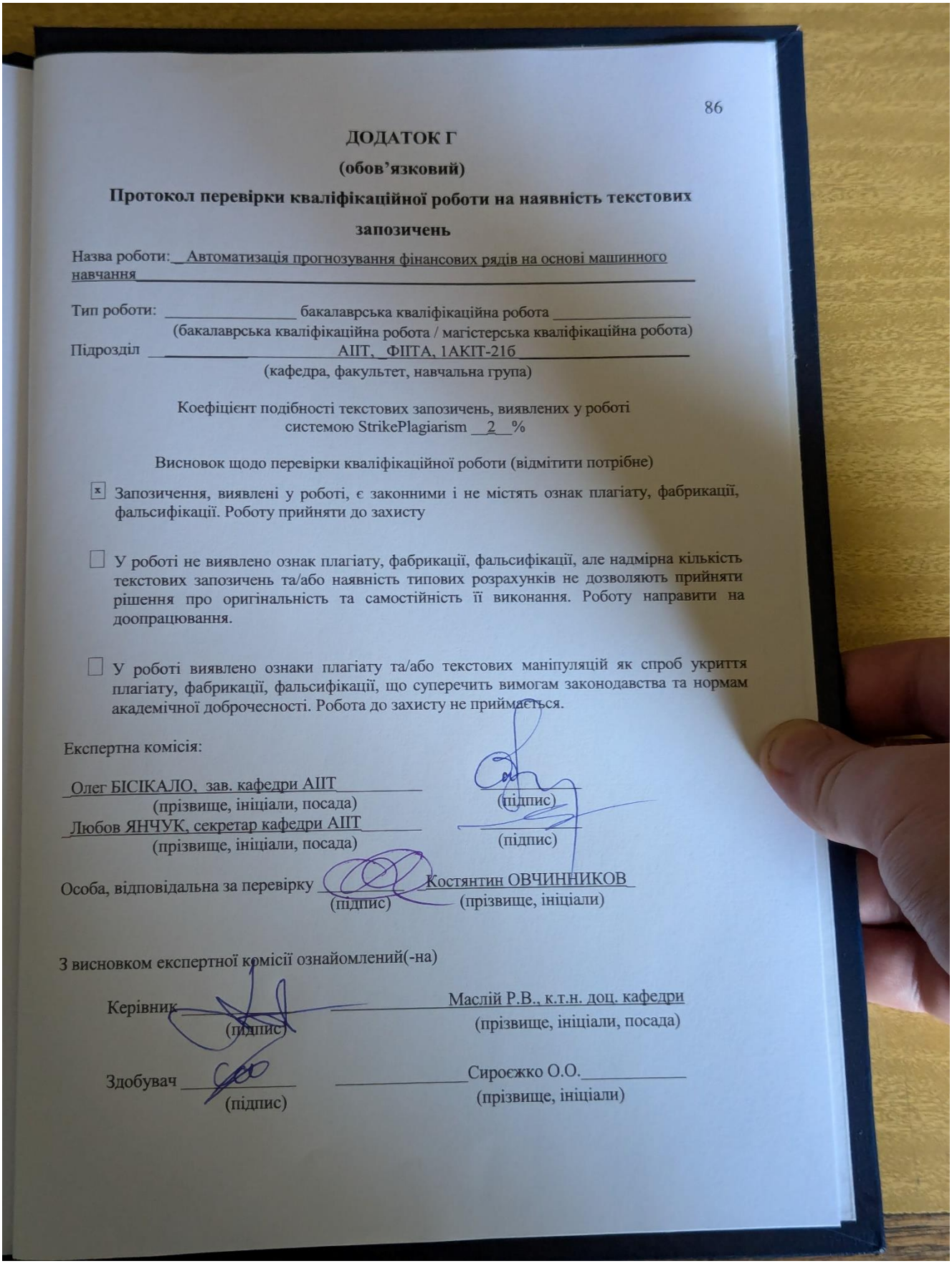
```
            # Додати нижню межу довіри
            fig.add_trace(go.Scatter(
                x=df.index[-len(y_test):],
                y=lower_bound,
                mode='lines',
                name='Нижня межа довіри',
                line=dict(dash='dot', color='orange'),
                hovertemplate='Дата: %{x}<br>Нижня межа: %{y:.2f} ' + currency + '<extra></extra>'
            ))

            # Оновлення заголовка графіка з додаванням валюти
            fig.update_layout(
                title=f"{ticker} - Прогноз ціни ({currency})",
                xaxis_title="Дата",
                yaxis_title=f"Ціна ({currency})",
                legend_title="Легенда"
            )

            return result_text, fig, {'display': 'block'}
        except Exception as e:
            return f"Помилка: {str(e)}", go.Figure(), {'display': 'none'}

@app.callback(
    Output('help-tooltip', 'style'),
    Input('help-button', 'n_clicks'),
    prevent_initial_call=True
)
```

```
def toggle_help_tooltip(n_clicks):
    if n_clicks % 2 == 1:  # Показати підказку
        return {
            'display': 'block',
            'backgroundColor': '#f9f9f9',
            'padding': '10px',
            'borderRadius': '5px',
            'boxShadow': '0px 0px 10px rgba(0, 0, 0, 0.1)',
            'position': 'absolute',
            'zIndex': 1000,
            'width': '300px',
            'top': '0px',  # Вирівнювання по вертикалі
            'left': '40px'  # Відступ праворуч від кнопки
        }
    else:  # Приховати підказку
        return {'display': 'none'}
if __name__ == '__main__':
    app.run(debug=True)
```



ДОДАТОК Г

(обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Автоматизація прогнозування фінансових рядів на основі машинного навчання

Тип роботи: _____ бакалаврська кваліфікаційна робота _____
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ _____ АІІТ, ФІІТА, 1АКІТ-21б _____
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 2 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☒ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Олег БІСІКАЛО, зав. кафедри АІІТ _____
(прізвище, ініціали, посада) (підпис)

Любов ЯНЧУК, секретар кафедри АІІТ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Костянтин ОВЧИННИКОВ
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Маслій Р.В., к.т.н. доц. кафедри
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Сироєжко О.О.
(підпис) (прізвище, ініціали)