ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Застосування ШІ на фондовій біржі для прогнозування ціни акцій, мовою Python з використанням технологій Tensorflow, Keras»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Станіслав МИРОНЕНКО
(підпис)

Виконав: здобувач вищої освіти групи ПД-41

Станіслав МИРОНЕНКО

Керівник: Ірина ЗАМРІЙ
д.т.н., доцент

Рецензент:

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Мироненку Станіславу Олександровичу

1. Тема кваліфікаційної роботи: «Застосування ШІ на фондовій біржі для прогнозування ціни акцій, мовою Python з використанням технологій Tensorflow, Keras»
керівник кваліфікаційної роботи д.т.н., доцент, зав.кафедри ІПЗ Ірина ЗАМРІЙ,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи роботи на фондовому ринку, опис методів прогнозування цін акцій за допомогою штучного інтелекту, документація з описом бібліотек для роботи з штучним інтелектом.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз та огляд існуючих інструментів для прогнозування ціни на фондовому ринку.
2. Технічний аналіз фондового ринку.
3. Дослідження та аналіз штучного інтелекту.
4. Розробка програмного продукту.
5. Тестування програмного продукту.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів
2. Вимоги до програмного забезпечення
3. Технічні засоби реалізації
4. Програмні засоби реалізації
5. Діаграма варіантів використання
6. Екранні форми

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Аналіз та дослідження існуючих аналогів	13.03.-18.03.2024	
4	Визначення вимог	19.03.-20.03.2024	
5	Аналіз засобів реалізації	21.03.-04.03.2024	
6	Розробка програмного забезпечення	05.03.-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Станіслав МИРОНЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Ірина ЗАМРІЙ
(підпис)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 49 стор., містить 20 рис., 1 табл., 3 дод., 13 джерел.

Об'єкт дослідження – процес прогнозування цін акцій на фондовму ринку.

Предмет дослідження – програмне забезпечення для прогнозування цін акцій на фондовому ринку з використанням штучного інтелекту.

Мета дослідження – підтримка робочих процесів аналітика фондового ринку.

Короткий зміст роботи: В роботі проаналізовано фондовий ринок та методи прогнозування на ньому за допомогою штучного інтелекту. Проаналізовано інструментальні засоби для формування тематичних словників: KERAS, TENSORFLOW, TALIB, NUMPY, PANDAS. Розроблено програмне забезпечення, яке допомагає створювати та аналізувати моделі штучного інтелекту спрямовані на регресійний аналіз часових рядів. Програмно реалізовані ключові функціональні можливості, зокрема: завантаження даних, їх зміна, стандартизація, стаціонарізація, вибір метрик навчання, вибір шарів штучного інтелекту, конфігурацій компілювання, збереження моделей та історії навчання моделі. Проведено тестування додатку. В роботі використано бібліотеки Python, Tensorflow для створення штучного інтелекту, NumPy, Pandas, SkLearndля роботи з даними, shutil для роботи з файлами та папками, yfinance для роботи з АПІ, talib для розрахунку індикаторів, FastAPI для створення АПІ. Також використовується React та MUI для фронтенд частини.

Сферою використання застосунку є прогнозування цін акцій на фондовому ринку.

КЛЮЧОВІ СЛОВА: PYTHON, PYCHARM, KERAS, TENSORFLOW, TALIB, NUMPY, PANDAS, ШТУЧНИЙ ІНТЕЛЕКТ.

ЗМІСТ

ВСТУП.....10
1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ.....13
1.1 Особливості та проблеми прогнозування на фондовому ринку.....13
1.2 Особливості роботи зі штучним інтелектом.....15
1.3 Переваги і недоліки застосування штучного інтелекту для прогнозування цін на фондовому ринку.....16
1.4 Перший погляд та аналіз фондової біржі.....20
1.5 Огляд штучного інтелекту з технічної точки зору.....22
1.5.1 Що таке нейрон в контексті штучного інтелекту.....22
1.5.2 Структура та принцип роботи повнозв'язної моделі штучного інтелекту.....23
1.5.3 Інші моделі штучного інтелекту.....29
1.5.3.1 Згорткова нейронна мережа.....29
1.5.3.2 Автокодер.....29
1.5.3.3 Звичайна рекурентна нейронна мережа (RNN).....30
1.5.3.4 Варіації рекурентних нейронних мереж.....31
1.5.3.4.1 Long Short-Term Memory (LSTM).....32
1.5.3.4.2 Gated Recurrent Unit (GRU).....32
1.6 Визначення вимог до програмного забезпечення.....33
2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....35
2.1 Огляд програмних засобів реалізації.....35
2.1.1 Мова програмування Python.....35
2.1.2 Середовище розробки PyCharm.....36
2.1.3 Бібліотеки NumPy та Pandas.....37
2.1.4 Бібліотека MatPlotLib.....39

2.1.5 Бібліотеки Tensorflow та Keras.....40
2.1.6 Бібліотека TaLib.....41
2.1.7 Бібліотека sklearn.....42
2.1.8 Фреймворк FastAPI.....43
2.1.9 Бібліотека React.....43
2.1.10 Бібліотека MUI/MUI X.....44
2.2 Моделювання вимог до програмного забезпечення.....45
2.3 Розробка програмного забезпечення для створення моделей для прогнозування часових рядів.....47
2.3.1. Розробка архітектури.....47
2.3.2. Розробка класів.....48
2.3.3. Створення моделі нульового прогнозування.....49
2.3.4. Застосування індикаторів.....50
2.3.5. Підготовка даних.....51
2.3.6. LSTM модель зі збереженням стану та без.....52
3. СТВОРЕННЯ, ОГЛЯД ТА АНАЛІЗ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ.....53
ВИСНОВКИ.....57
ПЕРЕЛІК ПОСИЛАНЬ.....59
ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація).....60
ДОДАТОК Б. РЕАЛІЗАЦІЯ БЕКЕНДУ.....68
ДОДАТОК В. РЕАЛІЗАЦІЯ ФРОНТЕНДУ.....91

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ШІ - штучний інтелект

ПЗ - програмне забезпечення

АПІ (API - Application Programming Interface) - це набір інструментів та протоколів, які дозволяють різним програмним додаткам взаємодіяти між собою. API визначає методи та формати запитів і відповідей, що використовуються для взаємодії між програмними компонентами.

ВСТУП

У сучасному світі фінанси мають великий вплив на якість і спосіб життя кожної людини. Кожен хоче жити в достатку і мати хороший дохід, який дозволить забезпечити себе, близьких і рідних. Маючи багато багатства, ви можете реалізувати свої мрії, подорожувати чи піклуватися про себе. Тому кожен намагається знайти пасивний заробіток, який не займе багато часу.

Щоб заробіток був достатнім, потрібно правильно вибрати галузь. У багатьох відразу виникає ідея спробувати себе в фінансах, адже це пряма дорога в безпеку - де ще будуть великі гроші в сфері фінансів? Одним із способів заробітку в сфері фінансів є торгівля на біржі.

У світі фондового ринку, де панують потрясіння та нестабільність, завдання прогнозування цін на валюту стало невід'ємною частиною життя трейдерів та інвесторів. Будь-яка дрібна подія чи новина, будь то нормативна зміна чи технологічний прорив, миттєво впливає на ціни цифрових активів. У такому динамічному середовищі, де зміни відбуваються миттєво, прогнозування стає ключем до успішної торгівлі.

Застосування методів глибокого навчання на основі нейронних мереж з використанням Python, TensorFlow і Keras відкриває нові перспективи для аналізу ринку. Ці інструменти дозволяють аналізувати великі обсяги даних, визначати складні закономірності та створювати точні моделі для прогнозування цін на валюту. Такі підходи можуть допомогти трейдерам приймати більш обґрунтовані рішення та покращити результати торгівлі.

В останні роки інтерес до використання штучного інтелекту на фондових ринках значно зріс. Прагнення до використання передових технологій для аналізу даних і прогнозування тенденцій призвело до поширення машинного навчання та нейронних мереж. Python, TensorFlow і Keras стали інструментами для багатьох дослідників і трейдерів завдяки їх ефективності та гнучкості.

Це дослідження зосереджено на застосуванні штучного інтелекту на ринках з метою прогнозування цін на валюту. Ми розглянемо різні методи глибокого навчання та оцінимо їх ефективність у прогнозуванні курсів валютних активів. Аналізуючи їхні переваги та обмеження, ми можемо краще зрозуміти, які методи найбільш підходять для прогнозування цін на фінансових ринках.

Використання штучного інтелекту для прогнозування цін має свої проблеми та ризики. Однак, при правильному підході та адекватному налаштуванні моделей машинного навчання можна досягти високої точності прогнозування. Це відкриває нові можливості для трейдерів та інвесторів, дозволяючи їм приймати більш обґрунтовані рішення та успішно торгувати.

Тим не менш, варто зазначити, що ринкова динаміка часто піддається зовнішнім впливам, включаючи регулювання, технологічні зміни та геополітичні події. Тому прогнозування цін завжди залишається викликом, який вимагає постійного моніторингу та аналізу даних.

Немалою проблемою є те, що неможливо написати алгоритм, який добре працюватиме для всіх валют і часів. Зі зміною ринку змінюються і алгоритми роботи з цим ринком. Тому багато алгоритмів не показують хороший результат на тривалому часовому інтервалі.

Це питання буде вирішено через деякі моменти в процесі розробки. По-перше, алгоритм на основі штучного інтелекту не потрібно переписувати щоразу при зміні даних, його можна перенавчати без втручання в код, що значно підвищує його ефективність. По-друге, цей проект буде з відкритим вихідним кодом, і кожен, хто хоче вдосконалити, змінити або застосувати алгоритм у своєму проекті, матиме таку можливість.

Об’єкт дослідження – процес прогнозування цін акцій на фондовому ринку.

Предмет дослідження – програмне забезпечення для прогнозування цін акцій на фондовому ринку з використанням штучного інтелекту.

Мета дослідження – підтримка робочих процесів аналітика фондового ринку.

Щоб досягти цієї мети необхідно:

1. Провести аналіз фондового ринку, зібрати історичні дані, оцінити ринкову капіталізацію, обсяги торгів.
2. Провести технічний аналіз на основі індикаторів.
3. Проаналізувати використання штучного інтелекту на фондовому ринку, які є підходи та стратегії, які моделі кращі, а які не варто розглядати.
4. Сформувати вимоги до програмного забезпечення.
5. Створити декілька моделей штучного інтелекту, порівняти між собою та оцінити ефективність.

1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Особливості та проблеми прогнозування на фондовому ринку

Прогнозування на фондовому ринку - складна задача, яка потребує великого розуміння, досвіду та рішучості. При роботі на фондовому ринку є багато проблем з якими зустрічаються трейдери. Давайте розглянемо деякі з них:

- Високий ступінь невизначеності. Фондовий ринок піддається впливу багатьох факторів, таких як політичні події, економічні показники, зміни в законодавстві, і навіть психологія інвесторів. Це створює високий рівень невизначеності, що робить точне прогнозування складним.
- Складність моделювання. Для прогнозування на фондовому ринку потрібен облік багатьох факторів, включаючи макроекономічні показники, фінансові звіти компаній, технічні аналізи та навіть соціальні тенденції. Побудова точних стратегій, здатних враховувати всі ці аспекти, є складним завданням.
- Непередбачуваність поведінки інвесторів. Поведінка інвесторів на ринку часто непередбачувана і може бути схильною до впливу емоцій, страху та ейфорії. Це може призвести до несподіваних коливань цін на акції, які можуть зірвати будь-які прогнози.
- Можливість втручання. Фондові ринки можуть бути піддані втручанню з боку урядів, центральних банків та інших великих гравців, що може спотворити природні тенденції ринку.
- Асиметрія інформації: На ринку є асиметрія інформації, коли деякі учасники ринку мають більше інформації, ніж інші. Це може призвести до того, що деякі інвестори можуть приймати рішення на основі конфіденційної інформації, яка недоступна іншим.

Вирішення цих проблем потребує розробки складних моделей, аналізу великих обсягів даних та врахування багатьох факторів при прогнозуванні на фондовому ринку. Тим не менш, навіть при використанні передових методів прогнозування, існує високий рівень ризику та невизначеності.

Але попри всі недоліки є і достатки, давайте їх розглянемо:

- Потенційно висока прибутковість. Фондовий ринок історично забезпечував високі дохідності порівняно з іншими видами інвестицій, такими як облігації чи заощадження. Інвестори можуть заробити на зростанні цін акцій, дивідендах та капіталовкладеннях.
- Ліквідність. Фондовий ринок зазвичай має високий ступінь ліквідності, що дозволяє інвесторам швидко купувати та продавати акції за поточними ринковими цінами. Це робить фондовий ринок привабливим для тих, хто шукає можливість швидкої конверсії активів у готівку.
- Диверсифікація портфеля. Фондовий ринок надає широкий спектр інвестиційних можливостей, від акцій великих компаній до акцій малих та середніх підприємств, а також галузевих фондів. Це дозволяє інвесторам диверсифікувати свій портфель та розподілити ризики між різними активами.
- Володіння частковими акціями великих підприємств. На фондовому ринку інвестори можуть придбати частку у власності великих компаній, не обов'язково купуючи всю компанію. Це дозволяє інвесторам отримувати доходи від прибутку компанії та брати участь у зростанні її вартості.
- Доступ до інформації. Фондовий ринок є відкритим і прозорим, що забезпечує доступ інвесторів до великої інформації про компанії, ринкові тенденції, фінансові звіти тощо. Це допомагає інвесторам приймати свідомі інвестиційні рішення на основі фундаментального та технічного аналізу.
- Участь у економічному зростанні. Інвестування в акції компаній дозволяє інвесторам брати участь в економічному зростанні та розвитку, оскільки

прибуток компаній та зростання їх вартості безпосередньо пов'язані із загальним станом економіки.

Ці фактори роблять фондовий ринок місцем можливостей для трейдерів та інвесторів. Правильна робота з фондовим ринком - це ключ до фінансової незалежності, тому так багато людей прагнуть опанувати цю галузь.

1.2 Особливості роботи зі штучним інтелектом

Штучний інтелект - дуже потужний інструмент, який останнім часом став дуже популярним. Коли з’явився перший штучний інтелект він був примітивним, тому ніхто не звернув увагу. Хоча за весь час існування штучного інтелекту не було кардинальних змін у підході до навчання моделі, ця галузь знову набирає оберти. Але чому так сталось?

Справа в тім, що за останні приблизно 20-25 років в мережі інтернет з’явилось безліч даних, на яких стало можливим краще навчати модель. Тобто підхід до навчання не змінився, але завдяки великим об’ємам даних штучний інтелект знову набирає популярність і активно використовується у всіх сучасних галузях.

Тому першою особливістю при роботі зі штучним інтелектом є те, що потрібно мати багато даних для гарного прогнозування. Дані мають бути належної якості, щоб в них можна було знайти закономірність.

Обов’язково потрібно зазначити й те, що дані потрібно правильно опрацьовувати та підготовлювати для того, щоб використовувати їх для навчання. Тобто наступна особливість - опрацювання та підготовка даних. Виявлення та усунення шумів, викидів, сезонностей, тенденцій та іншого - це необхідний крок для роботи зі штучним інтелектом.

Ще одним важливим етапом при написанні штучного інтелекту є вибір правильної моделі. На сьогоднішній день існує безліч видів моделей і не всі вони

підходять під конкретну задачу. При написанні моделі потрібно чітко розуміти, що має робити модель, з якими даними працює, що ми очікуємо в результаті. Все це впливає на складність, розмір та тип моделі.

Ще одна особливість при написанні штучного інтелекту - це обладнання. Ходить думка, що конкуренти в сфері галузі штучного інтелекту - це конкуренти не тільки в обізнаності, а й в обладнанні. І я погоджуюсь з цією думкою. Навчання моделі - дуже довгий та ресурсозатратний процес. Так як модель опрацьовує великі об’єми даних багато разів вона може навчатись тижнями, або навіть місяцями, тому швидкість обробки даних - це один з ключових елементів при розробці штучного інтелекту. Для великих моделей потрібна потужна станція, яка може опрацьовувати багато інформації.

Останнє, про що я хотів би розповісти в цьому розділі - розуміння моделі. При роботі зі стандартними аналітичними інструментами завжди зрозуміло звідки береться результат і що він означає. При роботі з штучним інтелектом - це не завжди так. Дуже часто буває так, що модель занадто незрозуміла і важко точно трактувати її поведінку і значення, які вона видає. Це заважає правильно оцінити та зрозуміти, наскільки модель ефективна.

1.3 Переваги і недоліки застосування штучного інтелекту для прогнозування цін на фондовому ринку

Прогнозування на фондовому ринку за допомогою штучного інтелекту має як свої особливості, так і проблеми. У цьому розділі ми розглянемо основні переваги та недоліки цього підходу. При аналізі цього методу прогнозування було виявлено наступні недоліки:

- Складність моделювання. Фондові ринки характеризуються складними та непередбачуваними моделями. Штучний інтелект можна використовувати для побудови моделей, які можуть впоратися з цією складністю та адаптуватися до неї, але все одно може бути важко врахувати всі фактори, що впливають на рух цін на акції.

- Невизначеність і ризики. Навіть найточніші моделі прогнозування не можуть передбачити майбутні рухи ринку з абсолютною точністю. Фінансові ринки залежать від багатьох факторів, включаючи політичні рішення, макроекономічні події, зміни споживчого попиту, рішення впливових людей тощо, що робить прогнози завжди певним ступенем ризику.
- Перенавчання моделі. Використання складних алгоритмів машинного навчання може призвести до перенавчання моделі на історичних даних, що може зменшити її здатність узагальнювати нові дані та зробити прогнози менш точними.
- Високочастотний трейдинг і «шум». У високочастотній торгівлі навіть невеликі тимчасові затримки можуть стати критичними. Ринковий «шум», спричинений короткостроковими коливаннями цін і випадковими подіями, також може ускладнити прогнозування навіть для найдосконаліших моделей ШІ.
- Необхідність втручання людини. Хоча штучний інтелект може бути корисним інструментом для прогнозування фондового ринку, людський аналіз і досвід все ще важливі. Людський фактор дозволяє врахувати контекст, інтуїцію та нюанси, які можуть бути упущені алгоритмами.
- Проблема «чорної скриньки». Однією з проблем використання штучного інтелекту на фондовому ринку є те, що деякі закономірності важко зрозуміти та інтерпретувати. Це може призвести до проблеми «чорної скриньки», коли трейдери не можуть пояснити причину рішення алгоритму.
- Навчання та адаптація моделей. Незважаючи на те, що алгоритм здатний самонавчатися на нових даних, сьогодні це необхідність. Щоб забезпечити точність і релевантність прогнозів, моделі штучного інтелекту повинні постійно оновлюватися та адаптуватися до змін на ринку. Це вимагає постійного моніторингу та навчання моделей на нових даних.

- Залежність даних. Точність прогнозів фондового ринку залежить від якості та актуальності даних, які використовуються для навчання моделей штучного інтелекту. Недостатні або викривлені дані можуть призвести до неточних прогнозів і поганих результатів. Це одна з головних проблем роботи на фондовому ринку – велика частина даних значно спотворена через величезний природний шум фондового ринку, тому навчити модель на таких даних дуже складно. І навіть якщо ви навчитеся помічати загальні моделі поведінки, проблема шуму залишається, оскільки вони також існують у нових даних.

Попри всі недоліки є також і переваги:

- Обсяг даних. Фондові ринки щодня генерують величезну кількість даних, зокрема ціни на акції, обсяги торгів, новини, фінансові звіти тощо.
- Портфель і управління ризиками. Використання штучного інтелекту може полегшити управління портфелем і ризиками на фондовому ринку. Алгоритми можуть допомогти оптимізувати склад портфеля з урахуванням різних факторів, таких як ризики, доходи та ліквідність активів.
- Взаємодія з трейдерами. Штучний інтелект може працювати як інструмент підтримки для трейдерів, надаючи їм рекомендації та аналітику для прийняття рішень. Така взаємодія може підвищити ефективність торгівлі та знизити ризик помилок.
- Можливості предикативного аналізу. Використання штучного інтелекту дозволяє проводити прогнозний аналіз на фондовому ринку, що дозволяє ідентифікувати тенденції та тенденції, які можуть допомогти інвесторам приймати більш обґрунтовані рішення.
- Можливість автоматично вдосконалювати алгоритм на нових даних, що значно подовжує термін служби алгоритму.

Отже штучний інтелект - це не панацея, яка буде прогнозувати ціни акцій за аналітиків. Штучний інтелект - це ще один потужний інструмент, який при правильному використанні може дати перевагу перед використанням стандартних методів прогнозування.

1.4 Перший погляд та аналіз фондової біржі

Фондова біржа - це організація, де купуються та продаються цінні папери, такі як акції, облігації, кошти та інші фінансові інструменти. Це центральна платформа для торгівлі цими цінними паперами між різними учасниками ринку. Не плутати з фондовим ринком, фондовий ринок — це абстрактне поняття, яке описує всі біржі в цілому, а фондова біржа— це певна платформа, на якій можна купувати та продавати цінні папери.

Історично фондові біржі сформувалися як місця, де купці та продавці збиралися для торгівлі цінними паперами. Спочатку це могли бути звичайні зустрічі на вулицях або в кафе, а з часом для організації торгівлі створювалися спеціальні приміщення та будівлі.

Основною метою фондової біржі є забезпечення ефективної та прозорої торгівлі цінними паперами. Для цього біржа реалізує кілька ключових концепцій і принципів:

- Ціноутворення. Фондова біржа визначає ціни цінних паперів на основі попиту та пропозиції на ринку. Це відбувається через взаємодію учасників ринку – покупців і продавців.
- Прозорість. Біржа забезпечує прозорість угод шляхом публікації інформації про ціни та обсяги торгів. Це дозволяє учасникам ринку приймати обґрунтовані торгові рішення.
- Ліквідність. Біржа сприяє формуванню ліквідного ринку, на якому учасники можуть легко купувати та продавати цінні папери без істотного впливу на їхні ціни.
- Регулювання. Фондова біржа підлягає регулюванню з боку фінансових органів і регуляторів для забезпечення цілісності, стабільності та ефективності ринку.

Головною метою фондової біржі є створення умов для ефективного функціонування ринку цінних паперів, що забезпечує можливість купівлі-продажу цих цінних паперів на оптимальних умовах для всіх учасників.

У світі існує багато бірж, які дозволяють купувати та продавати цінні папери, ось список основних відмінностей:

- Географічне положення. Фондові біржі можуть бути розташовані в різних країнах або різних регіонах світу. Наприклад, існують американські біржі, розташовані в США, а також європейські, азіатські, африканські та інші регіональні біржі.
- Тип активів. Деякі біржі спеціалізуються на торгівлі акціями компаній, тоді як інші можуть зосередитися на товарних ф'ючерсах, облігаціях або інших фінансових інструментах.
- Розмір і ліквідність ринку. Деякі біржі можуть бути більшими та мати більший обсяг торгів, що робить їх більш ліквідними для інвесторів.
- Нормативні вимоги. Кожна фондова біржа має власні правила та нормативні вимоги, яким компанії повинні відповідати, щоб торгувати на ній.
- Години торгівлі. Часові пояси різних бірж можуть відрізнятися, що впливає на години роботи ринку та доступність для трейдерів.
- Технологічні можливості. Деякі біржі можуть мати досконалішу торгову технологію та інфраструктуру, що робить торгівлю швидшою та ефективнішою.

Ці різниці можуть впливати на те, наскільки привабливою є фондова біржа для різних видів інвесторів та компаній.

1.5 Огляд штучного інтелекту з технічної точки зору

Для того, щоб правильно застосувати штучний інтелект потрібно розуміти, як він працює, тому в цьому розділі подивимось, що таке штучний інтелект з точки зору реалізації, як він працює, які є моделі штучного інтелекту та коли їх потрібно застосовувати.

1.5.1 Що таке нейрон в контексті штучного інтелекту

Нейрон - це базовий обчислювальний вузол нейронної мережі. Він спроектований для моделювання функціонування нейронів у людському мозку. Нейрон має такі базові властивості:

- Вхідні сигнали - сигнали, які надходять до нейрону з нейронів попереднього шару. Кожен вхідний сигнал обчислюється за формулою
  $x * \omega$
- Функція активації - функція, яка акумулює всі вхідні сигнали. Вона рахує значення функції від суми вхідних сигналів. Наприклад найпростіша функція активації - лінійна, коли всі вхідні сигнали просто додаються.
- Вихідний сигнал - це результат функції активації, його нейрон посилає до нейронів наступного рівня.

Зауваження - нейрони вхідного шару не мають вхідного сигналу та функції активації.

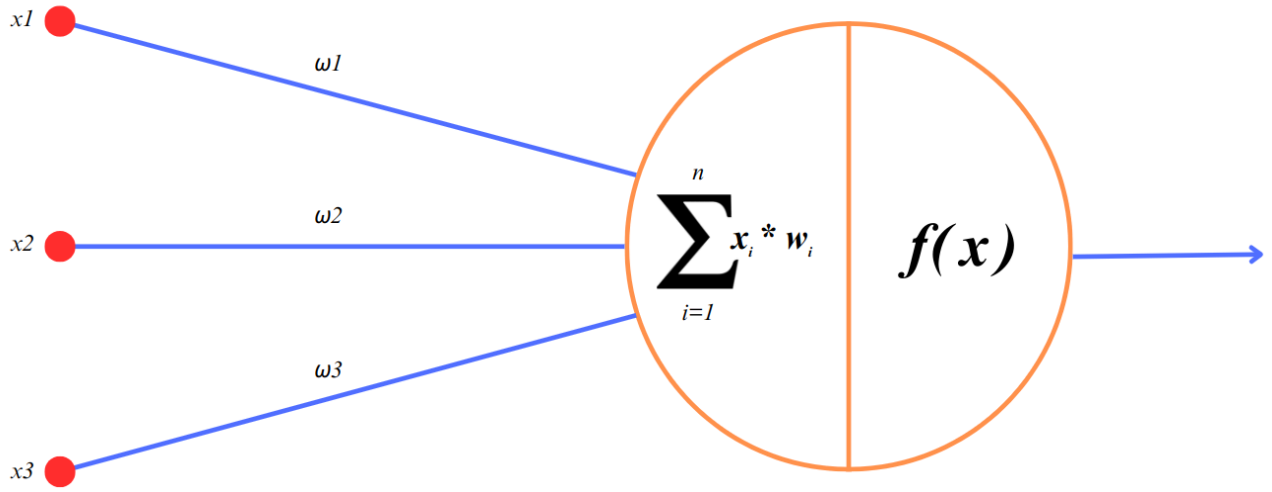


Рис. 1.1 Схема нейрону

1.5.2 Структура та принцип роботи повнозв'язної моделі штучного інтелекту

Повнозв'язна нейронна мережа прямого поширення - одна з найпопулярніших видів моделей штучного інтелекту. Суть її архітектури заключається у тому, що нейрони розбиваються на шари і кожен нейрон попереднього шару має зв'язок з кожним нейроном наступного шару.

На вхід подається $k + 1$ нейронів, де k - кількість вхідних параметрів. А останній нейрон - це нейрон зміщення, він дозволяє моделі вирішувати проблеми, такі як апріорна вага в кожному шарі мережі та здатність моделі адаптуватися до різних типів даних. Кількість прихованих шарів та кількість нейронів в кожному шарі обирає людина, яка створює нейронну мережу, точних правил їхнього створення немає. Кількість вихідних нейронів також обирає людина, але майже завжди вона залежить від задачі і очікуваного результату.

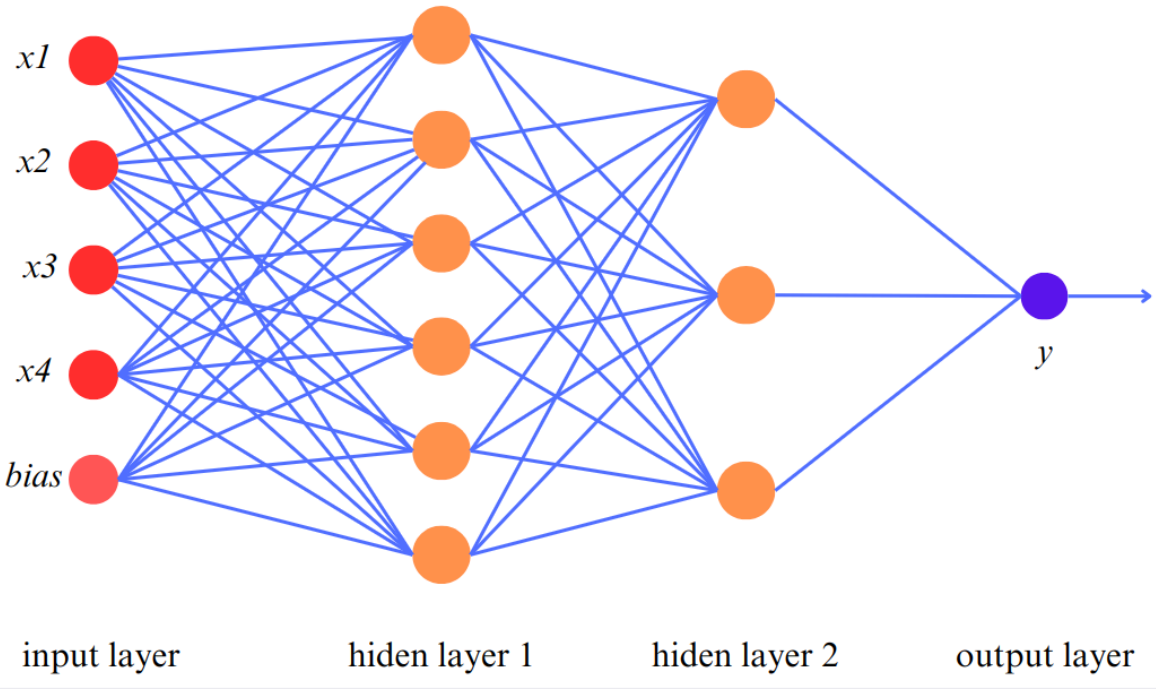


Рис. 1.2 Приклад повнозв’язної нейронної мережі

Важливо зазначити, що є зовнішні і внутрішні параметри нейронної мережі. Зовнішні - це ті, які обирає людина, яка пише модель. Наприклад кількість шарів, кількість нейронів, активаційні функції нейронів та інші. Внутрішні - це ті, які модель покращує під час навчання, а саме - ваги кожного з’єднання між нейронами. Тобто основною метою навчання нейромережі є знаходження таких вагових коефіцієнтів, при яких модель на виході буде давати значення, які задовольняють умови задачі.

Для рішення цієї задачі зараз найпопулярнішим є алгоритм зворотного розповсюдження (англ. back propagation). Він, в свою чергу, базується на алгоритмі градієнтного спуску.

Як працює алгоритм:

1. Спочатку шукаємо помилку відповіді по формулі

$$e = y - d, \quad (1)$$

де e - помилка, y - вихідне значення, d - очікуване вихідне значення.

2. Далі починається процес коригування вагових коефіцієнтів. Ми починаємо корегувати з останнього шару і поступово рухаємось до початку моделі.

Потрібно розрахувати локальний градієнт для останнього шару, він розраховується за такою формулою

$$\delta = e * f'(v) \quad (2)$$

де δ - локальний градієнт, e - помилка моделі, яку ми розрахували на попередньому кроці, f' - похідна функції активації нейрона від, v - вхідна сума на нейрон.

Так як ми беремо похідну від функції активації, то функція активації має бути диференційованою.

3. Якщо за функцію активації взяти логістичну функцію

$$f(x) = \frac{1}{1 + e^{x}}, \quad (3)$$

тоді

$$f'(x) = f(x) * (1 - f(x)), \quad (4)$$

якщо під x підставити вхідну суму v, яка приходить на нейрон, тоді f(x) - це функція активації нейрону від суми, яка приходить на цей нейрон, тобто - вихідне значення нейрону. Так як ми говоримо про нейрон вихідного шару, тоді f(v) = y - результат розрахунків моделі, тоді

$$f'(v) = f(v) * (1 - f(v)) \quad (5)$$

підставимо формулу 5 в формулу 2 і отримаємо

$$\delta = e * f(v) * (1 - f(v)) \quad (6).$$

4. Далі починаємо коригувати вхідні вагові коефіцієнти. Вони коригуються за формулою

$$w_{new} = w - \lambda * \delta * f_{prev}(v_{prev}), \quad (7)$$

де w_{new} - нове значення вагового коефіцієнта, w - поточне значення вагового коефіцієнта, λ - крок збіжності. Ще один зовнішній параметр, який обирають при навчанні для того, щоб наближатись до локального мінімуму. Загальна рекомендація така - використовувати значення 0.1, 0.01, 0.001 і подібні. f_{prev} - функція активації попереднього нейрону, v_{prev} - вхідна сума на попередній нейрон.

5. Далі потрібно порахувати локальний градієнт для нейрона попереднього шара δ_{prev}. Він рахується по формулі

$$\delta_{prev} = \delta * w_{new} \quad (8).$$

Далі повторюємо кроки 3-5 для всіх нейронів, поки не пройдемось по всій моделі

Розглянемо приклад застосування алгоритму для такої моделі

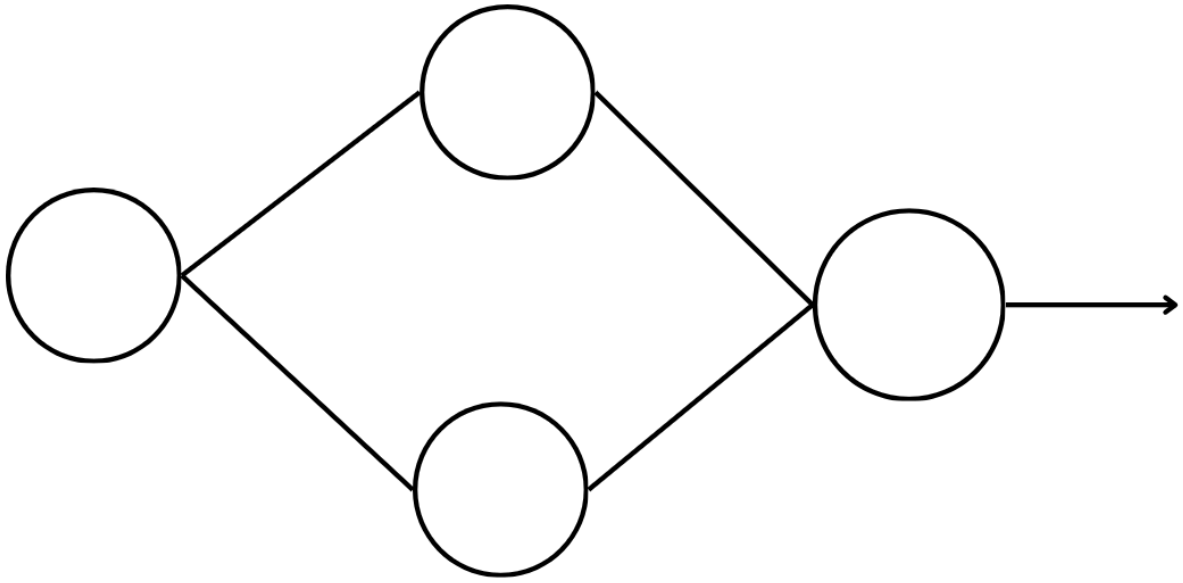


Рис. 1.3 Приклад застосування алгоритму back propagation

Позначимо вхідні суми, функції активації, вагові коефіцієнти для останніх шарів, також позначимо результат моделі і очікуваний результат моделі

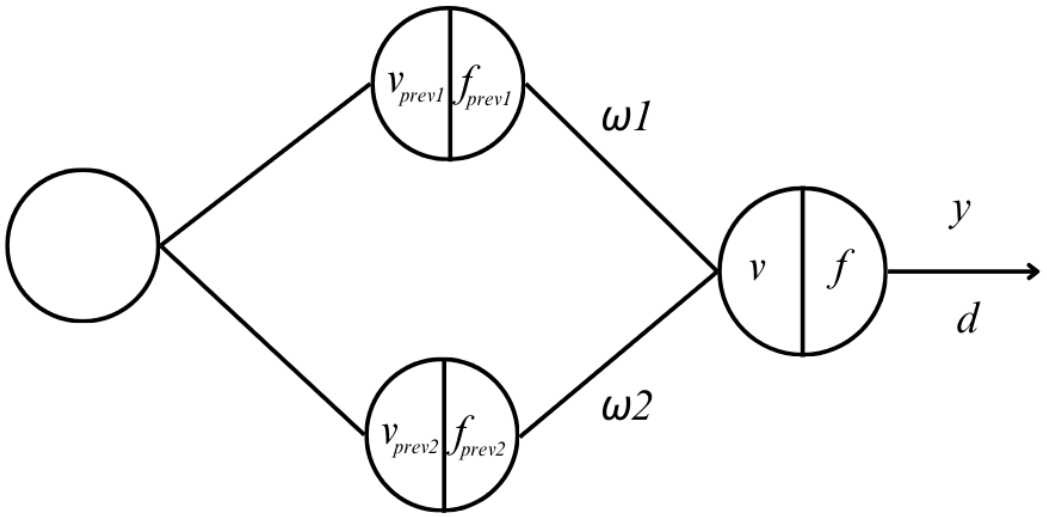


Рис. 1.4 Модель з позначеними математичними даними для двох останніх шарів

Розрахуємо помилку e по формулі (1)

$$e = y - d$$

Далі розрахуємо локальний градієнт δ по формулі (6)

$$\delta = e * f(v) * (1 - f(v))$$

Далі розрахужмо нові вагові коефіцієнти по формулі (7)

$$w1_{new} = w1 - \lambda * \delta * f_{prev1}(v_{prev1})$$
$$w2_{new} = w2 - \lambda * \delta * f_{prev2}(v_{prev2})$$

Порахуємо локальний градієнт для попереднього шару по формулі (8)

$$\delta_{prev1} = \delta * w1_{new}$$
$$\delta_{prev2} = \delta * w2_{new}$$

Тепер перейдемо до попереднього шару

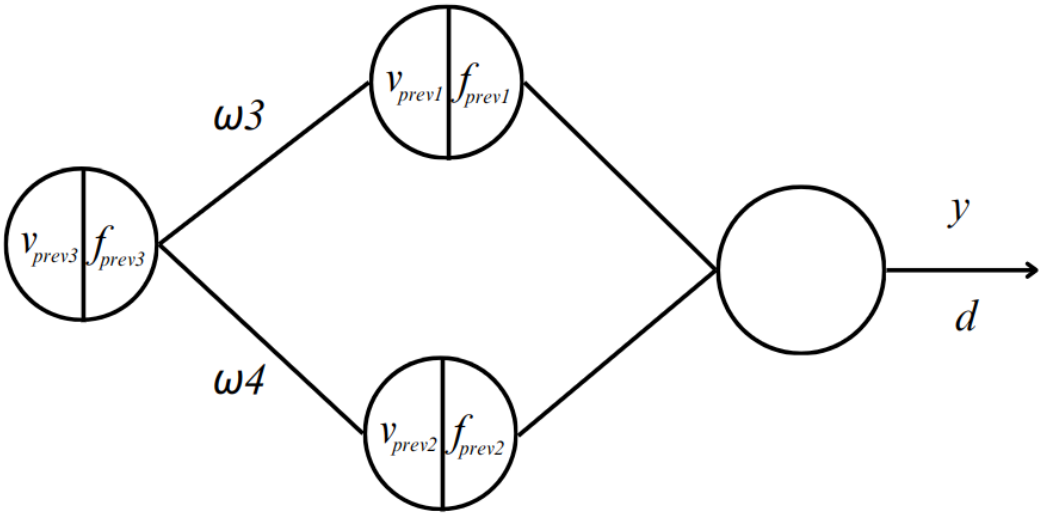


Рис. 1.5 Модель з позначеними математичними даними для двох перших шарів

По аналогії до попереднього кроку рахуємо вагові нові вагові коефіцієнти $w3$ та $w4$.

$$w3_{new} = w3 - \lambda * \delta_{prev1} * f_{prev3}(v_{prev3})$$
$$w4_{new} = w4 - \lambda * \delta_{prev2} * f_{prev3}(v_{prev3})$$

Тепер щоб порахувати градієнт нейрону prev3 потрібно порахувати градієнти на кожному ребрі w3 та w4:

$$\delta^{w3}_{prev3} = \delta_{prev1} * w3_{new}$$
$$\delta^{w4}_{prev3} = \delta_{prev2} * w4_{new}$$

Тоді градієнт нейрону prev3 буде їхньою сумою:

$$\delta_{prev3} = \delta^{w3}_{prev3} + \delta^{w4}_{prev3}$$

Хоча це і основний алгоритм коригування вагових коефіцієнтів він має деякі недоліки:

- Перший недолік пов’язаний з алгоритмом градієнтного списку. Алгоритм градієнтного спуску шукає не глобальний мінімум, а локальний. Тобто буває так, що ми коригуємо ваги і в результаті отримуємо не найкращий результат.

Щоб мінімізувати цей недолік потрібно запускати один і той самий алгоритм декілька разів з різними початковими ваговими коефіцієнтами.

- Якщо шарів нейронів дуже багато (більше 8) на останніх кроках коригування вагових коефіцієнтів стає дуже малим, що не є ефективним.

1.5.3 Інші моделі штучного інтелекту

1.5.3.1 Згорткова нейронна мережа.

Ця модель спеціалізується на обробці зображень. Вона використовує згорткові шари для виявлення різних ознак у вхідних зображеннях та пулінгові шари для зменшення просторових розмірів ознак, щоб зробити їх більш загальними. Таким чином ця модель робить висновки на знайдених чи не знайдених ознаках на малюнку.

1.5.3.2 Автокодер.

Це тип нейронної мережі, яка навчається репрезентувати вхідні дані у вигляді більш компактного коду. Основна ідея автокодера полягає в тому, щоб навчити модель зменшувати розмірність вхідних даних, а потім відновлювати їх з цього компактного коду з якомога меншою втратою інформації. Її структура включає дві основні частини - енкодер та декодер. Енкодер - це згорткова нейронна мережа, яка намагається стиснути вхідні дані без втрати інформації. Декодер - це розгорткова нейронна мережа, вона повертає дані в першопочатковий стан. Така мережа застосовується для компресії даних, відшумлення даних, візуалізації даних.

1.5.3.3 Звичайна рекурентна нейронна мережа (RNN).

Це тип нейронної мережі, який призначений для роботи з послідовними даними. Кожен нейрон у RNN приймає вхідні дані та внутрішні стани, і використовує їх для обчислення виходу, що також стає внутрішнім станом для наступного кроку часу. RNN можуть фіксувати короткочасні залежності в послідовних даних, але їм важко фіксувати довгострокові залежності.

Основна відмінність рекурентних нейронних мереж - рекурентні нейрони. Ці нейрони мають внутрішній стан та подають попередню відповідь на себе знову, що дозволяє використовувати попередні дані в наступний момент часу. Рекурентний нейрон можна розглядати, як декілька копій однакових нейронів.

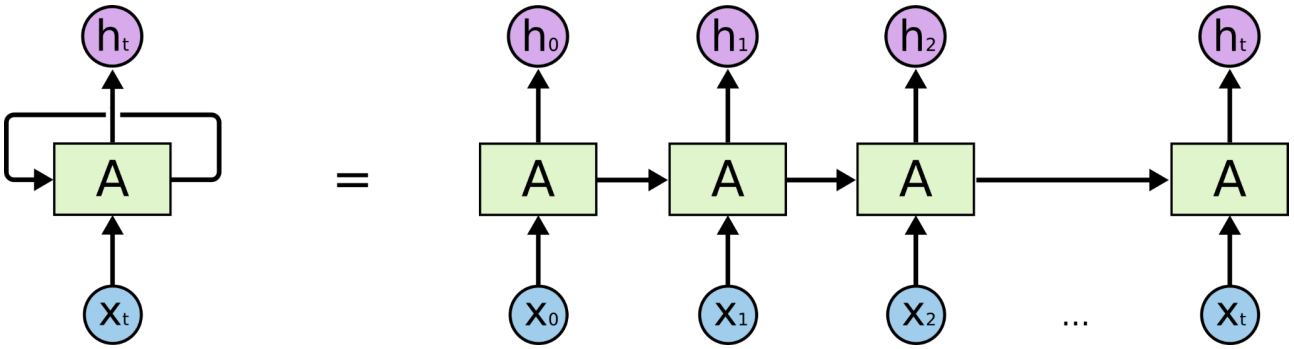


Рис. 1.6 Схематичне зображення розгорнутого рекурентного нейрону

На рисунку 1.6 бачимо загальну структуру будь-якої рекурентної моделі, вони всі відрізняються наповненням блоку A. наприклад у звичайній RNN він виглядає так:

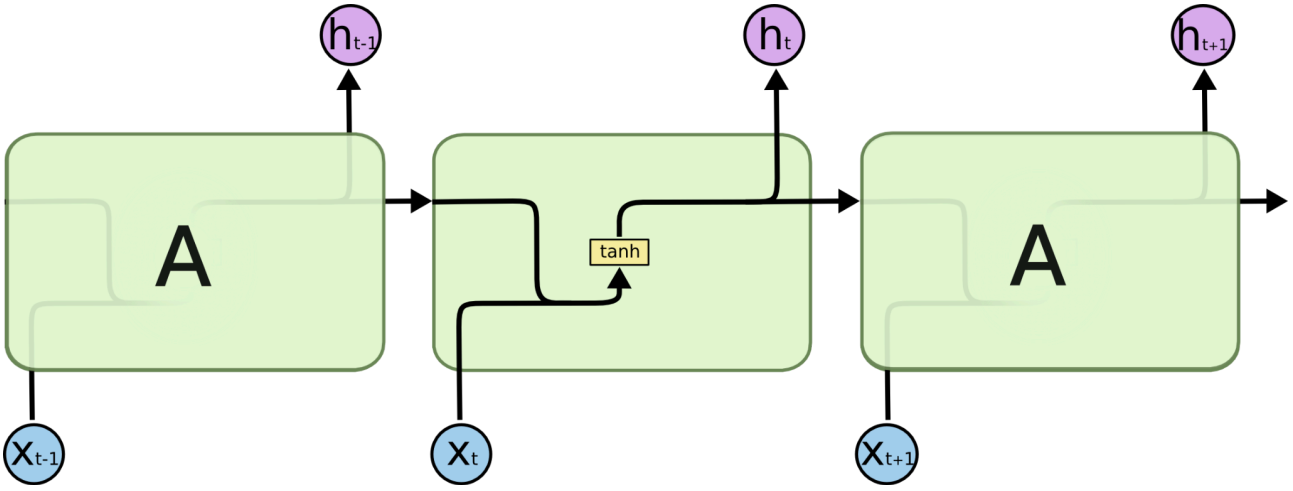


Рис. 1.7 Схематичне зображення рекурентного нейрону з деталізацією блоку A

Рис. 1.8 Умовні позначення на схемах

Як ми бачимо, у середині блок працює досить просто - конкатенує вхідне значення та значення з попередньої ітерації та заносить їх у нейрон з гіперболічну функцію активації і передає відповідь у наступний нейрон та вихід поточного нейрону.

1.5.3.4 Варіації рекурентних нейронних мереж.

Ці моделі, такі як LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit), вдосконалюють звичайні RNN, дозволяючи краще контролювати та зберігати інформацію на протязі довгого часу, що допомагає аналізувати послідовні дані, такі як часові ряди, текст і мова. Вони використовують комірку пам’яті та шлюзи для керування потоком інформації, що дозволяє їм вибірково

зберігати або відкидати інформацію за потреби та таким чином уникати проблеми зникнення градієнта, яка заважають традиційній RNN.

1.5.3.4.1 Long Short-Term Memory (LSTM).

Структура мережі LSTM складається з серії комірок LSTM, кожна з яких має набір шлюзів(вхідні забуття, вихідні забуття, шлюзи забуття), які контролюють потік інформації в комірку та з неї. Шлюзи використовуються для вибіркового забування, або збереження інформації з попередніх часових кроків, дозволяючи LSTM підтримувати довгострокові залежності у вхідних даних.

Три шлюзи вхідний, забуття і вихідний реалізовані за допомогою сигмоїдних функцій, які створюють вихідні дані між 0 і 1. Ці шлюзи навчаються за допомогою алгоритму зворотного поширення.

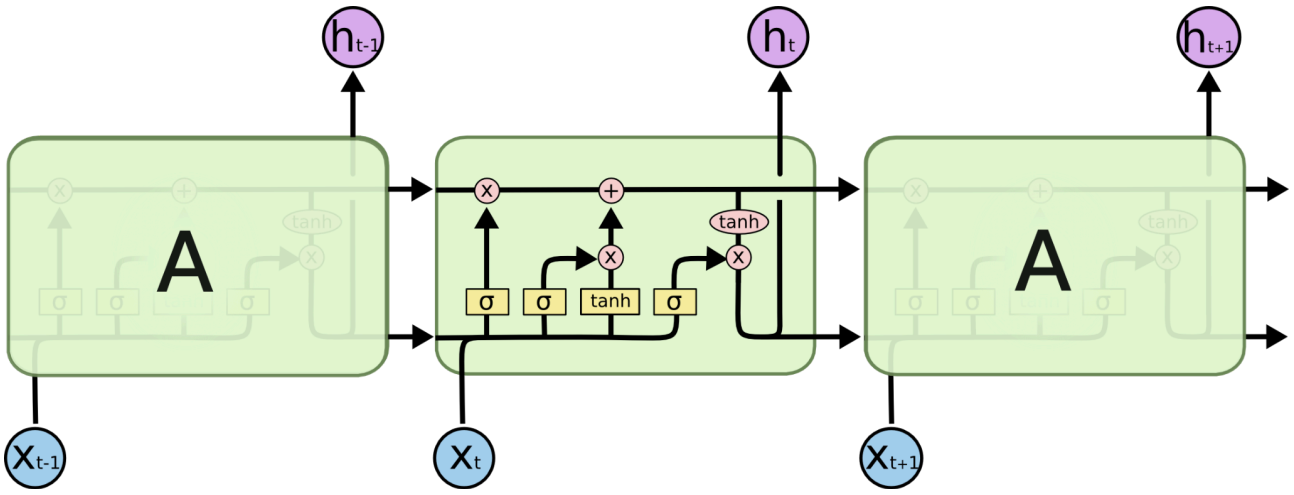


Рис. 1.9 Схематичне зображення шару LSTM

1.5.3.4.2 Gated Recurrent Unit (GRU).

Однією з новітніх варіацій блоку LSTM є блок GRU. Він об’єднує вхідний шлюз та шлюз забуття в єдиний шлюз оновлення. Також він об’єднує стан комірки та прихований стан і вносить деякі інші зміни. Отримана модель простіша ніж стандартна модель LSTM, вона краща на короткострокових залежностях та

швидше навчається, однак LSTM може бути більш ефективною для завдань, які вимагають довгострокових залежностей.

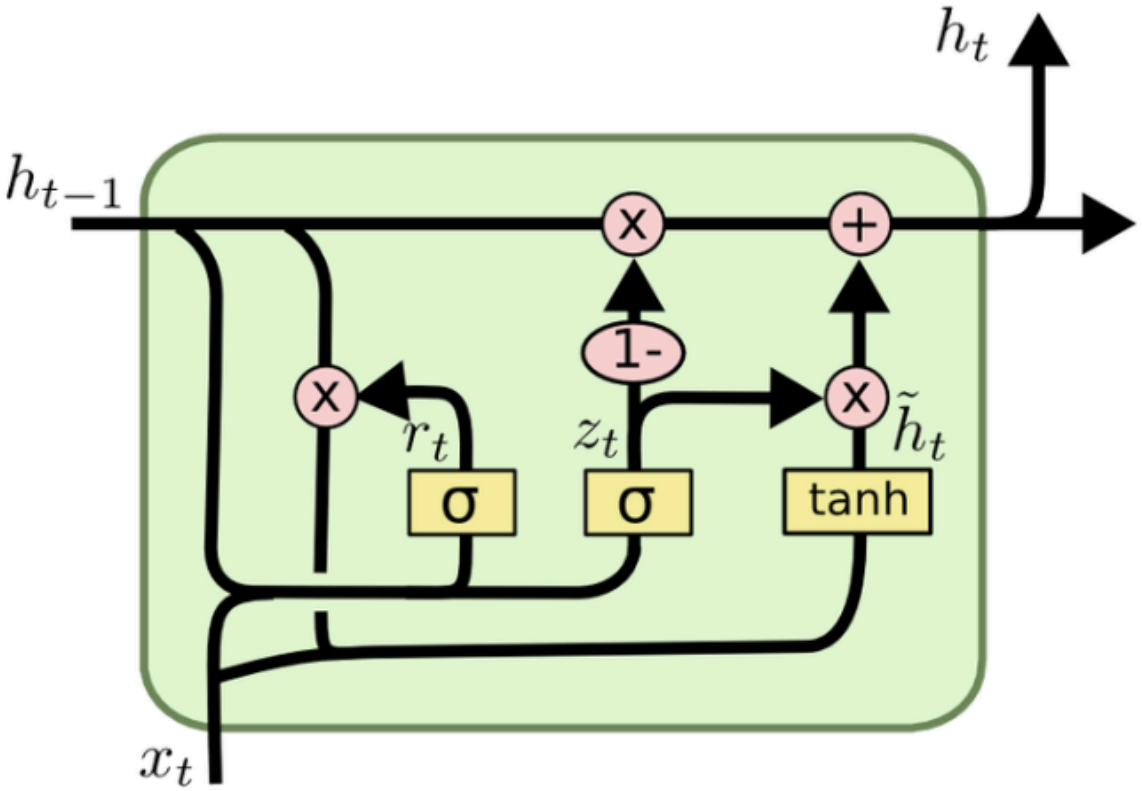


Рис. 1.10 Схематичне зображення шару GRU

1.6 Визначення вимог до програмного забезпечення

Функціональні вимоги:

- Згідно аналізу проведеному в цьому розділі потрібно створити алгоритм, який буде прогнозувати часові ряди на фондовому ринку на основі часових статистичних даних про купівлю та продаж за попередні роки.
- На їхній основі потрібно розрахувати індикатори та застосувати їх для навчання моделі.
- Розрахувати показники якості моделі.
- Створювати моделі нейронних мереж з конфігураціями користувача.
- Перегляд результатів навчання моделі нейронної мережі.
- Написати програмне забезпечення, спрямоване на прогнозування часових рядів на фондовому ринку, яке підтримує автоматичну підготовку даних,

збереження моделей, тренування моделей та інший функціонал, який спростить створення моделей.

Нефункціональні вимоги:

- Інтерфейс англійською мовою
- Підтримка основних браузерів - Chrome, Opera, Firefox

2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Огляд програмних засобів реалізації

2.1.1 Мова програмування Python

Python — це інтерпретована об’єктно-орієнтована мова програмування з простим і зрозумілим синтаксисом, що робить її доступною для розробників-початківців. Вона була створений Гвідо ван Россумом і вперше випущена у 1991 році.

Сильні сторони Python включають:

- Простота і зручність. Python відомий своєю простотою вивчення та читання коду. Це робить його чудовим вибором для початківців і дозволяє швидко розробляти програми.
- Багата стандартна бібліотека. Python поставляється з великою стандартною бібліотекою, яка включає багато корисних модулів і інструментів для різноманітних завдань, від роботи з файлами та мережами до обробки даних і створення веб-додатків.
- Широкі повноваження. Python підтримує різноманітні стилі програмування, включаючи процедурне, об’єктно-орієнтоване та функціональне програмування. Це дозволяє розробникам вибрати підхід, який найкраще підходить для конкретного завдання.
- Популярність і активна спільнота. Python є однією з найпопулярніших мов програмування у світі. Його активна спільнота розробників створює та підтримує різноманітні бібліотеки, фреймворки та інструменти, що робить його ідеальним вибором для розробки різних типів програм.

У світі штучного інтелекту, Python став однією з найпопулярніших мов програмування завдяки наступним причинам:

- Бібліотеки машинного та глибокого навчання. У Python є багато бібліотек, таких як TensorFlow, PyTorch, scikit-learn і Keras, які надають інструменти для проектування та реалізації алгоритмів машинного та глибокого навчання.
- Легкість прототипування. Python дозволяє швидко створювати прототипи та експериментувати з різними моделями машинного навчання завдяки його простоті та широкому вибору бібліотек.
- Інтеграція з іншими технологіями. Python легко інтегрується з іншими технологіями, такими як бази даних, веб-сервіси та хмарні платформи, що робить його універсальним інструментом для розробки та впровадження програм штучного інтелекту.
- Загальна доступність. Python — це відкрита та безкоштовна мова програмування, яка дозволяє дослідникам і розробникам вільно ділитися своїми знаннями, кодом і рішеннями, роблячи внесок у розвиток спільноти штучного інтелекту.

2.1.2 Середовище розробки PyCharm

PyCharm — це інтегроване середовище розробки (IDE) для мови програмування Python, розроблене JetBrains. Воно надає розробникам широкий набір інструментів і функцій для легкого створення, налагодження та керування проектами Python. Нижче наведено деякі сильні сторони PyCharm:

- Потужний редактор коду. PyCharm має потужний та інтуїтивно зрозумілий редактор коду з підсвічуванням синтаксису, автозавершенням, швидкими комітами та іншими функціями, які покращують процес написання коду.
- Підтримка кількох версій Python. PyCharm дозволяє легко перемикатися між різними версіями Python і створювати віртуальні середовища, що особливо корисно під час роботи над проектами, для яких потрібна певна версія мови.

- Інтегровані засоби налагодження. PyCharm надає широкий набір інструментів для налагодження коду, включаючи точки зупину, покрокове виконання, перегляд значень змінних тощо, допомагаючи розробникам швидко знаходити та виправляти помилки.
- Підтримка різних фреймворків і бібліотек. PyCharm інтегрується з популярними фреймворками та бібліотеками Python, такими як Django, Flask, SQLAlchemy, NumPy, Pandas та багатьма іншими, що полегшує роботу з ними та підвищує продуктивність розробника.
- Управління проектами. PyCharm надає зручні інструменти управління проектами, включаючи системи контролю версій, інтеграцію Git, підтримку віртуальних середовищ, керування залежностями та інші функції, які допомагають організувати та керувати проектами будь-якої складності.
- Можливість розширення та налаштування. PyCharm підтримує розширення функціональності за допомогою плагінів, що дозволяє налаштувати середовище розробки під конкретні потреби та вподобання розробника.

Загалом PyCharm — це потужний інструмент розробки додатків Python, який забезпечує зручність, продуктивність і ефективність для розробників.

2.1.3 Бібліотеки NumPy та Pandas

Бібліотеки NumPy і Pandas є двома ключовими інструментами для роботи з даними на мові програмування Python, особливо в галузі наукових обчислень, аналізу даних і машинного навчання.

NumPy — це основна бібліотека для обчислення багатовимірних масивів і матриць у Python. Він забезпечує високопродуктивні структури даних і функції для роботи з цими даними. NumPy надає багато операцій лінійної алгебри, випадкових чисел, перетворень масивів тощо. Його основним типом даних є багатовимірний масив ndarray.

Pandas — це бібліотека Python, створена на основі NumPy, яка надає високорівневі структури даних і інструменти аналізу даних. Основними структурами даних Pandas є Series (одновимірний масив з мітками) і DataFrame (двовимірна таблиця даних з мітками рядків і стовпців). Pandas надає потужні інструменти для об’єднання та обробки даних, фільтрації, агрегації, візуалізації тощо.

Основні переваги:

- Ефективність і продуктивність. NumPy і Pandas реалізовані на мові програмування C, що забезпечує високу продуктивність операцій з даними.
- Простота і зручність. Обидві бібліотеки забезпечують прості та інтуїтивно зрозумілі інтерфейси для роботи з даними, що робить їх доступними для широкого кола користувачів, у тому числі для початківців.
- Широкий спектр функцій. NumPy і Pandas надають широкий набір функцій для роботи з даними, включаючи математичні операції, фільтрацію, сортування, групування, агрегацію тощо.
- Інтеграція з іншими бібліотеками. Обидві бібліотеки інтегруються з іншими популярними бібліотеками Python, такими як scikit-learn, Matplotlib, Seaborn та іншими, що дозволяє використовувати їх у різноманітних сценаріях інтелектуального аналізу даних і машинного навчання.
- Підтримка часових рядів. Pandas має спеціалізовані структури даних і функції для роботи з часовими рядами, що робить його ідеальним інструментом для аналізу даних часу, таких як фінансові дані, дані про продажі тощо.

Загалом, NumPy і Pandas — потужні інструменти для роботи з даними на Python, які забезпечують високу продуктивність, простоту використання та широкий набір функцій для аналізу та обробки даних.

2.1.4 Бібліотека MatPlotLib

Бібліотека Matplotlib — це бібліотека Python для візуалізації даних. Він пропонує широкий спектр інструментів для створення різних типів графіків, діаграм та інших візуальних представлень даних.

Ось деякі з основних можливостей та переваг використання Matplotlib:

- Створення різноманітних графіків. Matplotlib дозволяє створювати багато типів графіків, включаючи лінійні графіки, точкові діаграми, гістограми, кругові діаграми, гістограми, контурні діаграми, теплові карти тощо. Це робить бібліотеку універсальним інструментом для візуалізації даних різних типів і форматів.
- Простота використання. Matplotlib надає простий та інтуїтивно зрозумілий інтерфейс для створення графіків. Він дозволяє легко налаштовувати різні аспекти візуалізації, такі як кольори, стилі ліній, маркери, шрифти тощо, що робить його гнучким і зручним інструментом для створення красивих та інформативних графіків.
- Кросплатформенність. Matplotlib працює на різних операційних системах, включаючи Windows, MacOS і Linux, що робить його доступним для широкого кола користувачів.
- Інтеграція з іншими бібліотеками. Matplotlib інтегрується з іншими популярними бібліотеками Python для аналізу даних і машинного навчання, такими як NumPy, Pandas і scikit-learn. Це забезпечує можливість використовувати його в різних сценаріях аналізу даних і візуалізації результатів.
- Гнучкі можливості налаштування. Matplotlib надає багато параметрів і функцій для налаштування зовнішнього вигляду графіків. Це включає в себе можливість додавати заголовки, мітки осей, легенди, сітки, різні типи штрихів тощо, що дозволяє створювати графіки професійного вигляду з мінімальними зусиллями.

Загалом, Matplotlib — це потужний інструмент візуалізації даних на Python, який дозволяє створювати красиві та інформативні графіки для аналізу та візуалізації різних типів даних.

2.1.5 Бібліотеки Tensorflow та Keras

Бібліотеки TensorFlow і Keras є двома ключовими інструментами для розробки та впровадження моделей глибокого навчання та штучних нейронних мереж на мові програмування Python.

TensorFlow — це бібліотека машинного навчання з відкритим кодом, розроблена Google. Він надає інструменти для побудови та навчання різних типів моделей машинного навчання, включаючи нейронні мережі, а також інструменти для розгортання моделей на різних платформах. TensorFlow пропонує низькорівневі операції для роботи з тензорами (багатовимірними масивами даних) і обчисленнями графів, що робить його потужним інструментом для створення складних моделей глибокого навчання.

Keras — це інтерфейс високого рівня для роботи з нейронними мережами, побудований на основі бібліотеки TensorFlow. Keras забезпечує простий та інтуїтивно зрозумілий інтерфейс для визначення та навчання нейронних мереж, що робить його ідеальним інструментом як для початківців, так і для досвідчених розробників. Він забезпечує абстракції високого рівня для побудови різних типів нейронних мереж, включаючи повні мережі, згорточні нейронні мережі (CNN), рекурентні нейронні мережі (RNN) і багатошарові персептрони (MLP).

Нижче наведено основні переваги використання TensorFlow та Keras:

- Гнучкість і зручність. TensorFlow і Keras надають гнучкі та зручні інструменти для створення та навчання моделей машинного навчання. Keras надає простий та інтуїтивно зрозумілий інтерфейс для визначення та навчання моделей нейронної мережі, тоді як TensorFlow надає більш низькорівневі операції для роботи з даними та обчислення графів.

- Продуктивність. TensorFlow і Keras забезпечують високу продуктивність у навчанні та виконанні моделей машинного навчання завдяки оптимізованим алгоритмам і можливості використання апаратного прискорення, такого як GPU і TPU.
- Широкий вибір моделей. TensorFlow і Keras підтримують різні типи моделей машинного навчання, включаючи нейронні мережі глибокого навчання, рекурентні нейронні мережі, згорточні нейронні мережі та інші, дозволяючи розробникам вибирати найбільш відповідний тип моделі для свого завдання.
- Спільнота та підтримка. TensorFlow і Keras мають активні спільноти розробників і обширну документацію, що забезпечує доступ до широкого спектру ресурсів і допомогу у використанні цих інструментів.

Загалом TensorFlow і Keras є потужними інструментами для розробки та навчання моделей машинного навчання, які забезпечують гнучкість, продуктивність і простоту використання під час побудови різних типів штучних нейронних мереж.

2.1.6 Бібліотека TaLib

Бібліотека TaLib для Python — це інструмент, який надає функції для технічного аналізу фінансових ринків. TaLib розшифровується як "Technical Analysis Library" (бібліотека технічного аналізу). Вона надає низку популярних і стандартних індикаторів, таких як RSI (індекс відносної сили), MACD (розбіжність ковзної середньої конвергенції), смуги Болліндже́ра (пов’язані з розподілом Гауса) та багато інших.

Ця бібліотека дозволяє аналізувати фінансові дані, такі як ціни на акції чи обмінні курси, і застосовувати різні технічні індикатори для прогнозування та прийняття рішень щодо торгівлі на ринках. Багато трейдерів і аналітиків використовують TaLib для створення торгових стратегій і аналізу ринку.

2.1.7 Бібліотека sklearn

Бібліотека scikit-learn (Sklearn) для мови програмування Python є однією з найпопулярніших бібліотек машинного навчання. Вона надає широкий спектр інструментів для класифікації, регресії, кластеризації та інших контрольованих і неконтрольованих методів навчання, включаючи опорні вектори, дерева рішень, метод опорних векторів (SVM), алгоритми кластеризації, контрольоване та неконтрольоване навчання та багато іншого

Sklearn також надає інструменти для попередньої обробки даних, включаючи вибір функцій, масштабування даних і зменшення розмірності. Вона має простий і зрозумілий інтерфейс, який дозволяє легко використовувати різні моделі машинного навчання для вирішення різноманітних завдань аналізу даних і прогнозування.

За допомогою scikit-learn ви можете легко виконати весь цикл роботи з даними, від підготовки даних до побудови моделі, оцінки продуктивності та налаштування параметрів. Це дозволяє досліджувати, розробляти та впроваджувати різні алгоритми машинного навчання в широкому діапазоні завдань.

2.1.8 Фреймворк FastAPI

FastAPI — це сучасний веб-фреймворк для створення API на Python. Його ключові особливості включають високу швидкість, простоту використання та підтримку асинхронного програмування. FastAPI побудовано на основі стандарту OpenAPI та JSON Schema, яка забезпечує автоматичне створення документації API та валідаторів даних.

Основні переваги FastAPI:

- Швидкість. Однією з головних переваг FastAPI є його продуктивність. Завдяки використанню асинхронних функцій і підтримці ASGI

(Asynchronous Server Gateway Interface) він може обробляти велику кількість запитів з мінімальною затримкою.

- Автоматичне формування документації. FastAPI автоматично створює інтерактивну документацію API на основі стандартів OpenAPI та Swagger. Це значно полегшує тестування та використання API.
- Типізація. Використання анотацій типу Python дозволяє FastAPI автоматично перевіряти вхідні дані та генерувати схему JSON. Це забезпечує високу надійність коду та спрощує налагодження.
- Простота використання. Незважаючи на потужні можливості, FastAPI дуже простий у використанні. Він має зрозумілий і добре задокументований інтерфейс, що робить його доступним як для початківців, так і для досвідчених розробників.

2.1.9 Бібліотека React

React — популярна бібліотека JavaScript для створення інтерфейсів користувача (UI). Він був створений Facebook і тепер підтримується як Facebook, так і спільнотою розробників. React дозволяє вам створювати компоненти — ізольовані багаторазові шматки коду, які керують власним станом і можуть бути зібрані в складніші інтерфейси.

Основні особливості React:

- Компонентний підхід. React заснований на компонентах. Кожен компонент може мати власний стан і властивості. Компоненти можна легко комбінувати та повторно використовувати, що полегшує розробку модульного та масштабованого коду.
- Віртуальний DOM. React використовує віртуальний DOM для підвищення продуктивності. Замість того, щоб безпосередньо маніпулювати реальним DOM, React працює з його віртуальним представленням. Коли стан компонента змінюється, React оновлює віртуальний DOM, визначає мінімальні зміни, необхідні для реального DOM, і застосовує їх.

- Односпрямований потік даних. У React дані передаються від батьківських компонентів до дочірніх через властивості. Це забезпечує передбачуваність і полегшує розуміння структури даних у програмі.
- JSX. React використовує JSX, синтаксичне розширення JavaScript, яке дозволяє писати HTML-подібний код у JavaScript. JSX спрощує створення ієрархій компонентів і робить код більш читабельним.

2.1.10 Бібліотека MUI/MUI X

MUI, раніше відомий як Material-UI, — це популярна бібліотека компонентів для React, яка надає готові до використання компоненти, оформлені відповідно до системи дизайну Google Material Design. Ця бібліотека дозволяє розробникам швидко створювати красиві та функціональні інтерфейси користувача.

Основні компоненти MUI:

- Компоненти інтерфейсу користувача. MUI надає великий набір готових компонентів, таких як кнопки, текстові поля, діалогові вікна, таблиці, панелі навігації та багато інших. Ці компоненти легко налаштовуються та підтримують теми.
- Тематизація. MUI дозволяє створювати власні теми, які можна застосовувати до всієї програми або до окремих компонентів. Це забезпечує єдиний стиль для всього проекту.
- Сумісність із Material Design. Компоненти MUI відповідають рекомендаціям Material Design, що допомагає створювати інтерфейси, які виглядають сучасно та привабливо.

MUI X — це розширення бібліотеки MUI, яке включає додаткові компоненти та функції, які не входять до базового набору MUI.

MUI X фокусується на більш складних компонентах, таких як:

DataGrid. Потужний компонент для роботи з таблицями даних. Підтримує сортування, фільтрацію, розбивку на сторінки, редагування, групування та інші додаткові функції.

Засоби вибору дати та часу. Компоненти для засобів вибору дати й часу, які підтримують різні формати та локалі.

Діаграми. Компоненти графіків і діаграм для легкої візуалізації даних.

Компоненти преміум-класу: компоненти, які надають додаткові функції та функції, доступні в платних версіях бібліотеки.

2.2 Моделювання вимог до програмного забезпечення

Діаграма варіантів використання — це один із типів діаграм UML (Unified Modeling Language), які використовуються для моделювання взаємодії між користувачами (акторами) системи та самою системою. Він дозволяє відобразити, як користувачі взаємодіють із системою для досягнення певних цілей. Основна мета діаграми використання — описати функціональні можливості системи з точки зору користувача та визначити різні сценарії використання (випадки використання). Кожен варіант використання описує конкретні функціональні можливості системи та взаємодії з акторами.

Елементи Use Case Diagram:

- Актори (Actors). Актори — це будь-які зовнішні сутності, такі як користувачі, системи, які взаємодіють із системою для виконання певних функцій.
- Використання (Use Cases). Використання описує певні функції або дії, які може виконувати система. Кожен варіант використання пов’язаний з одним або декількома акторами та показує, як актори взаємодіють із системою. Використання представлено у вигляді овалів або фігур з назвами.
- Зв'язки (Relationships). Відносини використовуються, щоб показати зв’язок між акторами та використанням. Основні з'єднання включають:

- Відношення між актором і використанням. Показує, як актор взаємодіє з використанням для виконання певної дії.
- Спадкування використання. Використовується для вказівки використання, яке успадковує функції від іншого використання.

- Системна межа (System Boundary). Це прямокутник або еліпс, який представляє межі системи та окремих користувачів.
- Зв'язок між використаннями. Це зв’язки, які можуть існувати між різними використаннями, такими як «включати», «розширювати» тощо.

Ці елементи спільно створюють структуру діаграми використання, яка допомагає відобразити функціональність системи та взаємодію з її користувачами.

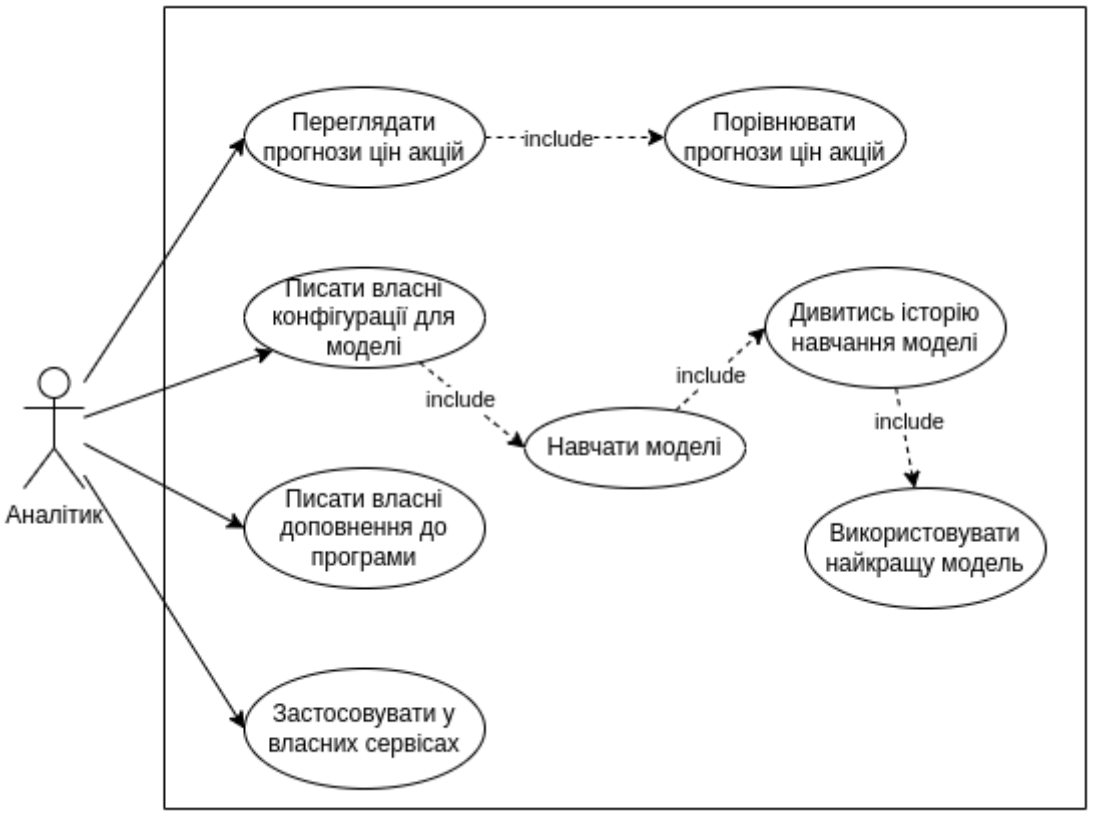


Рис. 2.1 Use Case діаграма

Дана програма спрямована на аналітиків та датасаєнтістів, які створюють моделі для задачі регресії на часові ряди, вона спрощує та допомагає створювати моделі з різними конфігураціями в найкоротший час.

2.3 Розробка програмного забезпечення для створення моделей для прогнозування часових рядів

2.3.1. Розробка архітектури

Архітектура програмного забезпечення — це високорівневий структурний дизайн програмної системи, який визначає її основні компоненти, їх взаємодію, а також принципи та шаблони, за якими система створюється та розвивається.

Архітектури створені для гарної структури та організації проекту, масштабованості, надійності та стабільності, продуктивності та зменшення складності.

Правильно розроблена архітектура значно спрощує розуміння проекту та процес розробки. Це також дозволяє зробити розробку швидше і дешевше, сам проект буде написаний якісно, що не призведе до старіння окремих модулів, хороша архітектура також підвищує продуктивність, масштабованість і безпеку.

На діаграмі зображена розроблена архітектура данного проєкту:

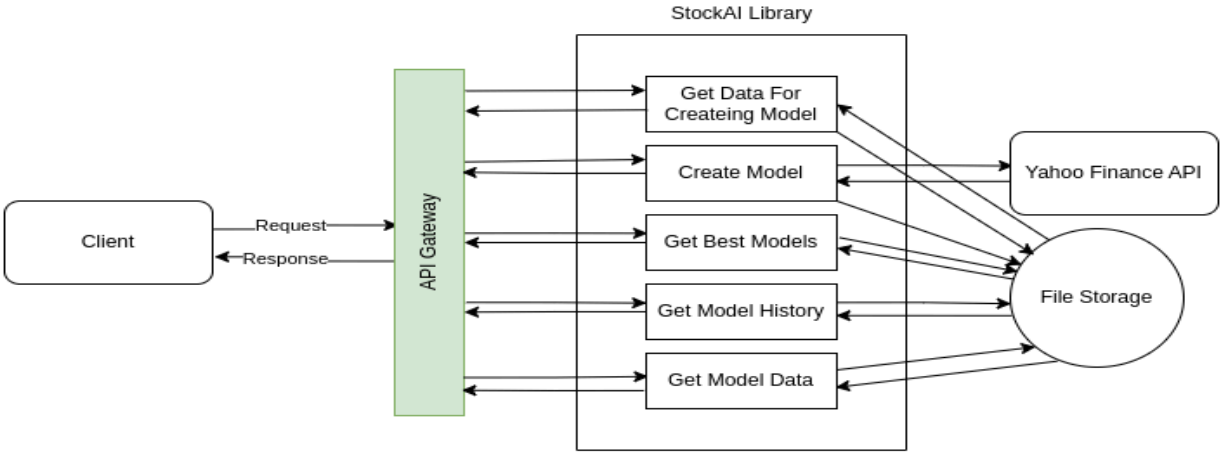


Рис. 2.2 Архітектура проєкту

З розробленої вище архітектури витікає наступна діаграма послідовності проєкту:

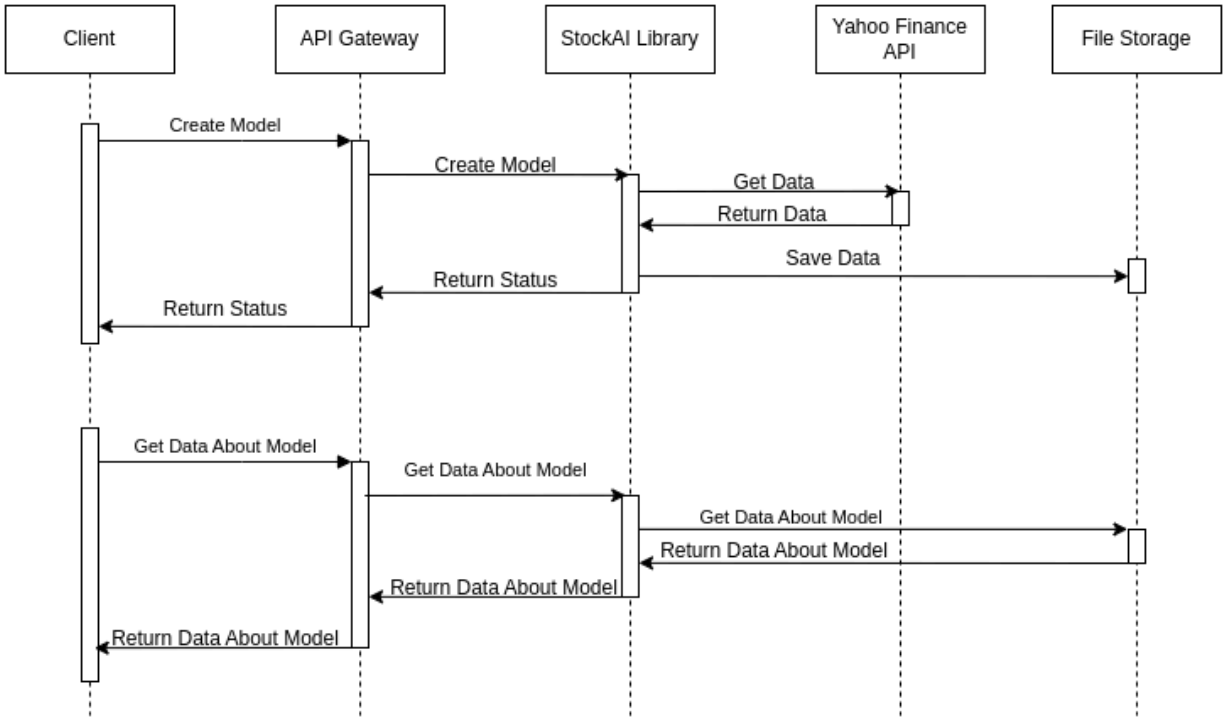


Рис. 2.3 Діаграма послідовності

2.3.2. Розробка класів

Діаграма класів — один із типів діаграм у мові моделювання UML. Він використовується для візуалізації структури системи шляхом відображення класів, їхніх зв’язків та атрибутів. До основних елементів діаграми класів належать:

- Класи. Являють собою структурні елементи системи або програми. Кожен клас може мати атрибути (змінні) і методи (функції), які визначають його характеристики та поведінку.
- Відносини між класами. Це відносини між класами, які показують, як класи взаємодіють один з одним. Найпоширеніші зв’язки включають асоціацію, залежність, агрегацію та композицію.
- Атрибути класів. Це властивості або дані, які належать до певного класу.
- Методи класів. Це функції або процедури, які виконуються в межах класу і можуть маніпулювати його атрибутами або взаємодіяти з іншими класами.

Діаграми класів допомагають розуміти структуру системи, взаємозв'язки між її складовими частинами та логіку програми або системи в цілому. Вона є потужним інструментом для аналізу та проектування програмного забезпечення.

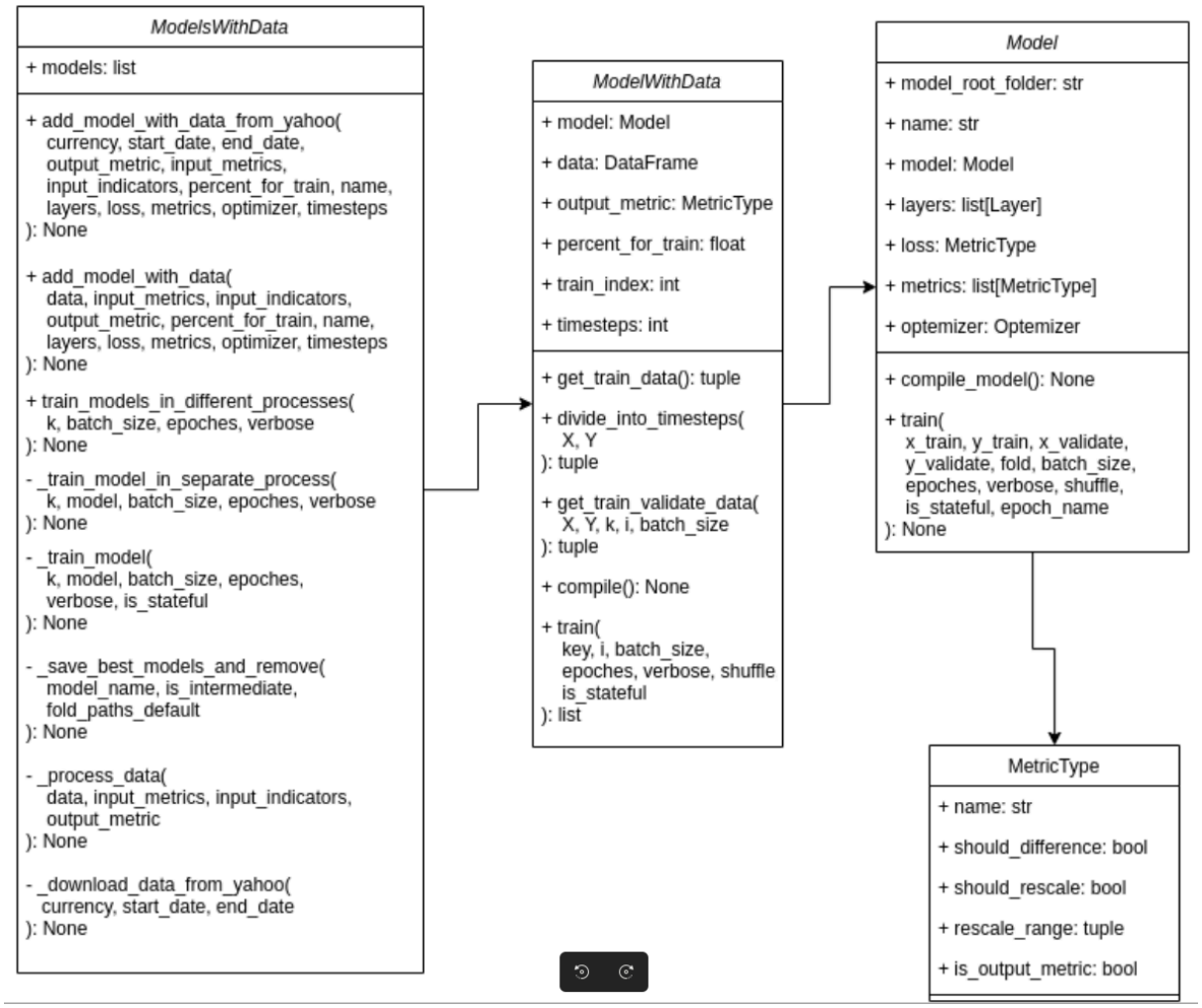


Рис. 2.4 Діаграма класів

2.3.3. Створення моделі нульового прогнозування

Модель нульового прогнозування - це базова модель, яка дає можливість порівняти з собою всі інші моделі, щоб зрозуміти ефективність їхньої роботи. Ця модель має бути простою в реалізації та зрозумілою в логіці роботи. Є декілька способів створення моделей нульового прогнозування. Наприклад створити алгоритм, який буде давати випадкове значення на виході.

Ця модель сама по собі не є ефективною, але вона дає розуміння про всі інші моделі. Якщо якась модель буде гірша, ніж нульова, то таку модель немає сенсу використовувати. Щоб порівняти моделі, можна порівняти суму помилок цих моделей, або середню помилку цих моделей.

Алгоритм випадкового прогнозування зручний у використанні при бінарною, або багатокласової класифікації, але при задачі регресії вона є занадто неефективною, тому що не зрозуміло в яких межах брати випадкові числа.

Для задач регресії є ще один спосіб - алгоритм, який завжди повертає середнє значення за весь період. Коефіцієнт, який рахує різницю між такою моделлю і реалізованою за допомогою нейронної мережі називається R^2 (R-квадрат). Якщо $R^2 = 0$, це означає, що алгоритми працюють однаково, якщо $R^2 < 0$, то створена модель гірша за модель нульового прогнозування, а якщо $R^2 = 1$, то створена модель працює зовсім без помилок.

2.3.4. Застосування індикаторів

У торгівлі індикатори — це інструменти аналізу, які допомагають трейдерам зрозуміти ринок і приймати рішення про вхід або вихід. Зазвичай вони будуються на основі ціни та всебічної ринкової інформації та можуть надавати сигнали про тенденції, нестабільність, обсяги та інші фактори.

Індикатори використовуються для аналізу ринку і можуть бути класифіковані на кілька категорій, включаючи:

- Тенденційні індикатори: Ці індикатори допомагають визначити напрямок руху ціни на ринку. Наприклад, Moving Average (плаваюче середнє) або MACD (Moving Average Convergence Divergence).
- Осциляторні індикатори: Вони вказують на ступінь перекупленості або перепроданості ринку, що може сигналізувати про зміну напрямку ціни. Наприклад, індекс відносної сили (RSI) або стохастичний осцилятор.
- Об'ємні індикатори: Ці індикатори аналізують обсяг угод, які відбуваються на ринку, що може допомогти у з’ясуванні сили та стабільності тренду. Наприклад, балансовий обсяг (OBV) або накопичення/розподіл.

Індикатори працюють за допомогою математичних формул, які обчислюються на основі вхідних даних, таких як ціни закриття, відкриття,

найвищі та низькі ціни, обсяги торгів тощо. Ці формули використовуються для створення індикаторів, які можна візуалізувати на графіку цін або іншому графічному представленні даних . Трейдери аналізують ці показники, щоб розпізнавати моделі та тенденції на ринку та приймати відповідні торгові рішення.

При створенні штучного інтелекту важливо розуміти, що потрібно гарно підбирати індикатори, які ми даємо на вхід. Одне з основних правил - не подавати на вхід дані, які є лінійно залежними. Якщо на вхід подаються такі дані, то моделі важко зрозуміти, що саме потрібно використовувати в тому чи іншому випадку, якщо ці дані змінюються однаково. Тому не рекомендується подавати більше ніж 1 індикатор з кожної категорії.

2.3.5. Підготовка даних

При моделюванні штучного інтелекту важливим аспектом є підготовка даних. Способів підготовки даних є дуже багато, це дуже індивідуальний процес, який залежить від вхідних, вихідних даних, моделі, задачі.

При роботі з часовими рядами важливо працювати зі стаціонарними даними. Статистична стаціонарність вказує на те, що властивості даних, такі як середнє значення, дисперсія і коваріація, не змінюються з часом. Це означає, що даний часовий ряд вважається стаціонарним, якщо він має стабільні статистичні характеристики, такі як постійне середнє та сталу дисперсію, незалежно від часу. Стаціонарність даних вказує на те, що в них немає тенденції та сезонності.

Щоб позбутись тенденції в часових рядах потрібно від кожного значення відняти попереднє значення. Таким чином різність значень не зміниться, але тенденція пропаде.

Щоб перевірити чи стаціонарні дані використовується тест Дікі-Фуллера. Тест Дікі-Фуллера - це статистичний тест, що використовується для перевірки гіпотези про наявність одиничних коренів у часовому ряді. Основна ідея тесту полягає в тому, щоб визначити, чи є корінь одиниці у моделі авторегресії.

Також важливим є масштабування даних. При роботі з LSTM моделями рекомендується масштабувати дані в проміжок від -1 до 1, бо в середині LSTM використовується гіперболічна функція активації, яка найкраще працює в цьому проміжку.

Саме через ці причини в класі MetricType є 2 показники should_difference та should_rescale, які і відповідають за зведення до стаціонарних даних та масштабування.

2.3.6. LSTM модель зі збереженням стану та без

При навчанні рекурентних моделей важливу роль відіграє стан моделі. В звичайному режимі стан моделі оновлюється кожен батч. Це спрощує формування моделі та її використання, однак для задач на часові ряди це не завжди добре. При роботі з часовими рядами стан моделі потрібно зберігати на всю епоху. Для таких випадків потрібно використовувати stateful LSTM. В такому випадку налаштовувати модель дещо складніше, але є і свої плюси - модель може запам’ятати важливу інформацію, яка була подана на вхід багато ітерацій тому. Коли використовуються Stateful моделі важливо між епохами не забувати скидати стан моделі.

3. СТВОРЕННЯ, ОГЛЯД ТА АНАЛІЗ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ

Для цього розділу буде створено декілька моделей з різними конфігураціями для їхнього порівняння. Ці моделі будуть намагатись передсказати ціни акцій таких компаній як Meta, Amazon, Google, Nvidia, Apple. За вхідні дані було обрано 3 індикатори - SMA, RSI, AD, аз а вихідні дані буде один показник - ціна закриття (close) акції на наступному часовому проміжку. Далі будуть наведені схеми кожної моделі:

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 128)	67,584
dense (Dense)	(None, 512)	66,048
dense_1 (Dense)	(None, 512)	262,656
dropout (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 128)	65,664
dense_3 (Dense)	(None, 128)	16,512
dropout_1 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 1)	129

Рис. 3.1 Схема моделі lstm_1

У моделі lstm_1 шар droput має коефіцієнт 0.5, а шар droput_1 має коефіцієнт 0.2. Усі шари Dense, окрім вихідного, мають функцію активації relu. Вихідний шар має функцію активації linear.

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 20, 256)	266,240
lstm_1 (LSTM)	(None, 128)	197,120
dense (Dense)	(None, 512)	66,048
dense_1 (Dense)	(None, 512)	262,656
dropout (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 128)	65,664
dense_3 (Dense)	(None, 128)	16,512
dropout_1 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 1)	129

Рис. 3.2 Схема моделі lstm_2

У моделі lstm_2 шар droput має коефіцієнт 0.5, а шар droput_1 має коефіцієнт 0.2. Шар lstm має параметри return_sequences=True, dropout=0.4. Всі шари Dense, окрім вихідного, мають функцію активації relu. Вихідний шар має функцію активації linear.

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 20, 128)	67,584
lstm_1 (LSTM)	(None, 20, 256)	394,240
lstm_2 (LSTM)	(None, 128)	197,120
dense (Dense)	(None, 512)	66,048
dense_1 (Dense)	(None, 512)	262,656
dropout (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 256)	131,328
dense_3 (Dense)	(None, 128)	32,896
dropout_1 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 1)	129

Рис. 3.3 Схема моделі lstm_3

У моделі lstm_3 шар droput має коефіцієнт 0.5, а шар droput_1 має коефіцієнт 0.2. Шар lstm та шар lstm_1 мать параметри return_sequences=True, dropout=0.4. Всі шари Dense, окрім вихідного, мають функцію активації relu. Вихідний шар має функцію активації linear.

Layer (type)	Output Shape	Param #
lstm (LSTM)	(64, 128)	67,584
dense (Dense)	(64, 512)	66,048
dense_1 (Dense)	(64, 512)	262,656
dropout (Dropout)	(64, 512)	0
dense_2 (Dense)	(64, 256)	131,328
dense_3 (Dense)	(64, 128)	32,896
dropout_1 (Dropout)	(64, 128)	0
dense_4 (Dense)	(64, 1)	129

Рис. 3.4 Схема моделі stateful_lstm_1

У моделі stateful_lstm_1 шар droput має коефіцієнт 0.5, а шар droput_1 має коефіцієнт 0.2. Шар lstm має параметри stateful=True. Всі шари Dense, окрім вихідного, мають функцію активації relu. Вихідний шар має функцію активації linear.

Далі в таблиці наведені результати моделей, буде відображено показник r-квадрат, який дає загальне розуміння якості моделі.

Таблиця 3.1

Результати навчання моделей

r2	META	AMZN	AAPL	NVDA	GOOG
lstm_1	0.019442211836576462	0.009038424119353294	0.020861750468611717	0.02151329442858696	0.026566904033819084
lstm_2	0.064556956291119873	0.028975486755371094	0.057762682437896673	0.031149446964263916	0.053905427455959021
lstm_3	0.059442639350891111	0.020791769027770996	0.024010479450225583	0.026093602180480957	0.021268844604492188
stateful_lstm_1	0.059837460517883 3	0.021041870117187 5	0.073677957057952 88	0.10548478364944458	0.036874115467071 53

З таблиці наведеної вище, добре видно, що всі моделі зовсім по-різному працюють з акціями різних компаній. Моделі LSTM покращились зі збільшенням числа шарів, але навіть найпростіша модель зі збереженням стану дає кращий результат ніж всі попередні моделі, навіть ті, де кількість нейронів набагато більша.

ВИСНОВКИ

1. Проведено аналіз фондового ринку, методів прогнозування на ньому та визначено особливості роботи з ним. Розглянуто способи прогнозування за допомогою штучного інтелекту. Виявлено 2 основні способи - прогнозування на новинах та прогнозування за допомогою індикаторів.
2. Виконано огляд застосунків, які прогнозують ціни акцій різних компаній на основі багатьох показників, в тому числі і з застосуванням штучного інтелекту.
3. Досліджено основні принципи роботи штучного інтелекту. Проаналізовано, як підготовлювати дані до подачі їх у модель штучного інтелекту. В даній роботі вирішена проблема стаціонаризації даних та масштабування для кращого навчання моделей.
4. Проаналізовано основні принципи роботи штучного інтелекту при прогнозуванні часових рядів. У ході дослідження виявлено, що для прогнозування цін акцій, на сьогодні, найкраще проявляють себе LSTM моделі нейронних мереж.
5. На основі аналізу галузі фондового ринку та штучного інтелекту було сформовано вимоги до програмного забезпечення, для спрощення процесу аналізу фондового ринку. Воно допомагає створювати та аналізувати моделі штучного інтелекту, які прогнозують ціни акцій на фондовому ринку.
6. Спроектовано та розроблено програмне забезпечення для створення, аналізу та порівняння моделей штучного інтелекту для прогнозування на фондовому ринку. Розроблено архітектуру системи, визначено механізми підключення зовнішніх API, розроблено структуру File Storage.
7. Створено декілька моделей штучного інтелекту, навчено їх на різних акціях та порівняно між собою, для виявлення найефективніших моделей нейронних мереж. Це дозволило в ході тестування виявити, що кращі результати показують LSTM моделі зі збереженням стану.
8. Робота пройшла апробацію на Всеукраїнській науково-технічній

конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях»:

1. Мироненко С.О., Замрій І.В. Визначення вимог до програми прогнозування ціни акцій на фондовому ринку на основі штучного інтелекту. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024, С. 332 - 333.

2. Мироненко С.О., Замрій І.В. Формулювання вимог до програми прогнозування цін акцій на фондовому ринку, яка базується на штучному інтелекті. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024, С. 317 - 319.

ПЕРЕЛІК ПОСИЛАНЬ

1. Brownlee J. *Deep Learning for Time Series Forecasting*, 2018. 575 с.
2. Burak Gülmez. Stock price prediction with optimized deep LSTM network with artificial rabbits optimization algorithm. *Expert Systems with Applications.* 2023. Vol 227. URL: https://www.sciencedirect.com/science/article/pii/S0957417423008485
3. Swaroop C. H. *A Byte of Python*, 2013, 189 с.
4. David J. Malan, Brian Yu. CS50's Introduction to Artificial Intelligence with Python URL: https://pll.harvard.edu/course/cs50s-introduction-artificial-intelligence-python
5. J. Hu, J.B. Gao, K.D. White. Estimating measurement noise in a time series by exploiting nonstationarity. *Chaos, Solitons & Fractals. 2004. Vol 22, issue 4, P. 807-819* URL: https://www.sciencedirect.com/science/article/abs/pii/S0960077904001055
6. Michael Keith. Exploring the LSTM Neural Network Model for Time Series. 2022 URL: https://towardsdatascience.com/exploring-the-lstm-neural-network-model-for-time-series-8b7685aa8cf
7. Олександр Летичевський. МЕТОДИ ШТУЧНОГО ІНТЕЛЕКТУ В СУЧАСНОМУ СВІТІ ТА ТЕХНОЛОГІЯХ *Світогляд*. 2023. URL: https://www.mao.kiev.ua/biblio/jscans/svitogliad/svit-2023-18-3/svit-3-2023-letichtv-02.pdf.
8. А.К. Погореленко. ШТУЧНИЙ ІНТЕЛЕКТ: СУТНІСТЬ, АНАЛІЗ ЗАСТОСУВАННЯ, ПЕРСПЕКТИВИ РОЗВИТКУ/ *Науковий вісник Херсонського державного університету*. 2019. с. 22-27. URL: https://ej.journal.kspu.edu/index.php/ej/article/view/405.
9. Майя Мар'єнко. ШТУЧНИЙ ІНТЕЛЕКТ ТА ВІДКРИТА НАУКА В ОСВІТІ. *Фізико-математична освіта*. 2023. Том 38. URL: https://fmo-journal.org/index.php/fmo/article/view/225.
10. coincodex URL: https://coincodex.com/.
11. altindex URL: https://altindex.com/.
12. danelfin URL: https://danelfin.com/.
13. wallstreetzen URL: https://www.wallstreetzen.com/.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Застосування ШІ на фондовій біржі для прогнозування ціни акцій, мовою Python з використанням технологій Tensorflow, Keras

Виконав студент 4 курсу
групи ПД-41
Мироненко Станіслав Олександрович
Керівник роботи
д.т.н., доцент, завідувач кафедри ІПЗ Замрій Ірина Вікторівна

Київ – 2024

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – підтримка робочих процесів аналітика фондового ринку.

Об’єкт дослідження – процес прогнозування цін акцій на фондовому ринку.

Предмет дослідження – програмне забезпечення для прогнозування цін акцій на фондовому ринку з використанням штучного інтелекту.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Проаналізувати та оглянути існуючі інструменти для прогнозування ціни на фондовому ринку.
2. Провести технічний аналіз фондового ринку.
3. Визначити функціональні та нефункціональні вимоги застосунку для аналізу та порівняння моделей штучного інтелекту для прогнозування на фондовому ринку.
4. Спроектувати та розробити застосунок для аналізу та порівняння моделей штучного інтелекту для прогнозування на фондовому ринку.
5. Протестувати застосунок для аналізу та порівняння моделей штучного інтелекту для прогнозування на фондовому ринку.

3

АНАЛІЗ АНАЛОГІВ

Характеристика	coincodex	altindex	danelfin	wallstreetzen	StockAI
Прогнозування з ШІ	+	+	+	+	+
Прогнозування на крипторинках	+	-	-	-	+
Прогнозуання на найбільших фондових біржах	-	+	+	+	+
Прогнозування на інших біржах	-	-	-	-	+
Можливість самостійно обирати вхідні/вихідні дані	-	-	-	-	+
Можливість обирати параметри нейромережі	-	-	-	-	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні:

1) Прогнозування часових рядів на фондовому ринку на основі статистичних даних про купівлю та продаж за попередні роки.
2) Розрахунок індикаторів та їхнє застосувати для навчання моделі нейронної мережі для прогнозування цін акцій на фондовому ринку.
3) Розрахунок показників якості моделі нейронної мережі для прогнозування цін акцій на фондовому ринку.
4) Можливість створювати моделі нейронної мережі для прогнозування цін акцій на фондовому ринку з власними конфігураціями.
5) Можливість перегляду результату навчання моделей нейронної мережі для прогнозування цін акцій на фондовому ринку.

Нефункціональні:

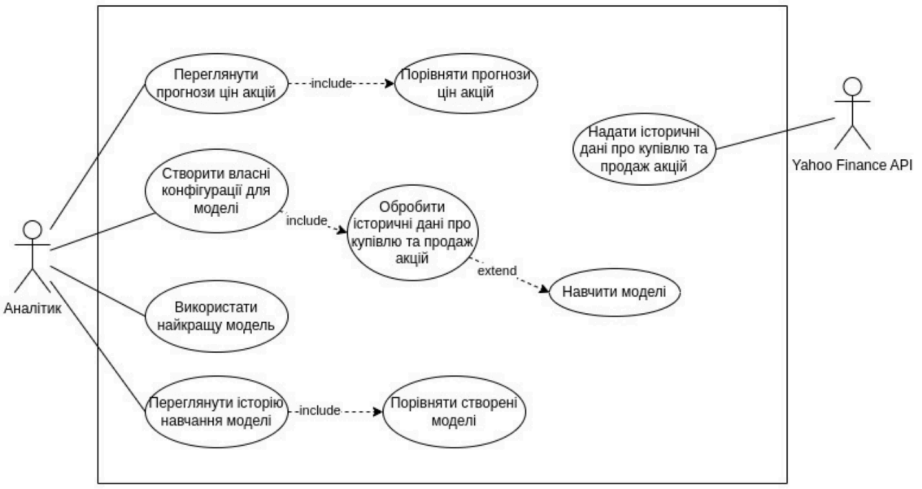
1) Інтерфейс англійською мовою.
2) Система має працювати в браузерах Chrome, Firefox, Opera.
3) Брати дані про купівлю та продаж за попередні роки з відкритиого АПІ yahoo finance.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

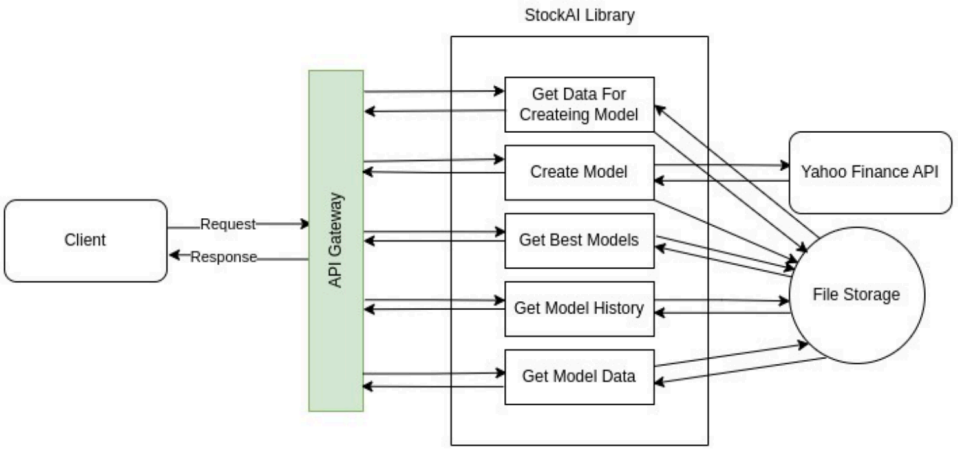
6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



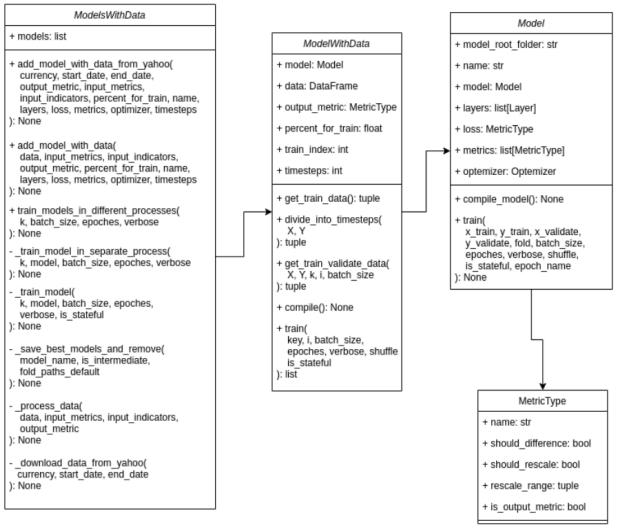
7

АРХІТЕКТУРА ПРОЄКТУ



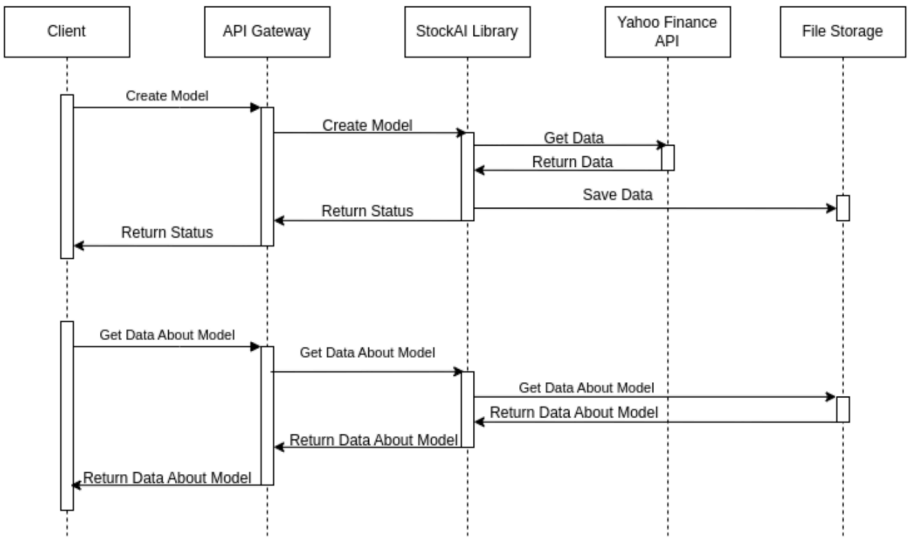
8

ДІАГРАМА КЛАСІВ БІБЛІОТЕКИ



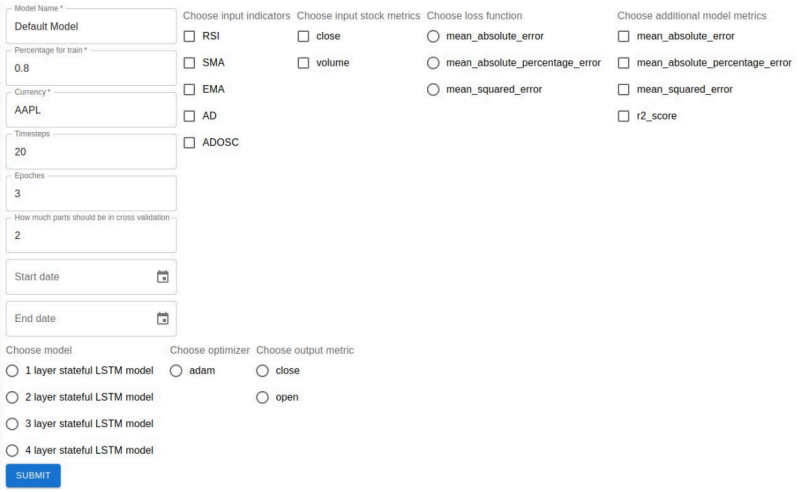
9

ДІАГРАМА ПОСЛІДОВНОСТІ



10

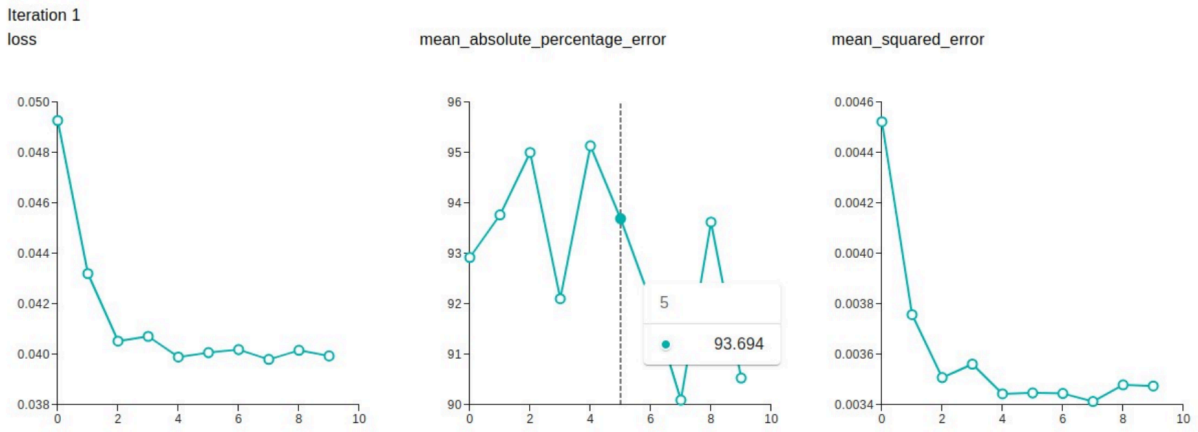
ЕКРАННІ ФОРМИ



Створення моделі

11

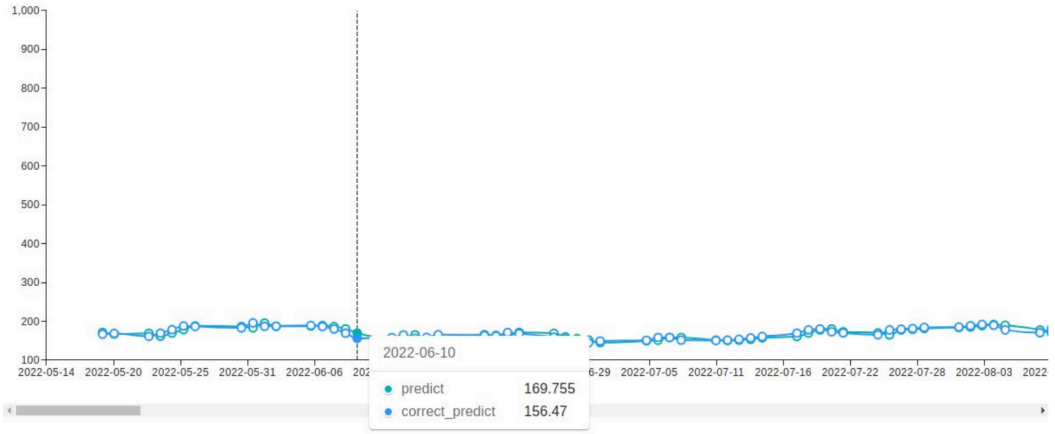
ЕКРАННІ ФОРМИ



Процес навчання

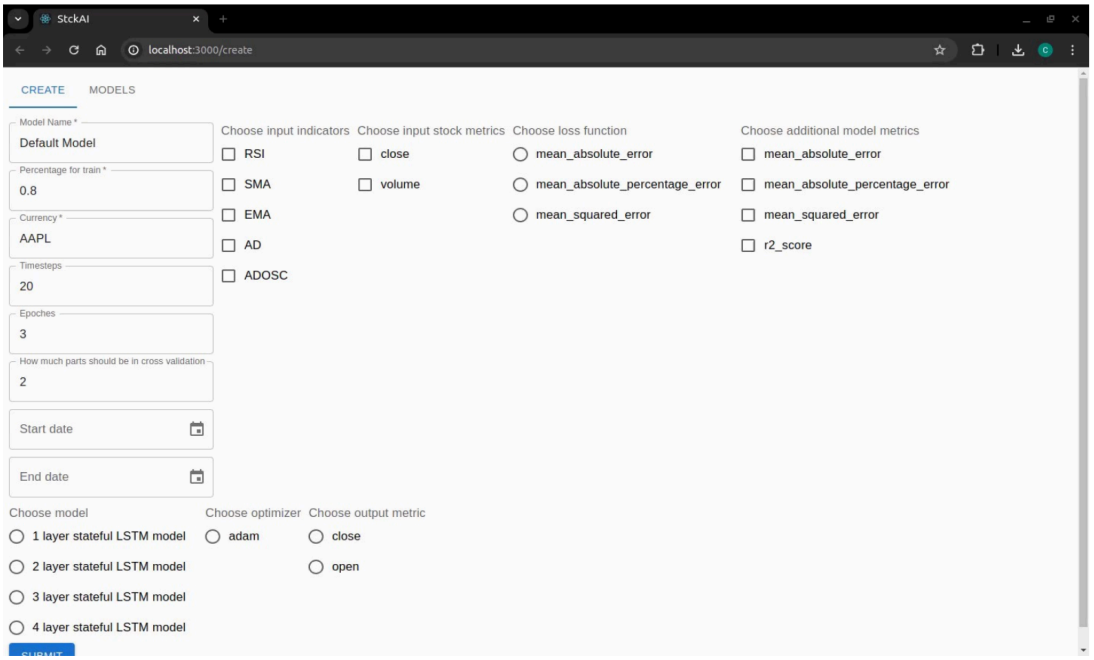
12

ЕКРАННІ ФОРМИ



Результати прогнозування

13



14

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Мироненко С.О., Замрій І.В. Визначення вимог до програми прогнозування ціни акцій на фондовому ринку на основі штучного інтелекту. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій.

Збірник тез. К.: ДУІКТ, 2024, С. 332 - 333.

2. Мироненко С.О., Замрій І.В. Формулювання вимог до програми прогнозування цін акцій на фондовому ринку, яка базується на штучному інтелекті. Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій.

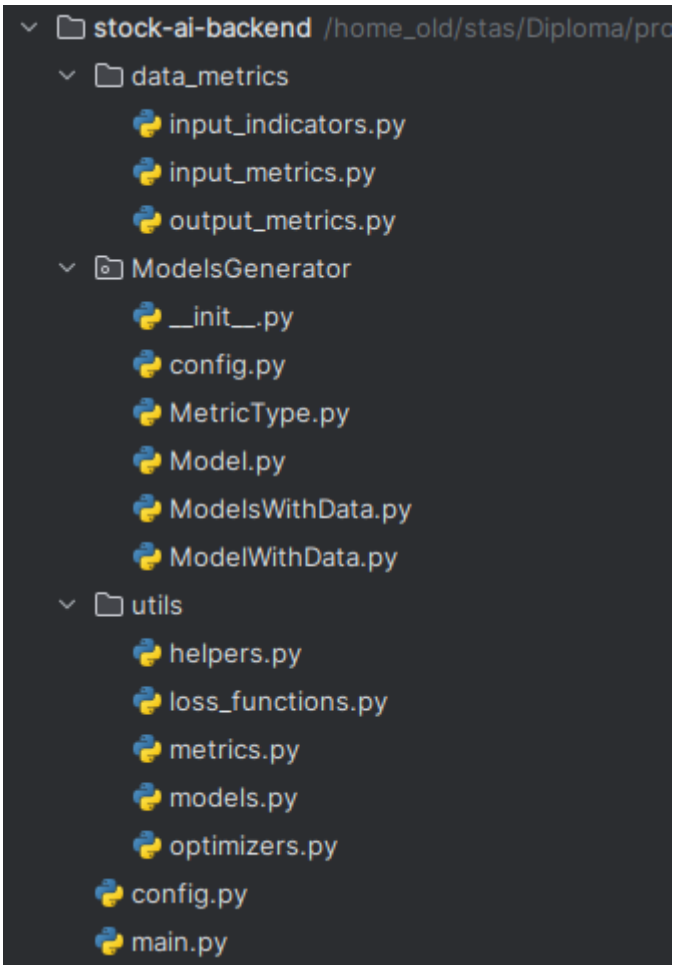
Збірник тез. К.: ДУІКТ, 2024, С. 317 - 319.

ВИСНОВКИ

1. Проаналізовано та оглянуто існуючі інструменти для прогнозування ціни на фондовому ринку, розглянуто їхні переваги та недоліки, також запропоновано способи рішення цих недоліків.
2. Проведено технічний аналіз фондового ринку, досліджено, як на ньому працювати, які є способи прогнозування цін акцій та як їх застосовувати.
3. За результатами порівняння аналогів сформовано функціональні та нефункціональні вимоги до програмного забезпечення.
4. Спроектовано та розроблено програмне забезпечення для створення, аналізу та порівняння моделей штучного інтелекту для прогнозування на фондовому ринку. Розроблено архітектуру системи, визначено механізми підключення зовнішних API, розроблено структуру File Storage.
5. Протестовано мануально клієнтську частину програмного забезпечення та API.

ДОДАТОК Б. РЕАЛІЗАЦІЯ БЕКЕНДУ

Структура файлів



Додаток Б рис 1 - структура проекту

main.py

```
import os.path
from typing import Annotated

from fastapi import Body, FastAPI,
HTTPException

from config import BEST_MODELS_FOLDER,
BEST_MODELS_FOLDER_INTERMEDIATE
from ModelsGenerator import ModelsWithData
from data_metrics.input_indicators import
indicators as all_input_indicators
from data_metrics.input_metrics import metrics
as all_input_metrics
from data_metrics.output_metrics import
metrics as all_output_metrics
from utils.helpers import current_best_models

from utils.models import all_models
from utils.loss_functions import
all_loss_funcs
from utils.metrics import all_metrics
from utils.optimizers import all_optimizers

import json
```

```
from fastapi.middleware.cors import
CORSMiddleware

app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=['*'],
    allow_methods=['GET', 'POST', 'PUT',
'DELETE'],
)

@app.get("/get_data_for_creating_model")
def get_data_for_creating_model():
    input_indicators_names = [ind.label for ind
in all_input_indicators]
    input_metrics_names = [ind.label for ind in
all_input_metrics]
    output_metrics_names = [ind['name'] for ind
in all_output_metrics]
    model_names = [model['name'] for model in
all_models]
    loss_func_names = [func.name for func in
all_loss_funcs]
    metric_names = [metric.name for metric in
all_metrics]
```

```
    optimizer_names = [optimizer['name'] for
optimizer in all_optimizers]

    return dict(

input_indicators=input_indicators_names,
        input_metrics=input_metrics_names,
        output_metrics=output_metrics_names,
        models=model_names,
        loss_functions=loss_func_names,
        metrics=metric_names,
        optimizers=optimizer_names,
    )


@app.post("/create_model")
def read_root(
        input_indicator_names: list[str],
        input_metric_names: list[str],
        output_metric_name: Annotated[str,
Body()],
        currency: Annotated[str, Body()],
        start_date: Annotated[str, Body()],
        end_date: Annotated[str, Body()],
        percent_for_train: Annotated[float,
Body()],
        name: Annotated[str, Body()],
        model_name: Annotated[str, Body()],
        loss_function_name: Annotated[str,
Body()],
        metric_names: list[str] = [],
        optimizer_name: Annotated[str, Body()]
= None,
        timesteps: Annotated[int, Body()] =
None,
        batch_size: Annotated[int, Body()] = 4,
        epoches: Annotated[int, Body()] = 3,
        k: Annotated[int, Body()] = 2,
):
    input_metrics = list(filter(lambda metric:
metric.name in input_metric_names,
all_input_metrics))
    input_indicators = list(filter(lambda
indicator: indicator.name in
input_indicator_names, all_input_indicators))
    output_metric = [metric for metric in
all_output_metrics if metric['name'] ==
output_metric_name][0]['get_metric']()
    model_data = [model for model in all_models
if model['name'] == model_name][0]
    model_function = model_data['layers']
    loss_function = [func for func in
all_loss_funcs if func.name ==
loss_function_name]

    metrics_and_loss = [*all_metrics,
*all_loss_funcs]
    metrics = [metric for metric in
metrics_and_loss if metric.name in
metric_names]
    optimizer = [optimizer for optimizer in
all_optimizers if optimizer['name'] ==
optimizer_name][0]['get']()

    models_generator = ModelsWithData()

models_generator.add_model_with_data_from_yaho
o(
        currency=currency,
        start_date=start_date,
        end_date=end_date,
        output_metric=output_metric,
        input_metrics=input_metrics,
        input_indicators=input_indicators,
        percent_for_train=percent_for_train,
        timesteps=timesteps,
        name=name,
        layers=model_function(batch_size,
timesteps, len(input_metrics) +
len(input_indicators)),
        loss=loss_function,
        metrics=metrics,
        optimizer=optimizer,
    )

models_generator.train_models_and_predict(k=k,
batch_size=batch_size, epoches=epoches,
is_stateful=model_data['is_stateful'])
    return {'status': 200}


@app.get("/get_best_models")
def get_models():
    models = current_best_models()
    return {'models': models}


@app.get("/get_best_epoch_by_r2")
def get_best_model_by_r2(
        name: str
):
    models = current_best_models()
    if name not in models:
        raise HTTPException(status_code=404,
detail='Model not found')

    path = os.path.join(BEST_MODELS_FOLDER,
name, 'r2')
    with open(os.path.join(path,
'history.json')) as f:
        history = json.load(f)
    files = os.listdir(path)
    model_file = [file for file in files if
file.endswith('.keras')][0]
    epoch_number =
int(model_file.split('.')[0]) - 1
    res = {}
    for key, value in history.items():
        res[key] = value[epoch_number]

    return res


@app.get("/get_model_history")
def get_model_history(
        name: str
):
    models = current_best_models()
    if name not in models:
        raise HTTPException(status_code=404,
detail='Model not found')

    path =
os.path.join(BEST_MODELS_FOLDER_INTERMEDIATE,
name)
    with open(os.path.join(path,
'histories.json')) as f:
        history = json.load(f)
    return history


@app.get("/get_model_predicted_data")
def get_model_predicted_data(name: str):
    models = current_best_models()
    if name not in models:
        raise HTTPException(status_code=404,
detail='Model not found')

    path = os.path.join(BEST_MODELS_FOLDER,
name, 'r2', 'predicted_data.json')
    with open(path) as f:
        data = json.load(f)
    return data
```

config.py

```
ROOT_FOLDER =
'/home_old/stas/Diploma/projects/'

MODELS_FOLDER: str = ROOT_FOLDER + 'models/'

BEST_MODELS_FOLDER_INTERMEDIATE = ROOT_FOLDER
+ 'best_models_intermediate/'
BEST_MODELS_FOLDER = ROOT_FOLDER +
'best_models/'
```

optimizers.py

```
from keras.optimizers import Adam

adam = {
   'name': 'adam',
   'get': lambda: Adam(0.001, name='adam')
}

all_optimizers = [adam]
```

models.py

```
from keras.layers import Input, Dense,
Dropout, LSTM

def get_stateful_lstm_1_layers(batch_size=128,
timesteps=25, input_dim=1, ):
   return [
       Input(batch_shape=(batch_size,
timesteps, input_dim,)),
       LSTM(128, stateful=True),
       Dense(512, activation='relu'),
       Dense(512, activation='relu'),
       Dropout(0.5),
       Dense(256, activation='relu'),
       Dense(128, activation='relu'),
       Dropout(0.2),
       Dense(1),
   ]


def get_stateful_lstm_2_layers(batch_size=128,
timesteps=25, input_dim=1, ):
   return [
       Input(batch_shape=(batch_size,
timesteps, input_dim,)),
       LSTM(256, stateful=True,
return_sequences=True, dropout=0.4),
       LSTM(256, stateful=True, dropout=0.4),
       Dense(512, activation='relu'),
       Dense(512, activation='relu'),
       Dropout(0.5),
       Dense(256, activation='relu'),
       Dense(128, activation='relu'),
       Dropout(0.2),
       Dense(1),
   ]


def get_stateful_lstm_3_layers(batch_size=128,
timesteps=25, input_dim=1, ):
   return [
       Input(batch_shape=(batch_size,
timesteps, input_dim,)),
       LSTM(256, stateful=True,
return_sequences=True, dropout=0.4),
       LSTM(256, stateful=True,
return_sequences=True, dropout=0.4),
       LSTM(256, stateful=True, dropout=0.4),
       Dense(512, activation='relu'),
       Dense(512, activation='relu'),
       Dropout(0.5),
       Dense(256, activation='relu'),
       Dense(256, activation='relu'),
       Dropout(0.2),
       Dense(1),
   ]
```

```
def get_stateful_lstm_4_layers(batch_size=128,
timesteps=25, input_dim=1, ):
   return [
       Input(batch_shape=(batch_size,
timesteps, input_dim,)),
       LSTM(256, stateful=True,
return_sequences=True, dropout=0.4),
       LSTM(256, stateful=True,
return_sequences=True, dropout=0.4),
       LSTM(256, stateful=True,
return_sequences=True, dropout=0.4),
       LSTM(256, stateful=True, dropout=0.4),
       Dense(512, activation='relu'),
       Dense(512, activation='relu'),
       Dropout(0.5),
       Dense(256, activation='relu'),
       Dense(256, activation='relu'),
       Dropout(0.2),
       Dense(128, activation='relu'),
       Dense(128, activation='relu'),
       Dense(1),
   ]


all_models = [
   {
       'name': '1 layer stateful LSTM model',
       'is_stateful': True,
       'layers': get_stateful_lstm_1_layers,
   },
   {
       'name': '2 layer stateful LSTM model',
       'is_stateful': True,
       'layers': get_stateful_lstm_2_layers,
   },
   {
       'name': '3 layer stateful LSTM model',
       'is_stateful': True,
       'layers': get_stateful_lstm_3_layers,
   },
   {
       'name': '4 layer stateful LSTM model',
       'is_stateful': True,
       'layers': get_stateful_lstm_4_layers,
   }
]
```

metrics.py

```
from keras.metrics import R2Score
import numpy as np

r2score = R2Score(
   class_aggregation="uniform_average",
num_regressors=0, name="r2_score",
dtype=np.float32
)

all_metrics = [r2score]
```

loss_functions.py

```
from keras.losses import
MeanAbsolutePercentageError, MeanSquaredError,
MeanAbsoluteError

mae =
MeanAbsoluteError(reduction="sum_over_batch_si
ze", name="mean_absolute_error")

mape = MeanAbsolutePercentageError(
   reduction="sum_over_batch_size",
name="mean_absolute_percentage_error"
)

mse = MeanSquaredError(
   reduction="sum_over_batch_size",
name="mean_squared_error"
)
```

```
all_loss_funcs = [mae, mape, mse]
```

helpers.py

```
import os

from config import BEST_MODELS_FOLDER


def current_best_models():
    return [f for f in
os.listdir(BEST_MODELS_FOLDER) if
os.path.isdir(os.path.join(BEST_MODELS_FOLDER,
f))]
```

ModelWithData.py

```
import json

import pandas as pd
import numpy as np

from .Model import Model
from .MetricType import MetricType


class ModelWithData:
    def __init__(
            self,
            model: Model,
            data: pd.DataFrame,
            initial_data: pd.DataFrame,
            output_metric: MetricType,
            percent_for_train: float = 0.8,
            timesteps: int=None
    ):
        self.model: Model = model
        self.data = data
        self.initial_data = initial_data
        self.output_metric = output_metric
        self.percent_for_train =
percent_for_train
        self.train_index = int(len(self.data) *
self.percent_for_train)
        self.timesteps = timesteps

    def get_train_data(self):
        """
        return all X and Y for train without
additional preparing (without timesteps etc)
        """
        X = self.data[:self.train_index]

        output_metric_name =
self.output_metric.name +
self.output_metric.output_suffix

        X[output_metric_name] =
X[output_metric_name].shift(-1)
        X.dropna(inplace=True)

        Y = pd.DataFrame(X[output_metric_name])
        X.drop(output_metric_name, axis=1,
inplace=True)
        return X, Y

    def get_all_data(self):
        X = self.data[:]


        output_metric_name =
self.output_metric.name +
self.output_metric.output_suffix

        X[output_metric_name] =
X[output_metric_name].shift(-1)
        X.dropna(inplace=True)

        Y = pd.DataFrame(X[output_metric_name])
        X.drop(output_metric_name, axis=1,
inplace=True)
        return X, Y

    def divide_into_timesteps(self, X, Y):
        if not self.timesteps:
            return X, Y

        y_train = Y.copy()

        x_train = np.array(X.copy())
        x_train =
np.array([np.array(x_train[i:i +
self.timesteps]) for i in range(0,
len(x_train) - self.timesteps + 1)])
        y_train = y_train[self.timesteps - 1:]
        return x_train, y_train

    @staticmethod
    def get_train_validate_data(X, Y, k, i,
batch_size=None):
        """
        :param X:
        :param Y:
        :param k: На скільки частин ділити
        :param i: Яку частину брати для валідації
        :param batch_size: uses for lstm stateful
models
        :return:
        """
        if not k:
            return (X, Y), (None, None)
        length = len(X)
        data_in_fold = length // k
        if i == k - 1:
            train_length = data_in_fold * i
            validation_end = length

            if batch_size:
                train_length -= train_length %
batch_size
                validation_end = length -
length % batch_size

            x_validate =
X[train_length:validation_end]
            y_validate =
Y[train_length:validation_end]
            x_train = X[:train_length]
            y_train = Y[:train_length]
        else:
            train_length = data_in_fold * (k -
1)
            if batch_size:
                train_length -= train_length %
batch_size
            validation_length = (length -
train_length)
            if batch_size:
                validation_length -=
validation_length % batch_size

            validation_start = data_in_fold * i
            validation_end = validation_start +
validation_length

            end_train = length - (length %
batch_size)
            x_validate =
X[validation_start:validation_end]
            y_validate =
Y[validation_start:validation_end]
            x_train =
np.concatenate([X[:validation_start],
X[validation_end:end_train]])
            y_train =
np.concatenate([Y[:validation_start],
Y[validation_end:end_train]])
```

```
        return (x_train, y_train), (x_validate,
y_validate)


    def compile(self):
        self.model.compile_model()

    def train(
            self,
            key,
            i,
            batch_size=128,
            epoches=1,
            verbose=1,
            shuffle=False,
            is_stateful=False
    ):
        """
        :param key: На скільки частин ділити
        :param i: Яку частину брати для валідації
        :param batch_size:
        :param epoches:
        :param verbose:
        :param shuffle:
        :param is_stateful:
        :return:
        """
        X, Y = self.get_train_data()
        X, Y = self.divide_into_timesteps(X, Y)
        train_data_batch_size = batch_size if
is_stateful else None
        (x_train, y_train), (x_validate,
y_validate) =
ModelWithData.get_train_validate_data(X, Y,
key, i, batch_size=train_data_batch_size)
        if not is_stateful:
            res = self.model.train(
                x_train=x_train,
                y_train=y_train,
                x_validate=x_validate,
                y_validate=y_validate,
                fold=f'{i:02}',
                batch_size=batch_size,
                epoches=epoches,
                verbose=verbose,
                shuffle=shuffle,
                is_stateful=is_stateful
            )
            return res
        else:
            history = {}
            for epoch in range(epoches):
                current_history =
self.model.train(
                    x_train=x_train,
                    y_train=y_train,
                    x_validate=x_validate,
                    y_validate=y_validate,
                    fold=f'{i:02}',
                    batch_size=batch_size,
                    epoches=1,
                    verbose=verbose,
                    shuffle=shuffle,
                    is_stateful=is_stateful,

epoch_name=f'{epoch+1:04}.keras'
                )
                for key, value in
current_history.items():
                    if key not in
history.keys():
                        history[key] = value
                    else:

history[key].extend(value)
                for layer in self.model.layers:
                    if 'lstm' in layer.name:
                        layer.reset_states()
            with
open(f'{self.model.model_root_folder}fold-{i:0
2}/' + 'history.json', 'w') as file:
                json.dump(history, file,
indent=2)
            return history

    def predict(self, batch_size,
is_stateful=False):
        X, Y = self.get_all_data()
        X, Y = self.divide_into_timesteps(X, Y)
        if is_stateful:
            start_point = len(X) % batch_size
            X = X[start_point:]
            Y = Y[start_point:]
        res = self.model.predict(X, batch_size)
        scaler = self.output_metric.scaler
        Y['predict'] = res
        if scaler:
            Y_scaled =
pd.DataFrame(scaler.inverse_transform(pd.DataF
rame(Y[self.output_metric.name +
self.output_metric.output_suffix])),
index=Y.index,
columns=[self.output_metric.name])
            Y['predict'] =
scaler.inverse_transform(pd.DataFrame(Y['predi
ct']))

            Y_actual =
pd.DataFrame(self.initial_data.loc[Y.index][se
lf.output_metric.name],
columns=[self.output_metric.name])
            Y['predict'] =
Y_actual[self.output_metric.name] +
Y['predict']
            Y['correct_predict'] = Y_scaled +
Y_actual

        output_metric_name =
self.output_metric.name +
self.output_metric.output_suffix
        Y.drop(output_metric_name, axis=1,
inplace=True)
        return Y
```

ModelsWithData.py

```
import json

import keras.src.saving

from .ModelWithData import ModelWithData
from .MetricType import MetricType

import yfinance as yf
from .Model import Model

from talib import abstract

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from multiprocessing import Process

import shutil
from pathlib import Path

import os

from config import MODELS_FOLDER,
BEST_MODELS_FOLDER_INTERMEDIATE,
BEST_MODELS_FOLDER


class ModelsWithData:
    def __init__(self):
        self.models: list[ModelWithData] = []
```

```
    def add_model_with_data_from_yahoo(
            self,
            currency,
            start_date,
            end_date,
            output_metric: MetricType,
            input_metrics: list[MetricType],
            input_indicators: list[MetricType],
            percent_for_train=0.8,
            name=None,
            layers=None,
            loss=None,
            metrics=None,
            optimizer=None,
            timesteps=None,
    ):
        data =
ModelsWithData._download_data_from_yahoo(curre
ncy, start_date, end_date)
        self.add_model_with_data(
            data=data,
            output_metric=output_metric,
            input_metrics=input_metrics,
            input_indicators=input_indicators,

percent_for_train=percent_for_train,
            name=name,
            layers=layers,
            loss=loss,
            metrics=metrics,
            optimizer=optimizer,
            timesteps=timesteps
        )

    def add_model_with_data(
            self,
            data: pd.DataFrame,
            input_metrics: list[MetricType],
            input_indicators: list[MetricType],
            output_metric: MetricType,
            percent_for_train=0.8,
            name=None,
            layers=None,
            loss=None,
            metrics=None,
            optimizer=None,
            timesteps=None,
    ):
        processed_data =
ModelsWithData._process_data(
            data=data,
            input_metrics=input_metrics,
            input_indicators=input_indicators,
            output_metric=output_metric
        )
        model = Model(
            name=name,
            layers=layers,
            loss=loss,
            metrics=metrics,
            optimizer=optimizer,
        )
        model_with_data = ModelWithData(
            model=model,
            data=processed_data,
            initial_data=data,
            output_metric=output_metric,

percent_for_train=percent_for_train,
            timesteps=timesteps
        )
        self.models.append(model_with_data)

    def train_models(self, k, batch_size=128,
epoches=1000, verbose=1, is_stateful=False):
        for model in self.models:
            ModelsWithData._train_model(
                model=model, k=k,
batch_size=batch_size, epoches=epoches,
verbose=verbose, is_stateful=is_stateful
            )
            #
ModelsWithData._save_best_models_and_remove(mo
del.model.name)

    def train_models_and_predict(self, k,
batch_size=128, epoches=1000, verbose=1,
is_stateful=False):
        for model in self.models:
            ModelsWithData._train_model(
                model=model, k=k,
batch_size=batch_size, epoches=epoches,
verbose=verbose, is_stateful=is_stateful
            )
            best_model_dir =
os.path.join(BEST_MODELS_FOLDER,
model.model.name, 'r2')
            model_file = [file for file in
os.listdir(best_model_dir) if
file.endswith('.keras')][0]

            best_model =
keras.src.saving.load_model(os.path.join(best_
model_dir, model_file))
            model.model.model = best_model
            predicted_data: pd.DataFrame =
ModelsWithData._predict_model(model,
batch_size, is_stateful)

predicted_data.to_json(os.path.join(best_model
_dir, 'predicted_data.json'), indent=2)

    def
train_models_in_different_processes(self, k,
batch_size=128, epoches=1000, verbose=1):
        processes = []
        for model in self.models:
            process = Process(

target=ModelsWithData._train_model,
                kwargs=dict(k=k, model=model,
batch_size=batch_size, epoches=epoches,
verbose=verbose),
                # daemon=True,
            )
            processes.append(process)

        for process in processes:
            process.start()

        for process in processes:
            process.join()


    @staticmethod
    def _train_model(k, model: ModelWithData,
batch_size=128, epoches=1000, verbose=1,
is_stateful=False):
        model.compile()
        histories = []
        weights =
model.model.model.get_weights()
        for i in range(k):
            history = model.train(key=k, i=i,
batch_size=batch_size, epoches=epoches,
verbose=verbose, is_stateful=is_stateful)
            histories.append(history)

model.model.model.set_weights(weights)

ModelsWithData._save_intermediate_best_models_
and_remove(model.model.name, True,
fold_paths_default=[f'fold-{i:02}'])
```

```
Path(os.path.join(BEST_MODELS_FOLDER_INTERMEDIATE,
model.model.model_root_folder)).mkdir(parents=True, exist_ok=True)
        with open(os.path.join(BEST_MODELS_FOLDER_INTERMEDIATE, model.model.name, 'histories.json'), 'a') as f:
            json.dump(histories, f, indent=2)

ModelsWithData._save_best_models_and_remove(model.model.name)

    @staticmethod
    def _predict_model(model: ModelWithData, batch_size=128, is_stateful=False):
        return model.predict(batch_size, is_stateful)

    @staticmethod
    def _save_intermediate_best_models_and_remove(model_name: str, is_intermediate=False, fold_paths_default: list[str]=None):
        best_r2 = None
        best_loss = None
        # best_mse = None
        path: str = os.path.join(MODELS_FOLDER, model_name)
        folds_paths = fold_paths_default or os.listdir(path)

        for fold in filter(lambda x: '.' not in x, folds_paths):
            fold_path: str = os.path.join(path, fold)
            with open(os.path.join(fold_path, 'history.json')) as f:
                history = json.load(f)

            keys = history.keys()
            r2 = history['val_r2_score'] if 'val_r2_score' in keys else []
            losses = history['val_loss'] if 'val_loss' in keys else []

            r2_index = r2.index(max(r2))
            losses_index = losses.index(min(losses))

            save_to_base = BEST_MODELS_FOLDER_INTERMEDIATE if is_intermediate else BEST_MODELS_FOLDER
            current_model_name = model_name
            if is_intermediate:
                current_model_name += f'/{fold}/'
            if not best_r2 or best_r2['value'] < r2[r2_index]:
                best_r2 = {
                    'model_path': os.path.join(fold_path, f'{r2_index + 1:04}.keras'),
                    'history': os.path.join(fold_path, 'history.json'),
                    'histories': os.path.join(path, 'histories.json'),
                    'value': r2[r2_index],
                    'model': f'{r2_index + 1:04}.keras',
                    'save_to': os.path.join(save_to_base, current_model_name)
                }

            if not best_loss or best_loss['value'] > losses[losses_index]:
                best_loss = {
                    'model_path': os.path.join(fold_path, f'{losses_index + 1:04}.keras'),
                    'history': os.path.join(fold_path, 'history.json'),
                    'histories': os.path.join(path, 'histories.json'),
                    'value': losses[losses_index],
                    'model': f'{losses_index + 1:04}.keras',
                    'save_to': os.path.join(save_to_base, current_model_name)
                }


        Path(os.path.join(best_r2['save_to'], 'r2')).mkdir(parents=True, exist_ok=True)
        Path(os.path.join(best_r2['save_to'], 'loss')).mkdir(parents=True, exist_ok=True)

        shutil.copyfile(best_r2['model_path'], os.path.join(best_r2['save_to'], 'r2', best_r2['model']))
        shutil.copyfile(best_r2['history'], os.path.join(best_r2['save_to'], 'r2', 'history.json'))


shutil.copyfile(best_loss['model_path'], os.path.join(best_loss['save_to'], 'loss', best_loss['model']))
        shutil.copyfile(best_loss['history'], os.path.join(best_loss['save_to'], 'loss', 'history.json'))

        if not is_intermediate:

shutil.copyfile(best_r2['histories'], os.path.join(best_r2['save_to'], 'r2', 'histories.json'))

shutil.copyfile(best_loss['histories'], os.path.join(best_loss['save_to'], 'loss', 'histories.json'))

        shutil.rmtree(path)

    @staticmethod
    def _save_best_models_and_remove(model_name: str):
        best_r2_index = None
        best_r2 = None
        best_loss_index = None
        best_loss = None
        path: str = os.path.join(BEST_MODELS_FOLDER_INTERMEDIATE, model_name)

        with open(os.path.join(path, 'histories.json')) as f:
            histories = json.load(f)

        for i, history in enumerate(histories):
            keys = history.keys()
            max_loss = history['val_loss'] if 'val_loss' in keys else None
            max_r2 = history['val_r2_score'] if 'val_r2_score' in keys else None
            if best_r2 is None or best_r2 < max_r2:
                best_r2 = max_r2
                best_r2_index = i

            if best_loss_index is None or best_loss_index < max_loss:
                best_loss_index = max_loss
                best_loss = i
```

```
        save_to =
os.path.join(BEST_MODELS_FOLDER, model_name)
        shutil.copytree(os.path.join(path,
f'fold-{best_r2_index:02}', 'r2'),
os.path.join(save_to, 'r2'))
        shutil.copytree(os.path.join(path,
f'fold-{best_loss:02}', 'loss'),
os.path.join(save_to, 'loss'))


    @staticmethod
    def _process_data(
            data: pd.DataFrame,
            input_metrics: list[MetricType],
            input_indicators: list[MetricType],
            output_metric: MetricType,
    ) -> pd.DataFrame:
        data_with_indicators =
pd.DataFrame(index=data.index)
        input_metrics_names =
[input_metric.name for input_metric in
input_metrics]

data_with_indicators[input_metrics_names] =
data[input_metrics_names]
        for indicator in input_indicators:
            try:
                current_indicator =
abstract.Function(indicator.name)(data)
                if
isinstance(current_indicator, pd.Series):

data_with_indicators[indicator.name] =
current_indicator
                else:
                    # DataFrame з декількома
стовпцямиа
                    for col in
current_indicator.columns:

data_with_indicators[col.upper()] =
current_indicator[col]
                        # print()
            except BaseException as e:
                print(f'key error
[{indicator}]')

        data_with_indicators[output_metric.name
+ output_metric.output_suffix] =
data[output_metric.name]
        metrics = [*input_metrics,
*input_indicators, output_metric]

        for metric in metrics:
            name = metric.name
            if metric.is_output_metric:
                name = name +
output_metric.output_suffix
            metric_data =
pd.DataFrame(data_with_indicators[name])
            if metric.should_difference:
                metric_data =
pd.DataFrame(metric_data.diff())


            if metric.should_rescale:
                scaler =
MinMaxScaler(metric.rescale_range)
                scaler.fit(metric_data)
                metric_data =
scaler.fit_transform(metric_data)

                if metric.is_output_metric:
                    metric.scaler = scaler

            data_with_indicators[name] =
metric_data
```

```
data_with_indicators.dropna(inplace=True)
        return data_with_indicators

    @staticmethod
    def _download_data_from_yahoo(currency,
start_date, end_date) -> pd.DataFrame:
        data = yf.download(currency,
start=start_date, end=end_date)
        data.rename(columns={
            'Open': 'open',
            'Close': 'close',
            'Low': 'low',
            'High': 'high',
            'Volume': 'volume'
        }, inplace=True)
        return data
```

Model.py

```
from keras.models import Sequential
from keras.optimizers import Adam
from keras.losses import
MeanAbsolutePercentageError, MeanSquaredError
from keras.metrics import R2Score
from keras.callbacks import ModelCheckpoint

import numpy as np
from config import MODELS_FOLDER
import json


class Model:
    def __init__(self, name='default_model',
layers=(), loss='mean_absolute_error',
metrics=(), optimizer=Adam()):
        self.model_root_folder = MODELS_FOLDER
+ name + '/'
        self.name: str = name
        self.model = Sequential()
        self.layers = layers
        for layer in layers:
            self.model.add(layer)

        self.loss = loss

        if len(metrics) == 0:
            metrics = [
                MeanAbsolutePercentageError(

reduction="sum_over_batch_size",
name="mean_absolute_percentage_error"
                ),
                MeanSquaredError(

reduction="sum_over_batch_size",
name="mean_squared_error"
                ),
                R2Score(

class_aggregation="uniform_average",
num_regressors=0, name="r2_score",
dtype=np.float32
                )
            ]

        self.metrics = metrics
        self.optimizer = optimizer

    def compile_model(self):
        self.model.compile(
            loss=self.loss,
            metrics=self.metrics,
            optimizer=self.optimizer,
        )

    def train(
            self,
```

```
            x_train,
            y_train,
            x_validate=None,
            y_validate=None,
            fold='00',
            batch_size=128,
            epoches=1,
            verbose=1,
            shuffle=False,
            is_stateful=False,
            epoch_name=None,
    ):
        """
        fold - рядок з 2 символів, наприклад 00,
01, 02
        """

        isValidationNone = x_validate is None

        fold_name = f'fold-{fold}'
        fold_folder =
f'{self.model_root_folder}{fold_name}/'

        epoch_name = epoch_name if is_stateful
else '{epoch:04d}.keras'
        checkpoint =
ModelCheckpoint(filepath=fold_folder +
epoch_name, monitor='loss', mode='min',

save_best_only=False)

        history = self.model.fit(
            x_train,
            y_train,
            batch_size=batch_size,
            epochs=epoches,
            validation_data=(x_validate,
y_validate) if not isValidationNone else None,
            validation_split=0.2 if
isValidationNone else 0,
            callbacks=[checkpoint],
            shuffle=shuffle,
            verbose=verbose,
        )

        if not is_stateful:
            with open(fold_folder +
'history.json', 'w') as file:
                json.dump(history.history,
file, indent=2)

        return history.history

    def predict(self, x, batch_size=128):
        res = self.model.predict(x,
batch_size=batch_size)
        return res
```

MetricType.py

```
from keras.models import Sequential
from keras.optimizers import Adam
from keras.losses import
MeanAbsolutePercentageError, MeanSquaredError
from keras.metrics import R2Score
from keras.callbacks import ModelCheckpoint

import numpy as np
from config import MODELS_FOLDER
import json


class Model:
    def __init__(self, name='default_model',
layers=(), loss='mean_absolute_error',
metrics=(), optimizer=Adam()):
        self.model_root_folder = MODELS_FOLDER
+ name + '/'
        self.name: str = name
        self.model = Sequential()
        self.layers = layers
        for layer in layers:
            self.model.add(layer)

        self.loss = loss

        if len(metrics) == 0:
            metrics = [
                MeanAbsolutePercentageError(

reduction="sum_over_batch_size",
name="mean_absolute_percentage_error"
                ),
                MeanSquaredError(

reduction="sum_over_batch_size",
name="mean_squared_error"
                ),
                R2Score(

class_aggregation="uniform_average",
num_regressors=0, name="r2_score",
dtype=np.float32
                )
            ]

        self.metrics = metrics
        self.optimizer = optimizer

    def compile_model(self):
        self.model.compile(
            loss=self.loss,
            metrics=self.metrics,
            optimizer=self.optimizer,
        )

    def train(
            self,
            x_train,
            y_train,
            x_validate=None,
            y_validate=None,
            fold='00',
            batch_size=128,
            epoches=1,
            verbose=1,
            shuffle=False,
            is_stateful=False,
            epoch_name=None,
    ):
        """
        fold - рядок з 2 символів, наприклад 00,
01, 02
        """

        isValidationNone = x_validate is None

        fold_name = f'fold-{fold}'
        fold_folder =
f'{self.model_root_folder}{fold_name}/'

        epoch_name = epoch_name if is_stateful
else '{epoch:04d}.keras'
        checkpoint =
ModelCheckpoint(filepath=fold_folder +
epoch_name, monitor='loss', mode='min',

save_best_only=False)

        history = self.model.fit(
            x_train,
            y_train,
            batch_size=batch_size,
            epochs=epoches,
            validation_data=(x_validate,
y_validate) if not isValidationNone else None,
```

```
            validation_split=0.2 if
isValidationNone else 0,
            callbacks=[checkpoint],
            shuffle=shuffle,
            verbose=verbose,
        )

        if not is_stateful:
            with open(fold_folder +
'history.json', 'w') as file:
                json.dump(history.history,
file, indent=2)

        return history.history

    def predict(self, x, batch_size=128):
        res = self.model.predict(x,
batch_size=batch_size)
        return res
```

output_metrics.py

```
from ModelsGenerator import MetricType

output_close = {
    'name': 'close',
    'get_metric': lambda: MetricType(
        name='close',
        should_difference=True,
        should_rescale=True,
        rescale_range=(-1, 1),
        is_output_metric=True,
    )
}

output_open = {
    'name': 'open',
    'get_metric': lambda: MetricType(
        name='open',
        should_difference=True,
        should_rescale=True,
        rescale_range=(-1, 1),
        is_output_metric=True,
    )
}


metrics = [output_close, output_open]
```

input_metrics.py

```
from ModelsGenerator import MetricType

input_close = MetricType(
    name='close',
    should_difference=True,
    should_rescale=True,
    rescale_range=(-1, 1),
    is_output_metric=False,
)
```

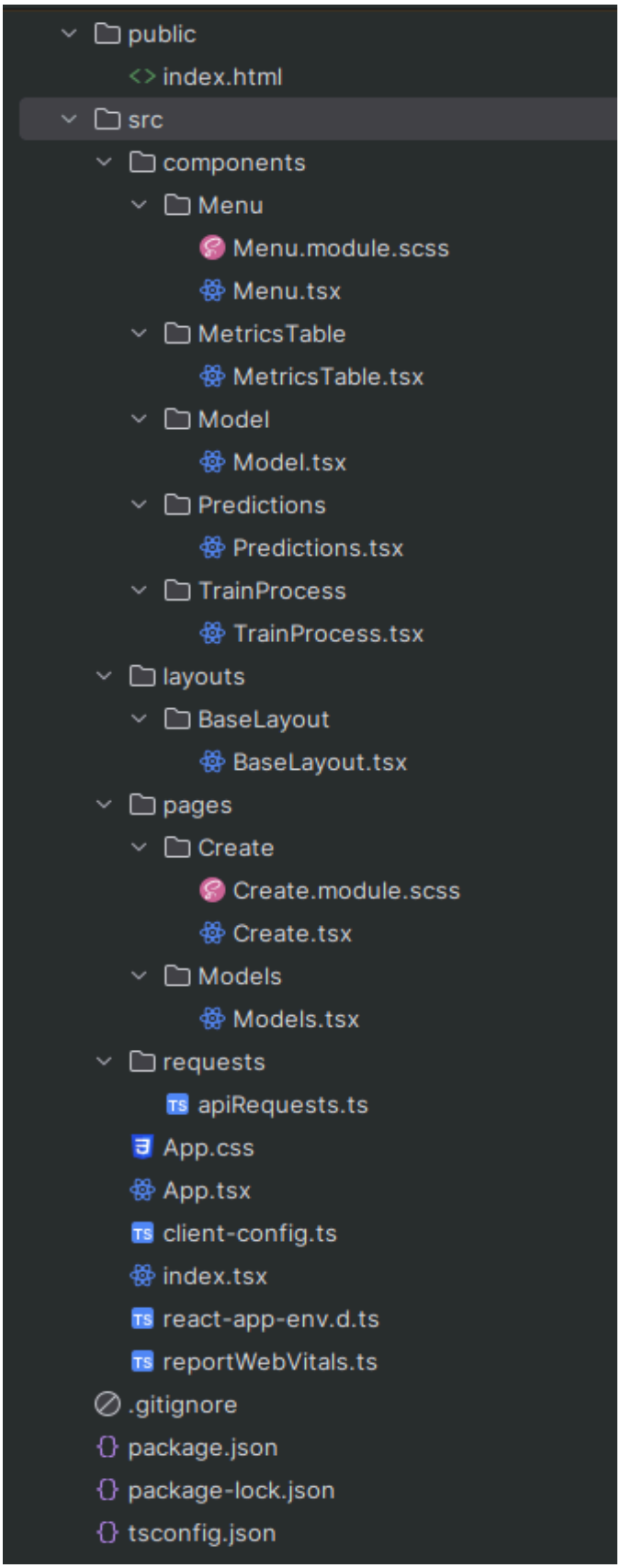
```
# todo research volume
input_volume = MetricType(
    name='volume',
    should_difference=False,
    should_rescale=True,
    rescale_range=(-1, 1),
    is_output_metric=False,
)

metrics = [input_close, input_volume]
```

input_indicators.py

```
from ModelsGenerator import MetricType

input_rsi = MetricType(
    name='RSI',
    should_difference=False,
    should_rescale=True,
    rescale_range=(-1, 1),
    is_output_metric=False,
)

input_sma = MetricType(
    name='SMA',
    should_difference=True,
    should_rescale=True,
    rescale_range=(-1, 1),
    is_output_metric=False,
)

input_ema = MetricType(
    name='EMA',
    should_difference=True,
    should_rescale=True,
    rescale_range=(-1, 1),
    is_output_metric=False,
)

input_ad = MetricType(
    name='AD',
    should_difference=True,
    should_rescale=True,
    rescale_range=(-1, 1),
    is_output_metric=False,
)

input_adosc = MetricType(
    name='ADOSC',
    should_difference=False,
    should_rescale=True,
    rescale_range=(-1, 1),
    is_output_metric=False,
)

indicators = [
    input_rsi,
    input_sma,
    input_ema,
    input_ad,
    input_adosc
]
```

ДОДАТОК В. РЕАЛІЗАЦІЯ ФРОНТЕНДУ

Структура проекту



Додаток В рис 1 - структура проекту

tsconfig.json

```
{
 "compilerOptions": {
   "target": "es5",
   "lib": [
     "dom",
     "dom.iterable",
     "esnext"
   ],
   "allowJs": true,
   "skipLibCheck": true,
   "esModuleInterop": true,
   "allowSyntheticDefaultImports": true,
   "strict": false,
   "forceConsistentCasingInFileNames": true,
   "noFallthroughCasesInSwitch": true,
   "module": "esnext",
   "moduleResolution": "node",
   "resolveJsonModule": true,
   "isolatedModules": true,
   "noEmit": true,
   "jsx": "react-jsx"
 },
 "include": [
   "src"
 ]
}
```

package.json

```
{
 "name": "stock-ai-frontend",
 "version": "0.1.0",
 "private": true,
 "dependencies": {
   "@emotion/react": "^11.11.4",
   "@emotion/styled": "^11.11.5",
   "@fontsource/roboto": "^5.0.13",
   "@mui/material": "^5.15.18",
   "@mui/x-charts": "^7.5.0",
   "@mui/x-date-pickers": "^7.4.0",
   "@testing-library/jest-dom": "^5.17.0",
   "@testing-library/react": "^13.4.0",
   "@testing-library/user-event": "^13.5.0",
   "@types/jest": "^27.5.2",
   "@types/node": "^16.18.97",
   "@types/react": "^18.3.2",
   "@types/react-dom": "^18.3.0",
   "dayjs": "^1.11.11",
   "react": "^18.3.1",
   "react-dom": "^18.3.1",
   "react-router-dom": "^6.23.1",
   "react-scripts": "5.0.1",
   "typescript": "^4.9.5",
   "web-vitals": "^2.1.4"
 },
 "scripts": {
   "start": "react-scripts start",
   "dev": "react-scripts build &&
react-scripts start",
   "build": "react-scripts build",
   "eject": "react-scripts eject"
 },
 "eslintConfig": {
   "extends": [
     "react-app",
     "react-app/jest"
   ]
 },
 "browserslist": {
   "production": [
     ">0.2%",
     "not dead",
     "not op_mini all"
   ],
   "development": [
     "last 1 chrome version",
     "last 1 firefox version",
     "last 1 safari version"
   ]
 },
 "devDependencies": {
   "sass": "^1.77.1"
 }
}
```

reportWebVitals.ts

```
import { ReportHandler } from 'web-vitals';

const reportWebVitals = (onPerfEntry?:
ReportHandler) => {
 if (onPerfEntry && onPerfEntry instanceof
Function) {
   import('web-vitals').then(({ getCLS,
getFID, getFCP, getLCP, getTTFB }) => {
     getCLS(onPerfEntry);
     getFID(onPerfEntry);
     getFCP(onPerfEntry);
     getLCP(onPerfEntry);
     getTTFB(onPerfEntry);
   });
 }
};

export default reportWebVitals;
```

index.tsx

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from
'./reportWebVitals';

const root = ReactDOM.createRoot(
 document.getElementById('root') as
HTMLElement
);
root.render(
 <React.StrictMode>
   <App />
 </React.StrictMode>
);

// If you want to start measuring performance
in your app, pass a function
// to log results (for example:
reportWebVitals(console.log))
// or send to an analytics endpoint. Learn
more: https://bit.ly/CRA-vitals
reportWebVitals();
```

client-config.ts

```
export const HOST = 'http://localhost:8000/'
```

App.tsx

```
import React from 'react';
import BaseLayout from
"./layouts/BaseLayout/BaseLayout";

function App() {
 return (
   <div className="App">
     <BaseLayout />
   </div>
 );
}

export default App;
```

apiRequests.ts

```
import {HOST} from "../client-config";
```

```
export const getExistingConfigs = async () =>
{
    return fetch(HOST +
'get_data_for_creating_model/')
}

export const createModel = async (requestBody)
=> {
    return fetch(HOST + 'create_model', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(requestBody),
    })
}

export const getModelNames = async () =>
fetch(HOST + 'get_best_models/')

export const getBestEpochByR2 = async
(modelName) => fetch(HOST +
`get_best_epoch_by_r2?name=${modelName}`)

export const getModelHistory = async
(modelName) => fetch(HOST +
`get_model_history?name=${modelName}`)

export const getModelPredictions = async
(modelName) => fetch(HOST +
`get_model_predicted_data?name=${modelName}`)
```

Models.tsx

```
import React, {useEffect, useState} from
'react';
import {getModelNames} from
"../../requests/apiRequests";
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import Box from '@mui/material/Box';
import Model from
"../../components/Model/Model";

const Models = () => {
    const [models, setModels] = useState([]);

    const [tab, setTab] = React.useState(0);
    const handleChange = (event:
React.SyntheticEvent, newValue: number) => {
        setTab(newValue);
    };

    useEffect(() => {
        (async () => {
            const response = await
getModelNames()
            const data = await response.json()
            setModels(data['models'])
        })()
    }, []);
    return (
        <Box sx={{display: 'flex', gap:
'20px'}}>
            <Box>
                <Tabs
                    orientation="vertical"
                    variant="scrollable"
                    value={tab}
                    onChange={handleChange}
                    sx={{ borderRight: 1,
borderColor: 'divider', width: '200px',
height: '100%' }}
                >
                    {!!models?.length &&
models?.map((item, index) => <Tab label={item}
value={index} key={'model-' + index}
sx={{width:'100%', maxWidth: '100%'}} />)}
                </Tabs>
            </Box>
            {!!models && <Model
modelName={models?.[tab]} />}
        </Box>

    );
};

export default Models;
```

Create.tsx

```
import React, {useEffect, useRef, useState}
from 'react';
import {createModel, getExistingConfigs} from
"../../requests/apiRequests";
import Checkbox from '@mui/material/Checkbox';
import FormGroup from
'@mui/material/FormGroup';
import FormControlLabel from
'@mui/material/FormControlLabel';
import FormLabel from
'@mui/material/FormLabel';
import RadioGroup from
'@mui/material/RadioGroup';
import Radio from '@mui/material/Radio';
import Box from '@mui/material/Box';
import styles from './Create.module.scss'
import TextField from
'@mui/material/TextField';
import { AdapterDayjs } from
'@mui/x-date-pickers/AdapterDayjs';
import { LocalizationProvider } from
'@mui/x-date-pickers/LocalizationProvider';
import { DatePicker } from
'@mui/x-date-pickers/DatePicker';
import Button from '@mui/material/Button';
import dayjs from "dayjs";

const Create = () => {
    const [formInputData, setFormInputData] =
useState(null)

    useEffect(() => {
        (async () => {
            try {
                const request = await
getExistingConfigs()
                const data = await
request.json()
                setFormInputData(data)
            } catch (e) {
                console.error(e)
            }
        })()
    }, []);
    const formRef =
useRef<HTMLFormElement>(null);

    const onFormSubmit = async () => {
        const inputIndicators = []

formRef.current?.['indicators']?.values()?.for
Each(input => input?.checked &&
inputIndicators.push(input?.value));

        const inputMetrics = []

formRef.current?.['input_metrics']?.values()?.
forEach(input => input?.checked &&
inputMetrics.push(input?.value));

        const loss =
formRef.current?.['loss']?.values()?.find(i =>
i.checked)?.value;

        const modelMetrics = [];

formRef.current?.['metrics']?.values()?.forEac
```

```
h(input => input?.checked &&
modelMetrics.push(input?.value));

       const model =
formRef.current?.['model']?.values()?.find(i
=> i.checked)?.value;

       const optimizer =
formRef.current?.['optimizer']?.length ?

formRef.current?.['optimizer']?.values()?.find
(i => i.checked)?.value :
formRef.current?.['optimizer']?.checked ?
formRef.current?.['optimizer']?.value : null;

       const output =
formRef.current?.['output']?.values().find(i
=> i.checked)?.value;

       window['optimizer'] =
formRef.current?.['optimizer']
       const requestBody = {
           name:
formRef.current?.name?.['value'],
           percent_for_train:
+formRef.current?.percentage?.['value'] ||
0.8,
           epoches:
+formRef.current?.epoches?.['value'] || 3,
           k: +formRef.current?.k?.['value']
|| 2,
           currency:
formRef.current?.currency?.['value'],
           timesteps:
+formRef.current?.timesteps?.['value'] || 20,
           start_date:
dayjs(formRef.current?.start_date?.['value']).
format('YYYY-MM-DD'),
           end_date:
dayjs(formRef.current?.end_date?.['value']).fo
rmat('YYYY-MM-DD'),
           input_indicator_names:
inputIndicators,
           input_metric_names: inputMetrics,
           loss_function_name: loss,
           metric_names: modelMetrics,
           model_name: model,
           optimizer_name: optimizer,
           output_metric_name: output,
       }

       const values =
Object.values(requestBody)
       if (values.some(value => typeof value
=== 'undefined')) {
           alert('All fields should be
fulfilled')
       } else {
           try {
               console.log(requestBody)
               const response = await
createModel(requestBody)
               console.log(response.status)
           } catch (e) {
               console.error(e)
           }

       }
   }
   return (
       <Box>
           <form id={'form'} ref={formRef}>
               <Box
className={styles['boxes']}>
                   <Box sx={{display: 'flex',
flexDirection: 'column', gap: '10px'}}>
                       <TextField
name={'name'} label="Model Name" required
variant="outlined" defaultValue={'Default
Model'} />
                       <TextField
name={'percentage'} label="Percentage for
train" defaultValue={0.8} required
variant="outlined" />
                       <TextField
name={'currency'} label="Currency"
defaultValue={'AAPL'} required
variant="outlined" />
                       <TextField
name={'timesteps'} label="Timesteps"
defaultValue={20} variant="outlined" />
                       <TextField
name={'epoches'} label="Epoches"
defaultValue={3} variant="outlined" />
                       <TextField name={'k'}
label="How much parts should be in cross
validation" defaultValue={2}
variant="outlined" />
                       <LocalizationProvider
dateAdapter={AdapterDayjs as any}>
                           <DatePicker
label="Start date" name={'start_date'} />
                       </LocalizationProvider>
                       <LocalizationProvider
dateAdapter={AdapterDayjs as any}>
                           <DatePicker
label="End date" name={'end_date'} />
                       </LocalizationProvider>
                   </Box>
                   <FormGroup>
                       <FormLabel>Choose input
indicators</FormLabel>

{(formInputData?.['input_indicators'] as
unknown as string[])
                           ?.map((item, index)
=>

<FormControlLabel control={<Checkbox
value={item} key={index} name={'indicators'}
/>} label={item}/>
                           )}

                   </FormGroup>
                   <FormGroup>
                       <FormLabel>Choose input
stock metrics</FormLabel>

{(formInputData?.['input_metrics'] as unknown
as string[])
                           ?.map((item, index)
=>

<FormControlLabel control={<Checkbox
value={item} key={index}
name={'input_metrics'} />} label={item}/>
                           )}

                   </FormGroup>
                   <RadioGroup>
                       <FormLabel>Choose loss
function</FormLabel>

{(formInputData?.['loss_functions'] as unknown
as string[])
                           ?.map((item, index)
=>

<FormControlLabel control={<Radio value={item}
key={index} name={'loss'} />} label={item}/>
                           )}

                   </RadioGroup>
                   <FormGroup>
                       <FormLabel>Choose
additional model metrics</FormLabel>
```

```
{([...(formInputData?.['loss_functions'] ||
[]) as unknown as string[],
...(formInputData?.['metrics'] || []) as
unknown as string[]])
                            ?.map((item, index)
=>

<FormControlLabel control={<Checkbox
value={item} key={index} name={'metrics'} />}
label={item}/>
                            )}

                    </FormGroup>
                    <RadioGroup>
                        <FormLabel>Choose
model</FormLabel>

{(formInputData?.['models'] as unknown as
string[])
                            ?.map((item, index)
=>

<FormControlLabel control={<Radio value={item}
key={index} name={'model'} />} label={item}/>
                            )}
                    </RadioGroup>
                    <RadioGroup>
                        <FormLabel>Choose
optimizer</FormLabel>

{(formInputData?.['optimizers'] as unknown as
string[])
                            ?.map((item, index)
=>

<FormControlLabel control={<Radio value={item}
key={index} name={'optimizer'} />}
label={item}/>
                            )}
                    </RadioGroup>
                    <RadioGroup>
                        <FormLabel>Choose
output metric</FormLabel>

{(formInputData?.['output_metrics'] as unknown
as string[])
                            ?.map((item, index)
=>

<FormControlLabel control={<Radio value={item}
key={index} name={'output'} />} label={item}/>
                            )}
                    </RadioGroup>
                </Box>
                <Button variant="contained"
onClick={onFormSubmit}>Submit</Button>
            </form>
        </Box>
    );
};

export default Create;
```

Create.module.scss

```
.boxes {
 display: flex;
 flex-wrap: wrap;
 gap: 10px
}
```

BaseLayout.tsx

```
import * as React from 'react';
import {
    createBrowserRouter,
    RouterProvider,
    Navigate
} from "react-router-dom";
import Menu from "../../components/Menu/Menu";
import Create from
"../../pages/Create/Create";
import Models from
"../../pages/Models/Models";

interface TabPanelProps {
    children?: React.ReactNode;
    index: number;
    value: number;
}

const router = createBrowserRouter([
    {
        element: <Menu />,
        children: [
            {
                path: "/",
                element: <Navigate
to={'/create'} replace />,
            },
            {
                path: "/create",
                element: <Create />,
            },
            {
                path: "/models",
                element: <Models />,
            },
        ]
    }
]);

export default function BaseLayout() {


    return (
        <div>
            <RouterProvider router={router} />
        </div>
    );
}
```

TrainProcess.tsx

```
import React, {useEffect, useState} from
'react';
import Box from '@mui/material/Box';
import {getModelHistory} from
"../../requests/apiRequests";
import { LineChart } from
'@mui/x-charts/LineChart';
import Typography from
'@mui/material/Typography';

interface ITrainProcess {
    modelName: string;
}

const TrainProcess: React.FC<ITrainProcess> =
({modelName}) => {
    const [history, setHistory] = useState([])
    useEffect(() => {
        (async () => {
            if (!modelName) {
                return ;
            }
            const response = await
getModelHistory(modelName)
            if (response.status === 200) {
                const data = await
response.json()
                setHistory(data)
            }
        })()
    }, [modelName]);
    return (
```

```
      <Box sx={{display: 'flex', gap: '20px',
flexDirection: 'column', overflowX:
'scroll'}}>
          {!!history?.length &&
history?.map((iteration, index) => {
              return <Box sx={{width:
'100%'}}>
                  <Typography>Iteration
{index + 1}</Typography>
                  <Box sx={{display: 'flex',
gap: '10px'}}>

{Object.entries(iteration).map(([key, value]:
[string, number[]], index) => {
                      return <Box>

<Typography>{key}</Typography>
                          <LineChart
xAxis={[{data:
Array.from(Array(value?.length).keys())}]}
series={[{curve: "linear", data: value}]}
width={400} height={400} />
                      </Box>
                  })}
              </Box>
              </Box>
          })}
      </Box>
  );
};

export default TrainProcess;
```

Predictions.tsx

```
import React, {useEffect, useState} from
'react';
import {getModelPredictions} from
"../../requests/apiRequests";
import dayjs from "dayjs";
import { LineChart } from
'@mui/x-charts/LineChart';
import Box from '@mui/material/Box';

interface IPredictions {
  modelName: string;
}

const Predictions: React.FC<IPredictions> = ({
modelName }) => {
  const [predictions, setPredictions] =
useState<object>(null)

  useEffect(() => {
      (async () => {
          const response = await
getModelPredictions(modelName)
          if (response.status === 200) {
              const data = await
response.json()
              setPredictions(data)
              // console.log(data)
          }
      })()

  }, [modelName]);

  const keys = Object.keys(predictions ||
{});
  const X =
Object.keys(predictions?.[keys?.[0]] ||
{})?.map(key => dayjs(+key))
  const Y: {key: string, value: number[]}[] =
Object.entries(predictions || {})?.map(([key,
value]) => ({key, value: Object.values(value)
|| []}))

  const len = X?.length > 500 ? 500 :
X?.length

  return (
      <Box sx={{overflowX: 'scroll'}}>
          {predictions && <LineChart
xAxis={[{ data: X?.slice(X?.length - 1 - len,
X?.length - 1), valueFormatter: (value) => {
                  return
dayjs(value).format('YYYY-MM-DD')
              } }]} series={Y?.map(({key,
value}) => ({data: value?.slice(value?.length
- 1 - len, value?.length - 1), label: key}))}
width={5 * X.length} height={500} />}
      </Box>
  );
};

export default Predictions;
```

Model.tsx

```
import React from 'react';
import Tab from "@mui/material/Tab";
import Tabs from "@mui/material/Tabs";
import MetricsTable from
"../MetricsTable/MetricsTable";
import Box from '@mui/material/Box';
import TrainProcess from
"../TrainProcess/TrainProcess";
import Predictions from
"../Predictions/Predictions";

const TABS = [
  {
      label: 'Metrics Table',
      component: MetricsTable,
  },
  {
      label: 'Train Process',
      component: TrainProcess,
  },
  {
      label: 'Predictions',
      component: Predictions
  }
];

interface IModel {
  modelName: string;
}

const Model: React.FC<IModel> = ({modelName})
=> {
  const [tab, setTab] = React.useState(0);

  const handleTabChange = (event:
React.SyntheticEvent, newValue: number) => {
      setTab(newValue);
  };

  const CurrentTabComponent =
TABS[tab].component;

  return (
      <Box sx={{width:'100%', overflowX:
'hidden' }}>
          <Tabs value={tab}
onChange={handleTabChange} sx={{marginBottom:
'20px'}}>
              {TABS.map((item, index) => (
                  <Tab label={item.label}
value={index} key={'model-tab-' + index} />
              ))}
          </Tabs>
          <CurrentTabComponent
modelName={modelName} />
      </Box>
  );
};
```

```
export default Model;
```

MetricsTable.tsx

```
import React, {useEffect, useState} from
'react';
import {getBestEpochByR2} from
"../../requests/apiRequests";
import Box from '@mui/material/Box';
import Table from '@mui/material/Table';
import TableBody from
'@mui/material/TableBody';
import TableCell from
'@mui/material/TableCell';
import TableContainer from
'@mui/material/TableContainer';
import TableHead from
'@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';

interface IMetricsTable {
   modelName: string;
}

const MetricsTable: React.FC<IMetricsTable> =
({modelName}) => {
   const [metrics, setMetrics] =
useState<{[key: string]: string |
number}>(null);
   useEffect(() => {
       (async () => {
           if (!modelName) {
               return;
           }
           const response = await
getBestEpochByR2(modelName)
           if (response.status === 200) {
               const data = await
response.json()
               setMetrics(data)
           }

       })()

   }, [modelName]);

   return (
       <Box>
           {!!metrics &&
               <TableContainer
component={Paper} sx={{ width: '100%' }}>
                   <Table sx={{ width: '100%'
}}>
                       <TableHead>
                           <TableRow>

<TableCell></TableCell>

{Object.keys(metrics).map((key, index) =>
                                   <TableCell
align="center"
key={`h-row-${index}`}>{key}</TableCell>)}
                           </TableRow>
                       </TableHead>
                       <TableBody>
                           <TableRow>

<TableCell>{modelName}</TableCell>

{Object.values(metrics).map((value, index) =>
                                   <TableCell
align="left"
key={`row-${index}`}>{value}</TableCell>)}
                           </TableRow>
                       </TableBody>
                   </Table>
               </TableContainer>
           }
       </Box>
   );
};

export default MetricsTable;
```

Menu.tsx

```
import React from 'react';
import {Link, Outlet} from "react-router-dom";
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import Box from '@mui/material/Box';
import styles from './Menu.module.scss'

const menuList = [
   {
       link: '/create',
       label: 'Create'
   },
   {
       link: '/models',
       label: 'Models'
   }
]
const Menu = () => {
   const [value, setValue] = React.useState(()
=> {
       return menuList.reduce((acc, item,
index) => {
           if
(window.location.href.endsWith(item.link)) {
               return index
           }
           return acc
       }, 0)
   });

   const handleChange = (event:
React.SyntheticEvent, newValue: number) => {
       setValue(newValue);
   };

   return (
       <Box >
           <Box className={styles['menu']}>
               <Tabs value={value}
onChange={handleChange}>
                   {menuList.map((item, index)
=> (
                       <Tab label={item.label}
value={index} component={Link} to={item.link}
key={index} />
                   ))}
               </Tabs>
           </Box>
           <Outlet />
       </Box>
   );
};

export default Menu;
```

Menu.module.scss

```
.menu {
 margin-bottom: 20px;
}
```

index.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="utf-8" />
   <link rel="icon"
href="%PUBLIC_URL%/favicon.ico" />
   <meta name="viewport"
content="width=device-width, initial-scale=1"
/>
```

```
    <meta name="theme-color" content="#000000"
/>
    <meta
      name="description"
      content="Web site created using
create-react-app"
    />
    <link rel="apple-touch-icon"
href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when
your web app is installed on a
      user's mobile device or desktop. See
https://developers.google.com/web/fundamentals
/web-app-manifest/
    -->
    <link rel="manifest"
href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the
tags above.
      It will be replaced with the URL of the
`public` folder during the build.
      Only files inside the `public` folder can
be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico",
"%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side
routing and a non-root public URL.
      Learn how to configure a non-root public
URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to
run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser,
you will see an empty page.

      You can add webfonts, meta tags, or
analytics to this file.
      The build step will place the bundled
scripts into the <body> tag.

      To begin the development, run `npm start`
or `yarn start`.
      To create a production bundle, use `npm
run build` or `yarn build`.
    -->
  </body>
</html>
```