**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ**
**ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ**
**КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ**

На правах рукопису
УДК 004.852

До захисту допущено
Завідувач кафедри ШІ
_____О.І. Чумаченко
«___»_____2023 р.

# Магістерська дисертація

**на здобуття ступеня магістра**

**зі спеціальності 122 «Комп'ютерні науки»**

**на тему: «Розробка моделей штучного інтелекту для аналізу та формування інвестиційного портфелю»**

Виконав:
студент ІІ курсу, групи КІ-11мп
Шевчук Олексій Сергійович _____

Керівник:
професор кафедри математичних методів системного аналізу,
д.т.н., проф. Кузнєцова Н. В. _____

Рецензент:
доцент кафедри системного проектування
КПІ ім. Ігоря Сікорського, к.т.н., Безносик О. Ю. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ
2022

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ**
**ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ**
**КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ**

Рівень вищої освіти – другий (магістерський)

Спеціальність – 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ О.І. Чумаченко

«___» _____ 2023 р.

**ЗАВДАННЯ**
**на магістерську дисертацію студенту**
**Шевчуку Олексію Сергійовичу**

1. Тема дисертації: «Розробка моделей штучного інтелекту для аналізу та формування інвестиційного портфелю», науковий керівник роботи Кузнєцова Наталія Володимирівна, професор кафедри ММСА, д.т.н., проф. затверджено наказом по університету від «03» листопада 2023 р. № 4046-с.

2. Термін подання студентом дисертації 15.12.2023

3. Об'єкт дослідження: інвестиційні процеси та портфелі, способи їх аналізу та формування.

4. Предмет дослідження: математичні методи та методи на основі штучного інтелекту для аналізу інвестиційних процесів та формування інвестиційного портфелю.

5. Перелік завдань, які потрібно зробити:

1) здійснити огляд технічної літератури за темою роботи;

2) дослідити актуальність обраної теми;

3) ознайомитись із існуючими методами та моделями формування інвестиційного портфелю;

4) здійснити порівняльний аналіз наявних методів, виявити їх переваги та недоліки;

5) розробити та реалізувати метод формування інвестиційного портфелю на основі штучного інтелекту;

6) провести експеримент, що засвідчує працеспроможність запропонованої моделі, виконати аналіз результатів, порівняти модель із класичними методами формування інвестиційного портфелю;

7) провести аналіз ринкових можливотсей запуску стартап проекту;

8) розробити концептуальні висновки;

9) підготувати ілюстративний матеріал;

10) оформити пояснювальну записку.

6. Перелік ілюстративного матеріалу.

7. Дата видачі завдання: 03 вересня 2023 р.

Календарний план

| № з/п | Назва етапів виконання магістерської дисертації | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Вивчення літератури за темою роботи. | 03.09.2023 – 13.09.2023 | Виконано |
| 2. | Підготовка першого розділу. | 14.09.2023 – 18.09.2023 | Виконано |
| 3. | Підготовка другого розділу. | 19.09.2023 – 26.09.2023 | Виконано |
| 4. | Розробка програмного продукту. | 27.09.2023 – 26.10.2023 | Виконано |

| | | | |
|---|---|---|---|
| 5. | Підготовка третього розділу | 27.10.2023 – 09.11.2023 | Виконано |
| 6. | Підготовка частини стартап-проєкту | 10.11.2023 – 16.11.2023 | Виконано |
| 9. | Концептуальні висновки. Перспективи розвитку отриманих рішень | 17.11.2023 – 21.11.2023 | Виконано |
| 10. | Оформлення пояснювальної записки | 22.11.2023 – 26.11.2023 | Виконано |

Студент  Олексій ШЕВЧУК

Керівник  Наталія КУЗНЄЦОВА

# РЕФЕРАТ

Дипломна робота: 134 с., 10 рис., 12 табл., 1 дод., 11 джерел.

ІНВЕСТИЦІЙНИЙ АНАЛІЗ, ФОРМУВАННЯ ІНВЕСТИЦІЙНОГО ПОРТФЕЛЮ, ШТУЧНИЙ ІНТЕЛЕКТ

Об’єкт дослідження – інвестиційні процеси та портфелі, способи їх аналізу та формування.

Предмет дослідження – математичні методи та методи на основі штучного інтелекту для аналізу інвестиційних процесів та формування інвестиційного портфелю.

Мета роботи – дослідити існуючі методи формування інвестиційного портфелю, розробити метод формування інвестиційного портфелю на основі штучного інтелекту, порівняти його із класичними методами.

Методи дослідження – теорія прийняття рішень при нечіткому відношенні переваги на множині альтернатив, модель Марковіца, мережі Байєса та методи на основі штучного інтелекту.

Актуальність – розробка методу формування інвестиційного портфелю на основі штучного інтелекту з метою підвищення прибутковості портфелю та зниження його ризикованості.

Результати роботи – було створено і протестовано метод формування інвестиційного портфелю на основі штучного інтелекту, проведено порівняння розробленого методу із класичними.

Шляхи подальшого розвитку предмету дослідження – використання нечітких нейронних мереж для розв’язку задачі формування інвестиційного портфелю, використанні інших метрик та різних комбінацій даних, для визначення справжньої ціни акцій.

# ABSTRACT

The theme: ‘Development of artificial intelligence models for investment portfolio analysis and creation’

Diploma work: 134 p., 10 fig., 12 tabl., 1 appendixes, 11 references.

INVESTMENT ANALYSIS, INVESTMENT PORTFOLIO FORMATING, ARTIFICIAL INTELLIGENCE

The object of research – investment processes and portfolio, approaches to analyze and to format them.

The subject of research – mathematics methods and artificial intelligence methods for investment process analyzation and investment portfolio creation.

The purpose of the work is explore the existing ways of investment portfolio formation, to develop alternative artificial intelligence based method for investment portfolio creation, explore an efficiency of developed approach in comparison with classic mathematical methods.

Research methods – decision-making theory with unclear advantages on many alternatives, Markovits` model, Bayesian networks and artificial network models, particularly neural networks.

Relevance – the task of developing artificial intelligence based method for creation of investment portfolio with the aim of increasing investment profit and decrease risks.

Results of work – an alternative AI based method to investment portfolio creation was created and tested, the method was compared with the classical mathematics methods.

Ways of further develop the subject of research – using unclear neural networks for investing portfolio creation task solving, using other metrics and data combinations for fair stock price calculation.

ЗМІСТ

# ВСТУП

У наш час, фінансово грамотні люди розуміють важливість правильного фінансового менеджменту, ціллю якого приведення активів у рух з метою їх примноження. Більш того, у сучасному світі будь-яка держава залежить від процесів інвестування.

Далеко не всі розуміють, що у сучасному світі інвестуванням займається кожен із нас. Навіть отримання освіти можна назвати певною інвестицією, адже це є внесок у майбутнє, і саме якісна освіта допоможе тобі знайти хорошу роботу із високою заробітною платнею.

Нажаль, у кожного із нас є лише обмежена кількість ресурсів, і це не стосується лише фінансів. Людина може інвестувати як гроші, так і свою роботу та час. Через все це дуже важливо вміти ефективно обирати об’єкт інвестування, щоб у майбутньому не пошкодувати за свій вибір.

Саме пошук найбільш оптимальних об’єктів та способів інвестування стимулював економістів та математиків розробляти та розвивати підходи та методи ефективного інвестування, аналізу різноманітних ринків, таких як фінансовий, земельний, ринок нерухомості та інші.

Об’єктом даного дослідження інвестиційні процеси та портфелі, а також способи їх аналізу та формування.

Предметом дослідження є математичні методи та методи на онові штучного інтелекту для аналізу інвестиційних процесів та формування інвестиційного портфелю.

Метою дослідження є дослідження існуючих методів формування інвестиційного портфелю та розробка методу формування інвестиційного портфелю на основі штучного інтелекту, а також порівняння його ефективності із класичними методами та розробка програмного продукту, що буде формувати інвестиційні портфелі.

Актуальність даного дослідження полягає у тому, що вибір більш ефективного способу інвестування, в залежності від наявних ресурсів, максимізує прибутки інвестора.

Структурно, робота складається з чотирьох розділів.

У першому розділі буде детально розглянуто предмет дослідження, теоретичне підґрунтя інвестиційних процесів та формування інвестиційних портфелів, проаналізовано існуючі методи формування інвестиційних портфелів, а також проведено їх порівняльну характеристику.

У другому розділі буде розглянута теоретичні основи класичних методів та розробленого методу на основі штучного інтелекту. Також будуть описані математичні основи та принципи практичної реалізації даних методів та проведено їх порівняльний аналіз.

У третьому розділі буде описано принцип роботи розробленого методу формування інвестиційних портфелів на основі штучного інтелекту. Також буде описано роботу модулів програмного продукту та проведено порівняльний аналіз класичного та альтернативного підходів.

У четвертому розділі буде проведено функціонально-вартісний аналіз роботи.

# ПОСТАНОВКА ЗАДАЧІ

1. Проаналізувати ІТ-ринок існуючих програмних засобів на предмет аналогічних чи схожих програмних продуктів, проаналізувати потенціал розробки продукту та нового методу формування інвестиційних портфелів.

2. На основі теорії формування і дослідження інвестиційних процесів, провести аналіз існуючих методів формування інвестиційного портфелю, визначивши основні їх переваги та недоліки.

3. Дослідити вимоги та обмеження різних методів формування інвестиційних портфелів. Розглянути способи оцінювання якості методів та моделей.

4. Розробити програмний продукт, що формує інвестиційний портфель класичними методами та розробленим методом на основі штучного інтелекту.

5. Провести порівняльний аналіз класичного та нового методів. Зробити висновки щодо ефективності розробленого методу на основі штучного інтелекту.

6. Зробити висновки щодо ефективності розробленого програмного продукту, способи його покращення та перспектив його використання.

# РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Актуальність задачі формування інвестиційного портфелю

Історія інвестицій бере свій початок в часи стародавнього Вавилону. Саме тими часами датуються перші писемні згадки, про інвестування та грамотного управління своїми фінансами. Варто зазначити, що тоді ці знання могли отримати лише особи знатного роду.

Також, велику увагу фінансам приділяли і в Античній Греції, зокрема в Афінах. Кожен громадянин міг отримати ділянку землі та фінансову підтримку від держави. При цьому, активний розвиток отримало кредитування. Зокрема, така послуга як іпотека була винайдена саме в Античній Греції.

Саме слово «інвестиція» з’явилося досить давно і має латинське походження. При цьому його початковий переклад був «одягаю». Лише з часом латинське слово «invest» набув значення «вкладати що-небудь».

У сучасному ж світі, зокрема в Україні, популярна думка, що інвестиційною діяльністю можуть займатися лише відповідні кредитні організації, державні органи і великі компанії. А серед фізичних осіб, отримувати прибутки від вкладання від вкладання коштів можуть лише заможні люди. Насправді це не так. Маючи відповідний хист та знання можна досягти значних висот за рахунок розумного фінансового менеджменту.

Прикладами успішного інвестування можуть бути як гравці фінансового ринку, такі як Джордж Сорас, Стенлі Дракенміллер та інші, так й інвестори у більш широкому сенсі цього слова. Так, іншими прикладом інвесторів може бути інвестор Бенджамін Грем, якого називають батьком вартісного інвестування. Він починав свій шлях з посади посильного у брокерській фірмі з заробітною платою 12 доларів. Крім прямих обов’язків, він повинен також був визначати курси облігацій та акцій на котировочній дошці. Через декілька років роботи, його заробітна платня підвищилася до 600 тисяч доларів.

Іншим прикладом успіху є відомий Воррен Баффет. Свій початковий капітал у розмірі 10 тисяч доларів він заробив, встановивши автомати для пінболу в перукарнях. Продовжуючи успішне ведення бізнесу, інвестування та гру на фінансових ринках, Баффет заробив свій перший мільйон до 32 років, в 52 роки його статки становили 376 мільйонів доларів, в 59 – 3.8 мільярди, а тепер в 91 рік – 96 мільярдів доларів США.

Але поряд із успіхом завжди ходить і невдача. Окрім провальних торгів на фінансовому ринку, прикладами яких є Джером Кервіл, що торгував європейськими ф'ючерсами, Браєа Хантер, що провалився при торгівлі газом та інших є і інші інвестори, що обрали неправильні об'єкти для своїх внесків. Так, вокаліст групи «U2» Пол Хьюсон, більш відомий як Боно, отримав «титул» «Найгірший інвестор США», за свої провальні інвестиції в компанії у 2010 році. Боно разом із Джоном Рікітелло, колишнім операційним директором Electronic Arts, створили інвестиційний фонд у 2004 році, який після низки провальних проектів підряд був закритим у 2015 році. Найбільшим провалом була інвестиція у компанію Palm Inc. що займалася розробкою мобільних телефонів і ціллю якої було «вбити Apple». Але сталося не так як гадалося і компанія опинилася на грані банкротства та була продана. Іншим великим провалом Боно була інвестиція в Forbes Inc., що займалася видачою друкованих видань, але які втрачали популярність через те, що люди віддали перевагу інтернету. Через ці та інші менші провали Пол Хьюсон втратив більше 900 мільйонів доларів.

Як видно з наведених прикладів, інвестування це завжди ризик, який може як забезпечити безбідну старість, так і залишити без копійки у кишені. Є випадки чистої удачі чи навпаки її відсутності, випадки проаналізувати які майже неможливо. Але цих випадків переважна меншість. Зазвичай, перемоги та невдачі можна пояснити аналітично і через це дуже важливо розуміти теорію інвестування.

## 1.2 Теорія інвестування: основні поняття та засади

Вчені-економісти не можуть дати однозначного визначення терміну «інвестиції». У загальному інвестиції – це майно, що вкладаються в економіку, економічні об’єкти та проекти, з метою забезпечення виробництва економічними ресурсами у майбутньому. За іншим означенням, інвестиції – це гроші, цінні папери, інше майно, включно із правами на власність та іншими правами, що мають грошову оцінку, які вкладаються в об’єкти підприємницької діяльності і (або) інші види діяльності з метою отримання прибутку і (чи) отримання іншого корисного ефекту. Але найбільш популярним означенням є наступне: інвестиції – це довгостроковий вклад капіталу в промисловість, сільське господарство, транспорт та інші різноманітні проекти з метою отримання прибутку [1].

Як видно з наведених вище означень, поняття інвестицій є дещо неоднозначним, але всі вони мають спільні ознаки.

Основними ознаками інвестицій є:

- здійснення вкладів інвесторами, що мають власні цілі, які не завжди співпадають з загальноекономічною вигодою;
- визначеність терміну, на який вкладаються кошти;
- потенційна спроможність інвестицій приносити прибуток;
- наявність ризику втратити капітал;
- використання різних інвестиційних ресурсів, що характеризуються попитом, пропозицією та ціною, у процесі інвестування;
- цілеспрямований характер вкладу капіталу в об’єкти та інструменти інвестування.

Самі ж інвестиції також бувають дуже різноманітними та мають розгорнуту класифікацію. Зокрема за формою власності виділяють:

- приватні
- державні
- змішані інвестиції.

Приватні інвестиції – це коли інвестором є фізична  чи юридична особа недержавної форми власності, наприклад громадяни чи недержавні підприємства. Державні інвестиції – це вклади, що здійснюються державними органами або підприємствами державної форми власності.

Інший спосіб класифікації інвестицій – за терміном вкладень. За цим критерієм виділяють наступні інвестиції:

- короткострокові;
- середньострокові;
- довгострокові.

До короткострокових належать інвестиції терміном до одного року, до середньострокових – від одного до трьох років, а до довгострокових – більше ніж три роки.

Також поділяють інвестиції за регіональними ознаками на такі групи:

- внутрішні або національні – ті інвестиції, що залучаються всередині країни;
- зовнішні або закордонні – інвестиції за кордоном нерезидентами в об’єкти та фінансові інструменти іншої країни;
- спільні – вкладання інвестором, що є громадянином країни, разом із іноземними інвесторами.

Дуже важливою класифікацією є поділ за об’єктом інвестування. За цим критерієм виділяють наступні види вкладень:

- реальні;
- фінансові;
- інтелектуальні.

Реальні інвестиції включають в себе вкладення коштів у матеріальні активи (матеріальний капітал: будівлі, прилади, товарно-сировинні запаси та ін.) і нематеріальні активи (патенти, ліцензії, «ноу-хау» та ін.) Під фінансовими вкладами мається на увазі інвестиції у цінні папери, такі як акції, векселі, облігації та інше, цільові банківські вклади, депозити, тощо. Зазвичай, саме купівля цінних паперів вважається формою портфельного інвестування.  Інтелектуальні інвестиції

– це вклади у творчий потенціал суспільства, об’єкти інтелектуальної власності, що слідує із авторського, винахідницького та патентного права.

Останній основний поділ інвестування – поділ за характером участі інвестора в інвестиційному процесі. За цим критерієм виділяють:

- прямі інвестиції – пряма, безпосередня участь інвестор а у вкладі капіталу в конкретний об’єкт інвестування, незалежно, чи то купівля реальних активів, чи то вклад капіталу в установчі фонди організації;
- опосередковані інвестиції – вклади капіталу інвестором в об’єкти інвестування не напряму, а з участю фінансових посередників за допомогою різноманітних фінансових інструментів.

Поняття інвестиції є атомарним у розглянутій темі і без ґрунтовного розуміння що таке інвестиція і які вони бувають не можна продовжувати подальший розгляд понять інвестиційного портфелю, способів його формування і тому подібне.

## 1.3 Поняття інвестиційного портфеля.

Розвиток інвестиційної діяльності спонукав економістів та інвесторів розробляти нові способи інвестування. Так з’явилося поняття портфельного інвестування. Суть портфельного інвестування полягає у покращенні можливостей інвестування шляхом надання сукупності об’єктів інвестування тих інвестиційних якостей, які неможливо досягти у випадку одного окремо взятого об’єкта інвестування, але цілком реальні у випадку поєднання цих об’єктів. Основним елементом портфельного інвестування є інвестиційний портфель.

Інвестиційним портфелем називається цілеспрямовано сформована сукупність вкладень в інвестиційні об’єкти, яка відповідає певній інвестиційній стратегії інвестора. Із цього означення випливає, що основною ціллю формування інвестиційного портфелю є забезпечення реалізації розробленої інвестиційної

політики шляхом підбору найбільш ефективних та надійних інвестицій. [2] Залежно від направленості обраної інвестиційної політики та особливостей ведення інвестиційної діяльності визначається система специфічних цілей, основними та найбільш поширеними з яких є:

- максимізація зростання капіталу;
- максимізація зростання прибутку;
- мінімізація інвестиційних ризиків;
- забезпечення ліквідності інвестиційного портфелю, що задовольняє вимогам.

Дані цілі формування інвестиційного портфелю, у більшій мірі, є альтернативними та взаємовиключними. Так, зростання ринкової вартості капіталу пов’язаний із певним зниженням поточного прибутку інвестиційного портфелю. Приріст вартості капіталу та підвищення прибутковості супроводжується зростанням інвестиційних ризиків. Задача досягнення певної ліквідності може бути перепоною для включення в інвестиційний портфель таких об’єктів інвестування, що забезпечує зростання капітальної вартості чи прибутковості, але, при цьому, характеризуються низькою ліквідністю. Враховуючи цю альтернативність цілей, інвестор, при формуванні інвестиційного портфелю, повинен визначити пріоритетність вище зазначених цілей чи продумати певну збалансованість окремих цілей.

Різноманіття видів об’єктів інвестування, цілей інвестування, їх пріоритетності, а також інших умов стало причиною створення величезного списку типів інвестиційних портфелів, що характеризуються певним співвідношенням прибутковості та ризикованості. Це знайшло своє відображення в різноманітних класифікаційних схемах, що наводяться в економічній літературі.

Класифікація інвестиційних портфелів по видам об’єктів інвестування перш за все пов’язана з направленістю та об’ємами інвестиційної діяльності. Так, підприємства, що займаються виробничою діяльністю, основним типом інвестиційного портфелю обирають портфель реальних інвестиційних об’єктів, а

для інституційних інвесторів більш притаманний портфель фінансових інструментів.

Це не виключає можливість формування змішаних інвестиційних портфелів, що об'єднують різні види відносно самостійних портфелів (субпортфелв), що характеризуються різними видами інвестиційних об'єктів і методами їх управління. При цьому спеціалізовані інвестиційні портфелі можуть формуватися як по об'єктам інвестицій, так і по більш приватним критеріям таких, як галузева чи регіональна приналежність, термінам інвестиції, видам ризику та ін.

Залежно від пріоритетних цілей інвестування можна виділити:

- портфель росту;
- портфель прибутку;
- консервативний портфель;
- портфель високоліквідних інвестиційних об'єктів.

Портфель росту та портфель прибутку переважно орієнтовані на вклади, що забезпечують відповідно або приріст капіталу, або отримання високих поточних прибутків. Обидва ці типи портфелів також характеризуються підвищеним рівнем ризикованості. Консервативний портфель формується за допомогою інвестиційних об'єктів, що мають нижчий рівень ризикованості, але, при цьому, і нижчим рівнем зростання капіталу та прибутковості. Портфель високоліквідних інвестиційних об'єктів припускає можливість швидкої трансформації портфелю в грошовий капітал без значної втрати вартості.

Усі ці види є основними, але між ними є низка проміжних видів портфелів. Наприклад, у рамках портфелю росту можуть бути виділені підвиди: портфель консервативного росту, портфель середнього росту, портфель агресивного росту.

Також є поділ портфелів за мірою відповідності цілям інвестора. Так, за цим критерієм виділяють:

- збалансований портфель;
- незбалансований портфель.

Збалансований портфель характеризується збалансованістю прибутків та ризиків, що відповідає критеріям, що були задані при його формуванні. В його

складі можуть бути різні об’єкти з швидко зростаючою ринковою вартістю, високо прибуткові та інші об’єкти, співвідношення яких визначається ринковою кон’юнктурою. При цьому комбінація різних інвестиційних вкладів дозволяє досягти приросту капіталу та отримання високих прибутків при зменшенні сумарних ризиків. Незбалансовані портфелі можуть розглядатися як портфелі, що не відповідають цілям, поставленим при його формуванні.

Оскільки підбір об’єктів для інвестиційного портфелю здійснюється із урахуванням вподобань інвестора, то існує зв’язок між типом портфелю та типом інвестора. Так, консервативному інвестору відповідає високонадійний, але низько прибутковий портфель, поміркованому – диверсифікований портфель, а агресивному інвестору – портфель з високими прибутками, але і високими ризиками.

## 1.4 Теорія формування інвестиційного портфеля.

Аналіз різних теорій портфельного інвестування свідчить про те, що в основі формування інвестиційного портфеля мають стояти певні принципи. В різних теоріях їх кількість і самі принципи відрізняються, але основними є:

- забезпечення реалізації інвестиційної політики, що випливає із необхідності досягнути відповідності між цілями формування інвестиційного портфеля і цілями розробленої та прийнятої інвестиційної політики;
- забезпечення відповідності між об’ємом та структурою інвестиційного портфелю та об’ємом та структурою джерел, що його формують, з ціллю підтримки ліквідності і стійкості;
- досягнення оптимального співвідношення доходності, ризику і ліквідності (враховуючи цілі формування інвестиційного портфелю) з метою збереження коштів та фінансової стабільності;

- диверсифікація інвестиційного портфелю, включення до його складу різних об’єктів інвестування, у тому числі й альтернативних інвестицій, з метою підвищення надійності, прибутковості та зниження рівня ризику;
- забезпечення керованості інвестиційним портфелем, що припускає обмеження кількості та складності інвестицій у відповідності із можливостями інвестора слідкувати за основними характеристиками інвестицій (прибутковості, ризикованості, ліквідності та ін.)

Формування інвестиційного портфелю відбувається лише після того, як буде сформовані цілі інвестиційної політики, визначені пріоритетні цілі формування інвестиційного портфеля, з урахуванням поточних умов інвестиційного клімату та кон’юнктури ринків.

Початковим етапом формування інвестиційного портфеля є комплексний аналіз можливостей інвестора та інвестиційної привабливості зовнішнього середовища. Ціллю такого аналізу є визначення прийнятного рівня ризику у світлі прибутковості та ліквідності балансу. Результаті такого аналізу є визначення основних характеристики інвестиційного портфеля таких, як міра допустимого ризику, розміри очікуваного прибутку, можливе відхилення від нього та інші. Потім відбувається оптимізація пропорцій різних видів інвестицій у рамках всього інвестиційного портфеля, з урахуванням об’єму та структури інвестиційних ресурсів.

Важливим етапом формування інвестиційного портфеля є вибір конкретних інвестиційних об’єктів, що будуть включені в оптимальний інвестиційний портфель. Для визначення таких об’єктів використовують основі оцінки їх інвестиційних якостей. Загальними критеріями включення різних об’єктів в інвестиційний портфель є співвідношення прибутковості, ризикованості та ліквідності інвестиційних вкладень. Разом із тим формування конкретних портфелів має свої особливості.

Так, портфель реальних інвестиційних об’єктів, на відміну від портфелів із іншими інвестиційними об’єктами, є зазвичай найбільш капіталомістким,

найменш ліквідним, високо ризикованим, а також найбільш складним у керуванні. Через це рівень вимог до формування такого портфеля та відбору інвестиційних об’єктів, що будуть до нього включені, дуже високий.

Портфель цінних паперів, у порівнянні з попередніми видами інвестиційних портфелів, має ряд характерних особливостей. До позитивних особливостей можна віднести високий рівень ліквідності та керованості, а до негативних – відсутність у деяких випадках можливості впливати на прибутковість портфеля, а також підвищені інфляційні ризики.

Для задачі формування інвестиційного портфеля в науковій літературі було знайдено величезну кількість рішень, більшість з яких беруть за основу класичну модель Марковіца. Саме теорія Марковіца допомагає інвесторам обрати інвестиційні об’єкти у свій портфель таким чином, щоб результуючий портфель задовольняв поставленим цілям інвестора.

Розвиток обчислювальних потужностей комп’ютерів та поява хмарних сховищ стали поштовхом для розвитку нових методів розв’язку задач оптимізації, класифікацій, кластеризації та інших. Зокрема, останні роки все більшої популярності набувають методи машинного навчання. Вони особливо ефективно проявили себе в задачах класифікації, оптимізації та вибору серед альтернатив. Не залишилась осторонь і фінансова сфера, де застосування машинного навчання також мало місце.

Іншими популярними методами формування інвестиційного портфелю є методи на основі штучного інтелекту. Ці методи допомагають примати рішення, які б відповідали певним початковим вимогам. Також вони проектуються таким чином, щоб мати можливість підлаштуватися під особливості сфери застосування та вирішити задачу найбільш оптимальним чином.

## 1.6 Формалізація постановки задачі дослідження

Проаналізувавши предметну область та вже існуючі методи аналізу в даній області, можна провести таку формалізацію постановки задачі:

— провести дослідження методів рішення задачі формування інвестиційних портфелів;
— розробити власний метод на основі штучного інтелекту для вирішення задачі формування інвестиційного портфелю;
— оцінити результати класичних та розробленого методів;
— зробити висновки щодо доцільності запропонованого підходу.

## 1.7 Висновки до розділу 1

У даному розділі було розглянуто теоретичне підґрунтя інвестиційних процесів, портфельного інвестування та методів формування інвестиційних портфелів. Також розглянуті та описані ці основні підходи. Проаналізовано актуальність даної теми. Підвищення ефективності методів формування інвестиційних портфелів дасть змогу формувати більш прибуткові та менш ризиковані портфелі, що, відповідно, збільшить доходи інвесторів.

Крім цього було проаналізовано основні методи формування інвестиційних портфелів. Окремим пунктом даного розділу є формалізація постановки задачі предметної області.

# РОЗДІЛ 2 МАТЕМАТИЧНІ ОСНОВИ РОЗВ'ЯЗАННЯ ЗАДАЧІ ФОРМУВАННЯ ІНВЕСТИЦІЙНОГО ПОРТФЕЛЯ

## 2.1 Аналіз математичних основ розв’язання задачі

### 2.1.1 Прийняття рішень при нечіткому відношенні переваги на множині альтернатив

Процес формування інвестиційного портфелю містить в собі ключовий етап прийняття рішень. Дуже важливо, щоб цей етап базувався на певних законах та принципах, а не випадковим чином. Звичайно, на цей процес можуть впливати особисті вподобання або передчуття, але основою для остаточного рішення все одно повинно бути математичне та статистичне підґрунтя. Саме для таких випадків виборів найкращої або декількох найкращих альтернатив і використовується метод прийняття рішень, що базується на теорії Лютфі Заде про нечіткі множини та їх властивості, яка була опублікована ще у 1965 році. Дана теорія допомагає формально описувати нечіткі поняття, якими користується людина, описуючи свої бажання, цілі та уявлення про систему [4].

#### 2.1.1.1 Основні поняття задачі багатокритеріального вибору.

Процес прийняття рішення щодо вибору найбільш оптимальної альтернативи серед множини усіх альтернатив може відбуватися при наданні різної кількості інформації про ці альтернативи. У деяких випадках цю інформацію можна описати за допомогою функції корисності, але таке можливо далеко не завжди. Універсальним ж способом опису вхідної інформації є метод запису у формі відношення переваги на множині альтернатив.

Розглянемо це відношення та його властивості. Нехай, на основі вхідної інформації на множині допустимих альтернатив $X$ було введено чітке відношення

нестрогої переваги $R$. Тоді для двох альтернатив можна висловити одне із тверджень:

- $x \succcurlyeq y$ – $x$ не гірше за $y$
- $y \succcurlyeq x$ – $y$ не гірше за $x$
- $(x, y) \notin R$ – $x$ та $y$ не порівняні між собою

Ця інформація дає змого звузити множину раціонального вибору, включивши до неї лише ті альтернативи, які не домінуються жодною іншою. Саме недоміновані альтернативи і є тими, що нас цікавлять.

Для визначення таких недомінованих альтернатив використовується відношення строгої переваги $R_S = R \backslash R^{-1}$, відношення нестрогої переваги $R$ і відношення байдужості $R_J = \{X \times X\}\{R \cup R^{-1}\} \cup \{R \cap R^{-1}\}$. Таким чином, множина недомінованих альтернатив визначається за виразом:

$$X_{\text{нд}} = \{x : x \in X, (y, x) \notin R \backslash R^{-1}, \forall y \in X\} \quad (2.1)$$

При моделюванні реальних систем випадки із чіткими відношеннями нестрогої переваги зустрічаються нечасто. Більш типовим є випадок нечітких відношень. Такі відношення описуються функцією належності $\mu_R(x, y)$, яка має властивість рефлексивності. На основі заданого на $X$ нечіткого відношення нестрогої переваги $R$ можна однозначно визначити три відповідних йому нечітких відношення:

1. байдужості $R_J = \{X \times X\}\{R \cup R^{-1}\} \cup \{R \cap R^{-1}\}.$ ;
2. еквівалентності $R_e = R \cap R^{-1}$;
3. строгої переваги $R_S = R \backslash R^{-1}$.

Таким чином, вирази для функцій належності цих відношень будуть наступними:

1. нечітке відношення байдужості

$$\mu_R^J(x, y) = \max\left[\min\{1 - \mu_R(x, y); 1 - \mu_R(y, x)\}; \min\{\mu_R(x, y); \mu_R(y, x)\}\right] \quad (2.2)$$

2. нечітке відношення еквівалентності:

$$\mu_R^e(x,y) = \min\{\mu_R(x,y); \mu_R(y,x)\} \tag{2.3}$$

3. нечітке відношення строгої переваги:

$$\mu_R^S(x,y) = \begin{cases} \mu_R(x,y) - \mu_R(y,x), \text{якщо } \mu_R(x,y) > \mu_R(y,x) \\ 0, \text{інакше} \end{cases} \tag{2.4}$$

Згідно із означенням операції перетину нечітких множин, вираз для функції належності множини недомінованих альтернатив буде мати наступний вигляд:

$$\mu_R^{\text{нд}} = 1 - sup_{y \in X}[\mu_R^S(x,y)], x \in X \tag{2.5}$$

Не можна забувати, що основною ціллю є пошук найбільш вигідної альтернативи $x_0$. Для цього потрібно, щоб виконувалася наступна властивість:

$$f_j(x_0) \geq f_j(y), \forall j = 1,..,m, \forall y \in X \tag{2.6}$$

Якщо умова виконується, то альтернатива $x_0$ називається ефективною або Парето-оптимальною. Саме ці альтернативи нас цікавлять. Але для розв'язання нашої задачі потрібно скористатися ще одним інструментом – згорткою багатьох критеріїв у скалярний.

Одним з найпоширеніших способів згортки є використання перетину:

$$Q_1 = \bigcap_{i=1}^{m} R_i \tag{2.7}$$

При використанні в рівняння (2.7) функції належності отримаєм наступне:

$$\mu_{Q_1}(x,y) = min\{\mu_1(x,y),\dots,\mu_m(x,y)\} \quad (2.8)$$

Але рівняння (2.8) справедливе лише коли усі критерії рівнозначні за своєю цінністю. Інакше матимемо наступне:

$$\mu_{Q_1}(x,y) = min\{w_1\mu_1(x,y),\dots,w_m\mu_m(x,y)\}, \quad (2.9)$$

де $w_1,\dots, w_m$ – ваги кожного з критеріїв.

Іншою згорткою, важливою для пошуку оптимальних альтернатив є згортка вихідних відношень $\{R_J\}$, яка записується у вигляді суми:

$$Q_2 = \sum_{j=1}^{m} w_j f_j(x,y)\,, \quad (2.9)$$

де $\sum_{j=1}^{m} w_j = 1, w_j \geq 0$.

У разі використання функцій належності, отримаєм рівняння наступного вигляду:

$$\mu_{Q_2}(x,y) = \sum_{j=1}^{m} w_j \mu_j(x,y) \quad (2.9)$$

Оскільки оптимальні значення лежать у просторі недомінованих альтернатив, то вони будуть лежати у множинах $Q_1^{\text{нд}}$ та $Q_2^{\text{нд}}$, отриманих у результаті згорток. Для отримання оптимальних альтернатив, потрібно відшукати такі елементи зі спільної множини недомінованих альтернатив $Q_{\text{нд}} = Q_1^{\text{нд}} \cap Q_2^{\text{нд}}$, для яких значення недомінованості буде максимальним.

2.1.1.1 Алгоритм пошуку ефективних альтернатив.

1. На універсальній множині альтернатив $X$ із заданими відношеннями переваги $R_1, \dots, R_m$ з функціями належності $\mu_j(x, y)$, а також ваговими коефіцієнтами відношень $w_j, j = 1, \dots m$, будуємо згортку $Q_1 = \bigcap_{j=1}^{m} R_j$, із функцією належності (2.8).
2. Визначаємо множину недомінованих альтернатив $Q_1^{\text{нд}}$ за формулою:

$$\mu_{Q_1}^{\text{нд}} = 1 - sup\{\mu_{Q_1}(y, x) - \mu_{Q_1}(x, y)\} \tag{2.10}$$

3. Використовуємо згортку критеріїв у вигляді суми та будуємо нечітке відношення переваги $Q_2$ за допомогою формули (2.9).
4. Знаходимо нечітку підмножину недомінованих альтернатив $Q_2^{\text{нд}}$ по відношенню $Q_2$ за формулою:

$$\mu_{Q_2}^{\text{нд}} = 1 - sup_{y \in X}\{\sum_{j=1}^{m} \mu_{Q_2}(y, x) - \mu_{Q_2}(x, y)\} \tag{2.11}$$

5. Знаходимо перетин множин $Q_1^{\text{нд}}$ та $Q_2^{\text{нд}}$, а також спільну множину недомінованих альтернатив $Q_{\text{нд}}$ з функцією належності:

$$\mu_{\text{нд}}(x) = min\{\mu_{Q_1}^{\text{нд}}(x), \mu_{Q_2}^{\text{нд}}(x)\} \tag{2.12}$$

Раціональним буде вважатися вибір із множини:

$$X_{\text{чнд}} = \{x^* : \mu_{\text{нд}}(x^*) = sup_x \mu_{\text{нд}}(x), x \in X\} \tag{2.13}$$

### 2.1.2 Модель Марковіца.

Модель для врахування відношення між ризиком інвестицій та їх потенційною доходністю була запропонована Гаррі Марковіцем ще в 1957 році і від того часу стала класичною для рішення задачі формування інвестиційного портфеля.

#### 2.1.2.1 Основна ідея моделі Марковіца.

Теорія формування інвестиційного портфелю за Г. Марковіцем засновується на поведінковій специфіці інвестора, який хоче вкласти свої ресурси в компанії із найменшим ризиком, і за цей ризик отримати певні дивіденди. Підхід Марковіца передбачає, що інвестор враховує лише два параметри: ризикованість та величину доходів.

Модель оптимального портфелю за Г. Марковіцем ґрунтується на наступних принципах [5]:

1. інвестор хоче максимізувати дохід за певного заданого рівня ризику;
2. інвестори завжди намагаються уникати ризику. Із двох активів із однаковою прибутковістю обирається той, ризикованість якого нижча;
3. ризик є невизначеністю майбутнього результату;
4. портфель інвестора складається з всіх його активів та зобов’язань;
5. інвестори приймають рішення про інвестиції, засновуючись на очікуваних доходах та ризикованості інвестицій. «Корисність інвестицій» вираховується за формулою:

$$U = E_R - A(E_Q)/2 \qquad (2.14)$$

де $A$ - ступінь неприйняття ризику інвестором;

$E_R$ – очікуваний дохід;

$E_Q$ – очікуваний ризик.

Модель диверсифікації, запропонована Марковіцем, стала найбільш розповсюдженою моделлю на практиці. Ідея цієї моделі полягає у тому, що портфель повинен складатися із цінних паперів, що відрізняються за різними критеріями, такими як види, терміни, емітенти та інше. При цьому диверсифікація здійснюється на основі двох основних характеристик: доходності та ризикованості.

Кожен окремий інвестор формує свій інвестиційний портфель, який залежить від особистих вимог інвестора. Такий підбір здійснюється методом «проб та помилок», для чого використовується метод побудови кривих байдужості (рис. 2.1).

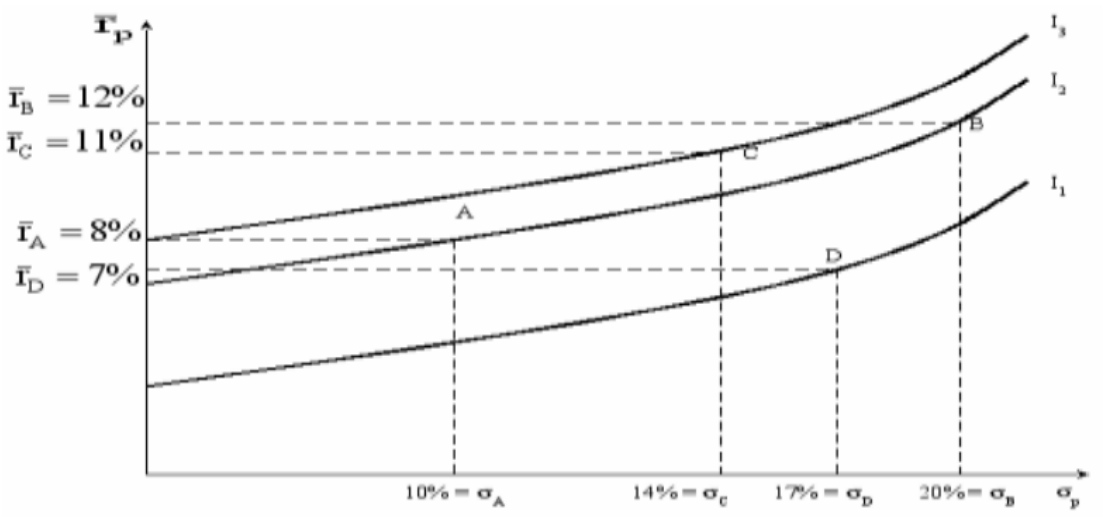


Рисунок 2.1 – Криві байдужості.

Для кривих байдужості притаманні такі властивості:

1. усі портфелі, що знаходяться на одній кривій байдужості, є рівнозначними для інвестора, а наслідком з цього є те, що криві байдужості не можуть перетинатися;

2. для інвестора можна побудувати нескінченну кількість кривих байдужості;
3. між двома портфелями, з однаковим стандартним відхиленням, інвестор вибере той, для якого очікувана дохідність буде вища;
4. портфель є набором різних цінних паперів, тобто очікувана прибутковість портфеля і його ризикованість повинні залежати від прибутковості та ризикованості його складових.

2.1.2.2 Формування інвестиційного портфелю за методом Марковіца.

За моделлю Марковіца, доходність портфеля – це середньозважена дохідність його складових, що визначається за формулою:

$$R_p = \sum_{i=1}^{N} W_i * r_i, \tag{2.15}$$

де $N$ – кількість цінних паперів,

$r_i$ – дохідність конкретного цінного паперу,

$W_i$ – частка даного цінного паперу у відсотках.

Для визначення ризикованості портфеля цінних паперів використовується наступна формула:

$$Risk = \sqrt{\sum_{a=1}^{N}\sum_{b=1}^{N}(W_a\sigma_a W_b\sigma_b r_{ab})}, \tag{2.16}$$

де $W_i$ – частка даного цінного паперу у відсотках,

$r_{ab}$ – коефіцієнт лінійної кореляції,

$\sigma_a$, $\sigma_b$ – ризики паперів $a$ та $b$ (середньоквадратичне відхилення).

Використовуючи модель Марковціа, маємо наступну задачу оптимізації:

$$\begin{cases} \dfrac{\sum_{i=1}^{N} W_i \times r_i}{\sqrt{\sum_{a=1}^{N}\sum_{b=1}^{N}(W_a\sigma_a W_b\sigma_b r_{ab})}} \leq \sigma_{req} \\ W_i \geq 0 \\ \sum W_i = 1 \end{cases} \quad (2.17)$$

Для моделі Марковіца важливою є теорема про ефективну множину, яка каже, що інвестор вибере свій оптимальний портфель серед безлічі портфелів , кожен з яких буде забезпечувати:

- максимальну прибутковість для заданого рівня ризику;
- мінімальний ризик для заданого значення очікуваної прибутковості.

Набір портфелів, що задовольняють теоремі, називають ефективною множиною або ефективною межею. В межах цієї множини або на межі знаходяться усі портфелі, що можуть бути сформовані з певної кількості цінних паперів. Ефективною множиною називають область, у якій розташовані точки, а ефективною межею називають лінію, що графічно окреслює цю множину.

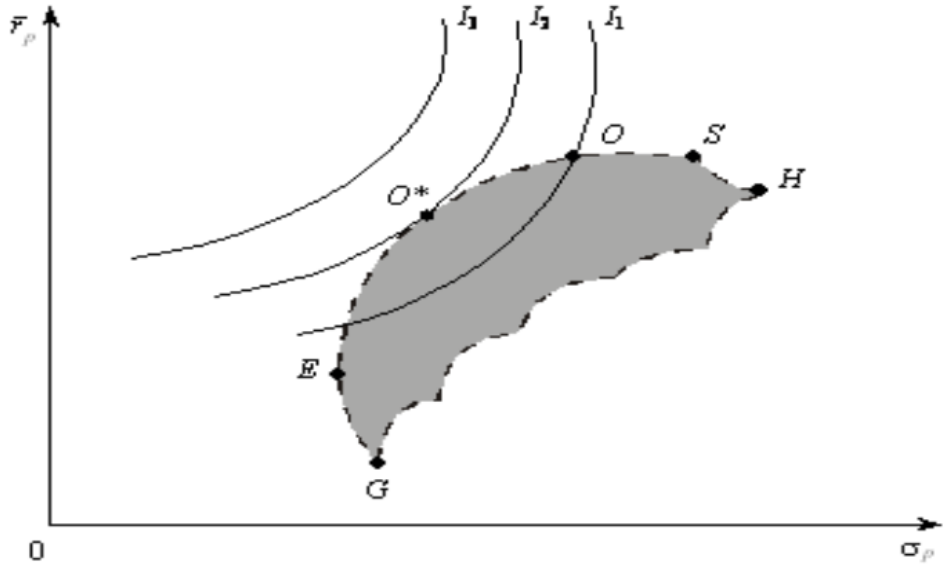

Рисунок 2.2 – Ефективна множина та межа.

Ціллю інвесторів є збільшити свій дохід, при цьому знижуючи ризики. Оптимальними, для інвестора, будуть портфелі, що знаходяться у межах ефективної множини. Але для кожного інвестора оптимальний інвестиційний портфель буде відрізнятися в залежності від його вимог та вподобань, хтось не бажає ризикувати, а комусь важливий лише дохід, не зважаючи на ризики. Оптимальним же за Парето портфель, для інвестора, за умови нехтування його вподобаннями та стратегією інвестування, буде той портфель, що знаходиться найвище та найбільш вліво, серед всіх точок множини. На рис. 2.2 такий портфель знаходиться в точці $O^*$.

2.1.2.3 Вибір портфелю за коефіцієнтом Шарпа.

Як вже було сказано, вибір об’єктів інвестиційного портфелю залежить не тільки від самих об’єктів, а також і від стратегії та цілей інвестора. Таким чином, найбільш очевидними є стратегії мінімального ризику та максимальної прибутковості. Іншим же способом формування інвестиційного портфелю є спосіб із максимізацією коефіцієнта Шарпа [6].

Даний коефіцієнт допомагає вирішити проблему вибору між двома об’єктами інвестування, шляхом порівняння їх дохідності із безризиковим об’єктом. Сам коефіцієнт Шарпа обраховується за формулою:

$$S = \frac{R_p - R_f}{\sigma_p}, \tag{2.18}$$

де $R_p$ – прибутковість портфелю,

$R_f$ – прибутковість безризикового об’єкта,

$\sigma_p$ – ризикованість портфеля.

Таким чином, критерій Шарпа покаже відносну прибутковість активу на одиницю ризику. Максимізуючи це значення, буде сформовано портфель із об’єктів з найбільш «вигідним» ризиком.

### 2.1.3 Мережа Байєса.

Мережі Байєса мають широке застосування мають широкий спектр застосування в інформаційних системах обробки статистичних даних, представлених часовими рядами. Особливе місце ці мережі посідають у задачі управління ризиками. Мережі Байєса допомагають встановити причинно-наслдікові зв’язки між певними ознаками та висновком, що отримується при таких умовах.

Побудова Байєсівських мереж пов’язана з необхідністю послідовного розв’язання низки задач, зокрема задач обчислювального характеру, що зустрічається при навчанні мережі. В загальному випадку задача навчання мережі відноситься до NP-повних задач, що означає, що об’єм обчислень зростає із збільшенням кількості змінних мережі.

#### 2.1.3.1 Математичні основи мережі Байєса.

Мережу Байєса можна розглядати як модель представлення взаємозв’язків між вершинами ациклічного графа, що представлені у вигляді ймовірнісних залежностей. Зв’язок $A \rightarrow B$ буде називатися причинним лише тоді, коли подія $A$ є причиною виникнення події $B$, тобто подія $A$ певною мірою впливає на виникнення події $B$. Сама ж мережа буде називатися причинною лише тоді, коли всі її зв’язки є причинними.

Формально, мережа Байєса – це трійка $N = < V, G, J >$, де $V$ – множинна змінних, $G$ – спрямований ациклічний граф, а $J$ – спільний розподіл ймовірностей змінних $V = \{X_1, \ldots, X_n\}$. Варто зазначити, що для множини змінних $V$ виконується марковська умова, а це значить, що кожна змінна мережі не залежить від усіх інших змінних, окрім батьківських попередників цієї змінної [7].

2.1.3.2 Теорема Байєса.

Основою мереж Байєса є теорія ймовірносте і зокрема теорема Байєса.

Із теорії ймовірносте відомо, що ймовірність одночасної появи двох незалежних подій $D$ та $S$ визначається за формулою:

$$p(D, S) = p(D)p(S) \tag{2.18}$$

Якщо події $D$ та $S$ залежні одна від одної, то поява однієї з них дає деяку інформацію про можливість появи іншої:

$$p(D, S) = p(D)p(S|D) \tag{2.19}$$

де $p(S|D)$ – ймовірність появи події $S$ у випадку, коли подія $D$ вже відбулась.

Оскільки рів. 2.19 комутативне то справедлива наступна серія рівностей:

$$p(D, S) = p(D)p(S|D) = p(S)p(D|S) \tag{2.20}$$

Саме з рівняння 2.20 і бере свій початок теорема Байєса, яку можна описати наступною формулою:

$$p(D|S) = \frac{p(D)p(S|D)}{p(S)} \tag{2.21}$$

Теорему Байєса можна розглядати як механізм прийняття рішень, і елементи рів. 2.21 можна пояснити наступним чином:

- $p(D|S)$ – ймовірність події $D$ при наявності збурника $S$, тобто події, відносно якої потрібно сформувати висновок;
- $p(D)$ – ймовірність цільової події $D$, цю величину можна оцінити за допомогою історичних даних;
- $p(S|D)$ – ймовірність зворотнього ходу подія, появи $S$ за наявності $D$;
- $p(S)$ – ймовірність появи події $S$, це значення також можна оцінити шляхом аналізу історичних даних, але це зазвичай не є необхідним.

Теорему Байєса можна розглядати як механізм, який об'єднує «апріорну» та «правдоподібну» інформацію. Це можна записати наступним рівнянням:

$$p(D|S) = \alpha p(D)p(S|D), \tag{2.22}$$

де $\alpha = \frac{1}{p(S)}$ – нормуюча константа.

Тепер $p(D)$ можна розглядати як апріорну інформацію, оскільки вона була відома до отримання будь-яких вимірів, а $p(S|D)$ – правдоподібна інформація, оскільки це значення вираховується у результаті аналізу вимірів. У деяких випадках апріорна ймовірність може бути обчислена на основі статистичних даних. Однак, для більшості випадків це зробити неможливо через суб'єктивні труднощі отримання статистичних даних, але апріорну інформацію можна представити у інших формах.

2.1.3.2 Опис простої мережі Байєса.

Розглянутий вище підхід до формування Байєсівського висновку не дає можливості застосовувати його у більш складних ситуаціях обробки апріорної інформації. В реальних прикладах, інформація може поступати із декількох джерел, а рів. 2.21 неспроможне впоратися із такою задачею. Саме тому теорема Байєса приймає більш загальний вигляд:

$$p(D|S_1,\ldots,S_n) = \frac{p(D)p(S_1,\ldots,S_n|D)}{p(S_1,\ldots,S_n)} \quad (2.23)$$

В такому випадку маємо проблему із оцінюванням умовної ймовірності для великих значень $n$. Але, якщо припустити, що всі події $S_1,\ldots,S_n$ незалежні при відомому $D$, то справедливим буде наступне:

$$p(S_1,\ldots,S_n|D) = p(S_1|D)\ldots p(S_n|D) \quad (2.24)$$

При подальшому нормуванні рівняння 2.23 можна позбутися знаменника $p(S_1,\ldots,S_n)$, що спростить задачу формування висновку. Таким чином, отримаємо узагальнене рівняння для формування висновку за теоремою Байєса:

$$p(D|S_1,\ldots,S_n) = \alpha p(D)p(S_1|D)\ldots p(S_n|D) \quad (2.25)$$

Рівняння 2.25 може бути представлено графічно у вигляді графа (Рис. 2.3).

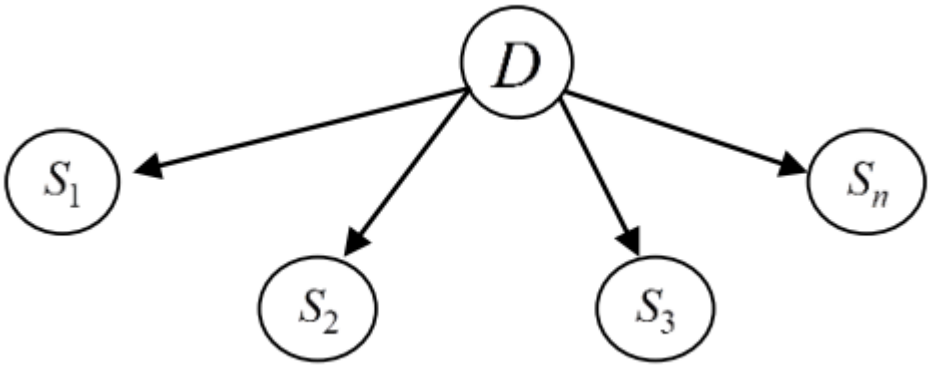


Рисунок 2.3 – Ефективна множина та межа.

На графі змінні представлені у вигляді кіл, а стрілки відображають зв’язок – умовну ймовірність між незалежними та залежними змінними. Незалежні змінні називаються батьківськими або попередниками, а залежні – дитячими або нащадками.

### 2.1.2 Моделі на основі штучного інтелекту.

Використання моделей на основі штучного інтелекті стає все більш популярним у різних сферах. Не є виключенням і сфера інвестування. При правильному виборі моделей та попередній обробці даних, такі моделі дають кращі результати, ніж їх класичні конкуренти. Різноманітних моделей штучного інтелекту є велика кількість, починаючи від різноманітних варіацій градієнтних бустингів і завершуючи повноцінними нейронними мережами.

Нерідко, для вирішення якоїсь задачі однієї моделі недостатньо. Для вирішення цієї проблеми використовують ансамбль моделей, тобто декілька алгоритмів машинного навчання, зібраних в єдине ціле. Такий підхід часто використовується для того, щоб підсилити позитивні якості окремих алгоритмів, які самі по собі можуть проявлятися слабко, а в групі показують чудовий результат. При використанні ансамблевих методів, алгоритми навчаються одночасно і можуть

виправляти помилки один одного. Ансамблі моделей зазвичай будуються на основі моделі дерева рішень. Дерева додаються по одному в ансамбль і навчаються для взаємного виправлення помилок прогнозування, що роблять попередні моделі. Такий тип ансамблів називають бустингом.

### 2.1.2.1 Модель градієнтного бустингу.

Градієнтний бутинг є однією із класичних моделей штучного інтелекту. Окрім цього він відноситься до ансамблевих моделей штучного інтелекту, тобто дана модель штучного інтелекту буде складатися із декількох моделей. Загальний метод бустингу є базовим і всі подальші модифікації та покращення так чи інакше базуються на ньому.

Розглянемо задачу розпізнавання об’єктів на багатовимірному просторі $X$ з простором міток $Y$. Нехай нам дана навчальна вибірка $\{x_i\}_{i=1}^{N}$, де $x_i \in X$. І нехай на цій вибірці відомі справжні значення міток кожного об’єкту $\{y_i\}_{i=1}^{N}$, де $y_i \in Y$. необхідно побудувати оператор розпізнавання, котрий зможе якнайточніше передбачити мітки для кожного нового об’єкта $x \in X$. Нехай нам задано деяке сімейство алгоритмів $H$, кожен елемент $h(x;\ a) \in H : X \to R$ якого визначається деяким вектором параметрів $a \in A$.

Будемо шукати фінальний алгоритм класифікації у вигляді наступної композиції:

$$F_M(\mathrm{x}) = \sum_{m=1}^{M} b_m h(x;\ a_m), b_m \in R, a_m \in A \qquad (2.26)$$

Але підбір оптимального набору параметрів $\{a_m, b_m\}_{m=1}^{M}$ – задача, що потребує великої кількості обрахунків. Через це, композиція 2.25 буде будуватися шляхом жадібного нарощування, кожен раз додаючи в суму доданок, який є найбільш оптимальним алгоритмом серед можливих. Припустимо, що вже було побудовано класифікатор $F_{m-1}$ довжини $m - 1$. Таким чином задача зводиться до пошуку пари найбільш оптимальних параметрів $\{a_m, b_m\}$ для класифікатора довжини $m$:

$$F_m(\mathrm{x}) = F_{m-1}(\mathrm{x}) + b_m h(x, a_m), b_m \in R, a_m \in A \tag{2.27}$$

Оптимальність тут розуміється у відповідності до принципу явної максимізації відступів, а це означає, що вводиться деяка функція втрат $L(y_i, F_m(x_i)), i \in 1, \ldots, N$, яка показує наскільки сильно передбачене значення $F_m(x_i)$ відрізняється від справжнього значення $y_i$. Для знаходження оптимальних параметрів мінімізується функціонал помилки:

$$Q = \sum_{i=1}^{N} L(y_i, F_m(x_i)) \rightarrow min \tag{2.28}$$

Варто зазначити, що функціонал помилки $Q$ – дійсна функція, яка залежить від точок $\{F_m(x_i)\}_{i=1}^{N}$ в $N$-вимірному просторі. Наша мета – це вирішити задачу мінімізації цього функціонала. Зробимо це можна шляхом виконання одного кроку методу градієнтного спуску. Розглянемо точку $F_{m-1}$ як ту, для якої будем шукати оптимальний приріст. Градієнт функціоналу помилок буде мати вигляд:

$$\nabla Q = [\frac{\partial Q}{\partial F_{m-1}}(x_i)]_{i=1}^{N} = [\frac{\partial(\sum_{i=1}^{N} L(y_i, F_{m-1}(x_i)))}{\partial F_{m-1}}(x_i)]_{i=1}^{N} =$$
$$= [\frac{\partial L(y_i, F_{m-1})}{\partial F_{m-1}}(x_i)]_{i=1}^{N} \tag{2.29}$$

Беручи до уваги вищезгадану інформацію та специфіку методу градієнтного спуску, найбільш ефективним буде рішення додавати новий доданок в класифікатор наступним чином:

$$F_m = F_{m-1} - b_m \nabla \mathrm{Q}, b_m \in \mathrm{R}, \tag{2.30}$$

де $b_m$ підбирається лінійним пошуком по дійсним числам із $R$:

$$b_m = argmin_{b \in R} \sum_{i=1}^{N} L(F_{m-1}(x_i) - b \nabla Q_i) \tag{2.31}$$

Але ∇Q являє собою лише вектор оптимальних значень для кожного об’єкта $x_i$, а не базовий алгоритм із сімейства $H$, що визначений $\forall x \in X$. Через це необхідно знайти $h(x, a_m) \in H$ найбільш подібний до ∇Q. Це знову ж таки можна зробити шляхом мінімізації функціоналу помилки, що заснований на принципі явної максимізації відступів:

$$a_m = argmin_{a \in A} \sum_{i=1}^{N} L(\nabla Q_i, h(x_i, a)) \equiv \text{навчити } (\{x_i\}_{i=1}^{N}, \{\nabla Q_i\}_{i=1}^{N}), \tag{2.32}$$

що просто відповідає базовому алгоритму навчання. Далі коефіцієнт $b_m$ можна знайти, використавши лінійний пошук:

$$b_m = argmin_{b \in R} \sum_{i=1}^{N} \sum_{i=1}^{N} L(F_{m-1}(x_i) - b \mathrm{h}(x_i, a_m)) \tag{2.32}$$

#### 2.1.2.2 Light Gradient Boosted Machine.

Light Gradient Boosted Machine (LightGBM) є розширенням алгоритму градієнтного бустингу, яке додає тип автоматичного вибору об’єктів, а також він фокусується на прикладах бустингу із великими градієнтами. LightGBM найчастіше використовується при роботі із табличними даними для задач регресійного та класифікаційного прогностичного моделювання.

Сама модель LightGBM біла описана Кі Голінем та його співавторами у статті 2017 року. Реалізація LightGBM відрізняється від звичайного градієнтного бустингу двома клячевими ідеями: GOSS та EFB.

Градієнтна одностороння вибірка (Gradient-based One-Side Sampling, GOSS) являє собою модифікацію градієнтного бустингу, яка фокусує увагу на тих навчальних прикладах, які приводять до більшого градієнта, що, в свою чергу, збільшує швидкість навчання шляхом зменшення обчислювальної складності моделі. За допомогою GOSS виключається велика кількість екземплярів даних з найбільшими градієнтами і у використання ідуть інші дані, які будуть використані для оцінки приросту інформації.

Об’єднання ознак, що взаємо виключаються (Exclusive Feature Bundling, EFB) – це підхід об’єднання розріджених, переважно нульових, взаємовиключних ознак, таких як категоріальні змінні вхідних даних, що закодовані унітарним кодуванням. Таким чином EFB компонує декілька ознак в одну, при чому ці ознаки не можуть бути всі нульовими одночасно, оскільки є взаємовиключними. Це зменшує кількість ознак в моделі.

Ці дві зміни GOSS та EFB дають значний приріс у швидкості навчання. У деяких випадках приріст складає 20 разів. При цьому, точність моделі нічим не гірша від моделі звичайного градієнтного бустингу.

#### 2.1.2.2 CatBoost.

CatBoost є ще одним покращенням для моделі градієнтного бустингу, яка підвищує швидкодію. В градієнтному бустингу прогнози робляться на основі ансамбля слабких алгоритмів, що начаються. При цьому, для ансамбля дерева будуються послідовно і попередні дерева в моделі не змінюються. Результати попереднього дерева використовуються для покращення наступного.

CatBoost, замість довільних дерев, використовує недбалі (oblivious) дерева рішень, щоб отримати в результаті збалансоване дерево. Одні і ті ж самі функції використовуються для створення лівих та правих розділень на кожному рівні дерева. В порівнянні із класичними деревами, недбалі дерева більш ефективні при реалізації на процесорі приладу, а також більш легко навчаються. Також, цей підхід підвищує результати прогнозування шляхом зменшення перенавчання моделі.

## 2.3 Висновки до розділу 2

Для розв’язання задачі формування інвестиційного портфелю, існує широкий спектр методів, від класичної теорії прийняття рішень при виборі альтернатив і до моделей, що базуються на штучному інтелекті. Кожен з методів має свох перевага та недоліки і мають свою нішу, де їх використання є найбільш пріоритетним та ефективним. Але різноманітні підходи до аналізу усе продовжують з’являтись та модифікуватись. У цій роботі розглянуто класичні методи формування інвестиційного портфелю і розроблений метод формування інвестиційного портфелю на основі штучного інтелекту.

# РОЗДІЛ 3 РОЗРОБКА МЕТОДУ ФОРМУВАННЯ ІНВЕСТИЦІЙНОГО ПОРТФЕЛЮ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ ТА ПРОГРАМНОГО ПРОДУКТУ

В даному розділі описано програмні продукти, що були використані при розробці програми для формування інвестиційного портфелю, загальний принцип роботи програми та опис конкретних її модулів. Також будуть представлені результати моделей, та проведено їх порівняння.

## 3.1 Опис використаних даних

Важливим етапом в моделювання є вибір та обробка даних. Для задачі формування інвестиційного портфелю ключовими даними є данні про компанію, а також про її фінансове становище та динаміку її розвитку. Таким чином, у роботі було використано дані трьох типів:

1. Базова інформацію про компанію;
2. Поквартальні звіти компанії;
3. Щоденна інформація про компанію та її акції.

Розглянемо ці дані більш детально.

До базових даних відноситься уся інформація про компанію, яка, переважно, не змінюється із часом. До таких даних відноситься інформація про індустрію, на якій спеціалізується компанія, сектор, який займає компанія, та інше.

Поквартальні звіти включають в себе різноманітну інформацію про динаміку розвитку компанії, а також типові метрики, які використовуються для оцінки фінансової характеристики компанії. До таких даних відносяться розмір боргу, розмір прибутку компанії, інформація про річні дивіденди акціонерів, прибуток без урахування виплати податків та інше.

Останнім типом інформації є щоденні дані. Вони описують щоденну динаміку фінансової привабливості компанії та її акцій. До таких даних відноситься інформація про капіталізацію компанії, а, також, ціна закриття курсу акцій компанії.

Увесь набір даних включає у себе проміжок в п’ять років. Для навчання моделей використовуються дані до «01-07-2022». Перевірочні дані закінчуються датою «28-09-2022». Кількість компаній, які використовуються для навчання моделей значна і їх кількість перевищує 500. Але кількість компаній, з яких буде формуватися портфель, обмежена кількість. Було взято по шість компаній із чотирьох різних індустрій: «Виробництво автомобілів», «Розробка програмного забезпечення», «Виробництво електроприладів», «Розважальний сектор». Отже, були обрані наступні компанії:

- «Виробництво автомобілів» - NKLA, TSLA, RACE, STLA, F, TM;
- «Розробка ПЗ» - MSFT, ADBE, ORCL, PAYO, BB, DBX;
- «Виробництво електроприладів» - AAPL, NVDA, AMD, SONY, KOSS, BOX
- «Розважальний сектор» - NFLX, DIS, IMAX, WWE, CNK, WMG

Основною метою даної роботи було розробити новий більш ефективний підхід до прогнозування курсу валюти і порівняти його з класичним. Зробимо загальний опис класичного та розробленого підходів детальніше.

У класичному варіанті для прогнозування курсу валют використовується аналіз часових рядів. Цей підхід зумовлює використання авторегресійних моделей із інтегрованим ковзним середнім. Ці моделі описують поведінку цільового курсу і дають можливість отримати прогноз.

Розроблений підхід ґрунтується на взаємозалежності усіх об’єктів фінансового ринку, тобто залежності рівнями курсів валют, цінами на ресурси та цінні папери та інше. Альтернативний підхід складається з двох етапів і використовує дві моделі.

На першому етапі використовуються, як і в класичному підході, авторегресійні моделі з ковзним середнім. Відмінністю з класичним підходом

полягає у тому, що ці моделі будуються не для цільового курсу, а для інших курсів та цін на різні ресурси. Можна сказати, що використовується класичний підхід до прогнозування цін певну кількість разів і в результаті отримується певна кількість прогнозів.

На другому етапі відбувається пошук залежності між нецільовими даними та цільовим курсом. Для цього використовуються методи регресійного аналізу, зокрема модель множинної лінійної регресії. В результаті отримується прогноз рівня цін цільового курсу.

## 3.2 Аналіз результатів

### 3.2.1 Модель прийняття рішень при нечіткому відношенні переваги на множині альтернатив

Розглянемо модель прийняття рішень, що була побудована для вибору оптимальних інвестиційних об’єктів в інвестиційний портфель. Дана модель на вхід отримувала ознаки, які описують фінансове становище компанії, та динаміку її росту. Найбільш важливими ознаками були індустрія, в якій працює компанія, зміна ціни на акції та зміна капіталізації компанії, а також розмір боргу. Данні бралися за останній квартал, а різницеві данні бралася як різниця останнього та передостаннього кварталів.

На вихід модель видає метрику важливості кожної альтернативи, яка знаходиться у межах від 0 до 1. Чим більшою є величина метрики, тим більш пріоритетним буде вибір даної альтернативи.

Кількість компаній, що будуть додані до портфелю встановлюється користувачем. Кількість акцій кожної компанії у відсотках знаходиться за формулою:

$$percent_i = \frac{metric_i}{\sum_{i=1}^{N} metric_i} * 100, \quad (3.1)$$

де $metric_i$ – метрика, що повернула модель для і-ї компанії,

$N$ – кількість компаній у портфелі.

Кількість компаній в інвестиційному портфелі була обрана як сім.. Таким чином, в результаті роботи моделі, було сформовано наступний інвестиційний портфель:

Таблиця 3.1 – Інвестиційний портфель моделі прийняття рішень

| Компанія | Різниця ціни закриття | Метрика | Відсоток | Прибуток |
|---|---|---|---|---|
| TSLA | -64.24 | 1 | 21.74 | -1396.52 |
| NKLA | 0.29 | 0.75 | 16.30 | 4.73 |
| WWE | 12.68 | 0.7 | 15.22 | 192.96 |
| BOX | -1.00 | 0.6 | 13.04 | -13.04 |
| STLA | -4.77 | 0.55 | 11.96 | -57.03 |
| BB | -0.76 | 0.5 | 10.87 | -8.26 |
| AMD | -1.33 | 0.5 | 10.87 | -14.46 |
| Сумарний прибуток | | -1291.63 | | |

Інвестиційний портфель (табл. 3.1) має незадовільні результати. Він приніс значні збитки -1291.63. Більшість об’єктів інвестування, обраних моделлю, втратили в ціні. Найбільших збитків завдала TSLA. Акції цієї компанії впали найбільше, із усього набору альтернатив, а модель обрала ці акції, як найбільш бажані. З позитивних моментів можна визначити вибір об’єкту WWE. Акції цієї компанії виросли найбільше з усього набору, але цього не достатньо, щоб компенсувати втрати цього портфелю.

Таким чином, модель прийняття рішень не дала задовільного результату. Вона обрала лише ті об’єкти інвестування, які до вподоби інвестору, майже не враховуючи фінансову доцільність. Задання ваг цінності кожного критерію

негативно впливає на ефективність моделі, адже вони засновуються на певних суб'єктивних судженнях а не аналітичних рішеннях.

### 3.2.2 Модель мереж Байєса

Модель мереж Байєса на вхід отримує різноманітні дані про компанії, з яких планується формувати портфель, а також про динаміку їх фінансових показників. На вихід модель повертає ймовірність того, що акції компанії будуть рости в ціні. У портфель будуть обрані лише ті компанії, ймовірність росту яких буде перевищувати певну меже відсікання, що задається інвестором. Відсотковий вміст кожного інвестиційного об'єкту визначається вираховується за формулою:

$$percent_{\ i} = \frac{prob_i}{\sum_{i=1}^{N} prob_i} * 100, \quad (3.2)$$

де $prob_i$ – ймовірність росту ціни акції і-ї компанії,

$N$ – кількість компаній, ймовірність росту акцій яких перевищила межу відсікання.

Для даної моделі, денні данні були перетворені в квартальні та з'єднані з квартальними даними. До них також було додано і базову інформацію про сектор та індустрію кожної компанії. Для числових даних було взято першу різницю з метою формування задачі визначення росту чи падіння курсу акцій за вхідними даними. У якості значення відсікання було обране значення 0.55, тобто усі компанії, ймовірність росту акцій яких перевищує 55%.

Таблиця 3.2 – Інвестиційний портфель моделі мереж Байєса

| Компанія | Різниця ціни закриття | Ймовірність | Відсоток | Прибуток |
|---|---|---|---|---|
| MSFT | 6.19 | 56.14 | 11.41 | 70.63 |
| ADBE | -87.1 | 56.12 | 11.41 | -993.42 |
| ORCL | 8.36 | 56.14 | 11.41 | 95.38 |
| PAYO | 2.08 | 56.12 | 11.41 | 23.72 |
| BB | -0.76 | 56.12 | 11.41 | -8.67 |
| DBX | 2.41 | 56.12 | 11.41 | 27.49 |
| SONY | 2.76 | 99.17 | 20.15 | 55.62 |
| BOX | -1 | 56.12 | 11.41 | -11.41 |
| Сумарний прибуток | | -740.65 | | |

Отриманий, моделлю мереж Байєса, портфель також виявився збитковим. У порівнянні з попереднім портфелем він має ряд переваг, окрім меншого збитку. До них можна віднести велику кількість прибуткових акцій. Лише три з восьми інвестиційних об’єктів впали в ціні і, нажаль, падіння одного з них було значущим. Для всіх інших об’єктів характерний ріст ціни. Варто зазначити, що у випадку підвищення величини значення відсікання, ризикованість портфелю впаде, і, в даному конкретному випадку, залишився б лише один інвестиційний об’єкт SONY, який мав приріст в ціні.

### 3.2.3 Модель Марковіца

Метод Марковіца був реалізований шляхом перебором ста мільйонів портфелів. В результаті, було отримано наступну хмару портфелів:

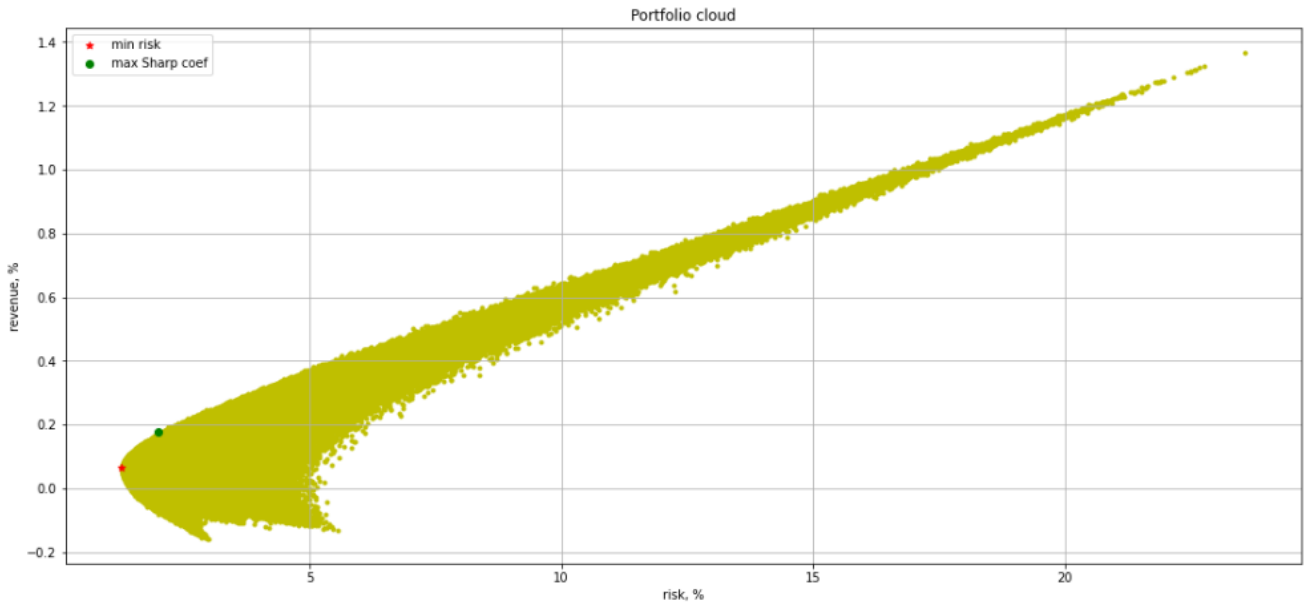


Рисунок 3.1 – Хмара портфелів за моделлю Марковіца

Використовуючи дві різні стратегії, стратегію мінімального ризику та стратегію максимізації коефіцієнта Шарпа, було отримано два портфелі табл 3.3 та табл. 3.4.

Таблиця 3.3 – Інвестиційний портфель моделі Марковіца мінімального ризику

| Компанія | Різниця ціни закриття | Відсоток | Прибуток |
|---|---|---|---|
| TSLA | -64.24 | 1.4 | -89.94 |
| STLA | -4.77 | 2.3 | -10.97 |
| TM | -8.15 | 33.3 | -271.40 |
| MSFT | 6.19 | 2.8 | 17.33 |
| ADBE | -87.1 | 1.4 | -121.94 |
| ORCL | 8.36 | 16.9 | 141.28 |
| DBX | 2.41 | 3.4 | 8.19 |
| NVDA | -37.17 | 0.9 | -33.45 |
| AMD | -1.33 | 2.5 | -3.33 |
| SONY | 2.76 | 3 | 8.28 |
| BOX | -1 | 2.6 | -2.6 |

Продовження таблиці 3.3

| Компанія | Різниця ціни закриття | Відсоток | Прибуток |
|---|---|---|---|
| NFLX | 5.66 | 1.1 | 6.23 |
| DIS | 7.22 | 2.6 | 18.77 |
| IMAX | 1.83 | 1.1 | 2.01 |
| WWE | 12.68 | 16.4 | 207.95 |
| CNK | 1.57 | 1.3 | 2.04 |
| WMG | 3.72 | 3.3 | 12.28 |
| Сумарний прибуток | | -105.67 | |

У табл.3.3 показано найбільш значущі елементи портфелю моделі Марковіца мінімального ризику. Портфель показав вже значно кращий результат ніж у двох попередніх варіантах, але він все одно є збитковим. Позитивними аспектами портфелю є те, що найбільш збиткові об’єкти інвестування, як ADBE, TSLA, NVDA взяті із дуже низьким відсотком. Крім того, найбільш прибуткові об’єкти, як WWE та ORCL, складають майже третину всього портфелю. Найбільшого збитку портфелю завдала купівля акцій TM із значним відсотком, зазвичай ці акції відносно стабільні, але цього разу вони сильно впали протягом кварталу.

Також варте уваги те, що ринок зараз має тенденцію до падіння. Якби були об’єктом інвестування, які б виросли приблизно так само як впали TSLA чи ADBE, то портфель міг би вийти в нуль, а то і взагалі принести прибуток.

Таблиця 3.4 – Інвестиційний портфель моделі Марковіца максимального коефіцієнта Шарпа

| Компанія | Різниця ціни закриття | Відсоток | Прибуток |
|---|---|---|---|
| TSLA | -64.24 | 3.9 | -250.54 |
| F | -0.62 | 15.5 | -9.61 |
| TM | -8.15 | 2.7 | -22.01 |

Продовження таблиці 3.4

| Компанія | Різниця ціни закриття | Відсоток | Прибуток |
|---|---|---|---|
| ADBE | -87.1 | 2.1 | -181.91 |
| ORCL | 8.36 | 1.3 | 10.87 |
| AAPL | 4.86 | 1.6 | 7.78 |
| NVDA | -37.17 | 2.7 | -100.36 |
| AMD | -1.33 | 1.6 | -2.13 |
| KOSS | 1.75 | 4.5 | 7.88 |
| BOX | -1 | 5.9 | -5.9 |
| DIS | 7.22 | 1.3 | 9.39 |
| IMAX | 1.83 | 1.3 | 2.38 |
| WWE | 12.68 | 48.9 | 620.05 |
| CNK | 1.57 | 3.8 | 5.97 |
| Сумарний прибуток | | 96.75 | |

У табл. 3.4 показано найбільш значущі об’єкти портфелю моделі Марковіца максимального коефіцієнта Шарпа. Даний портфель вже є прибутковим, але ця прибутковість досягнута за рахунок авантюрної покупки акцій WWE, які складають майже половину всього портфелю. Також зросли відсотки і для компаній TSLA, ADBE і NVDA, що знизило потенційні прибутки портфелю.

### 3.2.4 Розроблений метод формування інвестиційного портфелю на основі штучного інтелекту.

Розглянемо попередню обробку вхідних даних. На вхід модель отримує дані всіх форматів: базові, щоденні та поквартальні. Базові текстові дані при цьому конвертуються в числові. Для денних та квартальних даних, для кожної метрики

отримуються певні статистики за різні задані проміжки, такі як середнє відхилення, середнє значення за період та інші.

Розроблений метод працює у два етапи. На першому етапі він, за допомогою ансамблю моделей градієнтного бустингу, визначає «справжню» ціну акції. На другому етапі за формулою (Рів. 3.3) визначає, чи є компанія недооцінена, або навпаки – переоцінена. Компанії, для яких даний коефіцієнт буде перевищувати певну межу відсікання, будуть також відкидатися, як помилкові. Частка кожного з об'єктів інвестування визначається як відношення коефіцієнта об'єкта до суми коефіцієнтів усіх об'єктів, що були відібрані.

$$Coef = \frac{"Fair"\ Close}{Real\ Close} \tag{3.3}$$

Портфель, що був сформований розробленим методом на основі штучного інтелекту показав найкращі результати (Табл. 3.4). Сам портфель приніс найбільший прибуток. Крім того, усі найбільш збиткові об'єкти, як TSLA і ADBE, були відкинуті, а найбільш прибуткові, як ORCL чи WWE, навпаки були додані до портфелю. Також варто зауважити, що портфель є досить збалансованим. Частка кожного з об'єктів інвестування досить рівний, що робить портфель відносно стабільним, на відміну від портфелю моделі Марковіца максимального коефіцієнта Шарпа.

Таблиця 3.5 – Інвестиційний портфель розробленого методу на основі штучного інтелекту

| Компанія | Різниця ціни закриття | Коефіцієнт | Відсоток | Прибуток |
|---|---|---|---|---|
| NKLA | 0.29 | 1.52 | 8.96 | 2.60 |
| RACE | -0.71 | 1.03 | 6.08 | -4.32 |
| STLA | -4.77 | 1.82 | 10.72 | -51.13 |
| ORCL | 8.36 | 1.61 | 9.50 | 79.44 |
| PAYO | 2.08 | 2.34 | 13.80 | 29.71 |

Продовження таблиці 3.5

| Компанія | Різниця ціни закриття | Коефіцієнт | Відсоток | Прибуток |
|---|---|---|---|---|
| AMD | -1.33 | 1.67 | 9.81 | -13.05 |
| SONY | 2.76 | 1.63 | 9.61 | 26.53 |
| BOX | -1.00 | 1.46 | 8.62 | -8.62 |
| WWE | 12.68 | 1.22 | 7.20 | 91.35 |
| Сумарний прибуток | | 195.93 | | |

## 3.3 Порівняльний аналіз використаних методів

Після отримання результатів моделювання, можна зробити висновки, щодо ефективності розробленого методу формування інвестиційного портфелю, у порівнянні із класичними методами. Очевидно, що результати методів прийняття рішень при нечіткому відношенні переваги на множині альтернатив та мереж Байєса показали надто погані результати, тому не будемо додавати до порівняльної таблиці 3.6.

Таблиця 3.6 – Порівняння портфелів

| Компанія | Різн. ціни | Марковіца (мін. різн.) | | Марковіца (макс. коеф. Шарпа) | | Розроблений метод | |
|---|---|---|---|---|---|---|---|
| | | Відсоток | Приб. | Відсоток | Приб. | Відсоток | Приб. |
| TSLA | -64.24 | 1.4 | -89.94 | 3.9 | -250.54 | - | - |
| F | -0.62 | 0.6 | -0.37 | 15.5 | -9.61 | - | - |
| NKLA | 0.29 | 0.5 | 0.15 | 0.2 | 0.06 | 8.96 | 2.60 |
| RACE | -0.71 | 0.5 | -0.36 | 0.5 | -0.36 | 6.08 | -4.31 |
| STLA | -4.77 | 2.3 | -10.97 | 0.2 | -0.95 | 10.72 | -51.13 |
| TM | -8.15 | 33.3 | -271.40 | 2.7 | -22.01 | - | - |
| MSFT | 6.19 | 2.8 | 17.33 | 0.7 | 4.33 | - | - |
| ADBE | -87.1 | 1.4 | -121.94 | 2.1 | -181.91 | - | - |

Продовження таблиці 3.6

| Компанія | Різн. ціни | Марковіца (мін. різн.) | | Марковіца (макс. коеф. Шарпа) | | Розроблений метод | |
|---|---|---|---|---|---|---|---|
| | | Відсоток | Приб. | Відсоток | Приб. | Відсоток | Приб. |
| ORCL | 8.36 | 16.9 | 141.28 | 1.3 | 10.87 | 9.50 | 79.43 |
| PAYO | 2.08 | 0.8 | 1.66 | 0.3 | 0.62 | 13.80 | 29.71 |
| DBX | 2.41 | 3.4 | 8.19 | 0.5 | 1.21 | - | - |
| AAPL | 4.86 | 0.5 | 2.43 | 1.6 | 7.78 | - | - |
| NVDA | -37.17 | 0.9 | -33.45 | 2.7 | -100.36 | - | - |
| AMD | -1.33 | 2.5 | -3.33 | 1.6 | -2.13 | 9.81 | -13.05 |
| SONY | 2.76 | 3.0 | 8.28 | 0.1 | 0.28 | 9.61 | 26.53 |
| KOSS | 1.75 | 0.3 | 0.53 | 4.5 | 7.88 | - | - |
| BOX | -1.00 | 2.6 | -2.6 | 5.9 | -5.9 | 8.62 | -8.62 |
| NFLX | 5.66 | 1.1 | 6.23 | 0.1 | 0.57 | - | - |
| DIS | 7.22 | 2.6 | 18.77 | 1.3 | 9.39 | - | - |
| IMAX | 1.83 | 1.1 | 2.01 | 1.3 | 2.38 | - | - |
| WWE | 12.68 | 16.4 | 207.95 | 48.9 | 620.05 | 7.20 | 91.35 |
| CNK | 1.57 | 1.3 | 2.04 | 3.8 | 5.97 | - | - |
| WMG | 3.72 | 3.3 | 12.28 | 0.1 | 0.37 | - | - |
| Сумарний прибуток | | -105.67 | | 96.75 | | 195.93 | |

На суміжній таблиці гарно видно, що розроблений метод формування інвестиційного портфелю уміло відкинув найбільш неприбуткові об’єкти, які були включені в портфелі створені моделями Марковіціа мінімального ризику та максимального коефіцієнта Шарпа. Також гарно видно наступну перевагу розробленого методо – об’єкти його портфелю максимально рівнозначні, що його найбільш зваженим. Навіть у портфелі Марковіца мінімального ризику є об’єкт як ТМ, який явно відсотково переважає усі інші об’єкти портфелю.

Головним недоліком розробленого методу є його вимогливість до обчислювальних можливостей та до об’ємів вхідних даних. Метод використовує величезну кількість даних про різні компанії для визначення справжньої ціни акцій. Відповідно на це все потрібно витратити велику кількість часу. Той самий метод

Марковіца буде працювати значно швидше, оскільки йому не потрібно обробляти таку кількість даних. З іншого боку, навчена модель може сформувати портфель із нового набору компаній, дані про які будуть подані в модель, тим часом як моделі Марковіца потрібно починати формувати портфель з новими даними із самого початку заново. У такому випадку розроблена модель відпрацює швидше.

## 3.4 Платформа, мова програмування та бібліотеки

Перед початком розробки програмного продукту, важливим кроком є аналіз та вибір зручного середовища розробки, який буде відповідати поставленим цілям та ефективним для вирішення поставленої задачі. У випадку реалізації різноманітних моделей найбільшого розповсюдженим є використання мови програмування Python. Python – це об’єктно-орієнтована мова програмування високого рівня з строгою динамічною типізацією. Головними перевагами даної мови є її ефективність при обробці надвеликих масивів даних, а також наявність величезної кількості бібліотек, що допоможуть в обробці даних та реалізації маоделей, зокрема моделей штучного інтелекту. Для розробки даного продукту була використана версія Python 3.10.4.

Середовищем розробки було обрано платформу Anaconda. Anaconda – це дистрибутив для мов програмування Python та R. Також дана платформа допомагає із завантаженням різних бібліотек, що зокрема використовуються для аналізу та побудови моделей. Варто зазначити, що Anaconda має BSD ліцензію. Це дає можливість розробляти програмні продукти та використовувати їх на комерційній основі. Для розробки даного продукту була використана версія Anaconda 4.9.2.

Основними бібліотеками, що були використані у роботі, були:

— Numpy

— Pandas

— Sklearn

Основним функціоналом бібліотек Numpy та Pandas є робота із даними, представленими у різних формах. Також вони допомагають у більш зручному представленні та зручній роботі з різними типами даними, особливо з масивами та списками. Крім того, використання даних бібліотек забезпечує механізм конкретизації даний, що оптимізує код. Також, масиви, створені за допомогою Numpy займають менше місця у пам’яті, що також позитивно впливає на роботу програми.

Бібліотека Sklearn не може бути переоцінена для рішення задач моделювання. Дана бібліотека має відкритий вихідний код, що дає можливість модифікувати під свої потреби. У цій бібліотеці збережено величезну кількість моделей та способів оцінки цих моделей. Також, дана бібліотека має BSD ліцензію, що дає можливість використовувати програму на комерційній основі.

Крім зазначених вище бібліотек було використано інші допоміжні бібліотеки, що надають інструменти для виконання різноманітних математичних операцій та оптимізують роботу програми.

## 3.5 Опис модулів програмного продукту

Сам програмний продукт має складну структуру і складається з великої кількості файлів. Загалом, було створено дванадцять різних файлів, більшість з яких використовуються для розробленого методу формування інвестиційного портфелю на основі штучного інтелекту. Розглянемо ці файли більш детально.

Перш за все, варто згадати файл utils.ipynb, який містить в собі різноманітні додаткові функції, що використовуються в усіх інших файлах. Зокрема він мітить фільтри для коректного зчитування даних та функції для роботи із даними в цілому.

Перед початку роботи, важливим етапом є завантаження, зчитування та підготовка даних. За це все відповідають файли download.ipynb та data_loaders.ipynb. Файл download.ipynb відповідає за завантаження датасету із

середовища інтернет та запис його на пристрій. Файл data_loaders.ipynb займається зчитуванням даних з пристрою в оперативну пам’ять. За зчитування даних кожного з типів, базові, квартальні та щоденні дані, відповідає відповідний клас BaseData, QuarterlyData, DailyData.

Для розробленого методу формування інвестиційного портфелю потрібно специфічно підготувати дані. Цим займаються класи у файлах features.ipynb та targets.ipynb. Файл features.ipynb містить класи, які формують ознаки у потрібному форматі, а також комбінує ознаки різних типів, на основі яких має визначатися цільове значення. Для обробки цільового значення використовуються класи з targets.ipynb.

За формування моделі розробленого методу використовується одразу декілька файлів: metrics.ipynb, models.ipynb, pipeline.ipynb. Файл metrics.ipynb містить в собі функції, які можуть використовуватися моделлю градієнтного бустингу, як фінкція точності. У файлі models.ipynb знаходяться класи, які формують моделі з певними додатковими функціями. Зокрема, клас LogExpModel попередньо бере експоненту від цільового значення, GroupOOFModel групує дані перед навчанням моделі, а EnsembleModel допомагає у формуванні ансамблю моделей. У файлі  pipeline.ipynb знаходиться клас Pipeline, який допомагає у навчанні моделі та отриманні результатів визначення справжньої ціни компанії.

Самі експерименти, та результати знаходяться у файлах ai_model.ipynb, bayesian_model.ipynb, markovitz_model.ipynb, multicriterial_model.ipynb. У файлі ai_model.ipynb знаходяться експерименти для розробленого методу формування інвестиційного портфелю. Файл markovitz_model.ipynb представляє експерименти для моделі Марковіца мінімального ризику та максимального коефіцієнта Шарпа. Файл bayesian_model.ipynb репрезентує результати моделі Байєса, а multicriterial_model.ipynb – модель прийняття рішень при нечіткому відношенні переваги на множині недомінованих альтернатив.

## 3.6 Висновки до розділу 3

Результатом виконання роботи є розроблений програмний продукт для формування інвестиційного портфелю різним методами. Недоліком розробленої програми є значне споживання нею ресурсів пристрою користувача, зокрема для методів Байєса та розробленого методу прийняття рішень. Вирішити цю проблему можна шляхом оптимізації програми, а також шляхом завантаження програми на хмарні сервіси. Також доцільно використовувати паралельні обчислення у роботі програми.

Крім того, у результаті було розроблено метод формування інвестиційного портфелю на основі штучного інтелекту, а також проведено порівняння цього із класичними методами. Результат порівняння показав, що розроблений метод має вищу ефективність, ніж класичні методи. Сформований інвестиційний портфель приніс найбільший прибуток серед портфелів, а також був збалансованим, що показує його надійність. Покращити результати можна шляхом використання ще більшої кількості різноманітних метрик, та збільшення розміру датасету.

# РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, розробленого для вирішення задачі формування інвестиційного портфелю.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка 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 техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи формування інвестиційного портфелю. 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 аналізу та формування інвестиційного портфелю.

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 Обґрунтування функцій програмного продукту

Головна функція $F_0$ – розробка програмного продукту, який вирішує задачу формування інвестиційного портфелю. Беручи за основу цю функцію, можна виділити наступні:

$F_1$ – вибір мови програмування;

$F_2$ – вибір бібліотек, що допоможуть з реалізацією моделей;

$F_3$ – вибір середовища розробки.

Кожна з цих функцій має декілька варіантів реалізації:

Функція $F_1$:

а) Python

б) C#

Функція $F_2$:

а) Sklearn;

б) Accord.NET

Функція $F_3$:

а) Jupyter Notebook;

б) Visual Studio.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (Рис. 4.1).

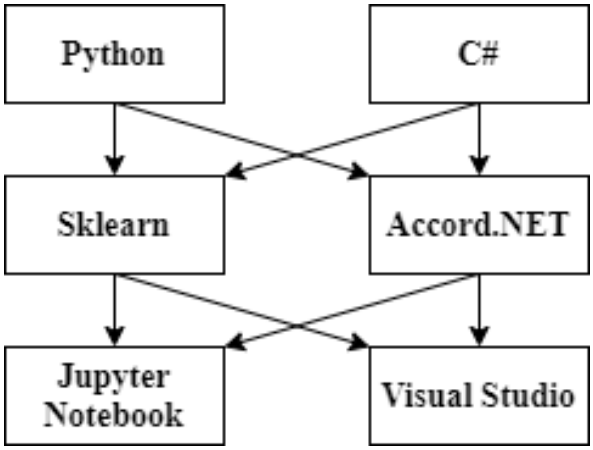


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину можливих варіанти основних функцій.

На основі цієї карти будуємо позитивно-негативну матрицю варіантів основних функцій (Таблиця 4.1). 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.

Таблиця 4.1 – Позитивно-негативна матриця

| Функції | Варіанти реалізації | Переваги | Недоліки |
|---|---|---|---|
| $F_1$ | *А* | Простота використання, наявність бібліотек у відкритому доступі | Швидкість обробки великої кількості даних |
| | *Б* | Швидкість компіляції та виконання коду | Розробка програми є більш трудомістким процесом |

Продовження таблиці 4.1

| Функції | Варіанти реалізації | Переваги | Недоліки |
|---|---|---|---|
| $F_2$ | *А* | Надійність роботи в складних проектах, популярність, легкість використання | Може використовуватися лише для невеликої кількості мов програмування |
| | *Б* | Надійність | Потребує додаткового часу для вивчення |
| $F_3$ | *А* | Зручність у використанні, легкість розробки | Складність розробки великих проектів |
| | *Б* | Ефективність при розробці великих проектів, надійність | Підтримує одночасно лише одну мову програмування |

Функція $F_1$*:*

Перевагу віддаємо швидкості вивчення, простоті використання та наявності стандартних бібліотек для обчислення. Для спрощення роботи по написанню коду та збільшення рентабельності розробки варіант Б має бути відкинутий.

Функція $F_2$*:*

Обидва варіанти можна використовувати в розробці, хоча для мови програмування Python зазвичай віддають перевагу варіанту А.

Функція $F_3$*:*

Віддаємо перевагу варіанту А в разі вибору мови програмування Python.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

$F_1$а – $F_2$а – $F_3$а

$F_1$а – $F_2$б – $F_3$а

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

## 4.3 Обґрунтування системи параметрів

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 обчислень та збереження даних;
- *X3* – час навчання даних;
- *X4* – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у таблиці 4.2.

Таблиця 4.2 – Основні параметри ПП

| Назва Параметра | Умовні позначення | Одиниці виміру | Значення параметра | | |
|---|---|---|---|---|---|
| | | | гірші | середні | кращі |
| Швидкодія мови програмування | X1 | оп/мс | 9000 | 15000 | 22000 |
| Об’єм пам’яті | X2 | Мб | 256 | 128 | 64 |
| Час попередньої обробки даних | X3 | мс | 600 | 240 | 90 |
| Потенційний об’єм програмного коду | X4 | кількість рядків коду | 25000 | 17000 | 12000 |

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

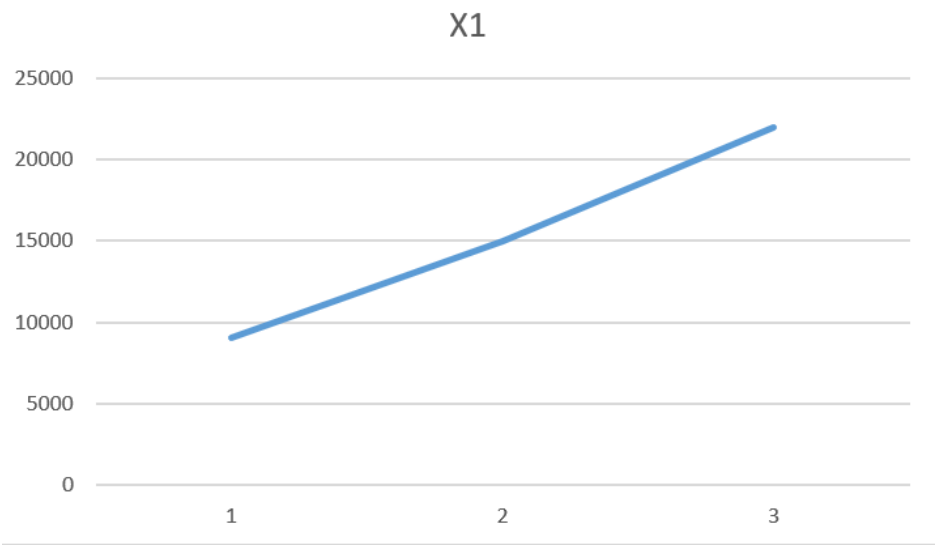


Рисунок 4.2 – X1, швидкодія мови програмування

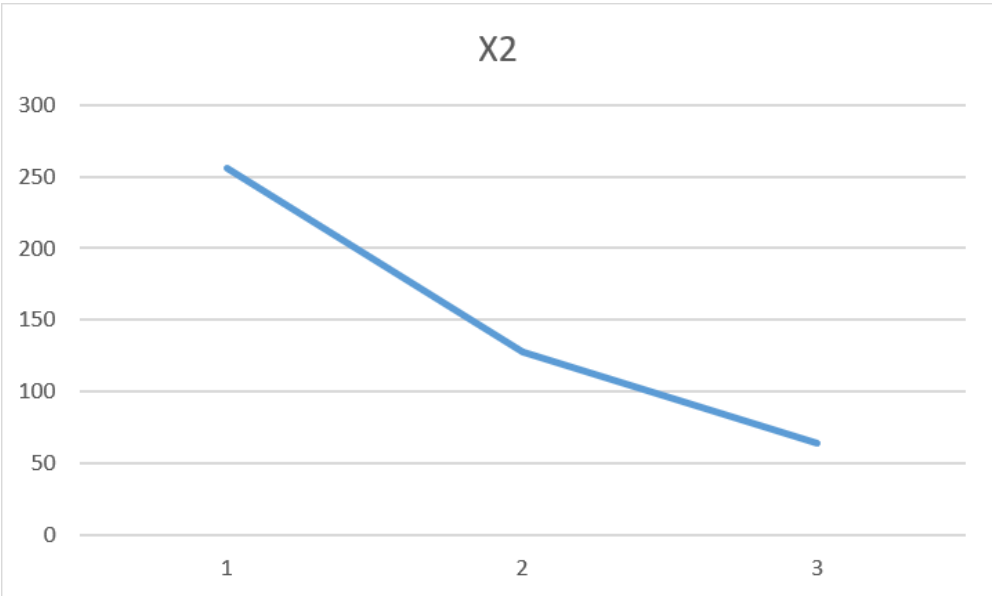


Рисунок 4.3 – X2, об’єм пам’яті

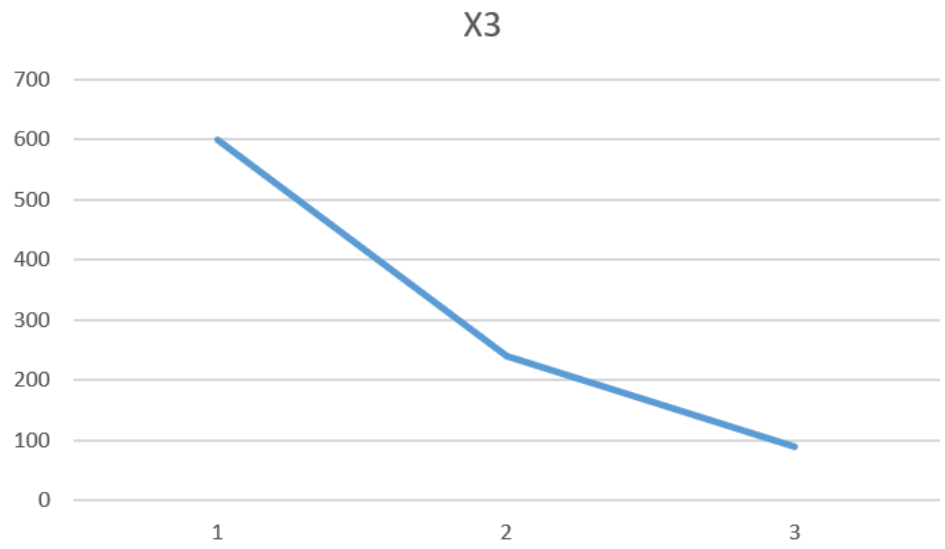


Рисунок 4.4 – X3, час попередньої обробки даних

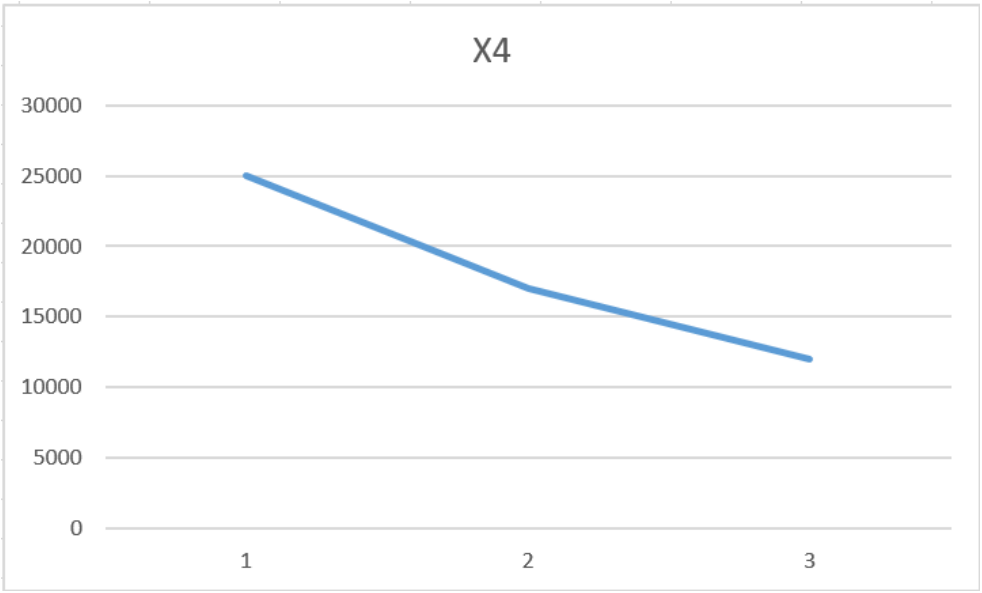


Рисунок 4.5 – X4, потенційний об’єм програмного коду

## 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.

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.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

| Позначення параметра | Назва параметра | Одиниці виміру | Ранг параметра за оцінкою експерта | | | | | | | Сума рангів $R_i$ | Відхи-лення $\Delta_i$ | $\Delta_i^2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | | |
| *X1* | Швидкодія мови програмування | Оп/мс | 3 | 5 | 4 | 5 | 3 | 4 | 5 | 29 | 4 | 16 |
| *X2* | Об’єм пам’яті | Mb | 1 | 3 | 3 | 2 | 1 | 2 | 2 | 14 | -11 | 121 |
| *X3* | Час попередньої обробки даних | мс | 5 | 3 | 5 | 5 | 4 | 5 | 3 | 30 | 5 | 25 |
| *X4* | Потенційний об’єм програмного коду | Кількість рядків коду | 3 | 3 | 5 | 3 | 4 | 4 | 5 | 27 | 2 | 4 |

Продовження таблиці 4.3

| Позначення параметра | Назва параметра | Одиниці виміру | Ранг параметра за оцінкою експерта | | | | | | | Сума рангів $R_i$ | Відхи-лення $\Delta_i$ | $\Delta_i^2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | | |
| | Разом | | 11 | 14 | 17 | 15 | 12 | 15 | 16 | 100 | 0 | 166 |

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 100, \tag{4.1}$$

де $N$ – число експертів;

$n$ – кількість параметрів.

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 25 \tag{4.2}$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T. \tag{4.3}$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 166. \tag{4.4}$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 136}{7^2(4^3 - 4)} = 0.68 > W_k = 0.67. \quad (4.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0.67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів.

| Параметри | Експерти | | | | | | | Кінцева оцінка | Числове значення |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | |
| X1 і X2 | < | > | < | > | = | < | > | = | 1.0 |
| X1 і X3 | < | = | < | = | < | < | > | < | 0.5 |
| X1 і X4 | = | = | < | > | < | > | > | > | 1.5 |
| X2 і X3 | < | > | < | > | > | > | = | > | 1.5 |
| 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 $K_{ві}$ за наступними формулами:

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

$$b_i = \sum_{j=1}^{N} a_{ij} \tag{4.8}$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

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

$$b_i' = \sum_{j=1}^{N} a_{ij} b_j \tag{4.10}$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, 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 | 1.0 | 1.0 | 0.5 | 1.5 | 4,0 | 0.26 | 16,0 | 0.27 | 64,0 | 0.26 |
| X2 | 0.5 | 1.0 | 1.5 | 0.5 | 3.5 | 0.22 | 12.25 | 0.20 | 42.88 | 0.18 |
| X3 | 1.5 | 0.5 | 1.0 | 1.5 | 4,5 | 0.29 | 20.25 | 0.33 | 91.13 | 0.37 |
| X4 | 0.5 | 1.5 | 0.5 | 1.0 | 3.5 | 0.23 | 12.25 | 0.20 | 42.88 | 0.18 |
| Всього: | | | | | 15,5 | 1 | 60.75 | 1 | 240.88 | 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 чином.

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

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

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 | 9000 | 7 | 0.26 | 1.82 |
| F2 | А | X2 | 64 | 4 | 0.18 | 0.72 |
| | Б | X2 | 128 | 3 | 0.18 | 0.54 |
| F3 | А | X3 | 240 | 5 | 0.37 | 1.85 |
| | А | X4 | 17000 | 6 | 0.18 | 1.08 |

За даними з таблиці 4.6 за формулою:

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

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 1.82 + 0.72 + 1.85 + 1.08 = 5.47,$$

$$K_{K2} = 1.82 + 0.54 + 1.85 + 1.08 = 5.29.$$

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

## 4.6 Економічний аналіз варіантів розробки ПП

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

$$Т_О = Т_Р \cdot К_П \cdot К_{СК} \cdot К_М \cdot К_{СТ} \cdot К_{СТ.М}, \quad (4.13)$$

де $Т_Р$ – трудомісткість розробки ПП;

$К_П$ – поправочний коефіцієнт;

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

$К_М$ – коефіцієнт рівня мови програмування;

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

$К_{СТ.М}$ – 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: $Т_Р$ = 120 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 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:

$$Т_1 = 120 \cdot 1.8 \cdot 0.9 = 194.4 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =48 людино-днів, $К_П$ =0.95, $К_{СК}$ = 1, $К_{СТ}$ =0.8:

$$Т_2 = 48 \cdot 0.95 \cdot 0.8 = 36.48 \text{ людино-днів.}$$

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

$T_I$ = (194.4 + 36.48 + 7.6 + 36.48) · 8 = 2199.68 людино-годин.

$T_{II}$ = (194.4 + 36.48 + 9.41 + 36.48) · 8 = 2214.16 людино-годин.

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 37000 грн., один аналітик в області даних з окладом 48500. Визначимо середню зарплату за годину за формулою:

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

де М – місячний оклад працівників;

$T_m$ – кількість робочих днів тиждень;

$t$ – кількість робочих годин в день.

$$С_ч = \frac{37000 + 37000 + 48500}{3 \cdot 21 \cdot 8} = 243.06 \text{ грн.} \quad (4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

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

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. $С_{ЗП}$ = 243.06 · 2199.68 · 1.2 = 641585.10 грн.

II. $С_{ЗП}$ = 243.06 · 2214.16 · 1.2 = 645808,50 грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 641585.10 \cdot 0.22 = 141148{,}70$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 645808{,}50 \cdot 0.22 = 142077.90$ грн.

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

Так як одна ЕОМ обслуговує одного програміста з окладом 37000 грн., з коефіцієнтом зайнятості 0.2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 37000 \cdot 0.2 = 88800 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 88800 \cdot (1 + 0.2) = 106560 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 106560 \cdot 0.22 = 23443.2 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 60000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 60000 = 17250 \text{ грн.},$$

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:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 40000 \cdot 0.05 = 3450 \text{ грн.},$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 102 – 14 – 16) \cdot 8 \cdot 0.9 =$$

$$= 1677.6 \text{ годин},$$

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:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1677.6 \cdot 0.3 \cdot 3.51 \cdot 2 = 3533.03 \text{ грн.},$$

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:

$$С_Н = Ц_{ПР} \cdot 0.67 = 60000 \cdot 0.67 = 40200 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$C_{ЕКС} = C_{ЗП} + C_{ВІД} + C_{А} + C_{Р} + C_{ЕЛ} + C_{Н}, \quad (4.17)$$

$C_{ЕКС}$ = 106560+ 23443.2 + 17250 + 3450 +3533.03+40200 = 194436,23 грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{М\text{-}Г} = C_{ЕКС} / T_{ЕФ} = 194436{,}23 / 1677.6 = 115.90 \text{ грн/год.}$$

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:

$$C_{М} = C_{М-Г} \cdot T, \quad (4.18)$$

I. $C_М = 115.90 \cdot 2199.68 = 254942.91$ грн.

II. $C_М = 115.90 \cdot 2214.16 = 256621.14$ грн.

Накладні витрати складають 67% від заробітної плати:

$$C_{Н} = C_{ЗП} \cdot 0.67, \quad (4.19)$$

I. $C_Н = 641585.10 \cdot 0.67 = 429862.02$ грн.

II. $C_Н = 645808{,}50 \cdot 0.67 = 432691.70$ грн.

Отже, вартість розробки ПП за варіантами становить:

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

I. $С_{ПП}$ = 641585.10 + 141148,70 + 254942.91 + 429862.02 = 1467538.73 грн.

II. $С_{ПП}$ = 645808.50 + 142077,90 + 256621.14 + 432691.70 = 1477199.24 грн.

## 4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР j} = К_{Кj} / С_{Фj}, \quad (4.21)$$

$К_{ТЕР1}$ = 5.47 / 1467538.73 = 3,73 · $10^{-6}$,

$К_{ТЕР2}$ = 5.29 / 1477199.24 = 3,58 · $10^{-6}$.

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1}$= 3,73 · $10^{-6}$.

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. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР}$ = 3,73 · $10^{-6}$.

## 4.8 Висновки до розділу 4

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. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР}= 3,73 \cdot 10^{-6}$.

Цей варіант реалізації програмного продукту має такі параметри:

— мова програмування – Python;

— бібліотека регресійного аналізу – Sklearn;

— середовище розробки – Jupyter Notebook.

Спосіб розробки саме з допомогою цих технічних засобів є найбільш оптимальним і комбінує у собі високу швидкість та зручність розробки, а також гарну якість кінцевого продукту.

# ВИСНОВКИ

У бакалаврській дипломній роботі було сформульовано та розв’язано задачу формування інвестиційного портфелю  Було проаналізовано основні існуючі методи до вирішення цієї проблеми.

— Здійснено аналіз основних методів для вирішення задачі формування та аналізу інвестиційного портфелю, розглянуто їх математичні основи, основні характеристики, а також переваги та недоліки. Розроблено метод формування інвестиційного портфелю на основі штучного інтелекту, проведено порівняння його ефективності із класичними методами.

— Було створено програмний продукт, із використанням мови програмування високого рівня Python.

— Важливість створеного продукту полягає у тому, що він формує інвесиційний портфель із вищим рівнем прибутковості та нижчим рівнем ризикованості. Розроблений програмний продукт може використовуватися інветорами та аналітиками для вибору потенційних об’єктів інвестування.

— Ефективність формування інвестиційного портфелю розробленою моделлю на основі штучного інтелекту може бути підвищення шляхом збільшення об’єму вхідних даних, а також використанням додаткових метрик, які описують інвестиційні об’єкти. Також швидкодія програми може бути підвищена шляхом додавання можливості паралельних обчислень у програмі, а також завантаженням програмного продукту на хмарний сервіс.

# ПЕРЕЛІК ПОСИЛАНЬ


1. Денисенко М., Федоренко В., Воронкова Т. Інвестування: підр. Алерта 2012. 272 с.
2. Замковий О.І. Портфельні теорії інвестування. метод.-наук. посіб. Дніпро: НТУ «ДП», 2020. – 70 с.
3. Шевченко О. Г., Майорова Т. В., Юркевич О. М., Урванцева С. В. [та ін.] / Портфельне інвестування : підручник ; за наук. ред. О. Г. Шевченко, Т. В. Майорової. — К.: КНЕУ, 2010. — 407 с.
4. Зайченко Ю.П. Дослідження операцій. Підручник. Сьоме видання, перероблене та доповнене. – К.:Видавничий Дім «Слово», 2006. – 816 с.
5. Скрипниченко М.В.Портфельные инвестиции: Учебное пособие. - СПб: Университет ИТМО, 2016 – 40 с.
6. Шарп У., Инвестиции. / У. Шарп, Г. Александер, Дж. Бейли. – М.: ИНФРА-М, 1997. – 172 с.
7. Бідюк П.І., Коршевнюк Л.О. Проектування комп'ютерних інформаційних систем підтримки прийняття рішень: Навчальний посібник. — Київ: ННК „ІПСА" НТУУ „КПІ", 2010. – 340 с.
8. Galkina S. Optimal Portfolio Selection Using Machine Learning Techniques / International Journal of Open Information Technologies ISSN: 2307-8162 vol. 2, no. 6, 2014. P. 14-19.
9. Тапсахалова Ф.М.-Г Инвестиции: учеб.-метод пособие. москва: 2010. – 173 с.
10. Бідюк П.І., Коршевнюк Л.О., Кузнєцова Н.В. Моделі і методи прикладної статистики: навч. посіб. з грифом МОН України. Київ: НТУУ «КПІ», 2014. 722 с.
11. Бидюк П.И., Кузнецова Н.В. Прогнозирование волатильности финансовых процессов с помощью моделей условной дисперсии. Проблемы управления и информатики. 2014. С. 47-54.

# ДОДАТОК А

utils.ipynb

```
import json
import os
import hashlib
import pandas as pd
import numpy as np
from copy import deepcopy
def get_na_values():
    return["",
        "#N/A",
        "#N/A N/A",
        "#NA",
        "-1.#IND",
        "-1.#QNAN",
        "-NaN",
        "-nan",
        "1.#IND",
        "1.#QNAN",
        "<NA>",
        "N/A",
        #"NA",
       "NULL",
        "NaN",
        "n/a",
        "nan",
        "null"]
def check_create_folder(file_path):
    if '/' in file_path:
```

```
        folder_path = '/'.join(file_path.split('/')[:-1])
        if not os.path.exists(folder_path):
            os.makedirs(folder_path)
def save_json(file_path, data):
    check_create_folder(file_path)
    with open(file_path, "w") as write_file:
        json.dump(data, write_file, ensure_ascii=False)
def load_json(path):
    with open(path, "r") as read_file:
        in_data = json.load(read_file)
    return in_data
def copy_repeat(data, cnt: int):
    result = [deepcopy(data) for _ in range(cnt)]
    return result
def int_hash_of_str(text:str):
    return int(hashlib.md5(text.encode('utf-8')).hexdigest()[:8], 16)
def nan_mask(arr):
    return pd.isna(arr) == False
def bound_filter_foo_gen(min_bound, max_bound):
    def foo(arr):
        result = np.isnan(arr) == False
        if min_bound is not None:
            result = result * (arr > min_bound)
        if max_bound is not None:
            result = result * (arr < max_bound)
        return result
    return foo
def get_quarter_idx(date: np.datetime64):
    date = np.datetime64(date)
    year = date.astype(object).year
```

```
    month = date.astype(object).month
    bounds = np.array([3, 6, 9, 12])
    idx = np.where(bounds >= month)[0][0] + 1
    q_idx = '{}q{}'.format(year, idx)
    return q_idx


        downloader.ipynb
import numpy as np
import time
import requests
import os
import copy
import yfinance as yf
from math import isnan
from ipynb.fs.full.data_loaders import BaseData
from ipynb.fs.full.utils import save_json
class DownloadRepository:
    def __init__(self, retry_cnt=10, sleep_time=1.4):
        self.secret_key = 't*****'
        self.retry_cnt = retry_cnt
        self.sleep_time = sleep_time
        self._save_dirpath = None
        self._base_url_route = None
    def _form_quandl_url(self, route):
        url = "https://www.quandl.com/api/v3/{}&api_key={}".format(
                route,
                self.secret_key)
        return url
    def _batch_ticker_download(self, tickers):
        time.sleep(np.random.uniform(0, self.sleep_time))
```

```
    url = self._base_url_route.format(ticker=','.join(tickers))
    url = self._form_quandl_url(url)
    for _ in range(10):
        try:
            response = requests.get(url)
            break
        except:
            time.sleep(np.random.uniform(0, self.sleep_time))
    if response.status_code != 200:
        print('Error downloading tickers: {}'.format(tickers))
        return
    data = response.json()
    datatable_data = np.array(data['datatable']['data'])
    if len(datatable_data) == 0:
        return
    ticker_seq = np.array([str(x[0]) for x in datatable_data])
    curr_data = copy.deepcopy(data)
    curr_data['datatable']['data'] = []
    for ticker in tickers:
        curr_datatable_data = datatable_data[ticker_seq == ticker].tolist()
        curr_data['datatable']['data'] = curr_datatable_data
        save_filepath = '{}/{}.json'.format(self._save_dirpath, ticker)
        save_json(save_filepath, curr_data)
def ticker_download(self,
                    base_url_route,
                    tickers,
                    save_dirpath: str,
                    skip_exists: bool=False,
                    batch_size: int=5):
    self._save_dirpath = save_dirpath
```

```
    self._base_url_route = base_url_route
    os.makedirs(save_dirpath, exist_ok=True)
    if skip_exists:
        exist_tickers = [x.split('.')[0] for x in os.listdir(save_dirpath)]
        tickers = list(set(tickers).difference(set(exist_tickers)))
    batches = [tickers[k:k+batch_size]
                   for k in range(0, len(tickers), batch_size)]
    for batch in batches:
        self._batch_ticker_download(batch)
def single_download(self, base_url_route, save_filepath):
    if '?' not in base_url_route:
        base_url_route = base_url_route + '?'
    url = self._form_quandl_url(base_url_route)
    for _ in range(10):
        try:
            response = requests.get(url)
            if response.status_code == 200:
                break
        except:
            print("request error")
            time.sleep(np.random.uniform(0, 2))
    if response.status_code == 200:
        data = response.json()
        save_json(save_filepath, data)
    else:
        print('single download error')
def zip_download(self, base_url_route, save_filepath):
    url = self._form_quandl_url(base_url_route)
    info_response = requests.get(url)
    zip_link = info_response.json()['datatable_bulk_download']['file']['link']
```

```
    data_response = requests.get(zip_link)
    if '/' in save_filepath:
        folder_path = '/'.join(save_filepath.split('/')[:-1])
        if not os.path.exists(folder_path):
            os.makedirs(folder_path)
    with open(save_filepath, 'wb') as f:
        f.write(data_response.content)
def close_daily_data_download(self, tickers, save_dirpath):
    for ticker in tickers:
        data = yf.download([ticker],period='6y')
        data = data.loc[:,['Close']]
        data.reset_index(inplace=True)
        data['ticker'] = ticker
        path = '{}/{}.csv'.format(save_dirpath, ticker)
        data = data.rename(columns = {'Date':'date', 'Close':'close'})
        data.to_csv(path, index=False)
    def download(data_path: str='data'):
downloader = DownloadRepository(sleep_time=0.8)
print('Start SF1 base downloading: {}'.format(
        str(np.datetime64(int(time.time() * 1000), 'ms'))))
downloader.zip_download(
        base_url_route='datatables/SHARADAR/TICKERS?qopts.export=true',
        save_filepath='{}/tickers.zip'.format(data_path))
print('Start SF1 snp500 downloading: {}'.format(
        str(np.datetime64(int(time.time() * 1000), 'ms'))))
downloader.zip_download(
        base_url_route='datatables/SHARADAR/SP500?qopts.export=true',
        save_filepath='{}/snp500.zip'.format(data_path))
base_df = BaseData(data_path).load()
tickers = base_df['ticker'].unique().tolist()
```

```
    print('Start SF1 quarterly downloading: {}'.format(
            str(np.datetime64(int(time.time() * 1000), 'ms'))))
    downloader.ticker_download(
            base_url_route='datatables/SHARADAR/SF1?ticker={ticker}',
            tickers=tickers,
            save_dirpath='{}/core_fundamental'.format(data_path),
            skip_exists=False,
            batch_size=2)
    print('Start SF1 daily downloading: {}'.format(
            str(np.datetime64(int(time.time() * 1000), 'ms'))))
    downloader.ticker_download(
            base_url_route='datatables/SHARADAR/DAILY?ticker={ticker}',
            tickers=tickers,
            save_dirpath='{}/daily'.format(data_path),
            skip_exists=False,
            batch_size=2)
    print('Start Close daily data downloading: {}'.format(
        str(np.datetime64(int(time.time() * 1000), 'ms'))))
    downloader.close_daily_data_download(tickers=tickers,
            save_dirpath='{}/close'.format(data_path))
download()


        data_loaders.ipynb
import os
import numpy as np
import pandas as pd
from typing import Optional, List
from ipynb.fs.full.utils import get_na_values, load_json
def _load_df(json_path: str) -> pd.DataFrame:
    data = load_json(json_path)
```

```
    df = pd.DataFrame(data['datatable']['data'])
    if len(df) == 0:
        columns = [x['name'] for x in data['datatable']['columns']]
        df = pd.DataFrame(columns=columns, dtype=object)
    else:
        df.columns = [x['name'] for x in data['datatable']['columns']]
    df = df.infer_objects()
    return df
def translate_currency(df: pd.DataFrame, columns: Optional[List[str]] = None):
    if columns is None:
        no_translate_cols = ['ticker', 'dimension', 'calendardate', 'datekey',
                             'reportperiod', 'date', 'marketcap', 'lastupdated']
        no_translate_cols += [x for x in df.columns if 'usd' in x]
        columns = [x for x in df.columns if x not in no_translate_cols]
    df = df.infer_objects()
    usd_cols = ['equityusd','epsusd','revenueusd','netinccmnusd',
            'cashnequsd','debtusd','ebitusd','ebitdausd']
    usd_cols = list(set(df.columns).intersection(set(usd_cols)))
    assert len(usd_cols) > 0
    rows = np.array([(df[col.replace('usd', '')] / df[col]).values
                     for col in usd_cols])
    df['trans_currency'] = np.nanmax(rows, axis=0).astype('float32')
    df['trans_currency'] = df['trans_currency'].interpolate()
    for col in columns:
        df[col] = df[col] / df['trans_currency']
    return df
class BaseData:
    def __init__(self, data_path: Optional[str]='data'):
        self.data_path = data_path
```

```
    def load(self, index: Optional[List[str]]=None) -> pd.DataFrame:
        path = '{}/tickers.zip'.format(self.data_path)
        tickers_df = pd.read_csv(path, na_values=get_na_values, keep_default_na=False)
        tickers_df = tickers_df[tickers_df['table'] == 'SF1']
        if index is not None:
            tmp = pd.DataFrame()
            tmp['ticker'] = index
            tmp['flag'] = True
            tickers_df = pd.merge(tickers_df, tmp, on='ticker', how='left')
            tickers_df['flag'] = tickers_df['flag'].fillna(False)
            tickers_df = tickers_df[tickers_df['flag']]
            del tickers_df['flag']
        return tickers_df.reset_index(drop=True)
    def existing_index(self):
        path = '{}/tickers.zip'.format(self.data_path)
        tickers_df = pd.read_csv(path, na_values=get_na_values, keep_default_na=False)
        tickers_df = tickers_df[tickers_df['table'] == 'SF1']
        index = list(tickers_df['ticker'].unique())
        return index
class QuarterlyData:
    def __init__(self,
                 data_path: Optional[str]='data',
                 dimension: Optional[str]='ARQ',
                 date_separator: Optional[str]=None):
        self.data_path = data_path
        self.dimension = dimension
        self.date_separator = date_separator
```

```
def load_train(self, index: List[str]) -> pd.DataFrame:
    result = self._load(index)
    if result is None:
        return None
    if (self.date_separator is None):
        return result
    return result[result['date'] <= self.date_separator]
def load_test(self, index: List[str]) -> pd.DataFrame:
    result = self._load(index)
    if result is None:
        return None
    if (self.date_separator is None):
        return result
    return result[result['date'] >= self.date_separator]
def existing_index(self):
    dir_path = '{}/core_fundamental'.format(self.data_path)
    index = [x.split('.json')[0] for x in os.listdir(dir_path)]
    return index
def _load(self, index: List[str]):
    result = []
    for ticker in index:
        path = '{}/core_fundamental/{}.json'.format(self.data_path, ticker)
        if not os.path.exists(path):
            continue
        close_data_path = '{}/close/{}.csv'.format(self.data_path, ticker)
        if not os.path.exists(close_data_path):
            continue
        df = _load_df(path)
        df = df[df['dimension'] == self.dimension]
```

```
            df['date'] = df['datekey']
            df = df.sort_values('date', ascending=False)
            close_daily_df = pd.read_csv(close_data_path, na_values=get_na_values,
keep_default_na=False)
            df = df.merge(close_daily_df, how = 'inner', on = ['ticker', 'date'])
            result.append(df)
        if len(result) == 0:
            return None
        result = pd.concat(result, axis=0).reset_index(drop=True)
        result = result.infer_objects()
        result['date'] = result['date'].astype(np.datetime64)
        result['marketcap'] = result['marketcap'].astype(float)
        result['close'] = result['close'].astype(float)
        return result
class DailyData():
    def __init__(self,
                 data_path: Optional[str]='data',
                 date_separator: Optional[str]=None):
        self.data_path = data_path
        self.date_separator = date_separator
    def load_train(self, index: List[str]):
        result = self._load(index)
        if result is None:
            return None
        if (self.date_separator is None):
            return result
        return result[result['date'] <= self.date_separator]
    def load_test(self, index: List[str]) -> pd.DataFrame:
        result = self._load(index)
        if result is None:
```

```
            return None
        if (self.date_separator is None):
            return result
        return result[result['date'] >= self.date_separator]
    def existing_index(self):
        dir_path = '{}/daily'.format(self.data_path)
        index = [x.split('.json')[0] for x in os.listdir(dir_path)]
        return index
    def _load(self, index: List[str]) -> pd.DataFrame:
        result = []
        for ticker in index:
            path = '{}/daily/{}.json'.format(self.data_path, ticker)
            if not os.path.exists(path):
                continue
            close_data_path = '{}/close/{}.csv'.format(self.data_path, ticker)
            if not os.path.exists(close_data_path):
                continue
            daily_df = _load_df(path)
            close_daily_df = pd.read_csv(close_data_path, na_values=get_na_values,
keep_default_na=False)
            df = daily_df.merge(close_daily_df, how = 'inner', on = ['ticker', 'date'])
            result.append(df)
        if len(result) == 0:
            return None
        result = pd.concat(result, axis=0).reset_index(drop=True)
        result = result.infer_objects()
        result['marketcap'] = result['marketcap'].astype(float)
        result['close'] = result['close'].astype(float)
        result['pe'] = result['close'].astype(float)
        result['date'] = result['date'].astype(np.datetime64)
```

```
        return result
```

metrics.ipynb

```
import pandas as pd
import numpy as np
def median_absolute_relative_error(gt, pred):
    mask = gt != 0
    pred = pred[mask]
    gt = gt[mask]
    vals = np.abs((gt - pred) / gt)
    vals = vals[~pd.isna(vals)]
    return np.median(vals)
def mean_absolute_relative_error(gt, pred):
    mask = gt != 0
    pred = pred[mask]
    gt = gt[mask]
    vals = np.abs((gt - pred) / gt)
    vals = vals[~np.isnan(vals)]
    return np.mean(vals)
```

features.ipynb

```
import copy
import numpy as np
import pandas as pd
from typing import Union, List, Dict, Callable
from multiprocessing import Pool, cpu_count
from ipynb.fs.full.utils import int_hash_of_str
np.seterr(divide='ignore', invalid='ignore')
def calc_series_stats(series: Union[List[float], np.array],
                      stats: Dict[str, Callable]={'mean': np.mean,
```

```
                                    'max': np.max,
                                    'min': np.min,
                                    'std': np.std},
                name_prefix: str='',
                norm: bool=False) -> Dict[str, float]:
    series = np.array(series).astype('float')
    series = series[~np.isnan(series)]
    series = list(series)
    if len(series) == 0:
        series = np.array([np.nan])
            result = {'{}_{}'.format(name_prefix, key): stats[key](series)
              for key in stats}
    if norm:
        result = {key: result[key] / np.abs(series[0]) for key in result}
        return result
class QuarterlyFeatures:
    def __init__(self,
                 data_key: str,
                 columns: List[str],
                 quarter_counts: List[int]=[2, 4, 10],
                 max_back_quarter: int=10,
                 min_back_quarter: int=0,
                 stats: Dict[str, Callable]={'mean': np.mean,
                                             'max': np.max,
                                             'min': np.min,
                                             'std': np.std},
                 calc_stats_on_diffs: bool=True,
                 data_preprocessing: Callable=None,
                 n_jobs: int=cpu_count()):
        self.data_key = data_key
```

```
    self.columns = columns
    self.quarter_counts = quarter_counts
    self.max_back_quarter = max_back_quarter
    self.min_back_quarter = min_back_quarter
    self.stats = stats
    self.calc_stats_on_diffs = calc_stats_on_diffs
    self.data_preprocessing = data_preprocessing
    self.n_jobs = n_jobs
    self._data_loader = None
def _calc_series_feats(self, data: pd.DataFrame,
                       str_prefix: str='') -> Dict[str, float]:
    result = {}
    for quarter_cnt in self.quarter_counts:
        for col in self.columns:
            series = data[col].values[:quarter_cnt][::-1].astype('float')
            name_prefix = 'quarter{}_{}'.format(quarter_cnt, col)
            feats = calc_series_stats(series=series,
                                      stats=self.stats,
                                      name_prefix=name_prefix)
            result.update(feats)
            if self.calc_stats_on_diffs:
                diff_feats = calc_series_stats(series=np.diff(series),
                                               stats=self.stats,
                                               name_prefix='{}_diff'\
                                                .format(name_prefix))
                result.update(diff_feats)
    return result
def _single_ticker(self, ticker:str) -> List[Dict[str, float]]:
    result = []
    quarterly_data = self._data_loader.load_train([ticker])
```

```
        if quarterly_data is None:
            return result
        if self.data_preprocessing is not None:
            quarterly_data = self.data_preprocessing(quarterly_data)
        max_back_quarter = min(self.max_back_quarter, len(quarterly_data) - 1)
        min_back_quarter = min(self.min_back_quarter, len(quarterly_data) - 1)
        assert min_back_quarter <= max_back_quarter
        for back_quarter in range(min_back_quarter, max_back_quarter):
            curr_data = quarterly_data[back_quarter:]
            feats = {
                'ticker': ticker,
                'date': curr_data['date'].values[0],
            }
            series_feats = self._calc_series_feats(curr_data)
            feats.update(series_feats)
            result.append(feats)
        return result
    def calculate(self, data: Dict, index: List[str]) -> pd.DataFrame:
        self._data_loader = data[self.data_key]
        with Pool(self.n_jobs) as p:
            X = []
            for ticker_feats_arr in p.imap(self._single_ticker, index):
                X.extend(ticker_feats_arr)
        if(len(X) == 0):
            return pd.DataFrame(X, columns=['ticker', 'date']).set_index(['ticker', 'date'])
        X = pd.DataFrame(X).set_index(['ticker', 'date'])
        return X
class HashingEncoder:
    def transform(self, vals):
        result = [int_hash_of_str(str(x)) for x in vals]
```

```
        return result
class BaseCompanyFeatures:
    def __init__(self,
                 data_key:str,
                 cat_columns:List[str]):
        self.data_key = data_key
        self.cat_columns = cat_columns
        self.he = HashingEncoder()
    def calculate(self, data: Dict, index: List[str]) -> pd.DataFrame:
        base_df = data[self.data_key].load(index)
        for col in self.cat_columns:
            base_df[col] = base_df[col].fillna('None')
            base_df[col] = self.he.transform(base_df[col])
        result = pd.DataFrame()
        result['ticker'] = index
        result = pd.merge(result, base_df[['ticker'] + self.cat_columns],
                          on='ticker', how='left')
        result = result.set_index(['ticker'])
        return result
class DailyAggQuarterFeatures:
    def __init__(self,
                 daily_data_key: str,
                 quarterly_data_key: str,
                 columns: List[str],
                 agg_day_counts: List[Union[int, np.timedelta64]] = [100, 200],
                 max_back_quarter: int=10,
                 min_back_quarter: int=0,
                 daily_index=None,
                 stats: Dict[str, Callable]={'mean': np.mean,
                                             'max': np.max,
```

```
                                'min': np.min,
                                'std': np.std},
            calc_stats_on_diffs: bool=True,
            norm: bool=True,
            n_jobs: int=cpu_count()):
    self.daily_data_key = daily_data_key
    self.quarterly_data_key = quarterly_data_key
    self.columns = columns
    self.agg_day_counts = agg_day_counts
    self.max_back_quarter = max_back_quarter
    self.min_back_quarter = min_back_quarter
    self.daily_index = daily_index
    self.stats = stats
    self.norm = True
    self.n_jobs = n_jobs
    self.calc_stats_on_diffs = calc_stats_on_diffs
    self._daily_data_loader = None
    self._quarterly_data_loader = None
def _calc_series_feats(self, data: pd.DataFrame,
                       str_prefix: str='') -> Dict[str, float]:
    result = {}
    if len(data) == 0:
        return result
    for day_cnt in self.agg_day_counts:
        if type(day_cnt) == int:
            curr_data = data[:day_cnt]
        elif type(day_cnt) == np.timedelta64:
            daily_dates = data['date'].values
            curr_data = data[daily_dates > daily_dates[0] - day_cnt]
```

```
            for col in self.columns:
                series = curr_data[col].values[::-1].astype('float')
                name_prefix = '{}_days{}_{}'.format(str_prefix, str(day_cnt), col)
                feats = calc_series_stats(series=series,
                                          stats=self.stats,
                                          name_prefix=name_prefix,
                                          norm=self.norm)
                result.update(feats)
                if self.calc_stats_on_diffs:
                    diff_feats = calc_series_stats(series=np.diff(series),
                                                   stats=self.stats,
                                                   name_prefix='{}_diff'\
                                                    .format(name_prefix))
                    result.update(diff_feats)
        return result
    def _single_ticker(self, ticker: str) -> List[Dict[str, float]]:
        result = []
        quarterly_data = self._quarterly_data_loader.load_train([ticker])
        if quarterly_data is None:
            return result
        daily_data = copy.deepcopy(self.daily_data)
        if self.daily_index is None:
            daily_data[''] = self._daily_data_loader.load_train([ticker])
        max_back_quarter = min(self.max_back_quarter, len(quarterly_data) - 1)
        min_back_quarter = min(self.min_back_quarter, len(quarterly_data) - 1)
        assert min_back_quarter <= max_back_quarter
        for back_quarter in range(min_back_quarter, max_back_quarter):
            curr_data = quarterly_data[back_quarter:]
            curr_date = np.datetime64(curr_data['date'].values[0])
```

```
            feats = {}
            feats['ticker'] = ticker
            feats['date'] = curr_date
            for idx in daily_data.keys():
                if daily_data[idx] is not None:
                    daily_dates = daily_data[idx]['date'].values
                else:
                    continue
                curr_daily_data = daily_data[idx][daily_dates < curr_date]
                daily_feats = self._calc_series_feats(curr_daily_data, idx)
                feats.update(daily_feats)
            result.append(feats)
        return result
    def calculate(self, data: Dict, index: List[str]) -> pd.DataFrame:
        self._daily_data_loader = data[self.daily_data_key]
        self._quarterly_data_loader = data[self.quarterly_data_key]
        self.daily_data = {}
        if self.daily_index is not None:
            for idx in self.daily_index:
                self.daily_data[idx] = self._daily_data_loader.load_train([idx])
        with Pool(self.n_jobs) as p:
            X = []
            for ticker_feats_arr in p.imap(self._single_ticker, index):
                X.extend(ticker_feats_arr)
        if(len(X) == 0):
            return pd.DataFrame(X, columns=['ticker', 'date']).set_index(['ticker', 'date'])
        X = pd.DataFrame(X).set_index(['ticker', 'date'])
        return X
class FeatureMerger:
    def __init__(self, fc1, fc2, on=Union[str, List[str]]):
```

```
        self.fc1 = fc1
        self.fc2 = fc2
        self.on = on
    def calculate(self, data: Dict, index) -> pd.DataFrame:
        X1 = self.fc1.calculate(data, index)
        X2 = self.fc2.calculate(data, index)
        X = pd.merge(X1, X2, on=self.on, how='left')
        X.index = X1.index
        return X


          targets.ipynb
import numpy as np
import pandas as pd
from multiprocessing import Pool, cpu_count
from typing import List, Dict, Tuple, Callable
class QuarterlyTarget:
    def __init__(self,
                 data_key: str,
                 col: str,
                 quarter_shift: int=0,
                 n_jobs: int=cpu_count()):
        self.data_key = data_key
        self.col = col
        self.quarter_shift = quarter_shift
        self.n_jobs = n_jobs
        self._data_loader = None
    def _single_ticker_target(self,
                              ticker_and_dates: Tuple[str,
                                                      List]) -> pd.DataFrame:
        ticker, dates = ticker_and_dates
```

```
        quarterly_data = self._data_loader.load_train([ticker])[::-1]
        quarter_dates = quarterly_data['date'].astype(np.datetime64).values
        vals = []
        for date in dates:
            assert np.datetime64(date) in quarter_dates
            curr_date_mask = quarter_dates == np.datetime64(date)
            curr_quarter_idx = np.where(curr_date_mask)[0][0]
            idx = curr_quarter_idx + self.quarter_shift
            if idx >= 0 and idx < len(quarterly_data):
                value = quarterly_data[self.col].values[idx]
            else:
                value = np.nan
            vals.append(value)
        result = pd.DataFrame()
        result['y'] = vals
        result['date'] = dates
        result['ticker'] = ticker
        return result
    def calculate(self, data: Dict, index: pd.DataFrame) -> pd.DataFrame:
        self._data_loader = data[self.data_key]
        grouped = index.groupby('ticker')['date'].apply(lambda x:
                  x.tolist()).reset_index()
        params = [(ticker, dates) for ticker, dates in grouped.values]
        with Pool(self.n_jobs) as p:
            result = []
            for ticker_result in p.imap(self._single_ticker_target, params):
                result.append(ticker_result)
        result = pd.concat(result, axis=0)
        result = result.drop_duplicates(['ticker', 'date'])
        result = pd.merge(index, result, on=['ticker', 'date'], how='left')
```

```
        result = result.set_index(['ticker', 'date'])
        result = result.infer_objects()
        return result
class DailyAggTarget:
    def __init__(self,
                 data_key: str,
                 col: str,
                 horizon: int=100,
                 foo: Callable=np.mean,
                 n_jobs: int = cpu_count()):
        self.data_key = data_key
        self.col = col
        self.horizon = horizon
        self.foo = foo
        self.n_jobs = n_jobs
        self._data_loader = None
    def _single_ticker_target(self,
                              ticker_and_dates: Tuple[str,
                                                      List]) -> pd.DataFrame:
        ticker, dates = ticker_and_dates
        result = pd.DataFrame()
        result['date'] = dates
        result['ticker'] = ticker
        result['y'] = None
        daily_data = self._data_loader.load_train([ticker])
        if daily_data is None:
            return result
        daily_data = daily_data[::-1]
        daily_dates = daily_data['date'].astype(np.datetime64).values
        vals = []
```

```
        for date in dates:
            if self.horizon >= 0:
                series = daily_data[daily_dates >= np.datetime64(date)]
                series = series[self.col].values[:self.horizon]
            else:
                series = daily_data[daily_dates < np.datetime64(date)]
                series = series[self.col].values[self.horizon:]
            vals.append(self.foo(series.astype(float)))
        result['y'] = vals
        return result
    def calculate(self, data: Dict, index: pd.DataFrame) -> pd.DataFrame:
        self._data_loader = data[self.data_key]
        grouped = index.groupby('ticker')['date'].apply(lambda x:
                x.tolist()).reset_index()
        params = [(ticker, dates) for ticker, dates in grouped.values]
        with Pool(self.n_jobs) as p:
            result = []
            for ticker_result in p.imap(self._single_ticker_target, params):
                result.append(ticker_result)
        result = pd.concat(result, axis=0)
        result = result.drop_duplicates(['ticker', 'date'])
        result = pd.merge(index, result, on=['ticker', 'date'], how='left')
        result = result.set_index(['ticker', 'date'])
        return result
```

models.ipynb

```
import pandas as pd
import numpy as np
from copy import deepcopy
from typing import List
```

```
from sklearn.model_selection import GroupKFold
class LogExpModel:
    def __init__(self, base_model):
        self.base_model = base_model
    def fit(self, X: pd.DataFrame, y):
        mask = (y > 0).values
        self.base_model.fit(X[mask], np.log(y[mask].astype('float')))
    def predict(self, X):
        return np.exp(self.base_model.predict(X)) class EnsembleModel:
    def __init__(self, base_models: List, bagging_fraction: float=0.8,
                 model_cnt: int=20):
        self.base_models = base_models
        self.bagging_fraction = bagging_fraction
        self.model_cnt = model_cnt
        self.models = []
    def fit(self, X: pd.DataFrame, y: pd.Series):
        for _ in range(self.model_cnt):
            idxs = np.random.randint(0, len(X),
                                     int(len(X) * self.bagging_fraction))
            curr_model = deepcopy(np.random.choice(self.base_models))
            curr_model.fit(X.iloc[idxs], y.iloc[idxs])
            self.models.append(curr_model)
    def predict(self, X):
        preds = []
        for k in range(self.model_cnt):
            try:
                model_pred = self.models[k].predict_proba(X)[:, 1]
            except:
                model_pred = self.models[k].predict(X)
```

```
            preds.append(model_pred)
        return np.mean(preds, axis=0)
class GroupedOOFModel:
    def __init__(self, base_model, group_column: str, fold_cnt: int=5):
        self.fold_cnt = fold_cnt
        self.group_column = group_column
        self.base_models = []
        for k in range(self.fold_cnt):
            self.base_models.append(deepcopy(base_model))
        self.group_df = None
        self.columns = None
    def fit(self, X: pd.DataFrame, y: pd.Series):
        groups = X.reset_index()[self.group_column]
        df_arr = []
        kfold = GroupKFold(self.fold_cnt)
        for k, (itr, ite) in enumerate(kfold.split(X, y, groups)):
            self.base_models[k].fit(X.iloc[itr], y.iloc[itr])
            curr_group_df = pd.DataFrame()
            curr_group_df['group'] = np.unique(groups[ite])
            curr_group_df['fold_id'] = k
            df_arr.append(curr_group_df)
        self.group_df = pd.concat(df_arr, axis=0)
        self.columns = X.columns
    def predict(self, X: pd.DataFrame) -> np.array:
        groups = X.reset_index()[self.group_column]
        predict_groups = pd.DataFrame()
        predict_groups['group'] = groups
        predict_groups = pd.merge(predict_groups, self.group_df,
                                  on='group', how='left')
        predict_groups.index = X.index
```

```
        # If group was not in train data -> put to 0th fold
        predict_groups = predict_groups.fillna(0)
        pred_df = []
        for fold_id in range(self.fold_cnt):
            X_curr = X[predict_groups['fold_id'] == fold_id]
            if len(X_curr) == 0:
                continue
            try:
                pred = self.base_models[fold_id].predict_proba(X_curr)[:, 1]
            except:
                pred = self.base_models[fold_id].predict(X_curr)
            curr_pred_df = pd.DataFrame()
            curr_pred_df['pred'] = pred
            curr_pred_df.index = X_curr.index
            pred_df.append(curr_pred_df)
        pred_df = pd.concat(pred_df, axis=0)
        pred_df = pred_df.loc[X.index]
        return pred_df['pred'].values


      pipeline.ipynb
import pandas as pd
import numpy as np
from typing import List, Dict
from ipynb.fs.full.utils import nan_mask, check_create_folder, copy_repeat
class Pipeline:
    def __init__(self, data: Dict, feature, target, model, out_name=None):
        self.core = {}
        self.data = data
        self.feature = feature
        if type(target) == list and type(model) == list:
```

```
            assert len(target) == len(model)
        if type(target) == list and type(out_name) == list:
            assert len(target) == len(out_name)
        self.target = target if type(target) == list else [target]
        target_len = len(self.target)
        self.core['model'] = model if type(model) == list else \
                            copy_repeat(model, target_len)
        if out_name is None:
            self.out_name = ['y_{}'.format(k) for k in range(target_len)]
        if type(out_name) is str:
            self.out_name = [out_name]
        if type(out_name) == list:
            self.out_name = out_name
    def fit(self, index: List[str], metric=None, target_filter_foo=nan_mask):
        if type(metric) == list:
            assert len(self.target) == len(metric)
        if type(target_filter_foo) == list:
            assert len(self.target) == len(target_filter_foo)
        metric = metric if type(metric) == list \
                            else [metric] * len(self.target)
        target_filter_foo = target_filter_foo if type(target_filter_foo) == list \
                            else [target_filter_foo] * len(self.target)
        metrics_result = {}
        X = self.feature.calculate(self.data, index)
        for k, target in enumerate(self.target):
            y = target.calculate(self.data,
                                 X.index.to_frame(index=False))
            leave_mask = target_filter_foo[k](y['y'].values)
            y_ = y[leave_mask]
            X_ = X[leave_mask]
```

```
            self.core['model'][k].fit(X_, y_['y'])
            if metric[0] is not None:
                pred = self.core['model'][k].predict(X_)
                metric_name = 'metric_{}'.format(self.out_name[k])
                metrics_result[metric_name] = metric[k](y_['y'].values, pred)
        return metrics_result
    def execute(self, index):
        result = pd.DataFrame()
        X = self.feature.calculate(self.data, index)
        for k, target in enumerate(self.target):
            pred = self.core['model'][k].predict(X)
            result[self.out_name[k]] = pred
        result.index = X.index
        return result
    def export_core(self, path=None):
        if path is None:
            now = time.strftime("%d.%m.%y_%H:%M", time.localtime(time.time()))
            path = 'models_data/pipeline_{}'.format(now)
        check_create_folder(path)
        with open('{}.pickle'.format(path), 'wb') as f:
            pickle.dump(self.core, f)
    def load_core(self, path):
        with open(path, 'rb') as f:
            self.core = pickle.load(f)
```

multicriterial_choice.ipynb

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
import pandas as pd
import numpy as np
```

```
import random
from pandas import DataFrame
import matplotlib.pyplot as plt
from tabulate import tabulate
from ipynb.fs.full.data_loaders import BaseData, QuarterlyData
from ipynb.fs.full.utils import int_hash_of_str
DATE_SEPARATOR = '2022-07-01'
QUARTER_COLUMNS = [
        "date",
        "ticker",
        "debt",
        "divyield",
        "marketcap",
        "close"]
QUARTER_DIFF_COLUMNS = [
   "marketcap_diff",
   "close_diff"
]
BASE_COLUMNS = [
   "ticker",
   "industry"]
relation_values = [0.15, 0.1, 0.1, 0.1, 0.2, 0.2, 0.15]
COMPANIES = ['NKLA', 'TSLA', 'RACE', 'STLA', 'F', 'TM',
            'MSFT', 'ADBE', 'ORCL', 'PAYO', 'BB', 'DBX',
            'AAPL', 'NVDA', 'AMD', 'SONY', 'KOSS', 'BOX',
            'NFLX', 'DIS', 'IMAX', 'WWE', 'CNK', 'WMG']
INDUSTRIES = [
   'Consumer Electronics',
   'Entertainment' ,
   'Software - Infrastructure',
```

```
    'Semiconductors',
    'Auto Manufacturers']
class ChoiseModel:
    def __init__(self):
        self.__init_dataloader()
    def __init_dataloader(self):
        self.dataloader = {}
        self.dataloader['base'] = BaseData()
        self.dataloader['quarterly'] = QuarterlyData(date_separator=DATE_SEPARATOR)
    def __prepare_data(self,
                       base_data,
                       quarter_data):
        new_quarter_data = DataFrame()
        for ticker in COMPANIES:
            ticker_base_data = base_data[base_data['ticker'] == ticker]
            ticker_quarter_data = quarter_data[quarter_data['ticker'] ==
ticker].head(2).drop(['date'], axis=1)
            ticker_quarter_data_diff = (-ticker_quarter_data
                                        .drop(['ticker', 'debt', 'divyield'], axis=1)
                                        .diff().dropna())
            result_ticker_data = (quarter_data[quarter_data['ticker'] ==
ticker].head(1).drop(['date'], axis=1))
            result_ticker_data['marketcap_diff'] =
ticker_quarter_data_diff['marketcap'].iloc[0]
            result_ticker_data['close_diff'] = ticker_quarter_data_diff['close'].iloc[0]
            result_ticker_data['ticker'] = ticker
            result_ticker_data['industry'] = ticker_base_data['industry'].iloc[0]
            new_quarter_data = pd.concat([new_quarter_data, result_ticker_data], axis=0)
        self.data = new_quarter_data.set_index('ticker')
    def __compare(self, v1, v2):
```

```
        if(type(v1) == str):
            v1 = INDUSTRIES.index(v1)
            v2 = INDUSTRIES.index(v2)
        if(v1 >= v2):
            return 1
        return 0
    def __calc_diagonal_diff(self, table):
        for i in range(len(COMPANIES)):
            for j in range(i + 1):
                diff = table[i][j] - table[j][i]
                table[i][j] = diff if diff > 0 else 0
                table[j][i] = -diff if -diff > 0 else 0
        return table
    def __create_relation_tables(self):
        relation_tables = []
        for column in self.data.columns:
            if column == "debt" or column == "close":
                column_values = -self.data.loc[:,column]
            else:
                column_values = self.data.loc[:,column]
            table = np.eye(len(COMPANIES))
            for el in range(len(COMPANIES)):
                for i in range(len(COMPANIES)):
                    if(el != i):
                        table[el][i] = self.__compare(column_values.iloc[el],
column_values.iloc[i])
            relation_tables.append(table)
        relation_table = np.zeros(len(COMPANIES))
        valued_relation_table = np.zeros(len(COMPANIES))
        for el in range(len(relation_tables)):
```

```
            relation_table = relation_table + relation_tables[el]
            valued_relation_table = valued_relation_table + relation_tables[el] *
relation_values[el]
        for i in range(len(COMPANIES)):
            for j in range(len(COMPANIES)):
                if(i != j and relation_table[i][j] < relation_table[i][i]):
                    relation_table[i][j] = 0
        self.relation_table = relation_table
        self.valued_relation_table = valued_relation_table
    def __calc_criteria(self):
        self.__create_relation_tables()
        relation_table_diff = self.__calc_diagonal_diff(self.relation_table)
        valued_relation_table_diff = self.__calc_diagonal_diff(self.valued_relation_table)
        relation_table_criteria = []
        valued_relation_table_criteria = []
        for i in range(len(COMPANIES)):
            relation_table_criteria.append(1 - max(relation_table_diff[:, i]))
            valued_relation_table_criteria.append(1 - max(valued_relation_table_diff[:, i]))
         return np.array(relation_table_criteria) * np.array(valued_relation_table_criteria)
    def load_data(self):
        base_data = self.dataloader['base'].load(COMPANIES).loc[:, BASE_COLUMNS]
        quarter_data = self.dataloader['quarterly'].load_train(COMPANIES).loc[:,
QUARTER_COLUMNS]
        self.__prepare_data(base_data, quarter_data)
    def get_portfolio(self, n):
        criteria = self.__calc_criteria()
        portfolio = DataFrame()
        droped_companies = DataFrame()
        for i in range(n):
            i_max = criteria.argmax(axis=0)
```

```
            i_portfolio = DataFrame([[COMPANIES[i_max], criteria[i_max]]],
columns=['ticker', 'metric'])
            criteria[i_max] = -1
            portfolio = pd.concat([portfolio, i_portfolio], axis=0)
        for i in range(len(COMPANIES)):
            if criteria[i] < 0:
                continue
            i_droped_company = DataFrame([[COMPANIES[i], criteria[i]]],
columns=['ticker', 'metric'])
            droped_companies = pd.concat([droped_companies, i_droped_company],
axis=0)
        metric_sum = sum(portfolio['metric'])
        portfolio['percent'] = 100 / metric_sum * portfolio['metric']
        return portfolio, droped_companies
def evaluate_portfolio(dataloader,
                       portfolio):
    future_close = dataloader['quarterly'].load_test(COMPANIES)
    future_close = future_close[['date', 'ticker', 'close']]
    prev_close = dataloader['quarterly'].load_train(COMPANIES)
    prev_close = prev_close[['date', 'ticker', 'close']]
    sum_revenue = 0
    for el in portfolio['ticker'].to_list():
        print(el)
        future_close_el = future_close[future_close['ticker'] == el].head(5)
        last_known_value = prev_close[prev_close['ticker'] == el].head(1)
        close_table = pd.concat([future_close_el, last_known_value])
        diff = close_table['close'].subtract(last_known_value.iloc[0]['close'])
        close_table.insert(close_table.shape[1],
                           'diff',
                           diff)
```

```
        metric = portfolio[portfolio['ticker'] == el]['metric'].iloc[0]
        close_table.insert(close_table.shape[1],
                           'metric',
                           metric)
        percent = portfolio[portfolio['ticker'] == el]['percent'].iloc[0]
        close_table.insert(close_table.shape[1],
                           'percent',
                           percent)
        revenue = diff * percent
        close_table.insert(close_table.shape[1],
                           'revenue',
                           revenue)
        sum_revenue += revenue.iloc[0]
        print(tabulate(close_table, headers='keys',
                       tablefmt='psql'))
        print()
    print("Summary revenue = {}".format(sum_revenue) )
def show_results(dataloader,
                 portfolio):
    future_close = dataloader['quarterly'].load_test(COMPANIES)
    future_close = future_close[['date', 'ticker', 'close']]
    prev_close = dataloader['quarterly'].load_train(COMPANIES)
    prev_close = prev_close[['date', 'ticker', 'close']]
    sum_revenue = 0
    for el in portfolio['ticker'].to_list():
        print(el)
        future_close_el = future_close[future_close['ticker'] == el].head(5)
        last_known_value = prev_close[prev_close['ticker'] == el].head(1)
        close_table = pd.concat([future_close_el, last_known_value])
        diff = close_table['close'].subtract(last_known_value.iloc[0]['close'])
```

```
        close_table.insert(close_table.shape[1],
                           'diff',
                           diff)
        probability = portfolio[portfolio['ticker'] == el]['metric'].iloc[0]
        close_table.insert(close_table.shape[1],
                           'metric',
                           probability)
        print(tabulate(close_table, headers='keys',
                       tablefmt='psql'))
        print()
cm = ChoiseModel()
cm.load_data()
portfolio, droped_companies = cm.get_portfolio(7)
evaluate_portfolio(cm.dataloader, portfolio)
show_results(cm.dataloader, droped_companies)
```

bayesian_model.ipynb

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
import pandas as pd
import numpy as np
import random
from pandas import DataFrame
import matplotlib.pyplot as plt
from tabulate import tabulate
from ipynb.fs.full.data_loaders import BaseData, QuarterlyData, DailyData
from ipynb.fs.full.utils import int_hash_of_str
from sklearn.naive_bayes import GaussianNB
DATE_SEPARATOR = '2022-07-01'
SCALE_MARKETCAP = ["3 - Small", "4 - Mid", "5 - Large", "6 - Mega"]
```

```
QUARTER_COLUMNS = [
        "date",
        "ticker",
        "debt",
        "divyield",
        "sps",
   "pe",
   "grossmargin"
        ]
BASE_COLUMNS = [
   "ticker",
   "sector",
   "industry",
   "currency"]
DAILY_COLUMNS = [
   "date",
   "ticker",
   "marketcap",
   "close"]
COMPANIES = ['NKLA', 'TSLA', 'RACE', 'STLA', 'F', 'TM',
             'MSFT', 'ADBE', 'ORCL', 'PAYO', 'BB', 'DBX',
             'AAPL', 'NVDA', 'AMD', 'SONY', 'KOSS', 'BOX',
             'NFLX', 'DIS', 'IMAX', 'WWE', 'CNK', 'WMG']
INDUSTRIES = ['Auto Manufacturers',
            'Software - Infrastructure',
            'Consumer Electronics',
            'Entertainment',
            'Semiconductors']
class BayesianModel:
   cut_off = 0.56
```

```
    def __init__(self):
        self.__init_dataloader()
        self.__init_model()
    def __init_dataloader(self):
        self.dataloader = {}
        self.dataloader['base'] = BaseData()
        self.dataloader['daily'] = DailyData(date_separator=DATE_SEPARATOR)
        self.dataloader['quarterly'] = QuarterlyData(date_separator=DATE_SEPARATOR)
    def __init_model(self):
        self.model = GaussianNB()
    def __get_tickers(self, base_data):
        tickers = list(base_data[base_data['scalemarketcap'].apply(lambda x: x in
SCALE_MARKETCAP) &\
                        base_data['industry'].isin(INDUSTRIES)]['ticker'].values)
        return tickers
    def __prepare_data(self,
                       tickers,
                       base_data,
                       daily_data,
                       quarterly_data):
        prepared_data = DataFrame()
        daily_quarterly_data = quarterly_data.merge(daily_data, how = 'inner', on =
['ticker', 'date'])
        for ticker in tickers:
            ticker_daily_quarterly_data =
(daily_quarterly_data[daily_quarterly_data['ticker'] == ticker]
                                        .drop(['date', 'ticker'], axis=1))
            ticker_daily_quarterly_data_diff = -ticker_daily_quarterly_data.diff().dropna()
            ticker_daily_quarterly_data_diff['closechng'] =
np.where(ticker_daily_quarterly_data_diff['close']> 0,
```

```
                                    'raise', 'fall')
            ticker_prepared_data = ticker_daily_quarterly_data_diff.drop(['close'], axis=1)
            ticker_base_data = base_data[base_data['ticker'] == ticker]
            industry = ticker_base_data['industry'].iloc[0]
            sector = ticker_base_data['sector'].iloc[0]
            ticker_prepared_data['sector'] = int_hash_of_str(str(industry))
            ticker_prepared_data['industry'] = int_hash_of_str(str(sector))
            ticker_prepared_data['ticker'] = ticker
            prepared_data = pd.concat([prepared_data, ticker_prepared_data], axis=0)
        self.prepared_data = prepared_data
    def __split_data(self, tickers):
        data_test = DataFrame()
        data_train = DataFrame()
        for ticker in tickers:
            ticker_test = self.prepared_data[self.prepared_data['ticker'] == ticker].head(1)
            data_test = pd.concat([data_test, ticker_test])
        data_train =  self.prepared_data.drop(data_test.index)
        self.data_test = data_test
        self.data_train = data_train
    def load_data(self):
        base_data = (self.dataloader['base']
                     .load())
        tickers = self.__get_tickers(base_data)
        daily_data = (self.dataloader['daily']
                      .load_train(tickers)
                      .loc[:, DAILY_COLUMNS])
        quarterly_data = (self.dataloader['quarterly']
                          .load_train(tickers)
                          .loc[:,QUARTER_COLUMNS])
        self.__prepare_data(tickers, base_data, daily_data, quarterly_data)
```

```
        self.__split_data(tickers)
    def fit(self):
        y = self.data_train['closechng'].values.ravel()
        x = self.data_train.drop(['ticker', 'closechng'], axis=1)
        self.model.fit(x, y)
    def create_portfolio(self, tickers):
        results = DataFrame()
        x = self.data_test.drop(['closechng'], axis=1)
        for ticker in tickers:
            ticker_x = x[x['ticker'] == ticker]
            prediction = self.model.predict(ticker_x.drop(['ticker'], axis=1))
            probability = self.model.predict_proba(ticker_x.drop(['ticker'], axis=1))[0][1]
            result_row = DataFrame([[ticker, probability]], columns=['ticker', 'probability'])
            results = pd.concat([results, result_row], axis=0)
        portfolio = results[results['probability'] > self.cut_off]
        droped_companies = results[results['probability'] <= self.cut_off]
        percent = 100 / sum(portfolio['probability']) * portfolio['probability']
        portfolio.insert(2, "percent", percent)
        return portfolio, droped_companies
def evaluate_portfolio(dataloader,
                       portfolio):
    future_close = dataloader['quarterly'].load_test(COMPANIES)
    future_close = future_close[['date', 'ticker', 'close']]
    prev_close = dataloader['quarterly'].load_train(COMPANIES)
    prev_close = prev_close[['date', 'ticker', 'close']]
    sum_revenue = 0
    for el in portfolio['ticker'].to_list():
        print(el)
        future_close_el = future_close[future_close['ticker'] == el].head(5)
        last_known_value = prev_close[prev_close['ticker'] == el].head(1)
```

```
        close_table = pd.concat([future_close_el, last_known_value])
        diff = close_table['close'].subtract(last_known_value.iloc[0]['close'])
        close_table.insert(close_table.shape[1],
                           'diff',
                           diff)
        probability = portfolio[portfolio['ticker'] == el]['probability'].iloc[0] * 100
        close_table.insert(close_table.shape[1],
                           'probability',
                           probability)
        percent = portfolio[portfolio['ticker'] == el]['percent'].iloc[0]
        close_table.insert(close_table.shape[1],
                           'percent',
                           percent)
        revenue = diff * percent
        close_table.insert(close_table.shape[1],
                           'revenue',
                           revenue)
        sum_revenue += revenue.iloc[0]
        print(tabulate(close_table, headers='keys',
                       tablefmt='psql'))
        print()
    print("Summary revenue = {}".format(sum_revenue) )
def show_results(dataloader,
                 portfolio):
    future_close = dataloader['quarterly'].load_test(COMPANIES)
    future_close = future_close[['date', 'ticker', 'close']]
    prev_close = dataloader['quarterly'].load_train(COMPANIES)
    prev_close = prev_close[['date', 'ticker', 'close']]
    sum_revenue = 0
for el in portfolio['ticker'].to_list():
```

```
        print(el)
        future_close_el = future_close[future_close['ticker'] == el].head(5)
        last_known_value = prev_close[prev_close['ticker'] == el].head(1)
        close_table = pd.concat([future_close_el, last_known_value])
        diff = close_table['close'].subtract(last_known_value.iloc[0]['close'])
        close_table.insert(close_table.shape[1],
                'diff',
                diff)
        probability = portfolio[portfolio['ticker'] == el]['probability'].iloc[0] * 100
        close_table.insert(close_table.shape[1],
                'probability',
                probability)
        print(tabulate(close_table, headers='keys',
                tablefmt='psql'))
        print()
bm = BayesianModel()
bm.load_data()
bm.fit()
portfolio, droped_companies = bm.create_portfolio(COMPANIES)
evaluate_portfolio(bm.dataloader, portfolio)
show_results(bm.dataloader, droped_companies)
```

markovitz_model.ipynb

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pandas import DataFrame
from tabulate import tabulate
from ipynb.fs.full.data_loaders import BaseData, DailyData, QuarterlyData
COMPANIES = ['NKLA', 'TSLA', 'RACE', 'STLA', 'F', 'TM',
```

```
            'MSFT', 'ADBE', 'ORCL', 'PAYO', 'BB', 'DBX',
            'AAPL', 'NVDA', 'AMD', 'SONY', 'KOSS', 'BOX',
            'NFLX', 'DIS', 'IMAX', 'WWE', 'CNK', 'WMG']
DATE_SEPARATOR = '2022-07-01'
class MarkovitzaModel:
    N = 10**8
    def __init__(self):
        self.__init_dataloader()
    def __init_dataloader(self):
        self.dataloader = {}
        self.dataloader['base'] = BaseData()
        self.dataloader['daily'] = DailyData(date_separator=DATE_SEPARATOR)
        self.dataloader['quarterly'] = QuarterlyData(date_separator=DATE_SEPARATOR)
    def __load_data(self):
        df = DataFrame()
        loaded_data = self.dataloader['daily'].load_train(COMPANIES)
        need_data = loaded_data.loc[:, ['ticker', 'date', 'close']]
        for company in COMPANIES:
            company_data = (need_data[need_data['ticker'] == company]
                .loc[:, ['date', 'close']]
                .rename(columns={"close": company}))
            temp_df = company_data.set_index('date')
            df = pd.concat([df, temp_df], axis=1)
        df = df.astype(float).dropna()
        self.data = df

    def __portfolioRevenue(self, mean_revenue, portfolio):
        return np.matmul(mean_revenue.values, portfolio)
    def __portfolioRisk(self, covariation, portfolio):
        return np.sqrt(np.matmul(np.matmul(portfolio, covariation.values),
```

```
                    portfolio))
def __generate_rand_portfolio(self):
    cnt = len(COMPANIES)
    portfolio = np.exp(np.random.randn(cnt))
    portfolio = portfolio / portfolio.sum()
    return portfolio.round(3)
def __create_portfolio_cloud(self):
    diff_data = self.data.pct_change().dropna()
    mean_diff_data = diff_data.mean()
    cov_diff_data = diff_data.cov()
    portfolio_list = np.zeros((self.N, len(COMPANIES)))
    risk_list = np.zeros(self.N)
    revenue_list = np.zeros(self.N)
    for i in range(self.N):
        portfolio = self.__generate_rand_portfolio()
        portfolio_list[i,:] = portfolio
        risk_list[i] = self.__portfolioRisk(cov_diff_data, portfolio)
        revenue_list[i] = self.__portfolioRevenue(mean_diff_data, portfolio)
    self.portfolio_list = portfolio_list
    self.risk_list = risk_list
    self.revenue_list = revenue_list
def fit(self):
    self.__load_data()
    self.__create_portfolio_cloud()
    self.min_risk = np.argmin(self.risk_list)
    self.max_Sharp_coef = np.argmax(self.revenue_list/self.risk_list)
def draw_portfolio_cloud(self):
    plt.figure(figsize=(18,8))
    plt.grid()
    plt.scatter(self.risk_list*100,self.revenue_list*100,c='y',marker='.')
```

```
    plt.xlabel('risk, %')
    plt.ylabel('revenue, %')
    plt.title("Portfolio cloud")
    plt.scatter([(self.risk_list[self.min_risk])*100],
                [(self.revenue_list[self.min_risk])*100],
                c='r',
                marker='*',
                label='min risk')
    plt.scatter([self.risk_list[self.max_Sharp_coef]*100],
                [self.revenue_list[self.max_Sharp_coef]*100],
                c='g',
                marker='o',
                label='max Sharp coef')
    plt.legend(loc="upper left")
    plt.show()
def get_min_risk_portfolio(self):
    portfolio = DataFrame([self.data.columns,
                           (self.portfolio_list[self.min_risk]*100)]).T
    portfolio = portfolio.rename(columns={portfolio.columns[0]: 'ticker',
                                          portfolio.columns[1]: 'percent'})
    risk = float(self.risk_list[self.min_risk])*100
    revenue = float(self.revenue_list[self.min_risk])*100
    return portfolio, risk, revenue
def get_max_Sharp_portfolio(self):
    portfolio = DataFrame([self.data.columns,
                           (self.portfolio_list[self.max_Sharp_coef]*100)]).T
    portfolio = portfolio.rename(columns={portfolio.columns[0]: 'ticker',
                                          portfolio.columns[1]: 'percent'})
    risk = float(self.risk_list[self.max_Sharp_coef])*100
    revenue = float(self.revenue_list[self.max_Sharp_coef])*100
```

```
        return portfolio, risk, revenue
def evaluate_portfolio(dataloader,
                       portfolio,
                       pred_risk,
                      pred_revenue):
    future_close = dataloader['quarterly'].load_test(COMPANIES)
    future_close = future_close[['date', 'ticker', 'close']]
    prev_close = dataloader['quarterly'].load_train(COMPANIES)
    prev_close = prev_close[['date', 'ticker', 'close']]
    sum_revenue = 0
    for el in COMPANIES:
        future_close_el = future_close[future_close['ticker'] == el].head(5)
        last_known_value = prev_close[prev_close['ticker'] == el].head(1)
        close_table = pd.concat([future_close_el, last_known_value])
        diff = close_table['close'].subtract(last_known_value.iloc[0]['close'])
        close_table.insert(close_table.shape[1],
                   'diff',
                   diff)
        percent = portfolio[portfolio['ticker'] == el]['percent'].iloc[0]
        close_table.insert(close_table.shape[1],
                   'percent',
                   percent)
        revenue = diff * percent
        close_table.insert(close_table.shape[1],
                   'revenue',
                   revenue)
        sum_revenue += revenue.iloc[0]
        print(tabulate(close_table, headers='keys',
                  tablefmt='psql'))
        print()
```

```
    print("Predicted revenue = {}".format(pred_revenue))
    print("Predicted risk = {}".format(pred_risk) )
    print("Summary revenue = {}".format(sum_revenue) )
model = MarkovitzaModel()
model.fit()
model.draw_portfolio_cloud()
min_risk_portfolio, min_risk_risk, min_risk_revenue = model.get_min_risk_portfolio()
max_Sharp_portfolio, max_Sharp_risk, max_Sharp_revenue =
model.get_max_Sharp_portfolio()
print("Min risk portfolio")
print(tabulate(min_risk_portfolio, headers='keys',
               tablefmt='psql', showindex=False))
print("\n Max Sharp companies")
print(tabulate(max_Sharp_portfolio, headers='keys',
               tablefmt='psql', showindex=False))
evaluate_portfolio(model.dataloader,
                   min_risk_portfolio,
                  min_risk_risk,
                  min_risk_revenue)
evaluate_portfolio(model.dataloader,
                   max_Sharp_portfolio,
                  max_Sharp_risk,
                  max_Sharp_revenue)


          ai_model.ipynb
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
import pandas as pd
import numpy as np
from pandas import DataFrame
```

```
import matplotlib.pyplot as plt
from tabulate import tabulate
from ipynb.fs.full.data_loaders import BaseData, QuarterlyData, DailyData
from ipynb.fs.full.targets import QuarterlyTarget, DailyAggTarget
from ipynb.fs.full.features import QuarterlyFeatures,
BaseCompanyFeatures,DailyAggQuarterFeatures, FeatureMerger
from ipynb.fs.full.models import LogExpModel, EnsembleModel, GroupedOOFModel
from ipynb.fs.full.pipeline import Pipeline
from ipynb.fs.full.metrics import median_absolute_relative_error
import lightgbm as lgbm
import catboost as ctb
from sklearn.metrics import median_absolute_error
DATE_SEPARATOR = '2022-07-01'
SCALE_MARKETCAP = ["3 - Small", "4 - Mid", "5 - Large", "6 - Mega"]
QUARTER_COLUMNS = [
        "revenue",
        "netinc",
        "ncf",
        "ebitda",
        "debt",
        "fcf",
        "divyield",
        "currentratio"]
QUARTER_COUNTS = [2, 4, 6, 8]
MAX_BACK_QUARTER = 20
MIN_BACK_QUARTER = 0
CAT_COLUMNS = ["sector", "industry"]
DAILY_COLUMNS = ["marketcap"]
AGG_DAY_COUNTS = [ 7, 30, 60, 90, 180, 360, 720]
BAGGING_FRACTION = 0.70
```

```
MODEL_CNT = 20
FOLD_CNT = 10
OUT_NAME = 'close'
COMPANIES = ['NKLA', 'TSLA', 'RACE', 'STLA', 'F', 'TM',
             'MSFT', 'ADBE', 'ORCL', 'PAYO', 'BB', 'DBX',
             'AAPL', 'NVDA', 'AMD', 'SONY', 'KOSS', 'BOX',
             'NFLX', 'DIS', 'IMAX', 'WWE', 'CNK', 'WMG']
def _create_data():
    data = {}
    data['base'] = BaseData()
    data['quarterly'] = QuarterlyData(date_separator=DATE_SEPARATOR)
    data['daily'] = DailyData(date_separator=DATE_SEPARATOR)
    return data
def _create_feature():
    fc1 = QuarterlyFeatures(data_key='quarterly',
                            columns=QUARTER_COLUMNS,
                            quarter_counts=QUARTER_COUNTS,
                            max_back_quarter=MAX_BACK_QUARTER,
                            min_back_quarter=MIN_BACK_QUARTER)
    fc2 = BaseCompanyFeatures(data_key='base',
                              cat_columns=CAT_COLUMNS)
    fc3 = DailyAggQuarterFeatures(daily_data_key='daily',
                                  quarterly_data_key='quarterly',
                                  columns=DAILY_COLUMNS,
                                  agg_day_counts=AGG_DAY_COUNTS,
                                  max_back_quarter=MAX_BACK_QUARTER,
                                  min_back_quarter=MIN_BACK_QUARTER)
    feature = FeatureMerger(fc1, fc2, on='ticker')
    feature = FeatureMerger(feature, fc3, on=['ticker', 'date'])
    return feature
```

```
def _create_target():
    target = DailyAggTarget(
        data_key='daily',
        col='close',
        horizon=30,
        foo=np.mean)
    return target
def _create_model():
    base_models = [LogExpModel(lgbm.sklearn.LGBMRegressor()),
                   LogExpModel(ctb.CatBoostRegressor(verbose=False))]
    ensemble = EnsembleModel(base_models=base_models,
                             bagging_fraction=BAGGING_FRACTION,
                             model_cnt=MODEL_CNT)
    model = GroupedOOFModel(base_model=ensemble,
                            group_column='ticker',
                            fold_cnt=FOLD_CNT)
    return model
def FairClose(max_back_quarter: int=None,
                  min_back_quarter: int=None) -> Pipeline:
    if max_back_quarter is not None:
        global MAX_BACK_QUARTER
        MAX_BACK_QUARTER = max_back_quarter
    if min_back_quarter is not None:
        global MIN_BACK_QUARTER
        MIN_BACK_QUARTER = min_back_quarter
    data = _create_data()
    feature = _create_feature()
    target = _create_target()
    model = _create_model()
    pipeline = Pipeline(feature=feature,
```

```
                target=target,
                model=model,
                data=data,
                out_name=OUT_NAME)
    return pipeline
def _process_results(close: DataFrame, fair_close: DataFrame):
    metrics = DataFrame(columns=['ticker', 'metric'])
    for company in COMPANIES:
        company_fair_close =
fair_close[fair_close.index.get_level_values('ticker').isin([company])]
        company_close = close[close['ticker'] == company]
        metric = _calculate_metric(company_close, company_fair_close)
        row = {'ticker':company, 'metric':metric}
        metrics = metrics.append(row, ignore_index=True)
    portfolio_companies = metrics[(metrics['metric'] >= 1) &\
                          (metrics['metric'] <= 3)]
    droped_companies = metrics[(metrics['metric'] < 1) |\
                       (metrics['metric'] > 3)]
    return portfolio_companies, droped_companies
def _calculate_metric(close: DataFrame, fair_close: DataFrame):
    date = fair_close.index.get_level_values('date')[0]
    last_close = close[close['date'] == date][OUT_NAME].values[0]
    last_fair_close = fair_close[fair_close.index.get_level_values('date') ==
date][OUT_NAME].values[0]
    metric = last_fair_close/last_close
    return metric
def _calcuate_portfolio_procentage(portfolio):
    metric_sum = portfolio['metric'].sum()
    portfolio.insert(portfolio.shape[1],
            'percent',
```

```
            portfolio['metric'] * 100 / metric_sum )
    return portfolio
def create_portfolio(pipeline):
    fair_close = pipeline.execute(COMPANIES)
    close = pipeline.data['quarterly'].load_train(COMPANIES)
    close = close[['date', 'ticker', 'close']]
    portfolio, droped_companies = _process_results(close, fair_close)
    portfolio = _calcuate_portfolio_procentage(portfolio)
    return portfolio, droped_companies
def _draw_graph(close: DataFrame, fair_close: DataFrame, company: str):
    plt.figure(figsize=(18,8))
    plt.grid()
    plt.plot(close['date'],
            close[OUT_NAME],
            label='close')
    plt.plot(fair_close.index.get_level_values('date'),
            fair_close[OUT_NAME],
            label='fair_close')
    plt.legend(loc="upper left")
    plt.title(company)
    plt.show()
def show_results(pipeline, tickers):
    fair_close = pipeline.execute(COMPANIES)
    future_close = pipeline.data['quarterly'].load_test(COMPANIES)
    future_close = future_close[['date', 'ticker', 'close']]
    prev_close = pipeline.data['quarterly'].load_train(COMPANIES)
    prev_close = prev_close[['date', 'ticker', 'close']]
    for el in tickers:
        fair_close_el = fair_close[fair_close.index.get_level_values('ticker') == el].head(5)
        future_close_el = future_close[future_close['ticker'] == el].head(5)
```

```
        prev_close_el = prev_close[prev_close['ticker'] == el].head(5)
        close_el = pd.concat([future_close_el, prev_close_el])
        _draw_graph(close_el, fair_close_el, el)
        last_known_value = prev_close_el.head(1)
        close_table = pd.concat([future_close_el, last_known_value])
        diff = close_table['close'].subtract(last_known_value.iloc[0]['close'])
        close_table.insert(close_table.shape[1],
                           'Diff',
                           diff)
        print(tabulate(close_table, headers='keys',
                       tablefmt='psql'))
def evaluate_portfolio(pipeline, portfolio):
    fair_close = pipeline.execute(COMPANIES)
    future_close = pipeline.data['quarterly'].load_test(COMPANIES)
    future_close = future_close[['date', 'ticker', 'close']]
    prev_close = pipeline.data['quarterly'].load_train(COMPANIES)
    prev_close = prev_close[['date', 'ticker', 'close']]
    sum_revenue = 0
    for el in portfolio['ticker'].to_list():
        fair_close_el = fair_close[fair_close.index.get_level_values('ticker') == el].head(5)
        future_close_el = future_close[future_close['ticker'] == el].head(5)
        prev_close_el = prev_close[prev_close['ticker'] == el].head(5)
        close_el = pd.concat([future_close_el, prev_close_el])
        _draw_graph(close_el, fair_close_el, el)
        last_known_value = prev_close_el.head(1)
        close_table = pd.concat([future_close_el, last_known_value])
        diff = close_table['close'].subtract(last_known_value.iloc[0]['close'])
        close_table.insert(close_table.shape[1],
                           'diff',
                           diff)
```

```
        metric = portfolio[portfolio['ticker'] == el]['metric'].iloc[0]
        close_table.insert(close_table.shape[1],
                'metric',
                metric)
        percent = portfolio[portfolio['ticker'] == el]['percent'].iloc[0]
        close_table.insert(close_table.shape[1],
                'percent',
                percent)
        revenue = diff * percent
        close_table.insert(close_table.shape[1],
                'revenue',
                revenue)
        sum_revenue += revenue.iloc[0]
        print(tabulate(close_table, headers='keys',
                tablefmt='psql'))
        print()
    print("Summary revenue = {}".format(sum_revenue) )
pipeline = FairClose()
base_df = pipeline.data['base'].load()
tickers = base_df[(base_df['scalemarketcap']
                .apply(lambda x: x in SCALE_MARKETCAP))]['ticker'].values
result = pipeline.fit(tickers, median_absolute_relative_error)
print(result)
portfolio, droped_companies = create_portfolio(pipeline)
evaluate_portfolio(pipeline, portfolio)
show_results(pipeline, droped_companies['ticker'])
```