МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
НН ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ

Кафедра математичних методів системного аналізу

«До захисту допущено»
Завідувач кафедри
Оксана ТИМОЩУК
«___» »_____2023 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»

на тему «Системне моделювання фондового ринку генеративно-змагальними нейронними мережами»

Виконав:
студент IV курсу, групи КА-91
Бєздєтний Данііл Дмитрович _____
Керівник:
професор, д.т.н. Данилов В.Я. _____
Консультант з економічного розділу:
доцент, к.е.н. Рощина Н. В. _____
Консультант з нормоконтролю:
к.ф.-м.н. Статкевич В.М. _____
Рецензент:
професор кафедри ІБ, д.т.н. Новіков О.М. _____

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

Київ 2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
НН Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 124 «Системний аналіз»
Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Оксана ТИМОЩУК
«__» _____ 2023 р.

ЗАВДАННЯ
на дипломну роботу студенту
Бєздєтному Даніїлу Дмитровичу

1. Тема роботи «Системне моделювання фондового ринку генеративно-змагальними нейронними мережами», керівник роботи Данилов Валерій Якович, професор кафедри ММСА, затверджені наказом по університету від «30» травня 2023 р. № 2065-с.

2. Термін подання студентом роботи 12.06.2023.

3. Вихідні дані до роботи: база даних про показники акцій компанії Shell.

4. Зміст роботи: вступ, визначення та принципи функціонування фондового ринку, генеративно-змагальні нейронні мережі, моделювання і прогнозування ціни акцій, функціонально вартісний аналіз програмного продукту, висновки.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): структура генеративно-змагальної мережі, вибір бази даних, отримані результати, значення функцій-втрати, аналіз отриманих результатів.

6. Консультанти розділів роботи

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

7. Дата видачі завдання: 1 лютого 2023

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Вивчення літератури за темою роботи	01.02.2023-28.02.2023	Виконано
2	Аналіз актуальності задач стосовно тематики дослідження	20.02.2023-28.02.2023	Виконано
3	Формулювання задачі дипломної роботи після теоретичного ознайомлення	01.03.2023-09.03.2023	Виконано
4	Математичне моделювання задачі	10.03.2023-24.04.2023	Виконано
5	Розробка програмного продукту	27.03.2023-21.04.2023	Виконано
6	Підготовка розділів під час переддипломної практики	24.04.2023-12.05.2023	Виконано
7	Оформлення записок та розділів в дипломній роботі відповідно до нормоконтролю	15.05.2023-29.05.2023	Виконано
8	Попередній захист дипломної роботи	30.05.2023-09.06.2023	Виконано
9	Захист дипломної роботи	19.06.2023-23.06.2023	

Студент _____ Даніїл БЄЗДЄТНИЙ

Керівник роботи _____ Валерій ДАНИЛОВ

РЕФЕРАТ

Дипломна робота: 75 с., 22 рис., 6 табл., 2 додатка, 18 джерел.

СИСТЕМНЕ МОДЕЛЮВАННЯ, ФОНДОВИЙ РИНОК, НЕЙРОННІ МЕРЕЖІ, ГЕНЕРАТИВНО-ЗМАГАЛЬНІ МЕРЕЖІ, ПРОГНОЗУВАННЯ

Об'єкт дослідження: поведінка цін акцій на фондовому ринку.

Предмет дослідження: використання генеративно-змагальних нейронних мереж для дослідження поведінки цін акцій на фоновому ринку.

Мета роботи: підвищення якості розуміння та прогнозування поведінки фондових ринків за допомогою генеративно-змагальних мереж для допомоги при розробці торгових стратегій.

В роботі розглянуто та реалізовано модель нейронної мережі, проаналізовано вибір параметрів використаної моделі, спосіб прогнозування нестаціонарних процесів.

Програмний продукт реалізовано мовою програмування Python.

Результат даної роботи доцільний для аналізу поведінки цін акцій на фондовому ринку та фінансових ринків загалом. Для проведення аналізу було використано дані, що надаються сервісом Yahoo Finance, який знаходиться у вільному доступі.

ABSTRACT

Thesis: 75 pages, 22 figures, 6 tables, 2 appendices, 18 references.

SYSTEM MODELING, STOCK MARKET, NEURAL NETWORKS, GENERATIVE ADVERSARIAL NETWORKS, FORECASTING

Research Object: Behavior of stock prices in the stock market.

Objective: To improve the quality of understanding and forecasting the behavior of stock markets using generative adversarial networks to assist in the development of trading strategies.

Subject of research: Using generative adversarial neural networks to study the behavior of stock prices in the background market.

The thesis examines and implements a neural network model, analyzes the selection of parameters used in the model, and the method of forecasting non-stationary processes.

The software product is implemented in the Python programming language.

The result of this work is relevant for analyzing the behavior of stock prices in the stock market and financial markets in general. The analysis was conducted using data provided by the Yahoo Finance service, which is freely accessible.

ЗМІСТ

ВСТУП

Фондовий ринок є складною динамічною системою, яка є значною складовою фінансового ринку та економіки в цілому. Через залученість багатьох факторів до його формування, починаючи з інформації про стан компанії та завершуючи політичними настроями суспільства, та великі обсяги даних, моделювання поведінки фондового ринку є нелінійною та досить складною задачею. З розвитком машинного навчання почалися численні спроби використати штучний інтелект заради спрощення прогнозування поведінки фінансових ринків. Однак традиційні статистичні підходи не мали змоги досить точно помічати закономірності, тому на допомогу до них з’явилися генеративно-змагальні нейронні мережі.

Поява генеративно-змагальних нейронних мереж стала свіжим подихом для задачі моделювання фінансових систем завдяки їх можливості генерувати реалістичні якісні імітовані дані. Завдяки використанню цих мереж у фінансовому моделюванні ми отримали змогу виділяти складні закономірності у даних фондового ринку, що дозволяє нам отримати цінну інформацію з приводу поведінки ринку та потенційно розширити можливості прогнозування.

Використовуючи потужність генеративно-змагальних нейронних мереж, ми прагнемо розробити системну модель для фондового ринку, яка може відобразити його складну динаміку та створити реалістичні сценарії. Згенеровані синтетичні дані можуть надати цінну інформацію для аналізу ризиків, оптимізації портфеля та процесів прийняття рішень на фінансових ринках. Крім того, модель системи може служити випробувальним майданчиком для оцінки різних інвестиційних стратегій і оцінки їх ефективності в різних ринкових умовах.

1 ВИЗНАЧЕННЯ ТА ПРИНЦИПИ ФУНКЦІОНУВАННЯ ФОНДОВОГО РИНКУ

1.1 Історія та розвиток фондового ринку

Історія фондової біржі налічує кілька століть, зародження біржової торгівлі відноситься до 16 століття. Далі розглянемо основні події у формуванні сучасного фондового ринку

Зародження концепції біржової торгівлі простежується з середньовічної Європи. Тоді були влаштовані перші торгові центри, на яких з метою купівлі та продажі акцій компаній збиралися купці та трейдери. Найяскравішим прикладом такого центру стала Амстердамська фондова біржа, яка була заснована в 1602 році та вважається однією з найперших офіційних бірж у світі.

Починаючи з 17-18 століття починається стрімкий розвиток фондових бірж у багатьох країнах світу. Фондові біржі з’явилися у Лондоні, Парижі та Нью-Йорку. Вже тоді біржі почали слугувати централізованим ринком для інвесторів, які приходили торгувати акціями та іншими цінними паперами. Задля забезпечення прозорості, справедливості та регулювання торгової діяльності були запроваджені стандартизовані правила торгівлі.

Промислова революція також далася взнаки. З початку 19 століття відбувся значний технологічний прогрес, стрімко зросли темпи індустріалізації. Це надало поштовх для розквіту великих корпорацій та розширення фондових ринків. Для отримання своєї частки у прибутках підприємств, які почали швидко розвиватися, та фінансування промислових проектів інвестори все частіше зверталися до фондового ринку.

З плином часу економіки країн ставали все більш взаємопов’язаними, через що фондові ринки почали виходити за рамки національних кордонів. Це

послугувало причиною того, що наприкінці 19 — на початку 20 століття почали з’являтися міжнародні фондові біржі. Такими стали Лондонська та Нью-Йоркська біржі, які завдяки цьому стали головними центрами світової торгівлі.

З технологічним прогресом у другій половині 20-го століття відбулися значні технологічні досягнення, пройшла революція в біржовій торгівлі, фондова біржа набула такого вигляду, як ми уявляємо її зараз. На заміну системі відкритих викликів прийшли електронні торгові платформи, які зробили торгівлю значно швидшою та ефективнішою. В свою чергу поява Інтернету та мереж комунікації зробило ще більший внесок у формування середовища фондового ринку.

Поява торгівельних онлайн платформ наприкінці минулого та початку теперішнього століття зробила доступ до фондового ринку значно більш простим та вільним. Зменшення залежності від традиційних брокерських фірм Індивідуальні інвестори отримали можливість торгувати акціями безпосередньо зі своїх комп’ютерів та мобільних пристроїв.

З розвитком фінансового ринку почали з’являтися і нові фінансові інструменти. Окрім звичних усім акцій на ринку почалися торги облігацій, опціонів, ф’ючерсів та паїв біржових фондів. Це урізноманітнило інвестиційні можливості та зробило більш гнучкими інструменти для управління ризиками.

Також з подальший розвиток технологічного прогресу в останні десятиліття спричинив появу кардинально нових підходів. Зародилися поняття високочастотної та алгоритмічної торгівлі. Ці автоматичні торгові стратегії покладаються на складні алгоритми та надшвидкісні комп’ютери для здійснення угод за частки секунди, що значно впливає на динаміку ринку.

Однак, слід зазначити, що окрім злетів також були і падіння. За свою історію фондовий ринок зазнав декількох крахів протягом свого існування. Одними з прикладів є велика депресія, яку спричинив крах Уолл-стріт 1929 року, та глобальна фінансова криза 2008 року. Впровадження регуляторних

заходів стали наслідком таких падінь. Вони були спрямовані на стабілізацію ринків та захист коштів інвесторів.

Загалом фондовий ринок є системою, що постійно розвивається та адаптується до нових технологій, економічних розробок і нормативних змін. Він відіграє вирішальну роль у світовій економіці, сприяючи накопиченню капіталу, відкриваючи можливості для інвестицій і слугуючи барометром економічного здоров'я. Історія фондової біржі відображає динамічний характер фінансових ринків і постійне прагнення до ефективних і прозорих механізмів торгівлі.

1.2 Визначення фондового ринку

Фондовий ринок є платформою, на якій покупці та продавці збираються разом для торгівлі акціями, які являють собою частку власності у публічно торгуючих компаніях. Це централізований ринок, на якому інвестори можуть здійснювати купівлю-продаж та обмін цінними паперами, а саме акціями, облігаціями, фондами взаємних інвестицій та похідними від них.

Фондовий ринок відіграє вирішальну роль в економіці, оскільки він дозволяє компаніям залучати капітал та надавати приватним особам можливість інвестувати та приймати участь у розвитку бізнесу.

Основні принципи фондового ринку:

- Фондовий ринок працює за принципом попиту та пропозиції, тобто ціна акції визначається взаємодією між покупцями, що створюють попит, та продавцями, які створюють пропозицію. Отже, коли певна акція має більшим попит, її ціна зазвичай зростає, і навпаки, падає, коли попит знижується.

- Ліквідність фондового ринку свідчить, наскільки легко можна виконати дії з цінним папером без значного впливу на його ціну. Ліквідний фондовий ринок сприяє ефективній торгівлі та справедливим цінам на ринку.
- Фондовий ринок має бути регульованим та знаходитися під наглядом керуючого органа для забезпечення чесної та прозорої торгівлі й для захисту інвесторів. Прикладом такого органу є Securities and Exchange у Сполучених Штатах Америки, він захищає цілісність ринку та запобігає шахрайству та маніпуляціям.
- На фондовому ринці мають брати участь різні гравці, трейдери, брокери, індвідуальні та інвестиційні інвестори.
- Ефективність фондового ринку означає, наскільки ціни на акції відображають усю доступну інформацію і свідчить що неможливо постійно випереджати ринок, спираючись виключно на загальнодоступну інформацію.
- Фондовому ринку також властива волатильність цін, тобто він за своєю природою є нестабільним, що спричиняє швидкі коливання цін на акції у відповідь на такі чинники, як результати діяльності компанії, події у світі та настрої інвесторів.

1.3 Ключові параметри цін акцій

Аналізуючи ціни на акціїї кілька важливих параметрів відіграють значну роль в оцінці ефективності та вартості акцій. Ці фактори дають змогу зрозуміти фінансовий стан компанії, прибутковість та настрої ринку. Далі розглянемо найважливіші параметри, які слід враховувати.

- Коефіцієнт ціни/прибутку: даний коефіцієнт порівнює ціну акцій компанії з прибутком на акцію. Це допомагає зрозуміти, скільки інвестори готові платити за кожен долар прибутку компанії. Вищі значення коефіцієнту вказують на вищі очікування зростання або переоцінку акцій, тоді як нижчий коефіцієнт може вказувати на то, що акції недооцінені або мають повільніші перспективи зростання.

- Дивідендна прибутковість являє собою відношення річного дивіденда на акцію до її поточної ціни. Він показує прибуток від інвестицій через дивіденди. Вища дивідендна прибутковість є привабливою для інвесторів, що прагнуть отримати прибуток, в той час, коли нижча прибутковість може характеризувати те, що компанія реінвестує свій прибуток у розширення та зростання.

- Ринкова капіталізація стосується загальної вартості акцій компанії в обігу та може бути розрахованою шляхом множення ціни акції на кількість акцій, що знаходяться в обігу. Вона допомагає класифікувати компанії за різними розмірами, наприклад, компанії з великою, середньою та малою капіталізацією. Інвестори часто розглядають ринкову капіталізацію, щоб оцінити розмір компанії та профіль ризики.

- Прибуток на акцію (також відомий як EPS) вимірює прибутковість компанії шляхом ділення її прибутку на кількість акцій в обігу. Вищий прибуток на акцію свідчить про більшу прибутковість. Щоб оцініти потенціал зростання, інвестори часто порівнюютьяс прибутки на акцію компанії з її минулими показниками, аналогами в галузі та очікуваннями ринку.

- Співвідношення заборгованості та власного капіталу порівнює власний капітал компанії та її заборгованість. Високий показник може означати вищий фінансовий ризик, у той час, як нижчий показник вказуватиме на більш консервативну структуру капіталу.

- Рентабельність власного капіталу вимірює прибутковість компанії, виражаючи ориманий чистий прибуток у відсотках від власного капіталу. Вона відображає наскількі ефективно компанія використовує інвестиції акціонерів для отримання прибутку. Очевидно, вища рентабельність капіталу свідчить про ефективніше його використання та більший потенціал прибутку.
- Грошовий потік також є важливим параметром, що характеризує певну компанію на ринку. Оцінка грошового потоку компанії, який скаладається з вільного та операційного грошових потоків, дає уявлення про здатність компанії генерувати готівку та капітал від операцій, інвестицій і дивідендів. Позитивний грошовий потік свідчить про здатність компанії підтримувати та розвивати свій бізнес.

1.4 Основні дані про акції фондового ринку

Для аналізу майбутньої поведінки ціни на акцію нам слід використовувати дані про нещодавню поведінку ціни у минулому. В цьому пункті наведемо приклад даних, які можуть допомогти у прогнозуванні ціни цінного паперу. Розглянемо ціну при відкритті, ціну при закритті, найвищу ціну, найнижчу ціну та обсяг.

Ціна при відкритті становить собою початкову ціну, за якою певна акція або фінансовий інструмент починає торгуватися на початку торгової сесії (або ж торгового дня). Це перша ціна, за якою покупці здійснюють операції після відкриття ринку. Ціна відкриття враховує у собі різні фактори, настрій ринку у минулу торгову сесію, ціну закриття попереднього дня та реакцію на події у світі.

Найвища ціна показує найвищу торгову ціну, за якою акція торгувалася протягом певної торгової сесії. Тобто, він показує пікове значення, досягнуте курсом акцій протягом заданого періоду часу. Найвища ціна цінного паперу є ключовим показником, на який звертають увагу інвестори та трейдери, задля оцінки їх продуктивності та визначення потенційниї тенденцій цін.

Найнижча ціна означає найнижчу торгову ціну, за якою акція або фінансовий інструмент торгувалися протягом певної торгової сесії. Він відображаю найнижче значення, досягнуте курсом акцій протягом заданого періоду часу. Як і найвища ціна, найнижча є важливим показником, який допомагає інвесторам і трейдерам визначити рівні підтримки та цінові тенденції.

Ціна при закритті означає кінцеву ціну торгів, за якою акція або фінансовий інструмент закриває торгівлю протягом певного періоду торгів. Вона являє собою останню ціну, за якою учасники ринку здійснюють операції перед закриттям ринку. Ціна при закритті має важливе значення, оскільки її зазвичай використовують для розрахунку різних показників технічного аналізу, таких як ковзні середні або співвідношення ціни та прибутку.

Обсяг надає інформацію про загальну кількість акцій або контрактів, які були випущені на певний цінний папер протягом певного періоду торгівлі. Це надає нам уявлення про рівень ринкової активності та ліквідності, а саме більший обсяг торгів як правило вказує на збільшення інтересу інвесторів до даного паперу, у той час, коли менший оборот вказує на зниження активності ринку. Цей показник є дуже важливим елементом технічного аналізу, оскільки він допомагає підтвердити рух цін і визначити потенційні тенденції ринку.

Використання цих даних разом дає учасникам ринку суттєве уявлення про динаміку цін і торгову активність цінних паперів на фондовому ринку. Ці характеристики зазнають широкого застосування серед інвесторів, аналітиків та трейдерів, які використовують їх для прийняття рішень та оцінки ефективності інвестицій.

1.5 Вплив різних факторів на аналіз фондового ринку

Аналіз фондового ринку передбачає оцінку різних факторів, які можуть вплинути на вартість і ефективність компанії та її акцій. Саме тому слід розглянути певні ключові фактори, які можуть внести свої зміни у поведінку фондового ринку з точки зору його аналізу.

При моделюванні фондового ринку варто зважати на економічні показники, наприклад зростання ВВП, рівень інфляції, відсоткові ставки та рівень безробіття. Позитивне значення економічних показників часто свідчить про сприятливе бізнес-середовище, що підвищує довіру інвесторів і ціни на акції.

В свою чергу гірші економічні показники створюють значну невизначеність, що може призвести до спаду ринку.

Також варто враховувати фінансові показники та дохід самої компанії, оскільки це відіграє значну роль у визначенні вартості її цінних паперів. Аналітики зважають на такі фактори, як норма прибутку, прибуток на акцію, грошовий потік та зростання доходу для оцінки фінансового стану компанії. Великі прибутки та стале зростання вважаються позитивним індикатором для інвесторів, що призведе до підвищення цін на акції. Окрім фінансових показників самої компанії слід враховувати тенденції її галузі, а саме ринковий попит, технологічний прогрес та галузеву конкуренцію, оскільки це впливає на перспективи компанії.

Окрім економічних факторів важливим є урахування макроекономічних подій та геополітичних факторів. Геополітична напруженість, стихійні лиха, економічні кризи та зміна торгової політики – усе це має свій вплив на фондовий ринок, а саме порушують стабільність та змінюють настрої інвесторів.

Іноді незалежно від фундаментальних факторів ціни акцій можуть змінюватися через настрої інвесторів та ринкову психологію. Наприклад,

позитивні настрої та поведінка можуть спровокувати купівельний ажіотаж, що зможе підвищити ціни, і навпаки.

Аналітики також не оминають новини про події у компанії. Запуск нових продуктів, злиття або поглинання іншою компанією та зміни в керівництві впливають на зміни цін на акції. Відповідно до подій учасникам торгівлі слід коригувати свої оцінки.

Висновки до розділу 1

У цьому розділі було наведено інформацію про виникнення та детально розглянуто історію розвитку фондового ринку, охарактеризовано ключові події та звороти у його формуванні. Ознайомившись з наданою інформацією, ми краще розуміємо особливості фондових ринків та природу показників їх аналізу, основні показники та фактори, які залучаються для моделювання його поведінки. Завдяки цим даним ми можемо детальніше зрозуміти природу певних змін у курсі цінних паперів певних компаній. Також було розглянуто вплив різноманітних факторів на показники діяльності окремих компаній, що надає нам інформацію про майбутні тенденції.

2 ГЕНЕРАТИВНО-ЗМАГАЛЬНІ НЕЙРОННІ МЕРЕЖІ

2.1 Основні дані про нейронні мережі та їх використання

Нейронні мережі, також відомі як штучні нейронні мережі, є основними компонентами систем машинного навчання та штучного інтелекту. Це обчислювальні моделі, натхненні структурою та функціями людського мозку. Нейронні мережі складаються з взаємопов'язаних вузлів, які називаються штучними нейронами або «вузлами», які організовані в шари.

Основним будівельним блоком нейронної мережі є штучний нейрон, який отримує вхідні сигнали, призначає їм ваги та обчислює вихідний сигнал за допомогою функції активації. Зв'язки між нейронами представлені вагами, які визначають силу та важливість вхідних сигналів.

Нейронні мережі навчаються за допомогою процесу, який називається «навчання». Під час навчання мережа регулює ваги своїх з'єднань на основі заданого набору вхідних даних і пов'язаних з ними бажаних виходів.

Мета полягає в тому, щоб мінімізувати різницю між прогнозованими і фактичними результатами, відому як «втрата» або «похибка». Цей процес зазвичай виконується за допомогою алгоритмів оптимізації, таких як градієнтний спуск.

Нейронні мережі можна використовувати для багатьох різнопланових задач, наприклад: задачі класифікації, розпізнавання шаблонів та образів, регресія, виявлення аномалій та системи рекомендацій. Далі розглянемо кожен із цих пунктів детальніше.

Доволі часто зустрічається використання нейронних мереж для задач класифікації. Результатом виконання задач такого типу є розподілення та

призначення вхідних даних до заздалегідь визначених класів або категорій. Такі завдання постають при розпізнаванні зображень, виявленні спаму, аналізу настрою або діагностики захворювань.

Варто зазначити, що нейронні мережі дуже добре виконують задачі ідентифікації складних шаблонів та пошуку закономірностей у даних. Їх здатність виявляти шаблони у зображенням, звуках і тексті дозволяє виконувати досить складні задачі розпізнавання предметів, розпізнавання мови та обробки нечітких вхідних сигналів.

Також нейронні мережі дуже добре показують себе у прогнозуванні безперервних значень та числових результатів. Такі задачі відносяться до задач регресії. Прикладом використання таких задач у реальному житті є оцінка стану певних показників у майбутньому (прогнозування цін на нерухомість, медична діагностика).

Виявлення аномалій є задачею, для вирішення якої нейронні мережі також стали у нагоді. Як було зазначено вище, вони якісно проявляють себе у завданнях розпізнавання шаблонів. Саме тому вони є дуже корисними в ідентифікації нехарактерних шаблонів у вхідних даних. Це корисно для розпізнавання шахрайства, виявлення мережевих вторгнень або виявлення виробничих дефектів.

Слід ще зазначити велику роль нейронних мереж для розв’язання проблеми складання системи рекомендацій. Аналізуючи поведінку та вподобання користувачів мережі надають механізми рекомендацій, щоб пропонувати відповідний продукт. Це широко використовується в електронній комерції, потокових платформах і персоналізованому маркетингу. Нині існує велика кількість сервісів, які здобули свою популярність в більшості за рахунок розробленої в цих застосунках якісної системи рекомендацій.

Нейронні мережі бувають різних форм, включаючи нейронні мережі прямого зв’язку, згорткові нейронні мережі (Convolutional neural network, CNN) для обробки зображень, рекурентні нейронні мережі (Recurrent Neural

Networks, RNN) для послідовних даних і більш просунуті архітектури, такі як глибокі нейронні мережі (Deep Neural Network, DNN) з багатьма прихованими шарами.

Розуміння нейронних мереж та їх застосування є ключовим для оцінки їх придатності для різних сценаріїв, визначення відповідної архітектури, оптимізації процесів навчання та оцінки потенційного впливу на продуктивність системи та ресурси.

Важливо відзначити, що нейронні мережі не позбавлені обмежень, включаючи потребу у великих обсягах позначених навчальних даних, обчислювальних ресурсів, необхідних для навчання та логічного висновку, а також потенційні проблеми інтерпретації та визначеності.

Однак вони залишаються потужним інструментом для вирішення складних проблем і розвитку машинного навчання та ШІ.

2.2 Історія створення генеративно-змагальних нейронних мереж

Походження генеративно-змагальних мереж (англійською Generative adversarial network, також позначається, як GAN) можна простежити з 2014 року, коли вчений Ян Гудфеллоу зі своїми колегами представили цю концепцію у своїй статті з відповідною назвою «Generative Adversarial Networks». Генеративно-змагальні нейронні мережі належать до моделей глибокого машинного навчання, що містять дві нейронні мережі, які називаються генератором та дискримінатором. Наведемо схему мережі (рис.2.1).

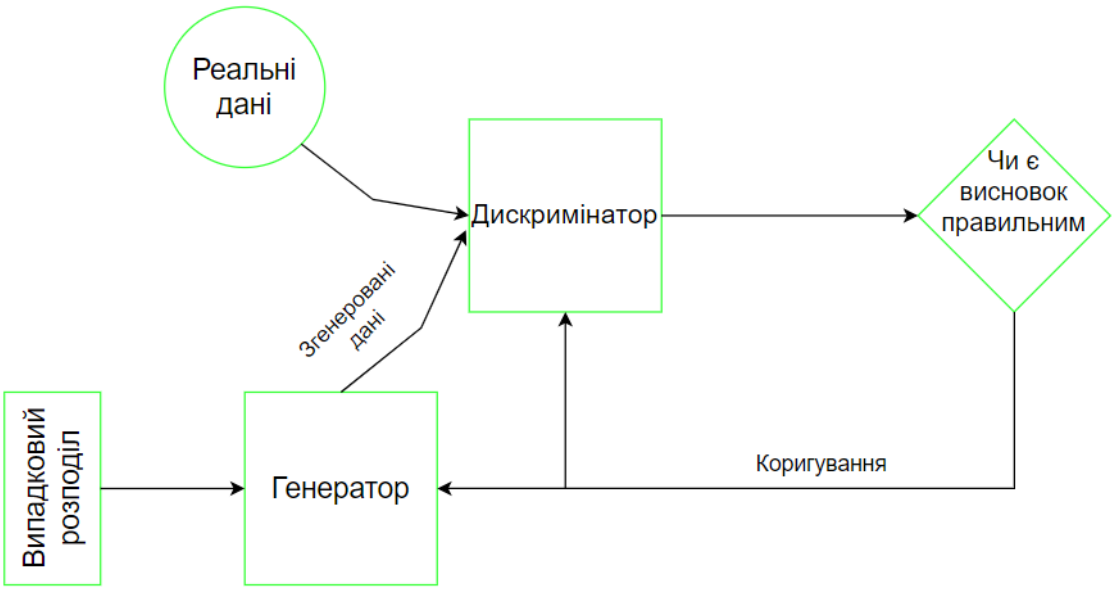


Рисунок 2.1 — Схема роботи варіанту генеративно-змагальної мережі

Генеративно-змагальні нейронні мережі працюються за принципом, що ці дві мережі навчаються одночасно в конкурентному середовищі. Метою генератора є створення реалістичних зразків даних, таких як зображення або текст, тоді як роль дискримінатора полягає в розрізненні реальних даних від згенерованих. Ця конкурентна взаємодія створює цикл зворотного зв'язку, коли обидві мережі періодично покращують свою продуктивність.

Революційна стаття Гудфеллоу представила новий підхід до навчання мереж з використанням зворотного поширення та стохастичного градієнтного спуску. Найважливішим нововведенням було введення змагальної системи навчання, в якій генератор і дискримінатор беруть участь у мінімаксній грі. Такий зв'язок дозволяє генератору навчатися на основі зворотнього зв'язку, що надає дискримінатор, та розробляти більш точні моделі.

З появою генеративно-змагальних мереж, вони одразу привернули значну увагу і знайшли широкий сперк галузей використання, включаючи комп'ютерний зір, обробку природної мови та синтез мовлення. Дослідники та

практики внесли значні вдосконалення та зміни в архітектуру мереж для вирішення проблем та підвищення продуктивності їх роботи.

Видатні розробки влючають глибоку згортку генеративно-змагальних мереж (Deep Convolutional GAN, DCGAN), запропоновану Редфортом разом із своєю командою у 2015 році, який представив згорткові шари для створення високоякісних зображень. Згодом були введені такі методи, як генеративно-змагальні мережі Васерштейна (Wasserstein GAN, WGAN) та прогресивні генеративно-змагальні мережі (Progressive GAN, PGAN), що покращили стабільність та роздільну здатність згенерованих вибірок.

Генеративно-змагальні мережі довели свою корисність у завданнях перекладу зображення в зображення та розширення даних. Вони надають можливість створювати реалістичні зображення та синтезувати нові дані, дозволяють подолати розрив між реальними та синтетичними областями.

Отже, історія генеративно-змагальних нейронних мереж зазначає їх еволюцію від початкового етапу створення у 2014 році до універсальної та потужної системи з багатьма додатками. Розуміння історичного контексту та еволюції мереж має вирішальне значення для оцінки потенційних варіантів використання обмежень і наслідків у різних сферах.

2.3 Застосування генеративно-змагальним нейронних мереж для моделювання фондових ринків

Генеративно-змагальні нейронні мережі, які також називають конкуруючими нейронними мережами, продемонстрували потенціал у моделюванні та прогнозуванні поведінки фондового ринку. Ці мережі використовують потужність двох взаємопов’язаних компонентів: генератора та дискримінатора.

У генеративно-змагальній нейронній мережі генератор відповідає за створення синтетичних зразків даних, які дуже нагадують реальні дані фондового ринку. Його мета полягає в тому, щоб створити точки даних, які фіксують базові закономірності та динаміку фондового ринку.

З іншого боку, дискримінатор бере на себе роль критика, розрізняючи синтетичні дані, створені генератором, і фактичні дані фондового ринку. Його основне завдання — точно визначити джерело даних і розрізнити два типи.

Генератор і дискримінатор працюють у змагальній обстановці, беручи участь у сценарії, схожому на гру. Оскільки генератор генерує синтетичні дані фондового ринку, дискримінатор стає все більш досвідченим у розрізненні синтетичних даних від реальних. Цей ітераційний процес заохочує генератор генерувати вибірки даних, які поступово стають більш реалістичними та точними.

Для досягнення конкурування між двома мережами використовується так звана функція втрат (loss-function). Існує декілька видів функцій втрат:

- втрата дискримінатора (discriminator loss): має на меті виміряти, наскільки добре дискримінатор виконує свою функцію розрізнення справжніх та згенерованих даних;
- втрата генератора (generator loss): вимірює, наскільки добре генератор синтезує дані;
- конкуруюча втрата (adversarial loss): враховує обидва показники, тобто враховує і наскільки добре дискримінатор розрізняє дані, і наскільки добре генератор може ввести дискримінатор у оману.

Наведемо відповідні функції втрати дискримінатора та втрати генератора:

$$loss_{gen} = E(\log\,(1 - D(G(z)))$$

$$loss_{dis} = -E(\log\big(D(x)\big) + E(\log(1 - D(G(z)))$$

де D(x) – імовірність неправильної класифікації справжніх даних дискримінатором, G(z) – згенеровані генератором дані, D(G(z)) – імовірність неправильної класифікації згенерованих даних дискримінатором.

2.4 Приклад пошуку фрактальної розмірності Хігучі для даних фондового ринку

Фрактальна розмірність являє собою математичний метод для оцінки складності та нерегулярності геометричних об’єктів та наборів даних. Фрактали – це складні візерунки, які виявляють самоподібність у різних масштабах, що вказує на узгоджені геометричні характеристики незалежно від збільшення.

Фрактальна розмірність пропонує засоби кількісної оцінки ступеня складності фрактального об’єкта або набору даних. Фрактальна розмірність є нецілим значенням, що описує, як фрактальний об’єкт займає або охоплює простір. Він надає метрику того, як змінюється складність об’єкта при збільшенні чи зменшенні масштабу. Цю концепцію започаткував математик Бенуа Мандельброт у 70-х роках XX століття під час дослідження фрактальної геометрії. Одним з методів вирахування цього показнику є підрахунок фрактальної розмірності за Хігучі.

Фрактальна розмірність Хігучі використовується для аналізу даних часових рядів. Вона ґрунтується на вимірюванні довжини кривої, що утворюється при побудові часового ряду у двовимірному просторі. Алгоритм Хігучі передбачає поділ часового ряду на сегменти різної довжини та обчислення довжини кривої для кожного сегмента. Знайшовши співвідношення між довжиною сегментів та відповідною кривою, ми маємо змогу оцінити фрактальну розмірність. Фрактальна розмірість Хігучі є дуже важливим інструментом для характеристики нерегулярності залежних від часу явищ, саме тому доцільно розглянути її і тут, на прикладі вибірки з фондового ринку.

Наведемо алгоритм дій методу знаходження фрактальної розмірності Хігучі:

1. Маємо деякий часовий ряд $X:\{1,\dots,N\}$ на множини R, що складається з N-кількості точок даних представляє масштаб аналізу.
2. Розділяємо часовий ряд на k_max рівних сегментів.
3. Для кожного з k-сегментів виконуємо підрахунок довжини кривої, що утворюється точками даних сегмента. Для цього маємо наступну формулу:

$$L_m(k) = \frac{N-1}{\left(\frac{N-m}{k}\right)k^2} \sum_{i=1}^{\left(\frac{N-m}{k}\right)} |X_n(m+ik) - X_n(m+(i-1)k)| ,$$

де $k \in \{1,\dots,k_{max}\}$ і $m \in \{1,\dots,k\}$,

4. Отриману довжину усереднюємо по всіх сегментах за формулою, наведеною далі:

$$L(k) = \frac{1}{k}\sum_{m=1}^{k} L_m(k)$$

5. Використовуємо метод регресії найменших квадратів для апроксимації лінії до графіка та визначаємо фрактальну розмірність.

Фрактальна розмірність Хігучі визначається як нахил найкращої лінійної функції через точки даних $(\log\frac{1}{k}; \log L(k))$.

Далі виконаємо пошук фрактальної розмірності Хігучі для певної вибірки даних про ціну акції з фондового ринку за 14 днів (рис.2.3).

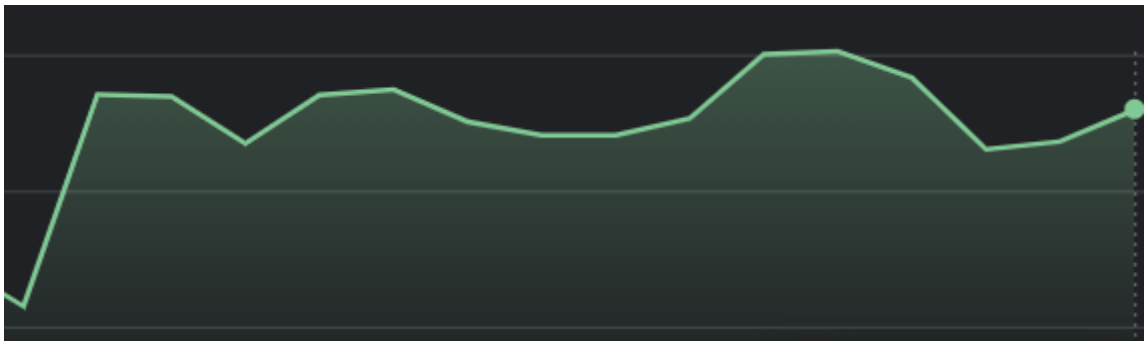


Рисунок 2.3 — Фрагмент графіку цін акції компанії «Apple» за 14 робочих днів

Отримані дані вибірки часового ряду:

X = [173.50, 171.77, 173.56, 173.75, 172.57, 172.06, 172.07, 172.69, 175.05, 175.16,174.20,171.56, 171.84,172.89].

Далі програмно виконуємо підрахунки згідно з алгоритмом методу Хігучі.

Отримане значення фрактальної розмірності приблизно становить: 1,467305. Це свідчить про те, що часовий ряд показує помірний ступінь нерегулярності.

2.5 Переваги використання генеративно-змагальних мереж у моделюванні фондового ринку

В останніх дослідженнях генеративно-змагальні нейронні мережі показують одні з найкращих результатів для задачі моделювання та прогнозування поведінки курсу цін акцій на фондовому ринку. Розрив між отриманими даними за допомогою реалізації методології Гудфелоу та іншими нейронними мережами становить близько 10% точності. Далі розглянемо, в чому саме полягають переваги у зручності генеративно-змагальних мереж для вказаної задачі.

Основним плюсом є генерація даних. Ці мережі надають можливість генерувати синтетичні дані фондового ринку, які точно віддзеркалюють характеристики та шаблони реальних даних. Це дозволяє дослідникам і аналітикам розглядати різни сценарії, тестувати стратегії та оцінювати потенційні результати, не спираючись виключно на історичні дані про поведінку.

Також згенеровані мережею дані досить слушно використовувати для оцінки та управління ризиками інвестицій на фондовому ринку. Аналітики мають змогу моделювати різні ринкові умови, оцінювати вплив різних факторів і оцінювати потенційні ризики, пов'язані з конкретними інвестиційними портфелями.

Окрім перерахованих вище переваг використання синтезовавих даних слід також визначити зручність оптимізації портфеля цінних паперів, оскільки можна розглянути низку ринкових сценаріїв. Завдяки аналізу ефективності портфеля за різних змодельованих умов інвестори можуть приймати обґрунтовані рішення щодо розподілу активів, управління ризиками та стратегії диверсифікації.

Аналіз ринкових тенденцій: ці мережі можуть допомогти в аналізі та прогнозуванні ринкових тенденцій, генеруючи синтетичні дані, які фіксують моделі та динаміку фондового ринку. Аналітики можуть використовувати ці дані для визначення потенційних ринкових тенденцій, прогнозування цінових коливань і прийняття обґрунтованих рішень на основі отриманої інформації.

Однак, попри переваги використання згенерованих даних, важливо зазначити, що застосування генеративно-змагальних нейронних мереж у моделюванні фондового ринку все ще розвивається, і є проблеми, які потрібно подолати. Ці складності включають вимогу до великих і різноманітних навчальних наборів даних, вирішення проблеми внутрішньої невизначеності та нестабільності фондового ринку та забезпечення надійності та точності згенерованих синтетичних даних.

Розуміння застосування генеративно-змагальних нейронних мереж у моделюванні фондового ринку може дати цінну інформацію про потенційні переваги та обмеження цього підходу. Це може допомогти оцінити придатність цих мереж для конкретних випадків використання, розробити ефективні стратегії моделювання та оцінити їхній вплив на процеси прийняття рішень у сфері фондового ринку.

2.6 Допоміжні метрики для генеративно-змагальних мереж

Середня квадратична похибка, корінь середньої квадратичної похибки та середня абсолютна похибка є важливими показниками для оцінки продуктивності генеративно-змагальних нейронних мереж у різних завданнях. Ці показники дають уявлення про якість і точність згенерованих даних у порівнянні з фактичними або реальними даними.

Середня квадратична похибка(Mean Squared Error, MSE) вимірює середню квадратичну різницю між згенерованими вибірками та реальними даними. Вона обчислює середнє значення квадратів похибок для кожної точки даних та задається наступною формулою:

$$MSE = \frac{1}{N}\sum_{1}^{N}\left(y_{\text{ген}} - y_{\text{реал}}\right)^2,$$

де $y_{\text{ген}}$ – значення генерованої вибірки, $y_{\text{реал}}$ – значення реальної вибірки.

Нижче значення цього значення означає кращу продуктивність, оскільки це вказує на те, що згенерована вибірка досить сильно відповідає реальним даним.

Квадратний корінь із середньої квадратичної похибки(Root Mean Squared Error, RMSE) являє собою корінь із середньоквадратичної різниці між згенерованими та реальними даними. Цей показник забезпечує вимірювання середньої відстані між значеннями та його пошук виконується за наступною формулою:

$$RMSE = \sqrt{\frac{1}{N}\sum_{1}^{N}\left(y_{\text{ген}} - y_{\text{реал}}\right)^2}$$

Середня абсолютна похибка (Mean Absolute Error, MEA) виконує підрахунок середньої абсолютної різниці між згенерованими та реальними

вибірками. Вона характеризує середню величину похибок без урахування їхнього напрямку. Виконати її пошук можна за формулою:

$$MAE = \frac{1}{N}\sum_{1}^{N}\left|y_{\text{ген}} - y_{\text{реал}}\right|$$

Ці показники надають кількісні оцінки продуктивності генеративно-змагальних мереж з точки зору генерування даних. Однак важливо зазначити, що вони не охоплюють всі аспекти якості згенерованих даних, таких як перцепційна подібність або семантична релевантність.

Висновки до розділу 2

В даному розділі було розглянуто основні поняття нейронних мереж та більш детально досліджено специфіку створення та використання генеративно-змагальних нейронних мереж. Проаналізувавши переваги у використанні цього типу мереж було доведено ефективність їх застосування для задачі прогнозування поведінки ринкових процесів. Після цього було розглянуто фрактальну розмірність за Хігучі, яка слугує допоміжним методом для аналізу подібності часових рядів та наведено приклад розрахунку цього показника на вибірці цін акційних паперів реальної компанії. Також було розглянуто приклади допоміжних метрик для оцінки якості роботи моделі, які є помічниками в аналізі точності прогнозування.

3 МОДЕЛЮВАННЯ І ПРОГНОЗУВАННЯ ЦІНИ АКЦІЙ

3.1. Пошук та вибір даних для аналізу

Для того, щоб провести дослідження варто спочатку визначитися з даними, на базі яких будемо проводити аналіз та навчати модель. Даних не має бути занадто багато, через значну кількість обчислень, однак і не має бути замало для правильного пошуку закономірностей та достатньої тренованості моделі, що збільшує точність прогнозу. Задля достовірності даних було обрано виконувати пошук за допомогою сервісу Yahoo Finance, який надає інформацію про показники акцій різних компаній протягом значних часових проміжків. Далі наведемо для прикладу фрагмент обраної бази даних (рис. 3.1).

Date	Open	High	Low	Volume	Close
1994-10-31	29.125	29.188	27.750	10081600	27.781
1994-11-07	27.938	28.125	27.188	8826400	27.375
1994-11-14	27.344	27.563	27.000	8109600	27.063
1994-11-21	27.000	27.375	26.375	7005600	26.563
1994-11-28	26.969	27.281	26.469	10472400	26.813
1994-12-05	26.625	26.781	26.219	7176400	26.438
1994-12-12	26.750	26.844	26.375	12481200	26.781
1994-12-19	26.719	27.000	26.594	5967600	26.750
1994-12-26	26.750	27.281	26.750	5313600	26.938
1995-01-02	26.938	27.063	26.813	4791200	26.875

Рисунок 3.1 — Фрагмент обраного датасету

Вибір впав на акції британської компанії «Shell», оскільки на даний момент вона є однією з найбільших транснаціональних компаній у світі. Обраний ресурс надає нам дані про акції обраної компанії починаючи з 31 жовтня 1994 року і до сьогодення. Оскільки варто проаналізувати зміну за значний проміжок часу, однак оброблювати великий обсяг даних не є простою задачею, було вирішено брати дані про показники акцій не кожного дня, а щотижнево. Загалом отримана збірка даних має 1493 записи, в кожному з яких зазначено наступні параметри: ціну при відкритті, ціну при закритті, найвищу ціну, найнижчу ціну та обсяг, опис кожного з яких було виконано у пункті 1.4. Попередньо вище було наведено фрагмент, а саме перші 10 записів датасету, але для більш детального огляду обраних даних краще переглянути їх графічну візуалізацію (рис.3.2).

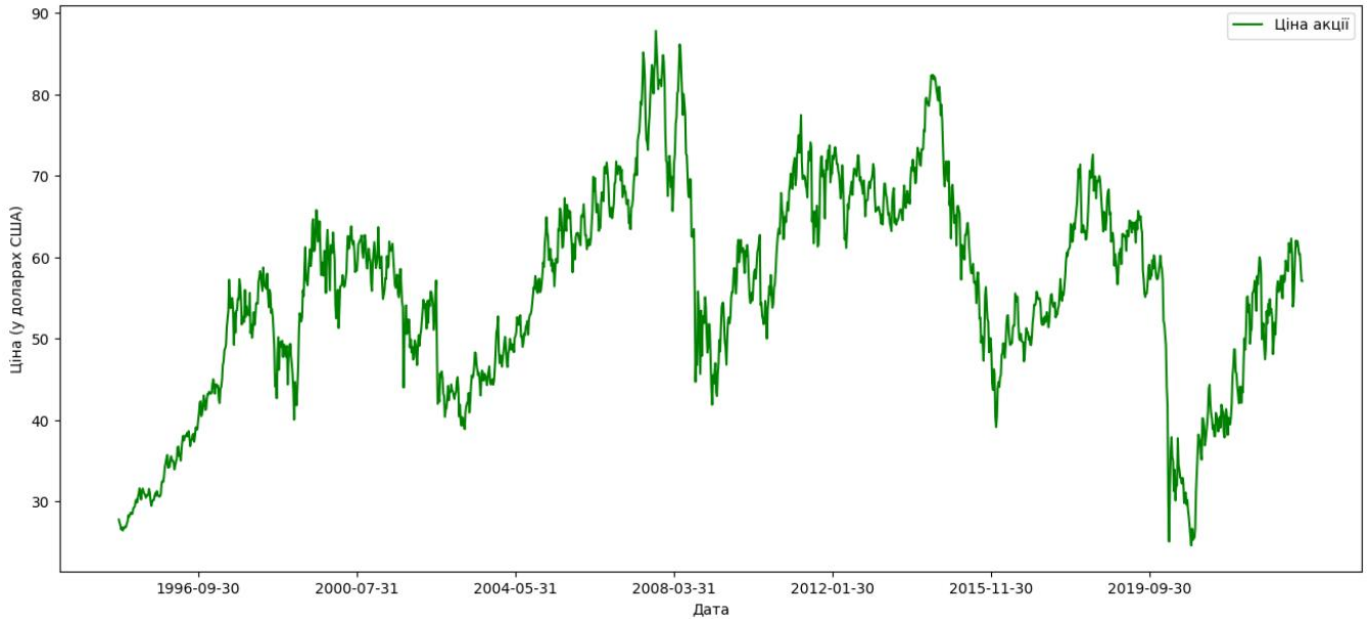


Рисунок 3.2 — Графічне представлення даних

На наведеному графіку помітно коливання у певні проміжку часу. Значні коливання припали на часи нестабільності у світі. Так, наприклад, значні спади курсу акцій відбулися під час глобальної економічної кризи у 2008 році, при російському вторгненні в Україну у 2014 році та початку пандемії коронавірусу у 2019 році.

3.2. Реалізація програмного продукту

Програму було вирішено реалізувати мовою програмування високого рівня Python. Під час написання програми було використано бібліотеки pandas, numpy, torch, matplotlib, sklearn та py-math. Сама програма виконує моделювання поведінки та прогнозування курсу акцій обраної компанії за допомогою виконаної генеративно-змагальної нейронної мережі.

З початку програма зчитує дані з файлу формату .csv, що використовується для представлення даних у вигляді таблиці. Далі отримана база даних представляється у формі графіку. Після цього робимо дії для підготовки даних до аналізу. Спочатку виконуємо розбиття вибірки на тренувальну та тестову, далі проводимо нормування даних, оскільки колонки нашої бази даних мають значні розбіжності у порядку даних, наприклад показник «об'єму» та показник «ціни відкриття», що погано впливає на швидкість навчання нейронної мережі. Далі реалізовуємо моделі генератора та дискримінатора.

Дискримінатор являє собою модель, що генерує дані на основі вхідного в нього вектору. Він складається з трьох шарів вентильних рекурентних вузлів(GRU) та трьох лінійних шарів. Сутність полягає у прийнятті вектора вхідних даних та послідовного використання перших трьох для обробки вхідних даних та других трьох для генерування вихідних даних. Також застосовується операціє dropout для запобіганню перенавченості моделі.

В свою чергу модель дискримінатора виконує розпізнавання даних та поділ їх на згенеровані та реальні та складається з трьох шарів згортки, трьох лінійних шарів та функцій активації. Вхідні дані у модель дискримінатора оброблюються через усі шари згортки та функції активації, а далі переходять до лінійних шарів разом з функціями активації. Після цього за допомогою сигмоїдної функції активації модель генерує вихід, яка являє собою імовірність реальності вхідних даних.

Перед початком тренування наших моделей, слід також задати оптимізатори для генератора та дискримінатора. Для кожного з них було використано вбудований у пакет torch оптимізатор Adam. Він являє собою один з найефективніших алгоритмів оптимізації, що залучає експоненційне ковзне середнє градієнта та квадратичний градієнт. Це дозволяє ефективно оновлювати ваги для моделей та сприяє більш ефективному навчанню. Після цього виконуємо навчання нашої нейронної мережі ітеративним проходженням по епохам. Відбувається генерація даних та їх об’єднання з реальними даними, далі відбувається передача даних до дискримінатора. Дискримінатор оброблює дані та дає вердикт для реальних та згенерованих даних. Після цього йде оновлення вагів для дискримінатора та генератора та виконується обрахунок значень loss-функцій окремо для кожного з них. На цьому навчання моделей завершується.

На наступному етапі модель генератора надає прогнози для навчальної та тестової вибірки, після чого повертаємо отримані дані до первинного вигляду, тобто до нормалізації. Зберігши результати прогнозування виводимо відповідні графіки та обраховуємо похибки MSE, RMSE та MAE, дані про які було наведено у розділі 2.5.

3.3. Аналіз роботи програми

У цьому пункті розглядається аналіз роботи програми та підбір параметрів, які найточніше допоможуть нам описати поведінку курсу акцій обраної компанії. Для початку намагаємося з’ясувати оптимальне розділення вибірок на навчальну та тестову. Для початку розглянемо поділ у співвідношенні 80% до 20% (рис.3.3 – рис.3.6) , відповідно навчальної до тестової вибірок. Після

цього ділимо початкову базу даних у співвідношенні 60% до 40% (рис.3.7 – рис.3.10). Наведемо відповідні зображення результатів роботи програми.

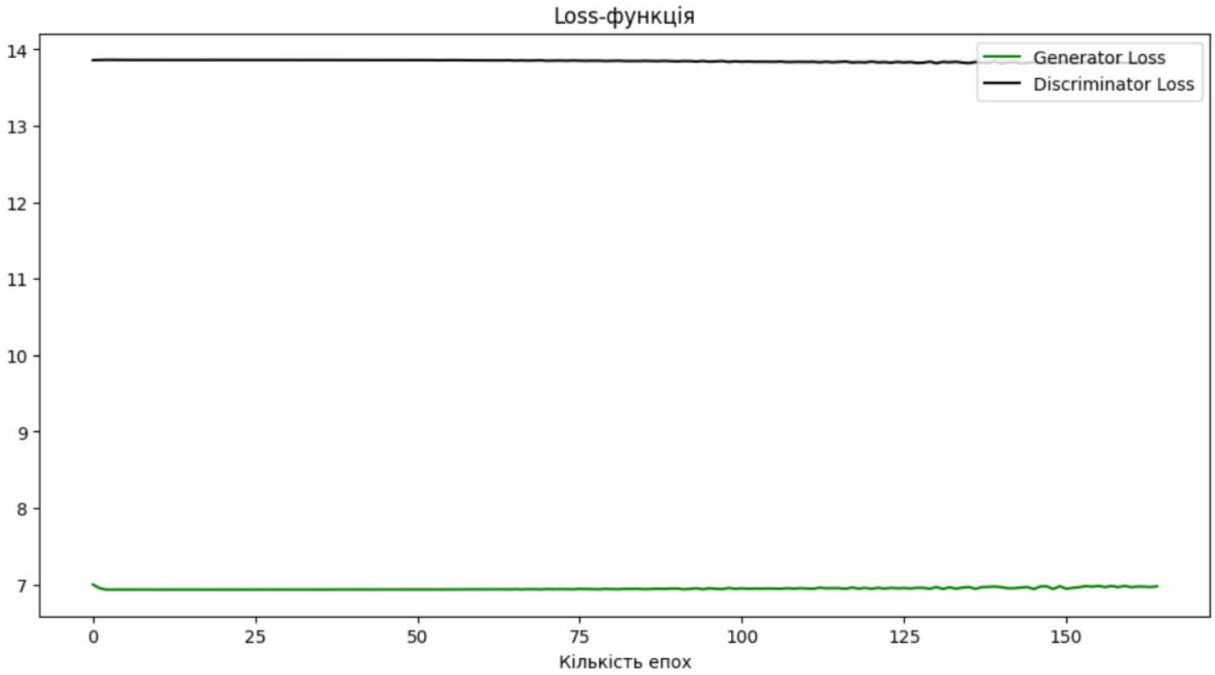


Рисунок 3.3 — Значення відповідних loss-функцій для навчальної вибірки 80%

```
Значення похибок:
RMSE:3.2592263147423286
MAE:2.4273724461176145
```

Рисунок 3.4 — Значення відповідних метрик для тестової вибірки 20%

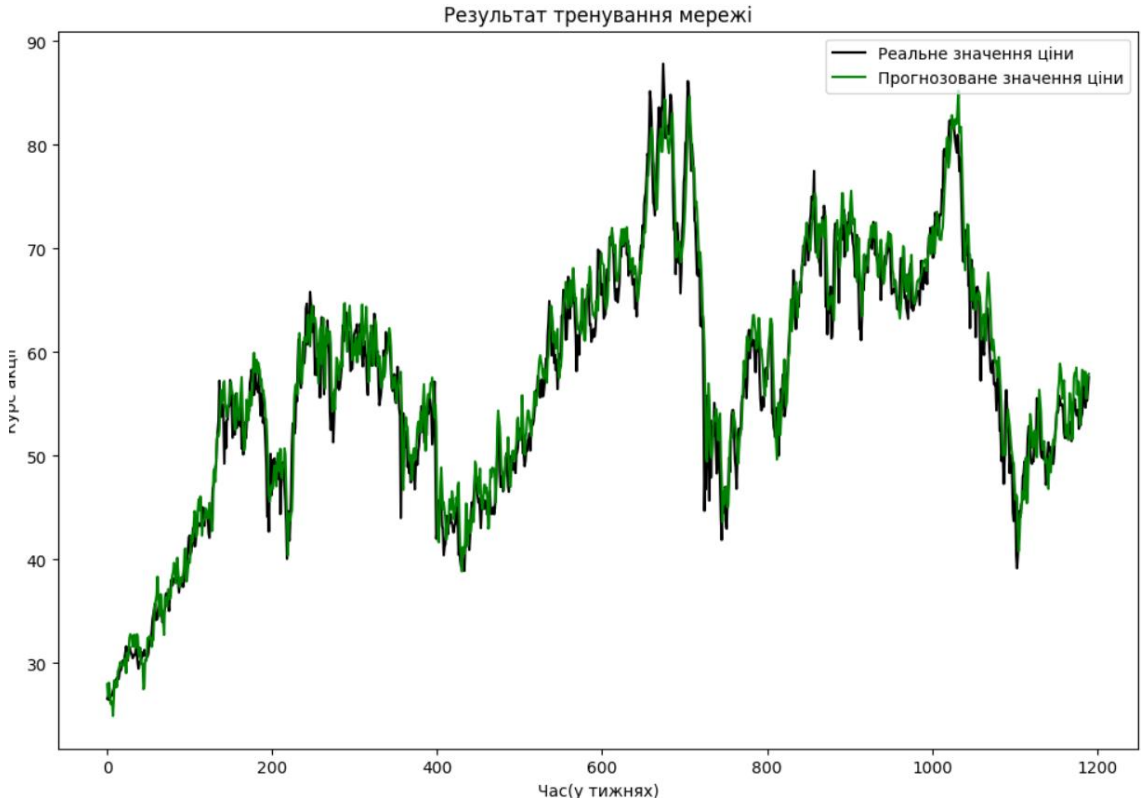


Рисунок 3.5 — Результат тренування мережі для навчальної вибірки 80%

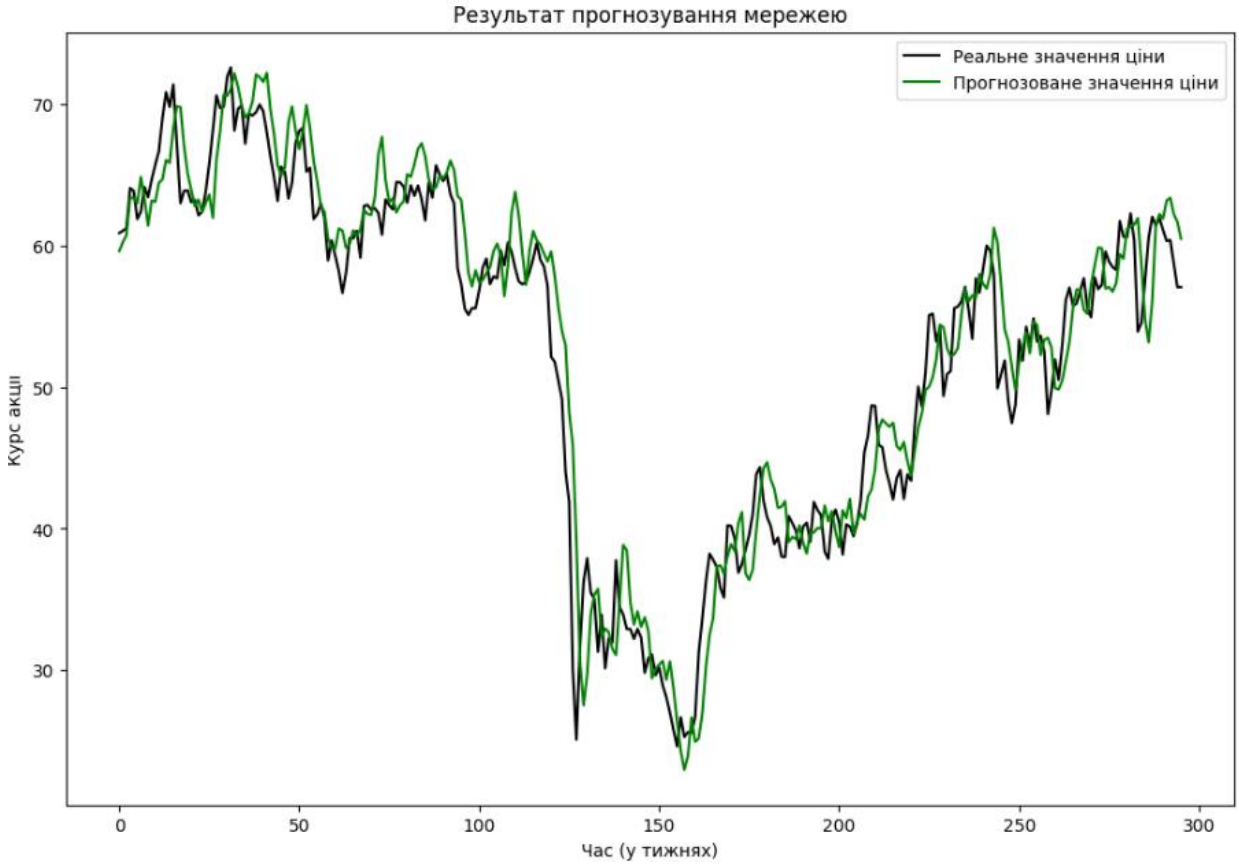


Рисунок 3.6 — Результат прогнозування мережі для навчальної вибірки 80%

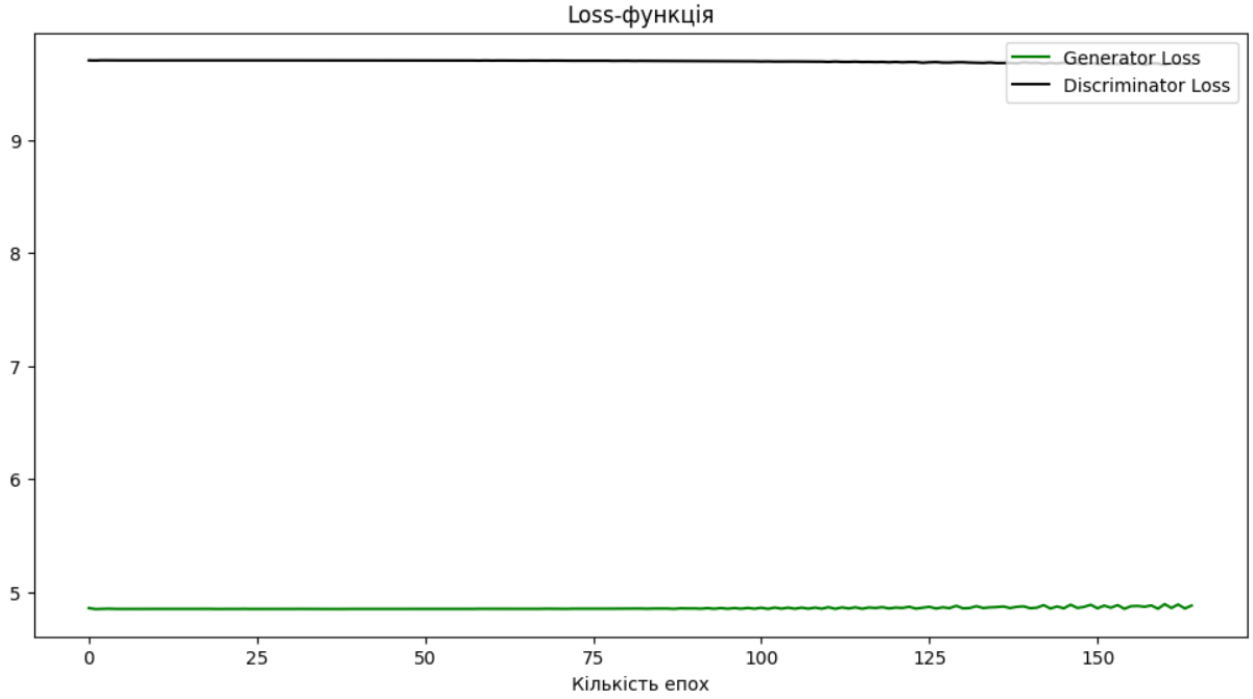


Рисунок 3.7 — Значення відповідних loss-функцій для навчальної вибірки 60%

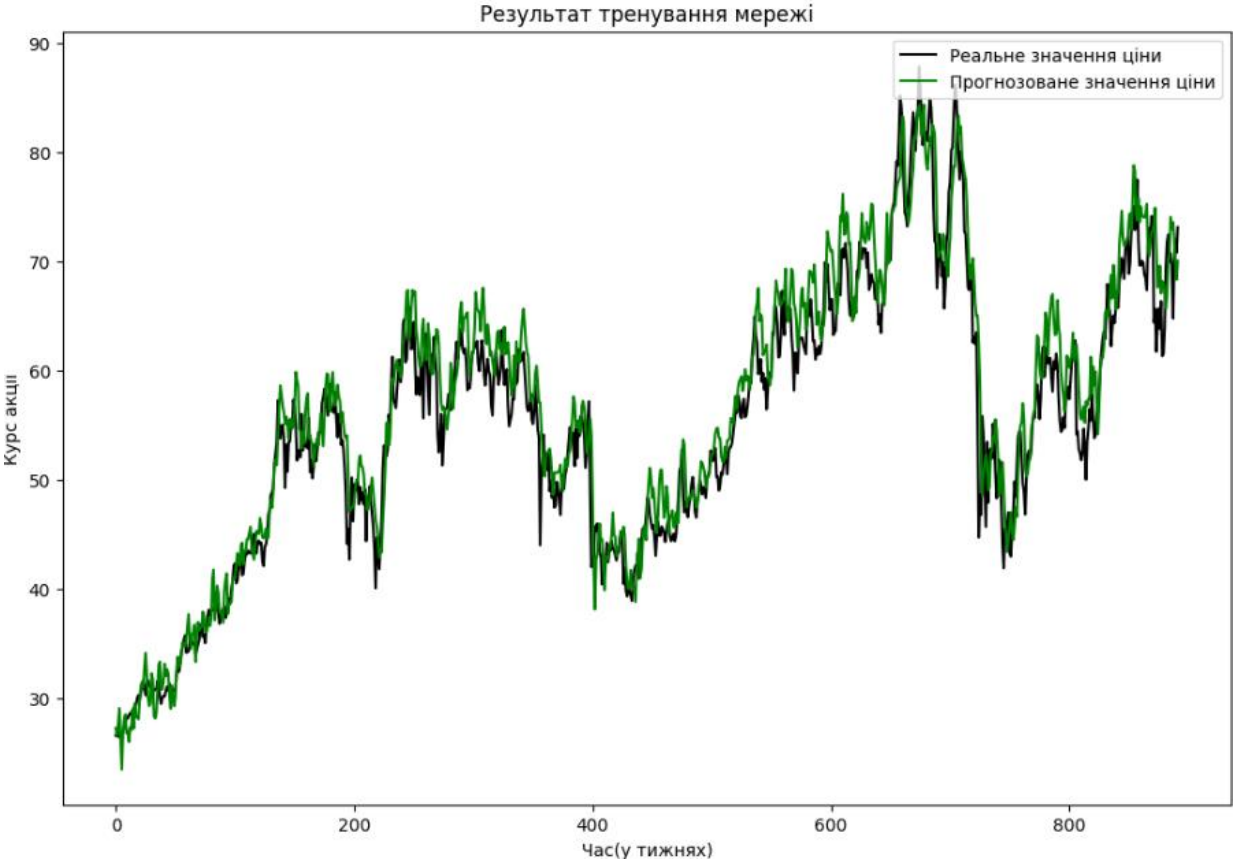


Рисунок 3.8 — Результат тренування мережі для навчальної вибірки 60%

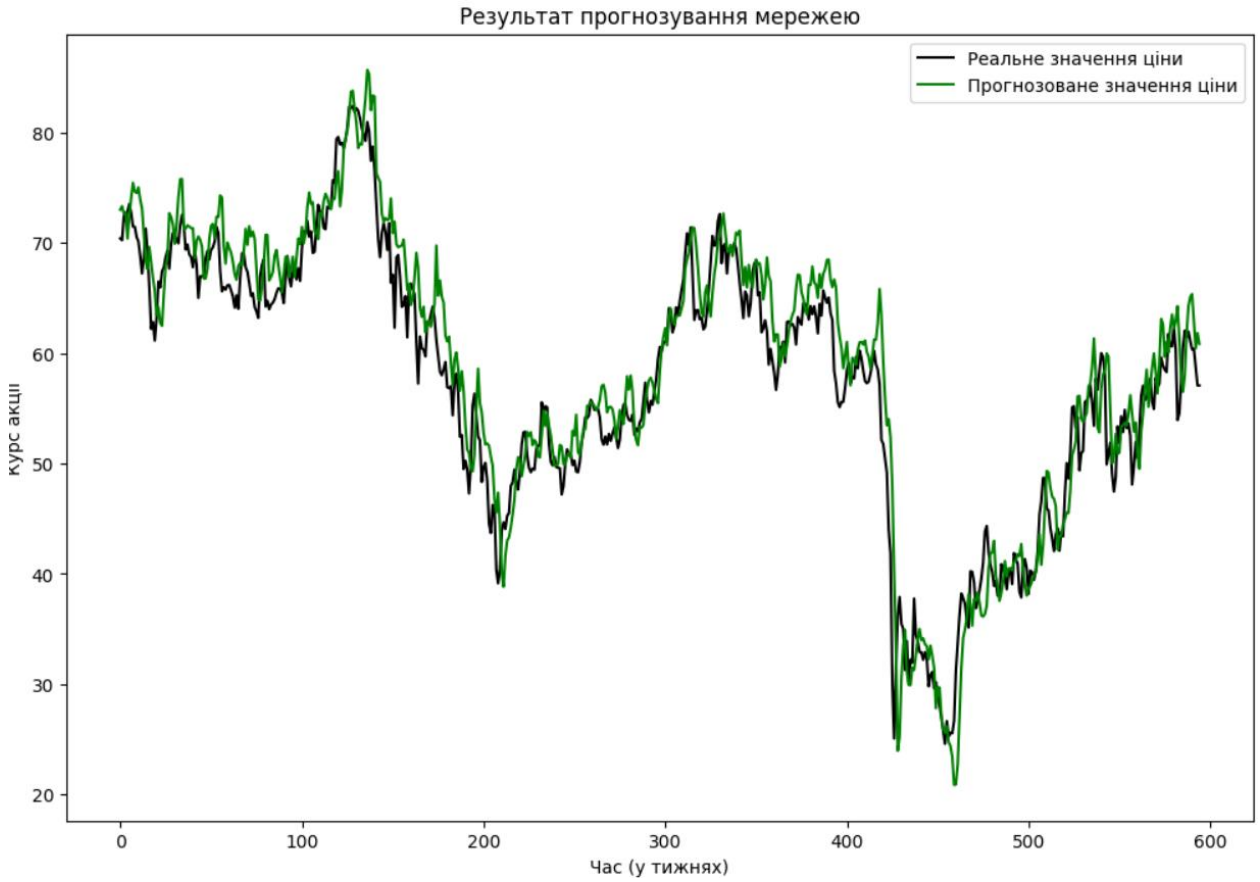


Рисунок 3.9 — Результат прогнозування мережі для навчальної вибірки 60%

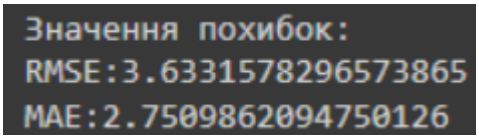


Рисунок 3.10 — Значення відповідних метрик для тестової вибірки 40%

З отриманих результатів роботи програми можна зробити висновок, що при використанні поділу на 80 та 20 відсотків, отримано менші середньоквадратичну та середню абсолютну похибки. Отже, доцільніше поділяти нашу базу даних саме першим запропонованим варіантом.

Наступним важливим параметром, який треба визначити це оптимальна кількість епох, яку слід обробити нашій мережі. Для попереднього дослідження співвідношення поділу вибірки для обох варіантів кількість епох становила 165. Однак, вона може виявитися завеликою, тобто мережа виконує зайву роботу, що і слід перевірити. Зараз запропоновано взяти меншу кількість епох, а саме 50. Далі наведемо відповідні результати роботи програми (рис. 3.11 — 3.14).

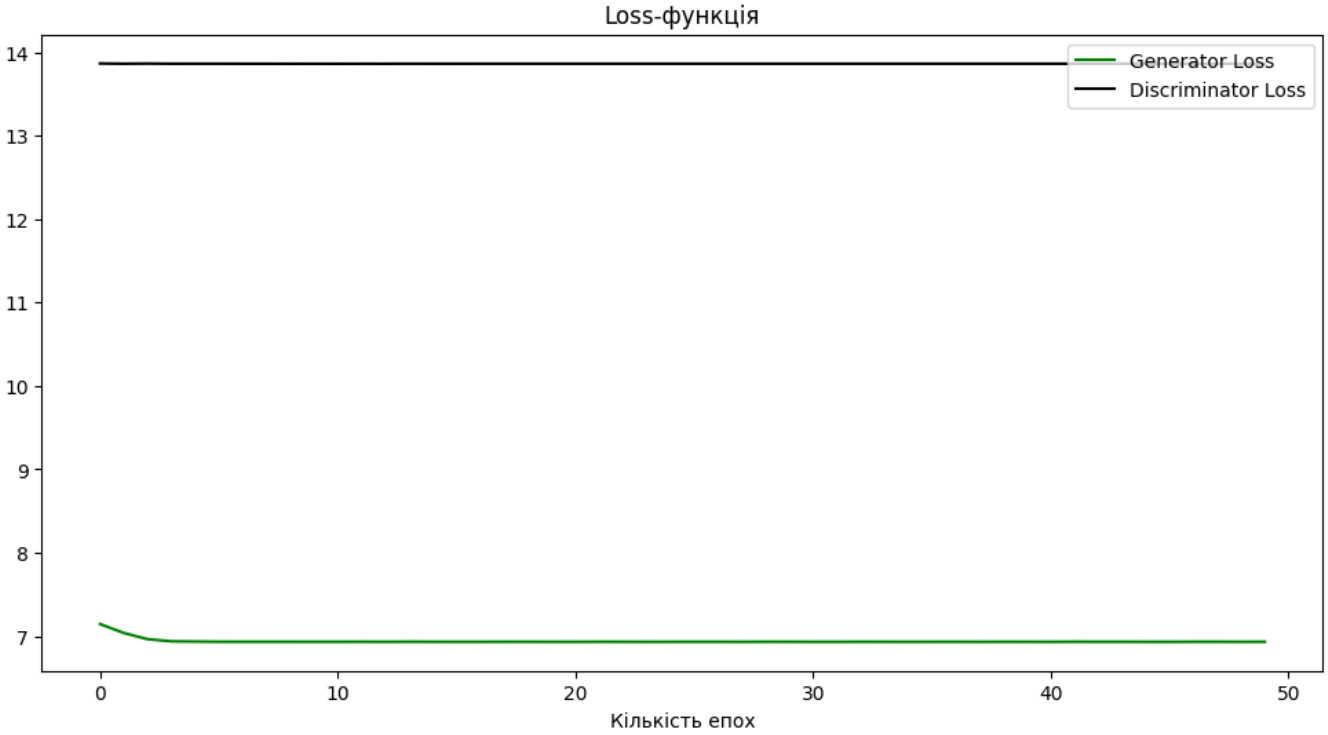


Рисунок 3.11 — Значення відповідних loss-функцій для 50 епох

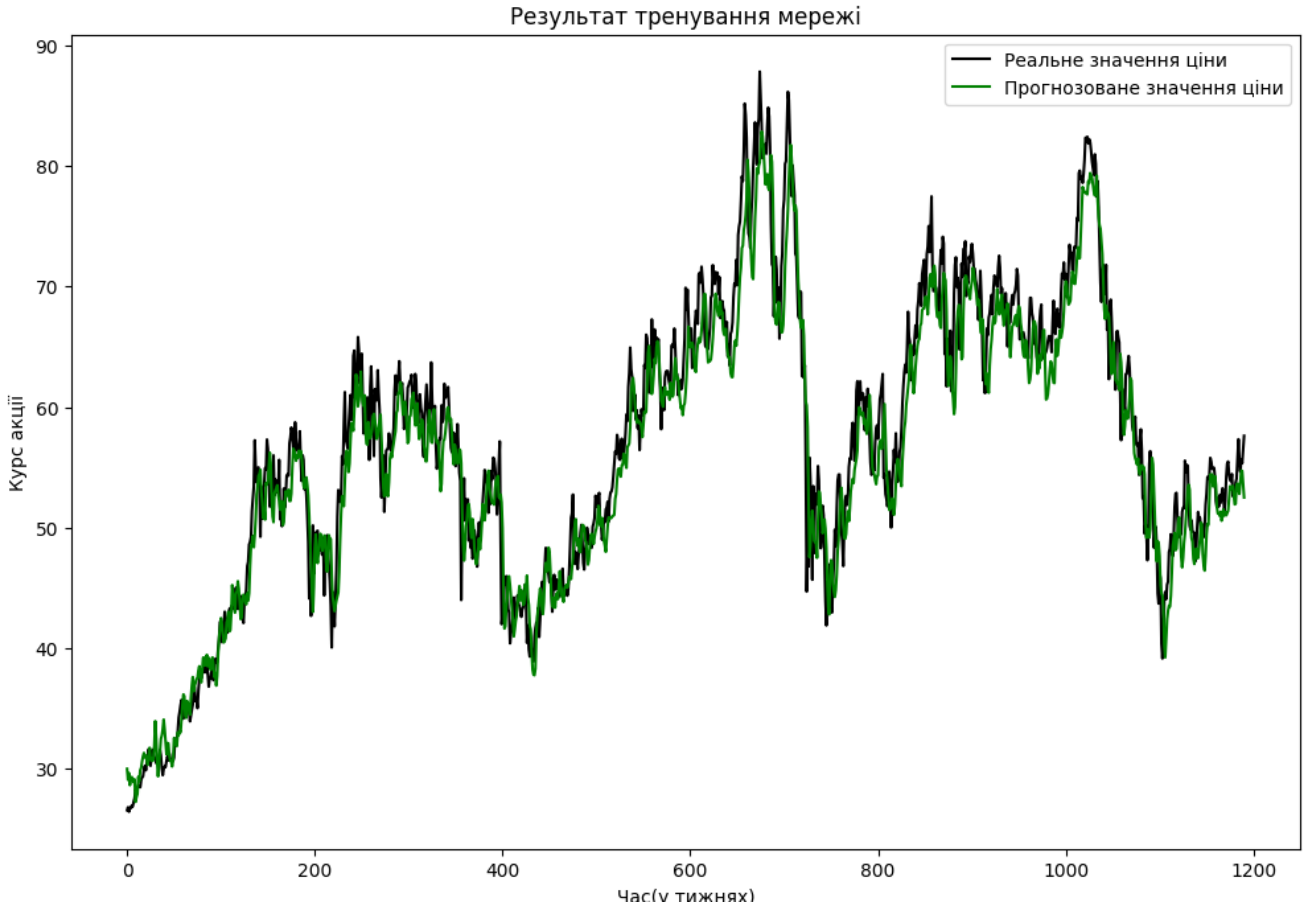


Рисунок 3.12 — Результат тренування мережі після 50 епох

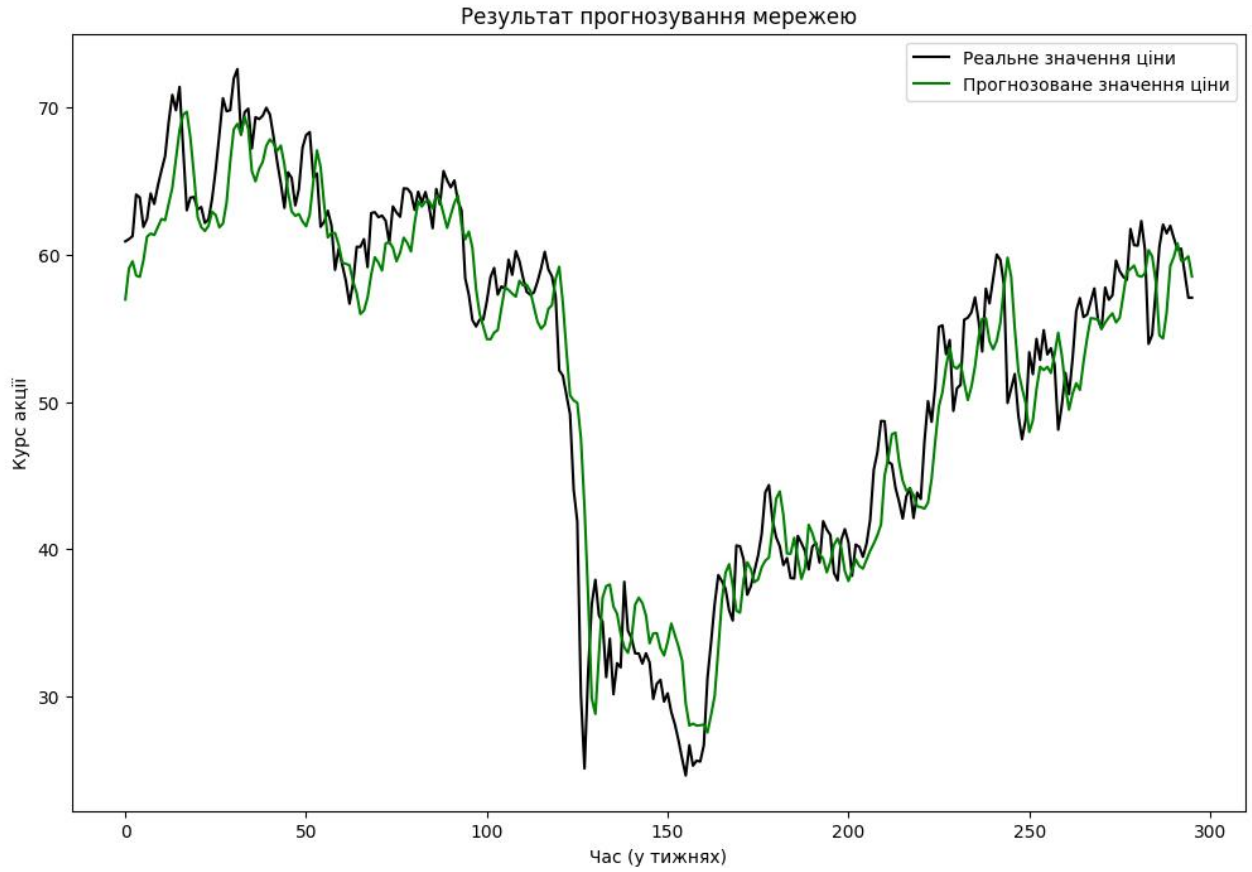


Рисунок 3.13 — Результат прогнозування мережі після 50 епох

```
Значення похибок:
RMSE:3.694750688818959
MAE:2.887110955896419
```

Рисунок 3.14 — Значення відповідних метрик після 50 епох

Після отриманих результатів роботи програми маємо, що зміна похибки при зменшенні кількості епох є незначною, та, враховуючи помітну різницю в часі роботи програми при 50 та 165 епохах, можемо зробити висновок, що більш вдалим рішенням буде навчати мережу протягом 50 епох.

Висновки до розділу 3

У ході роботи над цим розділом було розроблено програму на мові програмування Python для прогнозування та моделювання курсу цін акцій

компанії Shell. У ході виконання програми було реалізовано генеративно-змагальну нейронну мережу та виконано вивід результатів її роботи. Було проведено пошук та вибір бази даних. Вибір впав на акції британської нафтопереробної компанії Shell. База даних містить у собі дані про ціни акцій обраної компанії з початку її виходу на фондових ринок. При виконані досліджень було прийнято рішення з приводу оптимальних параметрів для роботи програми. Реалізований програмний продукт надає досить точні дані з приводу цін цінних паперів та дуже точно описує їх поведінку.

4 ФУНКЦІОНАЛЬНО ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі буде проведено оцінювання основних характеристик для програмного продукту, що спеціалізується на прогнозуванні цін на цінні папери.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

Застосовуємо метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу та моделювання фондового ринку. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного

продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є наступні:

- функціонування на персональних комп'ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обгрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту, який прогнозує поведінку цін на акції. Беручи за основу цю функцію, можна виділити наступні:

F_1 – вибір мови програмування.

F_2 – вибір середовища програмування.

F_3 – вибір основного допоміжного пакету для машинного навчання.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1:

а) Python.

б) R.

Функція F_2:

а) Google Colaboratory.

б) Jupyter Notebook.

Функція F_3:

а) tensorflow;

б) py-torch.

Варіанти реалізації основних функцій наведені у морфологічній карті системи. На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (рис. 4.1).

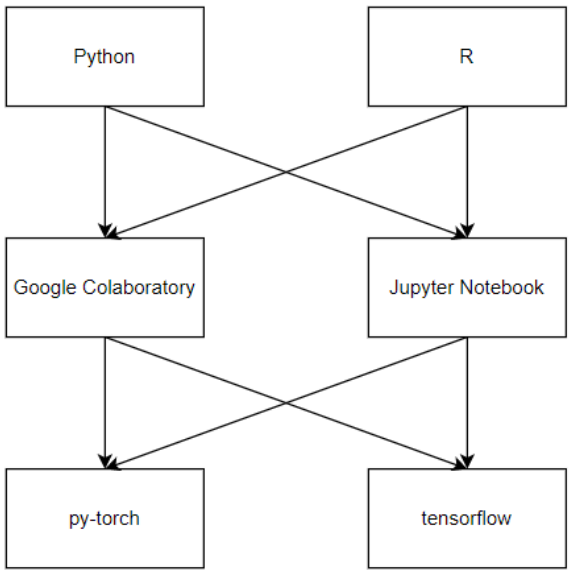


Рисунок 4.1 — Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантиів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

Таблиця 4.1 — Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	а	Більш універсальний та поширений	Обробка великих обсягів займає значну кількість оперативної пам’яті
	б	Більше сфокусований на статистичному аналізі	Менш продуктивний при обробці великої кількості даних

Продовження таблиці 4.1

F_2	а	Не потребує завантаження на локальний комп’ютер, підтримує технології для прискорення обчислень	Має бути стабільне підключення до інтернет-мережі.
	б	Має підтримку більшої кількості мов програмування	Проблема з продуктивністю при виконанні складних обчислень
F_3	а	Забезпечує кращу оптимізацію, має велику кількість визначених операцій та функцій	Складність навчання моделі
		Є більш гнучким в розробці моделі, більше можливостей для користувацьких моделей та операцій б	Не завжди демонструє оптимальну продуктивність

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1 *:*

Надаємо перевагу більшій поширеності та універсальності, через що обираємо варіант А.

Функція F_2:

Повна хмарність сервісу надає більше гнучкості для роботи декількох осіб у команді, однак доступ до мережі інтернет є зовсім не завжди, тому варіант А і Б є досить рівноцінними.

Функція F_3:

Дуже важливою є точніша розробка моделі, тому схиляємося до варіанту Б. Таким чином, будемо розглядати такий варіанти реалізації ПП:

F_1а - F_2а - F_3б

F_1а - F_2б- F_3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- X1 – швидкодія мови програмування;
- X2 – об’єм пам’яті для збереження даних;
- X3 – час навчання моделі;
- X4 – зручність використання програмного продукту

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту (табл.4.2).

Таблиця 4.2 — Основні параметри ПП

Назва параметра	Умовні позначення	Одиниця виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	Оп/мс	6000	13000	18000
Об'єм пам'яті для збереження даних	X2	Мб	64	32	16
Час навчання моделі	X3	с	600	300	100
Зручність використання програмного продукту	X4	%	30	60	100

На основі цих даних будуються графічні характеристики параметрів (рис. 4.2 — рис. 4.5).

Рисунок 4.2 — Параметр X1

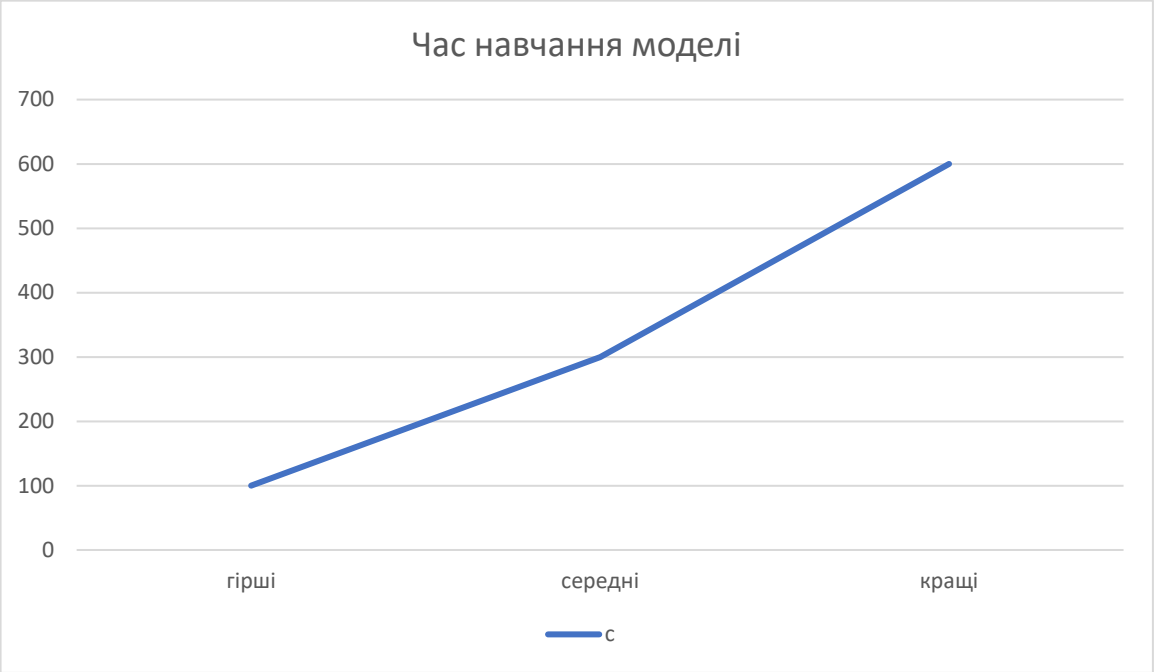


Рисунок 4.3 — Параметр X2

Рисунок 4.4 — Параметр X3

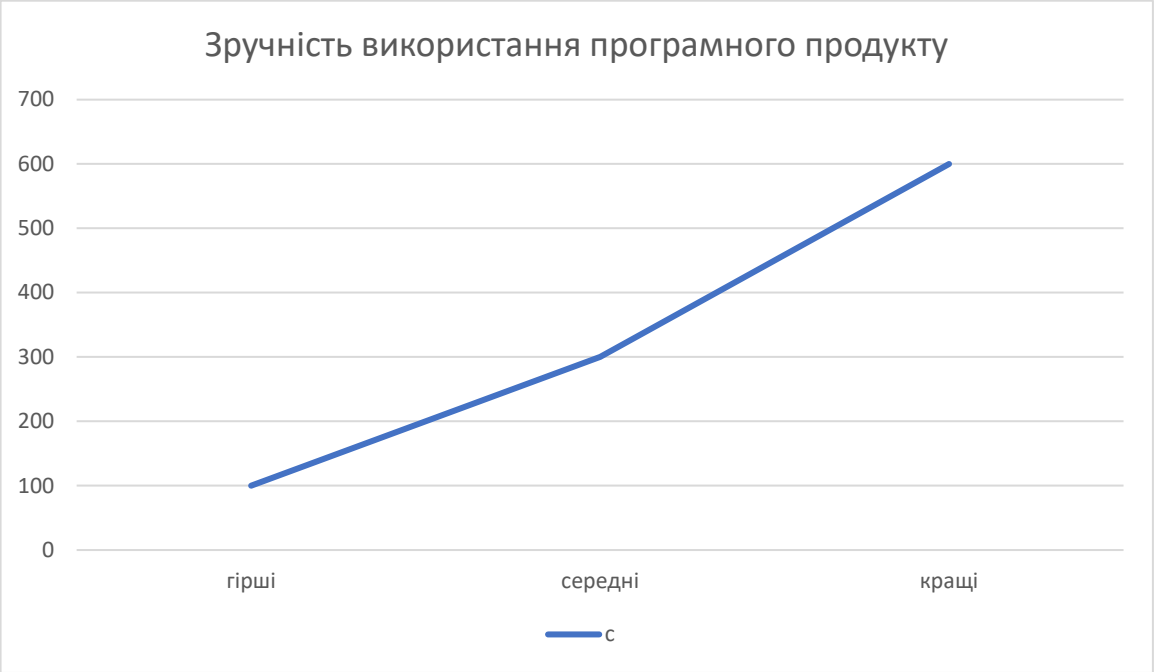


Рисунок 4.5 — Параметр X4

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень. Наведемо дані у таблиці 4.3.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;
– перевірку придатності експертних оцінок для подальшого використання;
– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Таблиця 4.3 — Експертна оцінка параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	2	1	1	1	1	1	1	8	-9,5	90,25
X2	Об'єм пам'яті для збереження даних	Мб	4	3	3	4	4	4	3	25	7,5	56,25
X3	Час навчання моделі	мс	1	2	2	3	2	2	2	14	-3,5	12,25
X4	Зручність використання програмного продукту	%	3	4	4	2	3	3	4	23	5,5	30,25
	Разом		10	10	10	10	10	10	10	70	0	189

Середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5$$

Відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T.$$

Сума відхилень по всім параметрам повинна дорівнювати 0; загальна сума квадратів відхилення становить:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 189.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{189 \cdot 12}{7^2(4^3 - 4)} = \frac{2268}{2940} \approx 0{,}77143 > W_k = 0{,}67.$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, який становить 0,67. Скориставшись

результатами ранжирування, проведемо попарне порівняння всіх параметрів (табл. 4.4).

Таблиця 4.4 — Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1,5
X1 і X3	<	>	>	>	>	>	>	>	1,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	<	>	>	<	<	<	>	<	0,5
X3 і X4	>	>	>	<	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}. \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю $A=\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (4.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На дру-

гому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \quad (4.9)$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \quad (4.10)$$

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

Таблиця 4.5 — Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1	1,5	1,5	1,5	5,5	0,34	21,25	0,36	77,875	0,36
X2	0,5	1	0,5	0,5	2,5	0,16	9,25	0,16	34,125	0,16
X3	0,5	1,5	1	1,5	4,5	0,28	16,25	0,27	59,125	0,28
X4	0,5	1,5	0,5	1	3,5	0,22	12,25	0,21	41,875	0,20
Всього:					16	1	59	1	213	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2* (Об’єм пам’яті), *X3* (час попередньої обробки даних) та *X4* (потенційний об’єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X1* (швидкість роботи мови програмування) обрано не найгіршим. Розрахунки наведемо у таблиці 4.6.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} \text{ ,} \qquad (4.11)$$

де: n – кількість параметрів; $K_{вi}$– коефіцієнт вагомості i–го параметра; B_i – оцінка i–го параметра в балах.

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

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	15000	8	0,36	2,88
F2	А	X3	250	7	0,28	1,96
	Б	X4	70	8	0,2	1,6
F3	Б	X2	32	5	0.16	0,8

За даними з таблиці 4.6 за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \qquad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_1 = 2{,}88 + 1{,}96 + 0{,}8 = 5{,}64;$$
$$K_2 = 2{,}88 + 1{,}6 + 0{,}8 = 5{,}28.$$

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

4.6 Економічний аналіз варіантів розробки програмного продукту

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;
2. Розробка програмної оболонки;

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

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

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

Загальна трудомісткість обчислюється як:

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.13)$$

де T_P – трудомісткість розробки ПП;

$K_П$ – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

$K_М$ – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

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

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =50 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.8. Поправочний коефіцієнт, який враховує складність контролю вхідної та

вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.9. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$Т_1 = 50 \cdot 1.8 \cdot 0.9 \cdot 1 = 81 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =25 людино-днів, $К_П$ =0.8, $К_{СК}$ = 1, $К_{СТ}$ =0.7:

$$Т_2 = 25 \cdot 0.8 \cdot 0.7 \cdot 1 = 14 \text{ людино-днів.}$$

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

$$Т_I = (81 + 14 + 4.8 + 14) \cdot 8 = 910{,}4 \text{ людино-годин.}$$

$$Т_{II} = (81 + 14 + 6.91 + 14) \cdot 8 = 927{,}28 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 35000 грн. та один аналітик в області даних з окладом 40000 грн. Визначимо середню зарплату за годину за формулою:

$$С_ч = \frac{М}{T_m \cdot t} \text{ грн.,} \qquad (4.14)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$С_ч = \frac{35000 + 35000 + 40000}{3 \cdot 21 \cdot 8} = 218{,}25 \text{ грн.} \qquad (4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д\,, \qquad (4.16)$$

де $С_ч$– величина погодинної оплати праці програміста;

$Т_i$ – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 218,25 \cdot 910,4 \cdot 1.2 = 238433,76$ грн.

II. $С_{ЗП} = 218,25 \cdot 927,28 \cdot 1.2 = 242854,63$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 202857,14 \cdot 0.22 = 52455,43$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 206873,38 \cdot 0.22 = 53428,02$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 35000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 35000 \cdot 0,2 = 84000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 84000 \cdot (1 + 0.2) = 100800 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 100800 \cdot 0,22 = 22176 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 45000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.4 \cdot 0.25 \cdot 45000 = 15750 \text{ грн.,}$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.4 \cdot 45000 \cdot 0.08 = 5040 \text{ грн.,}$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 16) \cdot 8 \cdot 0.75 =$$

$$= 1398,2 \text{ години,}$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1398,2 \cdot 0,35 \cdot 0,75 \cdot 4.87 = 1787,42 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 45000 \cdot 0,67 = 30150 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н, \quad (4.17)$$

$С_{ЕКС} = 100800 + 22176 + 15750 + 5040 + 1787,42 + 30150 = 175703,42$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 175703,42 / 1398,2 = 125,66 \text{ грн/год.}$$

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

$$С_М = С_{М-Г} \cdot Т, \quad (4.18)$$

I. $С_М = 125,66 \cdot 910,4 = 114400,86$ грн.

II. $С_М = 125,66 \cdot 927,28 = 116522$ грн.

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67, \quad (4.19)$$

I. $С_Н = 238433,76 \cdot 0,67 = 159750,62$ грн.

II. $С_Н = 242854,63 \cdot 0,67 = 162712,60$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_{М} + C_{Н}, \quad (4.20)$$

I. $C_{ПП}$ = 238433,76 + 52455,43 + 114400,86 + 159750,62 = 565040,67 грн.

II. $C_{ПП}$ = 242854,63 + 53428,02+ 116522 + 162712,60= 575517,25 грн.

4.7 Вибір кращого варіанту програмного продукту техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР\,j} = К_{Кj} / С_{Фj}$$

$$К_{ТЕР1} = 5,64 / 565040,67 = 9,99 \cdot 10^{-6},$$

$$К_{ТЕР2} = 5,28 / 575517,25 = 9,17 \cdot 10^{-6}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1}= 9,99 \cdot 10^{-6}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту.

Цей варіант реалізації програмного продукту має такі параметри:

- вибір мови програмування Python;
- вибір середовища програмування Google Colaboratory;
-.вибір допоміжного пакету для машинного навчання py-torch.

Даний варіант виконання програмного комплексу дає користувачу зрозумілий інтерфейс, швидку реалізацію програми та доступний функціонал для роботи.

Висновки до розділу 4

В даному розділі був проведений повний аналіз функцій та вартості програмного продукту. Також була здійснена оцінка основних функцій програмного продукту.

В результаті функціонально-вартісного аналізу розроблюваного програмного комплексу, було визначено та проведено оцінку основних функцій програмного продукту, а також знайдено параметри, що характеризують його.

На основі проведеного аналізу було обрано варіант реалізації програмного продукту.

ВИСНОВКИ

У цій дипломній роботі було досліджено та розроблено систему моделювання поведінки фондового ринку з використанням генеративно-змагальних нейронних мереж. Основною метою було підвищення якості розуміння та прогнозування фондових ринків, що є складними та нелінійними системами, з метою допомоги у розробці ефективних торгових стратегій.

У процесі роботи було спроектовано та розроблено систему, яка здатна аналізувати та прогнозувати нелінійні нестаціонарні процеси на фондовому ринку. Була розглянута та реалізована модель генеративно-змагальної нейронної мережі, а також проаналізовано вибір параметрів для її ефективного функціонування.

Отримані результати підтверджують доцільність використання генеративно-змагальних нейронних мереж у сфері фінансового аналізу. Система, розроблена в рамках цієї роботи, демонструє здатність прогнозувати поведінку цін акцій на фондовому ринку з високою точністю.

Однією з основних переваг такого підходу є можливість використання великого обсягу даних, що надаються сервісами, такими як Yahoo Finance, для побудови надійних прогнозів. Завдяки цьому, отримані результати можуть бути використані не тільки для аналізу фондового ринку, але й для аналізу фінансових ринків загалом.

У майбутньому, можливим напрямком подальших досліджень є розширення моделі та використання додаткових джерел даних, а також вдосконалення алгоритмів генеративно-змагальних нейронних мереж для ще більш точного прогнозування ринкових трендів та використання їх у практичній торгівлі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. https://www.sciencedirect.com/science/article/pii/S1877050919302789.

2. https://cs230.stanford.edu/projects_fall_2019/reports/26259829.pdf.

3. Robert H. Sgumway, David S. Stoffer. Time Series Analysis and Its Applications. Springer. C. 2020. 316–339.

4. Brownlee, Jason. "A Gentle Introduction to BigGAN the Big Generative Adversarial Network". Machine Learning Mastery. 2022.

5. Tero, Karras; Miika, Aittala; Janne, Hellsten; Samuli, Laine; Jaakko, Lehtinen; Timo, Aila. "Training Generative Adversarial Networks with Limited Data". Advances in Neural Information Processing Systems. 2020

6. Rolling Window Regression: a Simple Approach for Time Series Next value Predictoins URL: https://medium.com/making-sense-of-data/time-series-next-valueprediction-using-regression-over-a-rolling-window-228f0acae363. 2021.

7. https://developer.ibm.com/articles/generative-adversarial-networks-explained/.

8. "countries with largest stock markets". *Statista*. 2020.

9. Kvilhaug, Suzanne. Scott, Gordon "What Are Shares? Meaning and How They Compare to Stocks". Investopedia. 2022.

10. Games of GANs: Game-Theoretical Models for Generative Adversarial Networks, Monireh Mohebbi Moghadam, Bahar Boroomand, Mohammad Jalali, Arman Zareian, Alireza Daeijavad, Mohammad Hossein

11. https://guides.loc.gov/wall-street-history/exchanges

12. Ripley, Brian D. Pattern Recognition and Neural Networks. Cambridge University Press. 2007.

13. Wilson, Halsey. Artificial intelligence. Grey House Publishing. 2018.

14. https://developer.ibm.com/articles/generative-adversarial-networks-explained/

15.https://www.investopedia.com/terms/f/fundamentalanalysis.asp#:~:text=Fundamental%20analysis%20is%20a%20method,a%20buy%20recommendation%20is%20given.
16. “A Brief History of Generative Adversarial Networks”. Parth Sharma.
17.https://neptune.ai/blog/predicting-stock-prices-using-machine-learning
18.https://machinelearningmastery.com/how-to-evaluate-generative-adversarial-networks/.

ДОДАТОК А ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
from torch.utils.data import DataLoader, TensorDataset
import torch.nn.functional as F
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
import math
data = pd.read_csv('SHEL.csv', index_col = 'Date')
data = data[['Open','High', 'Low', 'Volume', 'Close']]

data.head(10)

df = data
df['Close'] = df['Close'].astype(float)
plt.figure(figsize=(16,7))
plt.plot(df.index, df['Close'].values, label = 'Ціна акції', color =
'green')
plt.xticks(np.arange(100,df.shape[0],200))
plt.xlabel('Дата')
plt.ylabel('Ціна (у доларах США)')
plt.legend()
plt.show()

data['y'] = data['Close']

x = data.iloc[:, :5].values
y = data.iloc[:, 5].values

split = int(data.shape[0]* 0.8)
train_x, test_x = x[: split, :], x[split:, :]
train_y, test_y = y[: split, ], y[split: , ]

print(f'trainX: {train_x.shape} trainY: {train_y.shape}')
print(f'testX: {test_x.shape} testY: {test_y.shape}')

x_scaler = MinMaxScaler(feature_range = (0, 1))
y_scaler = MinMaxScaler(feature_range = (0, 1))

train_x = x_scaler.fit_transform(train_x)
test_x = x_scaler.transform(test_x)

train_y = y_scaler.fit_transform(train_y.reshape(-1, 1))
test_y = y_scaler.transform(test_y.reshape(-1, 1))

class VAE(nn.Module):
    def __init__(self, config, latent_dim):
        super().__init__()
```

```
        modules = []
        for i in range(1, len(config)):
            modules.append(
                nn.Sequential(
                    nn.Linear(config[i - 1], config[i]),
                    nn.ReLU()
                )
            )

        self.encoder = nn.Sequential(*modules)
        self.fc_mu = nn.Linear(config[-1], latent_dim)
        self.fc_var = nn.Linear(config[-1], latent_dim)

        modules = []
        self.decoder_input = nn.Linear(latent_dim, config[-1])

        for i in range(len(config) - 1, 1, -1):
            modules.append(
                nn.Sequential(
                    nn.Linear(config[i], config[i - 1]),
                    nn.ReLU()
                )
            )
        modules.append(
            nn.Sequential(
                nn.Linear(config[1], config[0]),
                nn.Sigmoid()
            )
        )

        self.decoder = nn.Sequential(*modules)

    def encode(self, x):
        result = self.encoder(x)
        mu = self.fc_mu(result)
        logVar = self.fc_var(result)
        return mu, logVar

    def decode(self, x):
        result = self.decoder(x)
        return result

    def reparameterize(self, mu, logVar):
        std = torch.exp(0.5* logVar)
        eps = torch.randn_like(std)
        return eps * std + mu

    def forward(self, x):
        mu, logVar = self.encode(x)
        z = self.reparameterize(mu, logVar)
        output = self.decode(z)
        return output, z, mu, logVar
```

```
train_loader =
DataLoader(TensorDataset(torch.from_numpy(train_x).float()),
batch_size = 128, shuffle = False)
model = VAE([5, 400, 400, 400, 10], 10)

use_cuda = 1
device = torch.device("cuda" if (torch.cuda.is_available() & use_cuda)
else "cpu")
num_epochs = 300
learning_rate = 0.00003
model = model.to(device)
optimizer = torch.optim.Adam(model.parameters(), lr = learning_rate)

hist = np.zeros(num_epochs)
for epoch in range(num_epochs):
    total_loss = 0
    loss_ = []
    for (x, ) in train_loader:
        x = x.to(device)
        output, z, mu, logVar = model(x)
        kl_divergence = 0.5* torch.sum(-1 - logVar + mu.pow(2) +
logVar.exp())
        loss = F.binary_cross_entropy(output, x) + kl_divergence
        loss.backward()
        optimizer.step()
        loss_.append(loss.item())
    hist[epoch] = sum(loss_)
    print('[{}/{}] Loss:'.format(epoch+1, num_epochs), sum(loss_))

plt.figure(figsize=(12, 6))
plt.plot(hist)

model.eval()
_, VAE_train_x, train_x_mu, train_x_var =
model(torch.from_numpy(train_x).float().to(device))
_, VAE_test_x, test_x_mu, test_x_var =
model(torch.from_numpy(test_x).float().to(device))
def sliding_window(x, y, window):
    x_ = []
    y_ = []
    y_gan = []
    for i in range(window, x.shape[0]):
        tmp_x = x[i - window: i, :]
        tmp_y = y[i]
        tmp_y_gan = y[i - window: i + 1]
        x_.append(tmp_x)
        y_.append(tmp_y)
        y_gan.append(tmp_y_gan)
    x_ = torch.from_numpy(np.array(x_)).float()
    y_ = torch.from_numpy(np.array(y_)).float()
    y_gan = torch.from_numpy(np.array(y_gan)).float()
    return x_, y_, y_gan

train_x = np.concatenate((train_x,
VAE_train_x.cpu().detach().numpy()), axis = 1)
```

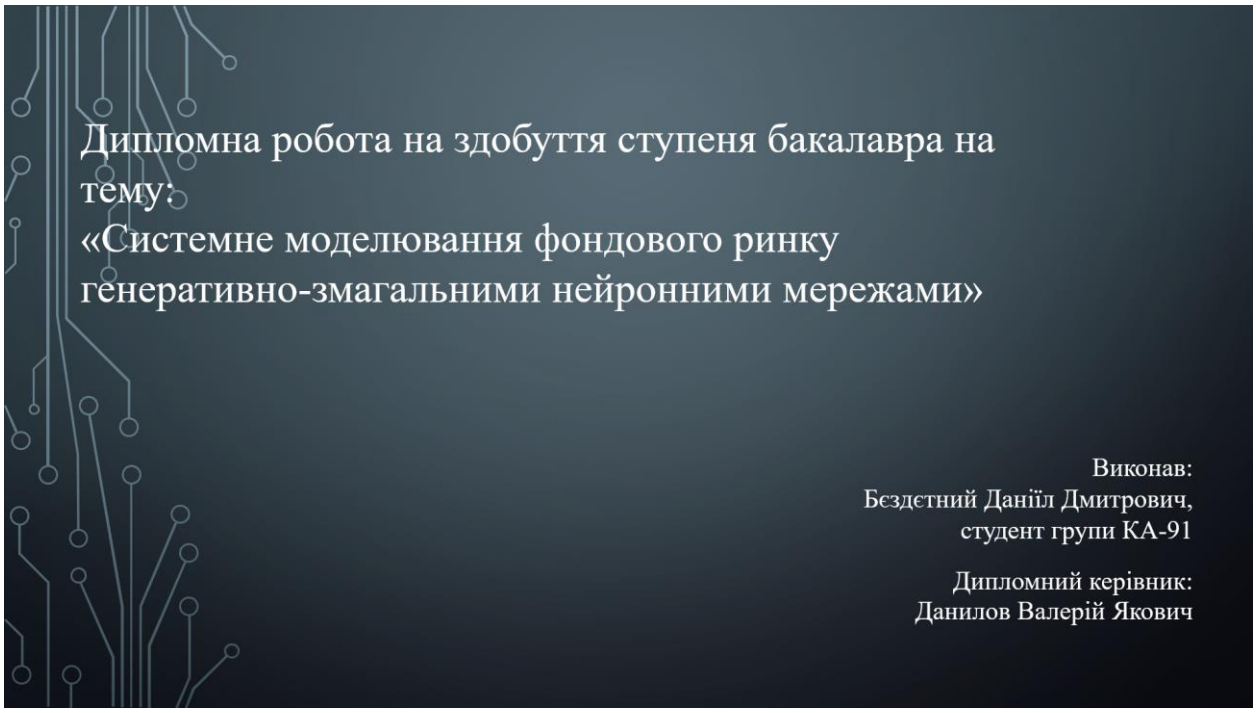
```
test_x = np.concatenate((test_x, VAE_test_x.cpu().detach().numpy()),
axis = 1)
train_x_slide, train_y_slide, train_y_gan = sliding_window(train_x,
train_y, 3)
test_x_slide, test_y_slide, test_y_gan = sliding_window(test_x,
test_y, 3)
print(f'train_x: {train_x_slide.shape} train_y: {train_y_slide.shape}
train_y_gan: {train_y_gan.shape}')
print(f'test_x: {test_x_slide.shape} test_y: {test_y_slide.shape}
test_y_gan: {test_y_gan.shape}')

class Generator(nn.Module):
    def __init__(self, input_size):
        super().__init__()
        self.gru_1 = nn.GRU(input_size, 1024, batch_first = True)
        self.gru_2 = nn.GRU(1024, 512, batch_first = True)
        self.gru_3 = nn.GRU(512, 256, batch_first = True)
        self.linear_1 = nn.Linear(256, 128)
        self.linear_2 = nn.Linear(128, 64)
        self.linear_3 = nn.Linear(64, 1)
        self.dropout = nn.Dropout(0.2)

    def forward(self, x):
        use_cuda = 1
        device = torch.device("cuda" if (torch.cuda.is_available() &
use_cuda) else "cpu")
        h0 = torch.zeros(1, x.size(0), 1024).to(device)
        out_1, _ = self.gru_1(x, h0)
        out_1 = self.dropout(out_1)
        h1 = torch.zeros(1, x.size(0), 512).to(device)
        out_2, _ = self.gru_2(out_1, h1)
        out_2 = self.dropout(out_2)
        h2 = torch.zeros(1, x.size(0), 256).to(device)
        out_3, _ = self.gru_3(out_2, h2)
        out_3 = self.dropout(out_3)
        out_4 = self.linear_1(out_3[:, -1, :])
        out_5 = self.linear_2(out_4)
        out = self.linear_3(out_5)
        return out

class Discriminator(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv1d(4, 32, kernel_size = 3, stride = 1,
padding = 'same')
        self.conv2 = nn.Conv1d(32, 64, kernel_size = 3, stride = 1,
padding = 'same')
        self.conv3 = nn.Conv1d(64, 128, kernel_size = 3, stride = 1,
padding = 'same')
        self.linear1 = nn.Linear(128, 220)
        self.batch1 = nn.BatchNorm1d(220)
        self.linear2 = nn.Linear(220, 220)
        self.batch2 = nn.BatchNorm1d(220)
        self.linear3 = nn.Linear(220, 1)
        self.leaky = nn.LeakyReLU(0.01)
        self.relu = nn.ReLU()
```

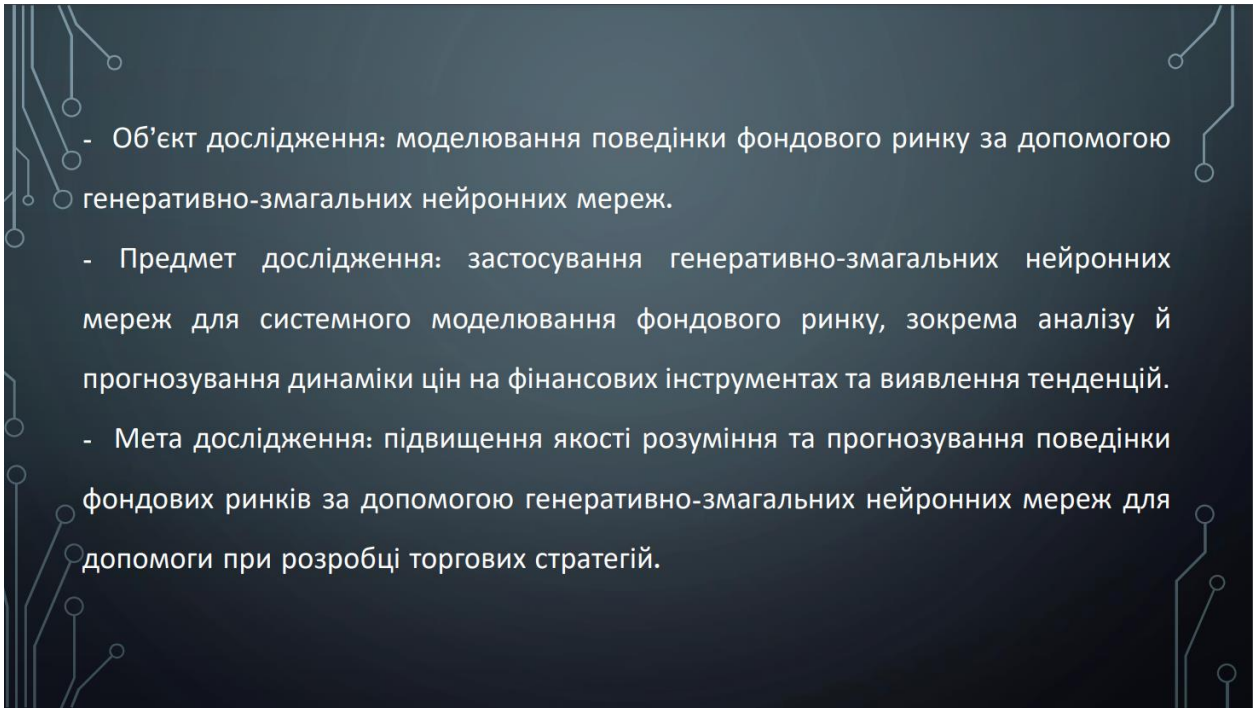
```
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        conv1 = self.conv1(x)
        conv1 = self.leaky(conv1)
        conv2 = self.conv2(conv1)
        conv2 = self.leaky(conv2)
        conv3 = self.conv3(conv2)
        conv3 = self.leaky(conv3)
        flatten_x = conv3.reshape(conv3.shape[0], conv3.shape[1])
        out_1 = self.linear1(flatten_x)
        out_1 = self.leaky(out_1)
        out_2 = self.linear2(out_1)
        out_2 = self.relu(out_2)
        out_3 = self.linear3(out_2)
        out = self.sigmoid(out_3)
        return out

use_cuda = 1
device = torch.device("cuda" if (torch.cuda.is_available() & use_cuda)
else "cpu")

batch_size = 128
learning_rate = 0.00010
num_epochs = 50

trainDataloader = DataLoader(TensorDataset(train_x_slide,
train_y_gan), batch_size = batch_size, shuffle = False)

modelG = Generator(15).to(device)
modelD = Discriminator().to(device)

criterion = nn.BCELoss()
optimizerG = torch.optim.Adam(modelG.parameters(), lr = learning_rate,
betas = (0.0, 0.9))
optimizerD = torch.optim.Adam(modelD.parameters(), lr = learning_rate,
betas = (0.0, 0.9))

histG = np.zeros(num_epochs)
histD = np.zeros(num_epochs)
count = 0
for epoch in range(num_epochs):
    loss_G = []
    loss_D = []
    for (x, y) in trainDataloader:
        x = x.to(device)
        y = y.to(device)

        fake_data = modelG(x)
        fake_data = torch.cat([y[:, :3, :], fake_data.reshape(-1, 1,
1)], axis = 1)

        dis_real_output = modelD(y)
        real_labels = torch.ones_like(dis_real_output).to(device)
        lossD_real = criterion(dis_real_output, real_labels)
```

```
        dis_fake_output = modelD(fake_data)
        fake_labels = torch.zeros_like(real_labels).to(device)
        lossD_fake = criterion(dis_fake_output, fake_labels)

        lossD = (lossD_real + lossD_fake)

        modelD.zero_grad()
        lossD.backward(retain_graph=True)
        optimizerD.step()
        loss_D.append(lossD.item())

        output_fake = modelD(fake_data)
        lossG = criterion(output_fake, real_labels)
        modelG.zero_grad()
        lossG.backward()
        optimizerG.step()
        loss_G.append(lossG.item())
    histG[epoch] = sum(loss_G)
    histD[epoch] = sum(loss_D)
    print(f'[{epoch+1}/{num_epochs}] LossD: {sum(loss_D)}
LossG:{sum(loss_G)}')

plt.figure(figsize = (12, 6))
plt.plot(histG, color = 'green', label = 'Generator Loss')
plt.plot(histD, color = 'black', label = 'Discriminator Loss')
plt.title('Loss-функція')
plt.xlabel('Кількість епох')
plt.legend(loc = 'upper right')

modelG.eval()
pred_y_train = modelG(train_x_slide.to(device))
pred_y_test = modelG(test_x_slide.to(device))

y_train_true = y_scaler.inverse_transform(train_y_slide)
y_train_pred =
y_scaler.inverse_transform(pred_y_train.cpu().detach().numpy())

y_test_true = y_scaler.inverse_transform(test_y_slide)
y_test_pred =
y_scaler.inverse_transform(pred_y_test.cpu().detach().numpy())

plt.figure(figsize=(12, 8))
plt.plot(y_train_true, color = 'black', label = 'Реальне значення
ціни')
plt.plot(y_train_pred, color = 'green', label = 'Прогнозоване значення
ціни')
plt.title('Результат тренування мережі')
plt.ylabel('Курс акції')
plt.xlabel('Час(у тижнях)')
plt.legend(loc = 'upper right')

MSE = mean_squared_error(y_train_true, y_train_pred)
MAE = mean_absolute_error(y_train_true, y_train_pred)
RMSE = math.sqrt(MSE)
print(f'Значення похибок:')
print(f'RMSE:{RMSE}')
```

```
print(f'MAE:{MAE}')

plt.figure(figsize=(12, 8))
plt.plot(y_test_true, color = 'black', label = 'Реальне значення
ціни')
plt.plot(y_test_pred, color = 'green', label = 'Прогнозоване значення
ціни')
plt.title('Результат прогнозування мережею')
plt.ylabel('Курс акції')
plt.xlabel('Час (у тижнях)')
plt.legend(loc = 'upper right')

MSE = mean_squared_error(y_test_true, y_test_pred)
MAE = mean_absolute_error(y_test_true, y_test_pred)
RMSE = math.sqrt(MSE)
print(f'Значення похибок:')
print(f'RMSE:{RMSE}')
print(f'MAE:{MAE}')
```

ДОДАТОК Б ПРЕЗЕНТАЦІЯ



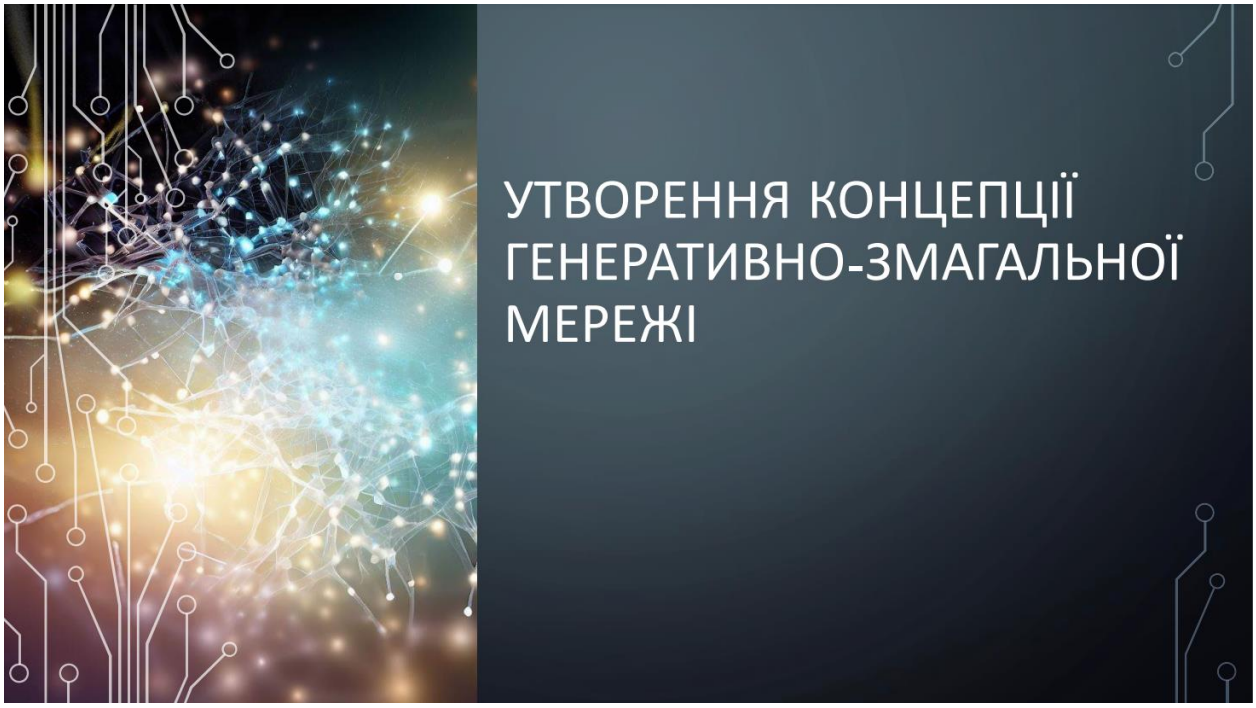


ПОСТАНОВКА ЗАДАЧІ

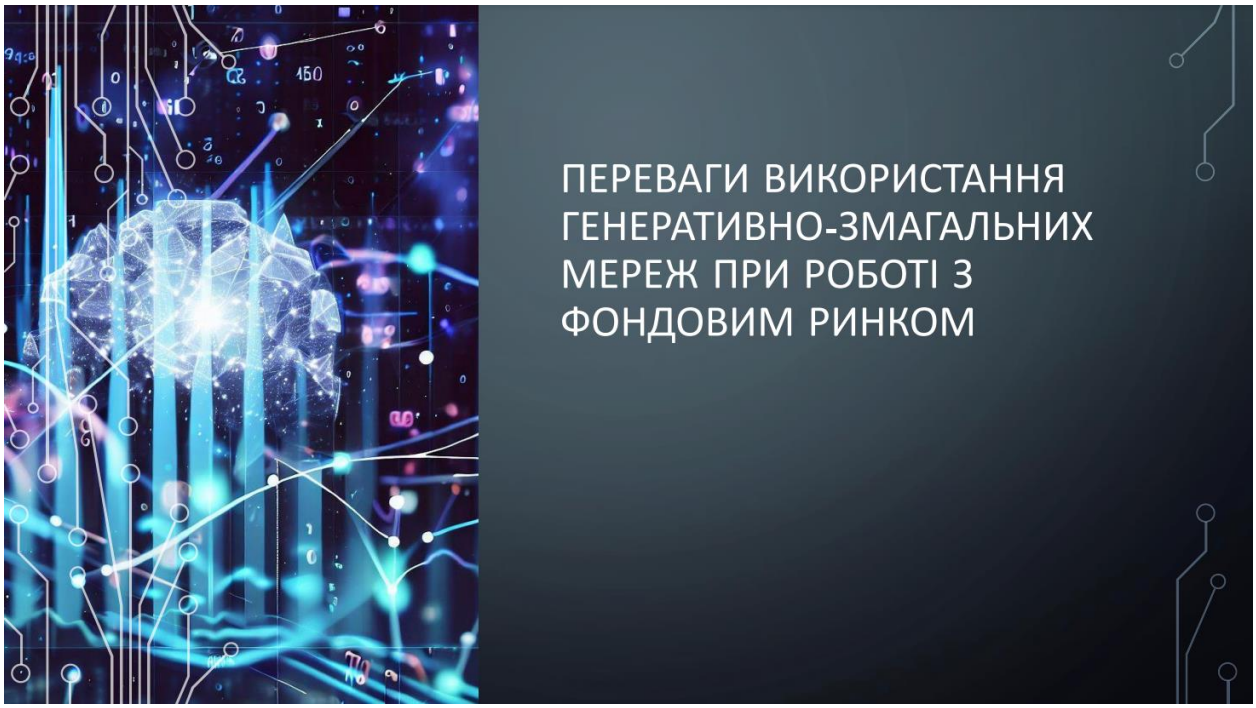
Для досягнення мети нам потрібно розв'язати наступні задачі: дослідити різні фактори та їх вплив на процеси фондового ринку, розглянути особливості генеративно-змагальних мереж, виконати пошук та попередню обробку даних про акції певної компанії, побудувати модель та реалізувати прогнозування вартості акцій за допомогою генеративно-змагальної нейронної мережі.

АКТУАЛЬНІСТЬ РОБОТИ

Системне моделювання фондового ринку генеративно-змагальними нейронними мережами є важливим інструментом для дослідників, аналітиків та інвесторів, які прагнуть отримати певні дані щодо поведінки фондового ринку та оцінити свої інвесторські стратегії. Через стрімкий розвиток міжнародної торгівлі та фінансів у сучасних реаліях прогнозування ринкових процесів є дуже актуальною задачею, яка потребує вирішення з використанням і вдосконаленням моделей прогнозування.


УТВОРЕННЯ КОНЦЕПЦІЇ ГЕНЕРАТИВНО-ЗМАГАЛЬНОЇ МЕРЕЖІ


СТРУКТУРА ГЕНЕРАТИВНО-ЗМАГАЛЬНОЇ МЕРЕЖІ
Реальні дані
Дискримінатор
Чи є висновок правильним
Згенеровані дані
Випадковий розподіл
Генератор
Коригування

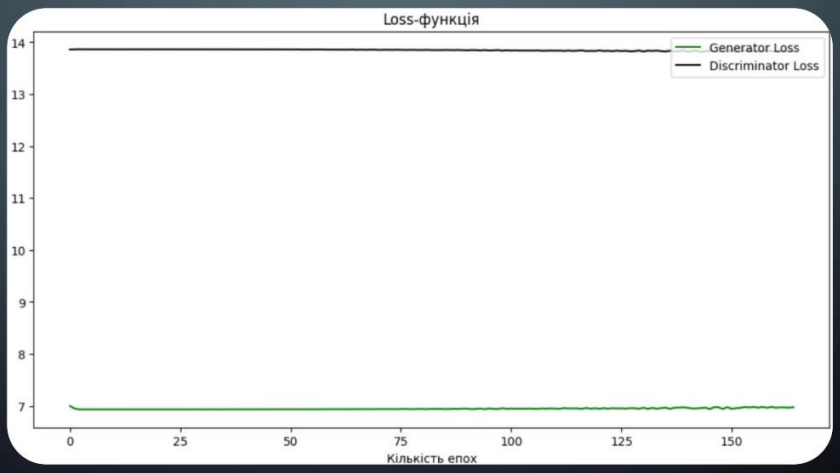

ПЕРЕВАГИ ВИКОРИСТАННЯ ГЕНЕРАТИВНО-ЗМАГАЛЬНИХ МЕРЕЖ ПРИ РОБОТІ З ФОНДОВИМ РИНКОМ


ВИБІР БАЗИ ДАНИХ
Проведення досліджень було вирішено виконати на базі даних про акції компанії Shell. Задля отримання достовірної інформації про вартість акцій компанії було залучено сервіс Yahoo Finance.
Shell
yahoo! finance

ОТРИМАНІ РЕЗУЛЬТАТИ

Результатами дослідження є прогноз генеративно-змагальною мережею на прикладі курсу акцій транснаціональної компанії «Shell».

ЗНАЧЕННЯ ФУНКЦІЙ ВТРАТИ



АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Отримавши результати після навчання нейронної мережі, ми аналізуємо точність нашого моделювання за допомогою різних метрик для виміру відхилень. Наведемо результат виміру точності прогнозування для одного з експериментів:

RMSE = 3.2259226

MAE = 2.4273724

ВИСНОВКИ

- У ході виконання роботи було розроблено генеративно-змагальну нейронну мережу для прогнозування поведінки цін на акції.
- Після проведення досліджень було виявлено оптимальні значення параметрів для побудованої моделі.
- Робота була виконана для покращення допоміжних засобів при аналізі поведінки фондових ринків.

ПОДАЛЬШІ ДОСЛІДЖЕННЯ

В подальшому для поліпшення результатів дослідження можна збільшити кількість наборів вхідних даних, узявши декілька компаній з одної галузі, розглянути різні архітектурні покращення моделі та виконати порівняльну характеристику їх роботи для прогнозування поведінки курсу цінних паперів компаній різного типу.

ДЯКУЮ ЗА УВАГУ!