НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

«На правах рукопису»

УДК 519.688:004.855.5

«До захисту допущено»
Завідувач кафедри
_____ Олег ЧЕРТОВ
«___» _____ 2023 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-науковою програмою «Наука про дані та математичне моделювання»
зі спеціальності 113 «Прикладна математика»
на тему: «Математичне та програмне забезпечення системи прогнозування вартості фінансових активів компанії»

Виконав: студент II курсу, групи КМ-11мн
Агафонов Дмитро Сергійович _____

Керівник:
кандидат технічних наук, доцент,
Сирота Сергій Вікторович _____

Консультант з нормоконтролю:
старший викладач,
Мальчиков Володимир Вікторович _____

Рецензент:
Посада, науковий ступінь, вчене звання,
Прізвище, ім’я, по батькові _____

Засвідчую, що в цій магістерській дисертації немає запозичень із праць інших авторів без відповідних посилань.
Студент _____

Київ — 2023

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра прикладної математики

Рівень вищої освіти — другий (магістерський)

Спеціальність – 113 «Прикладна математика»

Освітньо-наукова програма «Наука про дані та математичне моделювання»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Олег ЧЕРТОВ
«___» _____ 2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Агафонову Дмитру Сергійовичу

1. Тема дисертації: «Математичне та програмне забезпечення системи прогнозування вартості фінансових активів компанії», науковий керівник дисертації Сирота Сергій Вікторович, канд. техн. наук, доцент, затверджені наказом по університету від «30» березня 2023 р. № 1359-С.
2. Термін подання студентом дисертації: «15» травня 2023 р.
3. Об’єкт дослідження: Методи прогнозування вартості фінансових показників компанії з використанням моделей глибинного навчання, технік ансамблювання моделей машинного навчання, генеративних змагальних мереж. Методи технічного аналізу для вилучення необхідних патернів та прогнозування ймовірної зміни цін. Нейронні мережі для роботи з текстом spacy, nltk, TfidfVectorize, BERT та finBERT. Алгоритми пониження розмірності t-SNE, UMAP, Factor Analysis, Feature Selection methods, Autoencoders. Методи прогнозування часових рядів: багатовимірна лінійна регресія, ARIMA, марківська модель, ANN, RNN LSTM. Методи

ансамблювання базових моделей bagging, stacking, boosting. Генеративні змагальні мережі GAN.

Існуючі комерційні програмні рішення: StocksNeural, Stocksight, Deep Convolution Stock Technical Analysis.

4. Предмет дослідження: Фінансові показники з можливою сезонною компонентою: вартість фінансових активів компанії, прибуток, рентабельність, об’єм фін. засобів.

Вплив на вартість фінансових активів компанії таких показників, як:

- корельовані активи – показники залежних, схожих за економічною діяльністю або конкуруючих компаній;
- біржеіві товари – енергетична сировина, кольорові та дорогоцінні метали, промислова сировина тощо;
- курси валют;
- фондові індекси;
- кількість запитів в пошуковій системі;
- фінансові новини.

Можливості технічного аналізу для прогнозування ймовірних змін вартості фінансових показників.

Створення інформативних високорівневих ознак із застосуванням UMAP, t-SNE, Feature Selection methods, Autoencoders. Порівняльний аналіз базових моделей прогнозування, об’єднання їх в ансамбль методами bagging, stacking, boosting. Можливість застосування генеративної змагальної мережі GAN, підбор моделей для генератора та дискримінатора.

5. Перелік завдань, які потрібно розробити:

5.1 Огляд існуючих методів та рішень прогнозування фінансової інформації, їх порівняння та недоліки, дослідження особливостей предметної області. Розглядання текстової фінансової інформації, зокрема фінансових новин, як джерело впливу на показники вартості фінансових активів компанії. Аналіз наявних можливостей отримати прогноз ціни акцій на основі економічних,

фінансових показників та даних, отриманих в результаті семантичного аналізу фінансових новин, технічного аналізу і математичних перетворень

5.2 Розробка моделі системи аналізу та прогнозування фінансових показників, опис складових компонентів системи та функціональних компонентів. Побудова UML діаграм та діаграми послідовності.

5.3 Опис математичного забезпечення. Реалізація технічного аналізу фінансових показників, моделей оцінки емоційної забарвленості тексту фінансових новин, моделей прогнозування вартості цінних паперів компанії.

5.4 Розробка програмного забезпечення. Реалізація всіх програмних елементів, які необхідні для вирішення задачі дисертаційного дослідження, розробка інтерфейсу.

5.5 Верифікування та валідація результатів роботи системи прогнозування. Опис, оцінка та порівняння метрик.

6. Орієнтовний перелік ілюстративного матеріалу: модель системи, діаграми компонентів, діаграма послідовності, огляд мовної моделі для класифікації настроїв тексту, архітектура рекурентних нейронних мереж, діаграма взаємодії компонентів архітектури, ілюстративний матеріал.

7. Перелік публікацій:

- Маслянко, П. П., Агафонов Д. С. Порівняльний аналіз підходів та методів оцінювання емоційного забарвлення фінансової інформації про поточний стан компанії. Прикладна математика та комп’ютинг. ПМК-2022: п’ятнадцята науково-практична конференція магістрантів та аспірантів, Київ, 16-17 лист. 2022 р.: зб. Тез доп./ [редкол.: Дичка І. А. та ін.]. — К. : Просвіта, 2022. — С. 129-139.

8. Дата видачі завдання: «1» лютого 2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Ґрунтовне ознайомлення з предметною областю	15.09.2022	
2	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури	01.10.2022	
3	Робота над першим розділом магістерської дисертації	15.12.2022	
4	Проведення наукового дослідження; робота над другим розділом магістерської дисертації	15.01.2023	
5	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	15.02.2023	
6	Робота над третім розділом магістерської дисертації; підготовка статті за результатами наукового дослідження; розроблення програмного забезпечення	01.03.2023	
7	Завершення роботи над основною частиною магістерської дисертації; робота над розділом з охорони праці	15.05.2023	
8	Оформлення текстової і графічної частин магістерської дисертації	25.05.2023	

Студент _____ Дмитро АГАФОНОВ

Науковий керівник дисертації _____ Сергій СИРОТА

Реферат

Дисертацію виконано на 106 аркушах, вона містить 2 додатки та перелік посилань на використані джерела з 64 найменувань. У роботі наведено 19 рисунків та 2 таблиці.

Актуальність теми. Прогнозування вартості цінних паперів є надзвичайно важливим завданням для інвесторів і фінансових установ для прийняття обґрунтованих інвестиційних рішень, ефективного управлінні ризиками та загальної ефективності портфеля. Поєднання технічного аналізу активів ансамблювання моделей машинного та глибокого навчання, аналізу настроїв фінансових новин за допомогою мовних моделей має великий потенціал для підвищення точності та надійності прогнозів цін на акції. Технічний аналіз активів передбачає використання статистичних методів для аналізу історичних даних про ціни та обсяги для виявлення тенденцій і закономірностей на фондовому ринку. Цей метод десятиліттями використовувався трейдерами та інвесторами для прийняття інвестиційних рішень. Технічний аналіз можна автоматизувати та застосувати до великих наборів даних, дозволяючи робити більш точні та надійні прогнози. Методи глибокого навчання можуть вивчати складні взаємозв’язки між різними ринковими факторами, що дає точніші прогнози. В такі моделі можуть бути включені останні ринкові новини, економічні показники та інші відповідні фактори, щоб надати більш тонку та детальну інформацію. Ансамблювання моделей передбачає поєднання кількох моделей для підвищення точності та надійності прогнозів. Аналіз настроїв фінансових новин передбачає використання методів обробки природної мови для аналізу новинних статей і соціальних мереж, щоб визначити настрої ринку щодо певної акції чи компанії. Цей метод може

надати додаткову інформацію про ринкові тенденції та настрої інвесторів, що призводить до більш точних прогнозів.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконувалась згідно з планом науково-дослідних робіт кафедри прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Мета і задачі дослідження. Метою дисертаційної роботи є розробка математичного та програмного забезпечення для покращення прогнозу вартості фінансових активів компанії (порівняно з традиційними методами прогнозування часових рядів) із застосуванням семантичного аналізу тексту, автоенкодерів, ансамблю моделей, генеративних змагальних мереж GAN з метою підвищення конкурентоспроможності компанії на ринку, покращення оптимізації та управління її ресурсами, забезпечення інвесторів та клієнтів фінансовим прогнозом.

Для досягнення вказаної мети було розв'язано такі задачі:

- Збір та аналіз необхідних для навчання моделей даних;
- Тренування різних моделей машинного навчання, їх ансамблювання;
- Розробка інтерфейсу та забезпечення роботи програми в реальному часі.

Об'єктом дослідження є система прогнозування вартості фінансових показників компанії з використанням генеративних змагальних мереж GAN та навчання з підкріпленням RL. Методи технічного аналізу для прогнозування ймовірної зміни цін. Нейронні мережі для роботи з текстом spacy, nltk, BERT та finBERT. Алгоритми пониження розмірності t-SNE, UMAP, Factor Analysis, Feature Selection methods, Autoencoders. Методи прогнозування часових рядів багатовимірна лінійна регресія, ARIMA, марківська модель, fast-forward

NN, RNN LSTM. Методи ансамблювання базових моделей bagging, stacking, boosting. Генеративні змагальні мережі GAN. Існуючі комерційні програмні рішення: StocksNeural, Stocksight, Deep Convolution Stock Technical Analysis.

Предметом дослідження є техніки ансамблювання моделей машинного навчання, вплив на вартість фінансових активів компанії таких показників, як корельовані активи – показники залежних, схожих за економічною діяльністю або конкуруючих компаній; біржеіві товари – енергетична сировина, кольорові та дорогоцінні метали, промислова сировина тощо; курси валют; фондові індекси; кількість запитів в пошуковій системі; фінансові новини. Можливості технічного аналізу для прогнозування ймовірних змін вартості фінансових показників. Створення інформативних високорівневих ознак ззастосуванням t-SNE, UMAP, Feature Selection methods, Autoencoders. Порівняльний аналіз базових моделей прогнозування, об'єднання їх в ансамбль методами bagging, stacking, boosting. Можливість застосування генеративної змагальної мережі GAN, підбор моделей для генератора та дискримінатора.

Методи дослідження. Для розв'язання поставленої задачі використовувалися такі методи: методи машинного навчання з учителем, навчання з підкріпленням, генеративні змагальні мережі, методи зменшення розмірності та вибору важливих ознак.

Наукова новизна одержаних результатів полягає в потенціалі для покращення процесу прийняття інвестиційних рішень та управління ризиками на фінансових ринках за допомогою програмного забезпечення, створення якого є метою даної магістерської дисертації, що здатне інтегрувати набір методологій та алгоритмів в єдину систему, що дозволяє надавати більш точні та надійні прогнози. Програмне забезпечення здатне поєднувати велику кількість ознак (отриманих шляхом дата майнингу, інженірінгу ознак, технічного аналізу), ансамблі моделей машинного та

глибокого навчання, мовну модель визначення емоційної забарвленості фінансових текстів для надання більш точних та надійних прогнозів порівняно з наявними існуючими рішеннями. Тому підхід до прогнозування фінансової інформації, який полягає в використанні якомога більшої кількості ознак для навчання моделі і представлений в даній дипломній роботі є найбільш вдалим та актуальним.

Практичне значення одержаних результатів. Реалізовану систему можна застосовувати для проведення аналітики та прогнозування економічного стану окремої компанії; під час торгів на ринку акцій в режимі реального часу.

Апробація результатів дисертації. Деякі положення й результати роботи дисертації доповідались та опубліковані у матеріалах XV наукової конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг - ПМК-2022» (Київ 16-17 листопада 2022 року).

Публікації. Результати дослідження, викладені в одному з розділів дисертації, представлені в науковій праці: тези «Порівняльний аналіз підходів і методів оцінювання емоційного забарвлення фінансової інформації про поточний стан підприємства» на XV конференції ім. магістрантів та аспірантів «Прикладна математика та обчислювальна техніка – ПМК-2022».

Ключові слова: ціни акцій, корельовані активи, технічні індикатори, пониження розмірності, семантичний аналіз тексту, стаціонарний часовий ряд, рекурентні нейронні мережі, модель довгої короткострокової пам'яті.

ABSTRACT

The thesis is presented in 106 pages. It contains 2 appendixes and bibliography of 64 references. 19 figures and 2 table are given in the thesis.

Topic relevance. Forecasting the value of securities is an extremely important task for investors and financial institutions to make informed investment decisions, effective risk management and overall portfolio performance. The combination of technical asset analysis, ensemble machine and deep learning models, financial news sentiment analysis using linguistic models has great potential to improve the accuracy and reliability of stock price forecasts. Technical analysis of assets involves the use of statistical methods to analyze historical data on prices and volumes to identify trends and patterns in the stock market. This method has been used by traders and investors for decades to make investment decisions. Technical analysis can be automated and applied to large data sets, allowing for more accurate and reliable predictions. Deep learning techniques can learn complex relationships between various market factors, resulting in more accurate predictions. Such models may incorporate recent market news, economic indicators and other relevant factors to provide more nuanced and detailed information. Model ensemble involves combining several models to increase the accuracy and reliability of forecasts. Financial news sentiment analysis involves using natural language processing techniques to analyze news articles and social media to determine market sentiment about a particular stock or company. This method can provide additional information about market trends and investor sentiment, leading to more accurate forecasts.

Thesis connection to scientific programs, plans, and topics. The dissertation work was carried out in accordance with the plan of research works

of the Department of Applied Mathematics of the National Technical University of Ukraine "Ihor Sikorskyi Kyiv Polytechnic Institute".

Research goal and objectives. The aim of the dissertation is the development of mathematical and software for improving the forecast of the value of the company's financial assets (compared to traditional time series forecasting methods) using semantic text analysis, autoencoders, model ensembles, generative competitive GAN networks with the aim of increasing the company's competitiveness on the market, improving optimization and management of its resources, providing investors and clients with a financial forecast.

To achieve this goal, the following tasks were solved:

- Collection and analysis of data models necessary for training;
- Training of various machine learning models, their ensemble;
- Development of the interface and ensuring the operation of the program in real time.

The object of the study is a system for predicting the value of the company's financial indicators using generative adversarial GAN networks and RL reinforcement learning. Methods of technical analysis for forecasting likely price changes. Spacy, nltk, BERT and finBERT text neural networks. t-SNE, UMAP, Factor Analysis, Feature Selection methods, Autoencoders. Time series forecasting methods: multivariate linear regression, ARIMA, Markov model, fast-forward NN, RNN LSTM. Methods of assembling basic models of bagging, stacking, boosting. Generative adversarial GAN networks. Existing commercial software solutions: StocksNeural, Stocksight, Deep Convolution Stock Technical Analysis.

The subject of the research is the techniques of assembling machine learning models, the influence on the value of the company's financial assets of such indicators as correlated assets - indicators of dependent, similar in economic activity or competing companies; exchange goods – energy raw

materials, non-ferrous and precious metals, industrial raw materials, etc.; exchange rates; stock indices; the number of requests in the search engine; financial news. Possibilities of technical analysis for forecasting likely changes in the value of financial indicators. Creation of informative high-level features using t-SNE, UMAP, Feature Selection methods, Autoencoders. Comparative analysis of basic forecasting models, combining them into an ensemble using methods of bagging, stacking, boosting. The possibility of applying a generative competitive GAN network, selection of models for the generator and discriminator.

Methods of research. The following methods were used to solve this problem: machine learning methods, reinforced learning, generative competition networks, methods to reduce the dimension and the choice of important features.

Scientific contribution. The scientific novelty of the obtained results lies in the potential for improving the process of investment decision-making and risk management in financial markets with the help of software, the creation of which is the goal of this master's thesis, which is able to integrate a set of methodologies and algorithms into a single system that allows providing more accurate and reliable forecasts The software is able to combine a large number of features (obtained by data mining, feature engineering, technical analysis), ensembles of machine and deep learning models, a language model for determining the emotional coloring of financial texts to provide more accurate and reliable predictions compared to existing existing solutions. Therefore, the approach to forecasting financial information, which consists in using as many features as possible for training the model and presented in this thesis, is the most successful and relevant.

Practical value of obtained results. The implemented system can be used to analyze and forecast the economic condition of an individual company; during trading on the stock market in real time.

Approbation of the thesis results. The results of the research presented in one of the sections of the dissertation are presented in the scientific work: theses "Comparative analysis of approaches and methods for evaluating the emotional coloring of financial information about the current state of the company" at the XV conference of master's and postgraduate students "Applied mathematics and computing - PMK-2022".

Publications. The results of the research presented in one of the sections of the dissertation are presented in the scientific work: theses "Comparative analysis of approaches and methods for evaluating the emotional coloring of financial information about the current state of the company" at the XV conference of master's and postgraduate students "Applied mathematics and computing - PMK-2022".

Keywords: stock prices, correlated assets, technical indicators, dimensionality reduction, semantic text analysis, stationary time series, recurrent neural networks, long short-term memory model.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – програмний інтерфейс додатку.

AR – авторегресія.

ARIMA – авторегресійна інтегрована модель ковзаючого середнього, autoregressive integrated moving average.

BERT - Bidirectional Encoder Representations from Transformers.

FinBERT – financial Bidirectional Encoder Representations from Transformers

LSTM – модель довгої-короткострокової пам’яті, Long-Short Term Memory

MA – ковзаюче середнє

MACD – збіжність/розбіжність ковзаючих середніх, moving average convergence/divergence

MFI – індекс грошового потоку

MLM – маскована язикова модель, Masked Language Model

NLP - Natural Language Processing, обробка природної мови

RNN – рекурентна нейронна мережа, Recurrent Neural Network

SMA – просте ковзаюче середнє, simple moving average

Std – стандартне відхилення

TRIX – тройне експоненційне ковзаюче середнє

VAR – модель векторної авторегресії, Vector Autoregression

VWAP – середньозважена ціна за обсягом

ВСТУП

Торгівля цінними паперами на біржі має досить довгу та багатогранну історію. Якщо раніше біржові операції проводились досить вузьким колом фахівців, то останнім часом набрав популярність спосіб торгівлі акціями через Інтернет, тобто інтернет-трейдинг. Процедура купівлі-продажу сильно спрощується, тому все більше інвесторів та спекулянтів отримує можливість виконувати операції в режимі реального часу. Разом з тим, всі учасники торгівлі прагнуть до того, щоб отримати найбільш точний прогноз щодо розвитку ситуації на ринку цінних паперів. Зазвичай, для цього використовують різні інструменти, такі як: фундаментальний аналіз, технічний аналіз, алгоритми оцінки та машинне навчання, аналіз громадської думки та настроїв на ринку, застосування гібридних інструментів, тобто комбінація з вищеназваних інструментів [1].

В ході виконання роботи розглянуто декілька методів прогнозування багатовимірних часових рядів та обраний найбільш підходящий. Ключовим моментом для створення моделі та можливості ефективного прогнозування даних є збір та створення якомога більшої кількості ознак, які можуть бути пов’язані або впливати на прогнозовані дані, а також ефективне зменшення їх розмірності з мінімальними втратами залежностей.

Основними проблемами прогнозування вартості фінансових показників є висока стохастичність цін на акції, що ускладнює їх точне прогнозування; нелінійні та складні залежності в даних, що ускладнює їх моделювання за допомогою лінійних або традиційних статистичних методів, недостача доступу до високоякісних і повних джерел даних, які не завжди можуть бути доступними або точними – на якість прогнозу можуть впливати такі фактори, як зміщення даних, помилки вимірювання

та відсутність даних; обмеженість прогнозу короткими термінами та випадкові події – існує багато факторів, які знаходяться поза контролем інвесторів або аналітиків (наприклад несподівані події, зміни в нормативних актах і макроекономічні умови), що може призвести до непередбачуваних і раптових змін цін на акції; моделі прогнозування курсу акцій можуть бути схильні до перенавчання, що призводить до поганого узагальнення нових даних. [2]

Автором магістерської дисертації вже були започатковані дослідження щодо моделі BERT для вирішення фінансових задач в рамках написання тез «Порівняльний аналіз підходів і методів оцінювання емоційного забарвлення фінансової інформації про поточний стан підприємства» на XV конференції ім. магістрантів та аспірантів «Прикладна математика та обчислювальна техніка – ПМК-2022», а також щодо програмного забезпечення для прогнозування вартості фінансових паперів компанії при виконанні бакалаврської роботи і частково ці результати будуть використовуватись в цій магістерській дисертації.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність задачі прогнозування фінансової інформації

Прогнозування руху цін фінансових активів є важливою та актуальною задачею в інвестиційній, фінансовій, академічних діяльностях, зокрема:

- Прийняття інвестиційних рішень: інвестори, трейдери та менеджери фондів покладаються на прогнози цін на акції, щоб приймати обґрунтовані рішення щодо купівлі, продажу та зберігання акцій. Точні прогнози цін на акції можуть допомогти їм визначити недооцінені або переоцінені акції, керувати ризиками та оптимізувати ефективність свого портфеля.
- Корпоративні фінанси. Компанії використовують прогнози цін на акції, щоб оцінити вартість власних акцій, а також акцій своїх конкурентів і потенційних цілей придбання. Цю інформацію можна використовувати для прийняття стратегічних рішень, таких як злиття та поглинання, викуп акцій і виплата дивідендів.
- Економічний аналіз. Прогнозування курсу акцій може надати уявлення про загальний стан економіки та фінансових ринків. Їх можна використовувати для моніторингу економічних показників, таких як процентні ставки, інфляція та зростання ВВП, а також для оцінки впливу економічної політики та подій на фондовий ринок.
- Академічні дослідження: Прогнозування курсу акцій є активною сферою досліджень у сфері фінансів, економіки та інформатики. Дослідники використовують прогнози цін на акції для тестування та розробки нових теорій, моделей і алгоритмів, а також для глибшого розуміння складної динаміки фінансових ринків.

1.2 Огляд існуючих методів прогнозування часових рядів

Існує значна кількість методів, які можна успішно використовувати для аналізу та прогнозування часових рядів. Однак, у даній дипломній роботі акцент робиться на прогнозуванні фінансових даних, а саме, на прогнозуванні поведінки цін акцій на ринку. Слід детально розглянути моделі, що ефективно використовуються при роботі з багатовимірними фінансовими часовими рядами, а особливо ті методи, що здатні виявляти нелінійні залежності, приховані патерни та глобальні і локальні тренди, оскільки ці критерії є головними у виборі методу.

1.2.1 Векторна авторегресія

Модель векторної авторегресії (VAR - Vector Autoregression) є популярним методом прогнозування багатовимірних часових рядів, зокрема у фінансовому та економічному аналізі. Модель VAR була запропонована [4] як альтернатива системам економічних рівнянь, що визначають взаємозв’язок між економічними змінними, а саме системам одночасних рівнянь [5]. Модель VAR описує залежність між змінними часового ряду шляхом визначення кожної змінної як лінійної комбінації її попередніх значень та попередніх значень всіх інших змінних. Модель VAR порядку p можна записати у векторно-матричній формі. (формула 1.1):

$$y_t = a_0 + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + \varepsilon_t =$$

$$= a_0 + \sum_{m}^{p} A_m y_{t-m} + \varepsilon_t, \#(1.1)$$

де $y_t = y_t^1, y_t^2, \dots, y_t^k$ – вектор часових рядів;

A_i – матриця коефціцєнтів;

a_0 – константа;

ε_t – залишкова похибка (residuals).

Формула (1.1) є замкнутою, тому що в ролі екзогенних змінних виступають лаги ендогенних змінних. Модель в матричному вигляді, доповнена екзогенними змінними до порядку q має формулу (1.2):

$$y_t = a_0 + \sum_{m=1}^{\rho} A_m y_{t-m} + \sum_{n=0} B_n x_{t-n} + \varepsilon_t, \#(1.2)$$

де $y_t = y_t^1, y_t^2, \dots, y_t^k$ – вектор часових рядів;

A_i – матриця коефціцєнтів ендогенної змінної;

a_0 – константа;

B_i – матриця коефціцєнтів екзогенної змінної;

y_t – вектор екзогенних часових рядів;

ε_t – залишкова похибка (residuals).

Можна використовувати модель векторної авторегресії для аналізу як стаціонарних, так і нестаціонарних часових рядів, що мають інтегральний порядок I(1). При використанні цієї моделі для нестаціонарних рядів, необхідно зазначати порядок різниці. [6].

Модель векторної авторегресії є статистичною, її функціональна залежність визначається аналітично між майбутніми та фактичними значеннями часового ряду та зовнішніми факторами.

До переваг моделі VAR відносять її гнучкість у визначенні кількості змінних та порядку моделі, можливість врахування взаємодії між змінними та можливість отримання прогнозів на довготерміновий період.

1.2.2 Багатовимірна лінійна регресія

Багатовимірна лінійна регресія - це статистичний метод аналізу даних, що використовується для моделювання залежності між залежною змінною та кількома незалежними змінними [7].

У контексті прогнозування часових рядів, багатовимірна лінійна регресія може використовуватися для прогнозування майбутніх значень часових рядів на основі історичних даних та інших релевантних змінних.

Формула багатовимірної лінійної регресії є лінійною функцією (формула 1.3) [8]:

$$y_i = w_0 + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_P x_{ip} + \varepsilon = w_0 + wx_i, \qquad w = (w_1, w_2, \dots, w_P) \ \#(1.3)$$

де $i = 0, n$ – кількість елементів у вибірці;

y_i – залежна (пояснювана) змінна;

x_i – незалежна (пояснююча) змінна;

w_0 – вільний коефіцієнт (intercept);

w_P – коефіцієнти пояснюючих змінних (slope);

ε – залишкова похибка (residuals).

Метою навчання моделі є мінімізація функції втрат (формула 1.4):

$$L(w,b) = \sum_{i=1}^{n} \left(y_i - (w_0 + wx_i)\right)^2 . \#(1.4)$$

Для додання зміної w_0 до вектора коефіцієнтів $w \epsilon \mathbb{R}^p$, необхідно:

а) Додати одну ознаку, де кожен елемент дорівнює одиниці $\tilde{x} = (1, x) \in \mathbb{R}^{p+1}$;

б) Встановити $\tilde{w} = (w_0, w) \in \mathbb{R}^{p+1}$;

в) Тоді $y = w_0 + wx = \tilde{w}\tilde{x} = (w_0, w)(1, x) = w_0 + wx$.

В результаті, функцію втрат (1.4) можна записати як (формула 1.5):

$$L(w,b) = \sum_{i=1}^{n} (y_i - \tilde{w}\tilde{x}_i)^2 \ \#(1.5)$$

Формулу багатовиміної лінійної регресії та функцію втрат можна представити в матричній формі, оскільки для комп'ютерних обчислень є більш оптимальним матричне представлення.

Значення незалежних змінних представлені в матричному вигляді (1.6) розмірністю n x (p+1):

$$X = \begin{pmatrix} \leftarrow & \tilde{x}_1 & \rightarrow \\ \leftarrow & \tilde{x}_2 & \rightarrow \\ \leftarrow & \tilde{x}_n & \rightarrow \end{pmatrix} \#(1.6)$$

Значення залежних змінних матрицею (1.7) розмірністю n x 1:

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ y_n \end{pmatrix} \#(1.7)$$

Функція втрат (1.4) може бути записана (формула 1.8):

$$L(\widetilde{w}) = \|Y - X\widetilde{w}\|, \#(1.8)$$

де $X\widetilde{w} = \begin{pmatrix} \widetilde{w} * \tilde{x}_1 \\ \widetilde{w} * \tilde{x}_2 \\ \widetilde{w} * \tilde{x}_n \end{pmatrix}$;

Детальніше розписана формула 1.8 (формула 1.9):

$$Y - X\widetilde{w} = \begin{pmatrix} y_1 - \widetilde{w} * \tilde{x}_1 \\ y_2 - \widetilde{w} * \tilde{x}_2 \\ y_n - \widetilde{w} * \tilde{x}_n \end{pmatrix} = (2.6) = \|Y - X\widetilde{w}\|, \#(1.9)$$

Мінімізація коефіцієнтів пояснювальних змінних відбувається за допомогою матричних операцій, матриця має розмірність (p+1, n) (формула 1.10):

$$\widetilde{w} = (X^T X)^{-1}(X^T Y), \#(1.10)$$

де першим у відповіді буде значення вільного коефіцієнту w_0 (intercept) з формул (1.3 – 1.4).

При роботі з багатовимірними часовими рядами важливо враховувати можливість мультиколінеарності між незалежними змінними, що може призвести до неточних результатів моделювання та прогнозування. Одним з ключових припущень багатовимірної лінійної регресії є лінійність залежності між залежною змінною та незалежними змінними. Виходячи з поставленої в даній магістерській дисертації задачі необхідно, щоб модель могла виявляти нелінійні залежності в даних.

Функціональна залежність між фактичними значеннями часового ряду та значеннями часового ряду з лагом, а також зовнішніми факторами задана аналітично. Модель багатовимірної лінійної регресії є статистичною.

1.2.3 Модель ARIMA

Модель авторегресійного інтегрованого ковзаючого середнього (autoregressive integrated moving average, далі – ARIMA). Вона призначена для прогнозування нестаціонарних часових рядів $x(t)$, які мають наступні властивості [5]:

а) Ряд, що аналізується моделлю, повинен включати в себе адитивну складову, яка має вигляд алгебраїчного полінома, залежного від часу t;

б) Ряд $x_k(t)$, отриманий в результаті застосування до початкового ряду процедури методу послідовних різниць, може бути описаний за допомогою моделі авторегресіного ковзаючого середнього ARMA. Іншими словами, ряд $x(t)$ повинен мати властивість перетворення в стаціонарний в результаті послідовного взяття різниць [5].

ARIMA поєднує в собі трьохкомпонентну модель, що включає авторегресійну, інтегровану та ковзаючу середню складові. Модель ARIMA може бути записана як ARIMA(p, d, q), де p - порядок авторегресії, d - порядок інтегрування, а q - порядок ковзної середньої. Модель аналізованого процеса $x(t)$, де t = 1,2,..n може бути представлена формулою (1.11) [5]:

$$x_k(t) = \alpha_1 x_k(t-1) + \alpha_2 x_k(t-2) + \cdots + \alpha_p x_k(t-p) + \delta(t) - \#(1.11)$$
$$-\theta_1 \delta(t-1) - \cdots - \theta_q \delta(t-q),$$

де α_i, θ_i – коефіцієнти моделі, що змінюються під час навчання;

p – кількість зсунутих точок, порядок зсуву (для авторегресійної частини);

k – кількість застосувань різниць до початкового ряду, поки ряд не стане стаціонарним;

q – розмір вікна ковзаючого середнього або порядок ковзаючого середнього (для частини ковзаючого середнього);

t – момент часу;

$x_k(t)$ можна представити як (2.12):

$$x_k(t) = x(t)\Delta^k = x(t) - C_k^1 x(t-1) + C_k^2 x(t-2) - \cdots + (-1)^k x(t-k), \#(1.12)$$
$$t = k+1, k+2, \ldots, n$$

де Δ^k – оператор різниці часового ряду порядку k;

C_k^i – функція послідовного взяття різниць ряду.

Модель ARIMA має наступні компоненти:

- авторегресія (AR) – відповідає моделі, в якій значення з попередніх моментів часу використовуються в якості вхідних даних для прогнозування наступних значень;
- інтегрованість (I) – приведення ряду до стандартного вигляду;
- ковзаюче середнє (MA) – включає залежність між значенням та похибкою з моделі ковзаючого середнього, яка застосовується до попередніх значень часового ряду.

Метою використання моделі ARIMA (Autoregressive Integrated Moving Average) є прогнозування динаміки часового ряду шляхом аналізу різниць між його значеннями. У модель ARIMA можуть бути включені додаткові змінні як незалежні, що використовуються для навчання моделі.

При застосуванні моделі ARIMA на даних, де попереднє значення впливає на наступне, вона дає гарні результати. Однак, для задачі магістерської дисертації вона не є найкращим варіантом, оскільки не дозволяє виявляти складні нелінійні та приховані залежності між незалежними змінними.

Модель ARIMA є статистичною моделлю, де функціональна залежність між залежними та незалежними змінними задається аналітично.

1.2.4 Моделі на основі дерев прийняття рішень

Випадковий ліс та його варіації, такі як Gradient Boosted Trees, стають все більш популярними методами машинного навчання для прогнозування часових рядів. У моделі лісу ансамбль дерев рішень будується на різних підмножинах навчальних даних, а остаточний прогноз є середнім або зваженим середнім прогнозів усіх дерев у лісі [9]. Деревоподібні моделі навчені передбачати майбутні значення на основі вікна минулих значень і, можливо, інших характеристик.

Однією з переваг дерев’яних моделей для прогнозування часових рядів є те, що вони можуть обробляти нелінійні залежності між вхідними змінними та цільовою змінною. Вони також відносно швидкі для навчання і можуть бути застосовані до багатовимірних даних з багатьма функціями. Дерев’яні моделі також стійкі до викидів і відсутніх значень, оскільки вони можуть використовувавти інші дерева в лісі для надання точних прогнозів.

Наступна формула ілюструє передбачення моделі лісу [9]:

$$\hat{y}(t) = \frac{1}{K}\sum_{k=1}^{K} f_k(X(t), X(t-1), \dots, X(t-p), \mathbf{z}) \quad \#(1.13)$$

Де $\hat{y}(t)$ – передбачене значення часового ряду в час t

$X(t), X(t-1), \dots, X(t-p)$ – останні p значень часового ряду

$\mathbf{z}$ – вектор додаткових ознак

f_k – функція прогнозування на k-ом дереві

К – кількість дерев в лісі

З іншого боку, дерев'яні моделі мають кілька недоліків у застосуванні до даних часових рядів [10].

- По-перше, вони не призначені для моделювання часових залежностей у рамках часового ряду, тому вони можуть не врахувати важливі залежності та кореляції між минулими та майбутніми значеннями.
- По-друге, прогнози моделі лісу можуть бути менш точними, ніж передбачення спеціалізованих моделей часових рядів, особливо коли часові ряди мають складну часову структуру або сезонність.
- По-третє, інтерпретація моделі лісу може бути складною, оскільки внесок кожної функції в прогноз поширюється на всі дерева в лісі.

Дерев'яні моделі, такі як випадковий ліс і Gradient Boosted Trees, показали гарні результати в прогнозуванні часових рядів завдяки їхній здатності фіксувати нелінійні залежності та обробляти багатовимірні дані. Вони використовують методи ансамблювання для агрегування прогнозів з кількох дерев рішень, які можна навчати паралельно, підвищуючи ефективність обчислень. Однак ці моделі можуть страждати від

перенавчання та вимагати ретельного налаштування гіперпараметрів для досягнення оптимальної точності. Загалом лісові дерев'яні пропонують надійний і гнучкий підхід для завдань прогнозування часових рядів. [11]

1.2.5 Штучні нейронні мережі

Штучна нейронна мережа (Artificial Neural Network, ANN) – це паралельно розподілений процесор, який має властивість до збереження та репрезентації отриманих в ході навчання знань [12].

Модель ANN є структурною, при застосуванні ANN не обов'язково робити будь-які припущення про дані, як це було необхідно робити в статистичних методах. Штучні нейронні мережі можуть апроксимувати будь-яку функцію, можуть знаходити нелінійні залежності в даних, знаходити приховані поведінкові особливості та зв'язки [13].

Для прогнозування часових рядів зазвичай використовуються наступні види штучних нейронних мереж:

а) нейронні мережі з прямим зв'язком (Feed forward Neural Network);

б) нейронні мережі з затримкою по часу (Time delay neural network);

в) згорткові нейронні мережі (convolutional neural network);

г) рекурентні нейронні мережі (recurrent neural network).

В ході порівняння різних методів найбільш підходящим для вирішення задачі прогнозування вартості цінних паперів було обрано модифікацію рекурентних нейронних мереж – довга короткострокова пам'ять (Long-Short Term Memory, LSTM), що має властивість довгострокової пам'яті, що є необхідним при аналізі та прогнозуванні часових рядів.

1.3 Порівняння розглянутих методів

Порівняльна характеристика розглянутих вище методів прогнозування вартості цінних паперів компанії наведена в таблиці 2.1. Серед властивостей, що порівнюються: вимога до стаціонарності часових рядів, що подаються в модель, тип моделі, чи виявляє модель нелінійні залежності в даних та простота математичної та програмної реалізації.

Таблиця 2.1 – Порівняння математичних методів

	Стаціонарність ряду	Тип моделі	Нелінійні залежності в даних	Простота реалізації
VAR	Стаціонарний/ нестаціонарний	Статична	Не виявляє	+/-
Лінійна регресія	Стаціонарний	Статична	Не виявляє	+
ARIMA	Стаціонарний/ нестаціонарний	Статична	Не виявляє	+
Tree-based models	Стаціонарний/ нестаціонарний	Структурна	Виявляє	+/-
ANN (RNN LSTM)	Стаціонарний/ нестаціонарний	Структурна	Виявляє	+

Необхідно відмітити, що для отримання коректних результатів прогнозування, часові ряди потрібно приводити до стаціонарного виду при застосуванні будь-яких моделей прогнозування.

1.4 Особливості предметної області

Учасники фінансових ринків розглядають кілька теорій щодо ефективності ринків, що ціни відображають всю відому інформацію та пристосовуються до нової інформації або складаються випадковим чином. Одна з застосованих серед учасників фінансових ринків теорій – Гіпотеза ефективного ринку (Efficient Market Hypothesis EMH – гіпотеза, згідно з якою вся суттєва інформація негайно та повною мірою відбивається на ринковій курсовій вартості цінних паперів) Юджина Фами [14]. У цій теорії було описано три рівні ринкової ефективності: слабкої, середньої та сильної форми. Середня ефективність означає, що поточна ціна відображає всю загальнодоступну інформацію, і прихована інсайдерська інформація може призвести до зміни руху ціни. Виконуючи дану роботу, було припущено, що гіпотеза ефективного ринку – вірна, тому необхідно вказати декілька тверджень, що є формами ринкової ефективності [15]:

а) ринки не повністю випадкові;

б) історичні події можуть повторюватись;

в) вартість ринкового актива відображає попередню інформацію, яка стосується даного актива;

г) вартість ринкового актива відображає публічну інформацію, зокрема інформацію, представлену в пресі, звітах компаній, політичних заявах, економічних та фінансових показниках.

Прогнозування курсу акцій є складним завданням, яке пов’язане з кількома проблемами. Деякі з основних проблем прогнозування ціни акцій включають [2]:

- Волатильність і нелінійність: ціни на акції відомі своєю високою стохастичністю, що ускладнює їх точне прогнозування. Крім того,

ціни на акції можуть демонструвати нелінійні та складні залежності, що ускладнює їх моделювання за допомогою лінійних або традиційних статистичних методів.

- Якість і доступність даних: для прогнозування курсу акцій потрібен доступ до високоякісних і повних джерел даних, які не завжди можуть бути доступними або точними. На якість даних можуть впливати такі фактори, як зміщення даних, помилки вимірювання та відсутність даних.
- Обмеженість прогнозу короткими термінами та випадкові події: хоча деякі аспекти цін на акції можна передбачити з прийнятною точністю, існує багато факторів, які знаходяться поза контролем інвесторів або аналітиків, наприклад несподівані події, зміни в нормативних актах і макроекономічні умови. Ця обмежена передбачуваність може призвести до непередбачуваних і раптових змін цін на акції.
- Перенавчання та зміщення: моделі прогнозування курсу акцій можуть бути схильні до перенавчання, що призводить до поганого узагальнення нових даних. Крім того, на вибір відповідної моделі, функцій і гіперпараметрів може вплинути зміщення в даних.
- Інтерпретованість: деякі моделі прогнозування курсу акцій, такі як нейронні мережі або алгоритми машинного навчання, можуть бути дуже складними та важкими для інтерпретації, тому важко зрозуміти, як модель робить свої прогнози, і повідомити результати зацікавленим сторонам.

Загалом, ці проблеми роблять прогнозування курсу акцій складним завданням, що вимагає ретельного розгляду підходу до моделювання, якості даних та інтерпретації результатів.

1.5 Текстова фінансова інформація як джерело інсайдів

Як було зазначено в [17], широко використовуваним ринковим принципом є коригування цін за допомогою нової інформації для досягнення середньої ефективності. У фінансових текстах, таких як новини, звіти аналітиків та офіційні повідомлення компаній, міститься багато нової інформації. Однак, ручна обробка великих обсягів даних, які постійно оновлюються, та отримання корисних висновків є дуже складним завданням для будь-якої окремої організації. Автоматизований аналіз настроїв або полярності текстів, створених фінансовими суб’єктами за допомогою методів обробки природної мови (NLP), може вирішити цю проблему. Обробка природної мови - це напрямок штучного інтелекту та математичної лінгвістики, який досліджує проблеми комп'ютерного аналізу та синтезу текстів природними мовами. [16].

Використання методів машинного або глибокого навчання для обробки інформації є важливим та ефективним інструментом на фінансових ринках. Алгоритми аналізу текстів та торгівлі на основі новин використовуються для визначення, чи є новини про компанії та їхні ціни на акції, валюти чи товари позитивними чи негативними. У цьому контексті аналіз настроїв учасників фінансового ринку може бути застосований для оцінювання стабільності фінансової установи та коректності інформації в публікаціях про неї, а також для вирішення інших актуальних задач [17].

1.6 Підходи до аналізу настроїв у фінансах

Задача аналізу настроїв полягає у вилученні настроїв або думок із письмових текстів [29]. Існують два підходи до цієї задачі: 1) методи машинного навчання з ознаками, що отримуються з тексту за допомогою "word counting" [30], та 2) методи глибокого навчання, де текст представлений послідовністю ембедингів [31]. Перший метод має недоліки, такі як нездатність передати семантичну інформацію, що випливає з послідовності слів, тоді як другий вважається занадто "вимогливим до даних", оскільки він вивчає багато параметрів.

Аналіз фінансових настроїв має відмінності не лише за предметною областю, але й за метою порівняно з загальним аналізом настроїв. Зазвичай, метою аналізу фінансових настроїв є передбачення того, як ринки реагуватимуть на інформацію, яка міститься в тексті [32].

За допомогою машинного навчання та підходу "word counting", Loughran і McDonald (2016) провели огляд останніх робіт з аналізу фінансового тексту, використовуючи методи "bag-of-words" та лексиконів [33]. Вони створили словник фінансових термінів, які були присвоєні значенням "позитивний" та "невизначений", і вимірили тональність документів, підраховуючи слова з певним значенням зі словника [34]. Інший приклад - дослідження Pagolu та ін. (2016), де контрольовані алгоритми машинного навчання використовуються для виявлення настроїв щодо фінансової організації з використанням n-грам з твітів, що містять фінансову інформацію.

Один з перших досліджень, який використовував методи глибокого навчання для аналізу емоційної відтіненості тексту в фінансовій сфері, був проведений Kraus and Feuerriegel (2017) [35]. Вони використали нейронну мережу LSTM для аналізу спеціальних оголошень компаній з метою

передбачення руху фондового ринку і показали, що цей метод є більш точним, ніж традиційні методи машинного навчання. Також є кілька інших досліджень, в яких використовуються різні типи нейронних архітектур для аналізу фінансових настроїв. Lutz et al. (2018) [36] використовують підхід до використання doc2vec для створення векторних представлень речень, що стосуються конкретної компанії, та використовують багатократне навчання (multiple-instance learning) [37] для прогнозування результатів фондового ринку.

Можливість використання нейронних мереж для аналізу настроїв обмежена відсутністю великих позначених фінансових наборів даних. Навіть якщо вбудовування слів у перших шарах мережі ініціалізуються попередньо навченими значеннями, інші частини моделі все одно потрібно навчати на відносно невеликій кількості позначених даних для вивчення складних відносин. Ефективнішим рішенням може бути ініціалізація майже всієї моделі попередньо навченими значеннями та тонка настройка цих значень для вирішення задачі класифікації. [17]

1.7 Постановка задачі дослідження

Для проведення подальшого дослідження можна сформулювати наступні задачі:

- Провести дослідження існуючих методів формування прогнозів фінансових даних
- Виконати збір даних, згенерувати необхідні для прогнозування ознаки, обробити та статистично протестувати дані

- Провести оцінку емоційного забарвлення відповідних фінансових новин за допомогою мовних моделей
- Розробити та порівняти моделі прогнозування вартості фінансових активів компанії із застосуванням глибокого навчання, технік ансамблювання моделей машинного навчання, генеративних змагальних мереж GAN
- Провести оцінку роботи розроблених алгоритмів, використовуючи метрики
- Програмно реалізувати систему прогнозування з застосуванням найкращої розробленої моделі

1.8 Висновки до розділу

В розділі 1 було приведено актуальність задачі прогнозування фінансової інформації, проведено огляд існуючих методів прогнозування часових рядів з можливою стохастичністю, а саме:

- Векторна авторегресія
- Багатовимірна лінійна регресія
- Модель ARIMA
- Моделі на основі дерев прийняття рішень
- Штучні нейронні мережі

Здійснено порівняння методів.

У цьому розділі розглядалися особливості ефективного прогнозування вартості цінних паперів з використанням текстової фінансової інформації як можливого джерела впливу на фінансові деривативи конкретної компанії. Були розглянуті різні підходи до аналізу

настроїв в фінансовому секторі, а також була сформульована постановка задачі для досягнення ефективного прогнозування.

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

2.1 Сервіс StocksNeural

Онлайн-сервіс StocksNeural було розроблено з метою надання трейдерам можливості отримання прогнозів щодо руху цін на інструменти фондового ринку та рекомендацій щодо покупки або продажу без необхідності опанування технічного та фундаментального аналізу та стеження за фінансовими показниками компаній. Платформа StocksNeural є веб-орієнтованою та забезпечує доступ до прогнозів та рекомендацій у режимі реального часу. [18]

Математичне забезпечення програми StocksNeural базується на поєднанні інструментів технічного аналізу та глибокого машинного навчання. Програма використовує корельовані активи для побудови прогнозів щодо руху цін на інструменти фондового ринку.

Недоліками онлайн-сервісу StocksNeural є те, що прогноз можна отримати лише на п’ять днів вперед та для доступу до платформи необхідно оформити платну підписку. Також мінусом є те, що розробники не розкривають інформацію про вхідні дані.

2.2 Сервіс Stocksight

Stocksight - це програмне забезпечення з відкритим вихідним кодом, призначене для аналізу фондового ринку. Для збору та зберігання даних Stocksight використовує Elasticsearch, що дозволяє отримувати актуальну інформацію з Twitter та фінансових новин, які стосуються певної компанії. Однак, найбільшою перевагою програми є можливість визначення

загального фону навколо компанії, що дає можливість зрозуміти настрої ринку та прогнозувати можливі зміни у цінах акцій. [19]

Для аналізу настроїв в тексті, Stocksight використовує NLP та бібліотеки для аналізу настроїв, що дозволяє більш точно встановлювати загальний настрій навколо компанії. Програма дає безкоштовний доступ до свого вихідного коду, що дозволяє більш широкому колу користувачів працювати з нею.

Однак, необхідно зазначити, що програма має свої недоліки. Зокрема, прогнозування здійснюється на маленький проміжок часу, тому не можна гарантувати точність прогнозів на довгострокову перспективу. Також варто враховувати, що програма не використовує дані технічного та функціонального аналізу, що може значно підвищити точність прогнозів. Крім того, для роботи з програмою необхідні навички програмування, що може бути складно для деяких користувачів.

Загалом, Stocksight є корисним інструментом для аналізу фондового ринку, який дає можливість швидко та ефективно визначати загальний настрій навколо певної компанії.

2.3 Сервіс Deep Convolution Stock Technical Analysis

Deep Convolution Stock Technical Analysis [20] є програмним рішенням з відкритим вихідним кодом, яке використовує глибокі згорткові нейронні мережі та технічні індикатори для моделювання ринку акцій. Завдяки використанню згорткових нейронних мереж, програма може проаналізувати візуальне представлення часового ряду та відшукати патерни, необхідні для прогнозування напрямку або зміни тренду. Доступ до вихідного коду є безкоштовним.

Недоліком даного програмного рішення є те, що при прогнозуванні не використовуються корельовані активи та аналіз настроїв новин та інвесторів, а тільки щоденні показники торгівлі на біржі, такі як ціна відкриття, максимальна та мінімальна ціна, ціна закриття та обсяг торгів. Також, програмне рішення не використовує фундаментальний аналіз, який може допомогти врахувати економічні та фінансові фактори, що впливають на діяльність компаній, а також зрозуміти макроекономічні тенденції.

2.4 Висновки до розділу

Оглянуті програмні засоби – StocksNeural, Stocksight та Deep Convolution Stock Technical Analysis - призначені для аналізу фондового ринку та надання прогнозів щодо руху цін на інструменти фондового ринку. StocksNeural використовує корельовані активи для прогнозування цін, базуючись на поєднанні інструментів технічного аналізу та глибокого машинного навчання, Stocksight використовує NLP та аналіз настроїв у тексті для прогнозування цін, а Deep Convolution Stock Technical Analysis використовує згорткові нейронні мережі для аналізу, навчання та прогнозу на часовому ряді.

StocksNeural має обмеження, такі як доступність лише на платній підписці та змогу отримати прогнози лише на п'ять днів вперед, а також не розкриває інформацію про вхідні дані. З іншого боку, Stocksight є безкоштовним та має відкритий вихідний код, але має недоліки у своєму аналізі, такі як короткий проміжок часу для прогнозування та неможливість використання технічного та фундаментального аналізу. Програмне рішення Deep Convolution Stock Technical Analysis не

використовує фундаментальний аналіз, який може допомогти врахувати економічні та фінансові фактори, що впливають на діяльність компаній, а також зрозуміти макроекономічні тенденції.

Загалом, всі описані інструменти можуть бути корисними для трейдерів та інвесторів у фондовому ринку, але користувачам важливо розуміти, що ніякий програмний засіб не може гарантувати точність прогнозів цін на фондовому ринку та необхідно додатково аналізувати фінансові показники та інші фактори, що впливають на ціни на фондовому ринку.

3 МОДЕЛЬ СИСТЕМИ

3.1 Складові компоненти системи

Для забезпечення стабільної роботи загальної системи прогнозування цін фінансових активів та кожної з складових цієї системи, реалізована інкапсуляція кожного з сервісів , які є окремими компонентами або набором компонент, в окремий контейнер.

Компонентна система складається з наступних складових (рис. 2.1): веб-інтерфейсу користувача, сховища даних для зберігання табулярних та текстових даних, складової очистки та трансформації даних, моделі класифікації настроїв фінансових новин, моделі прогнозування та компоненти взаємодії.

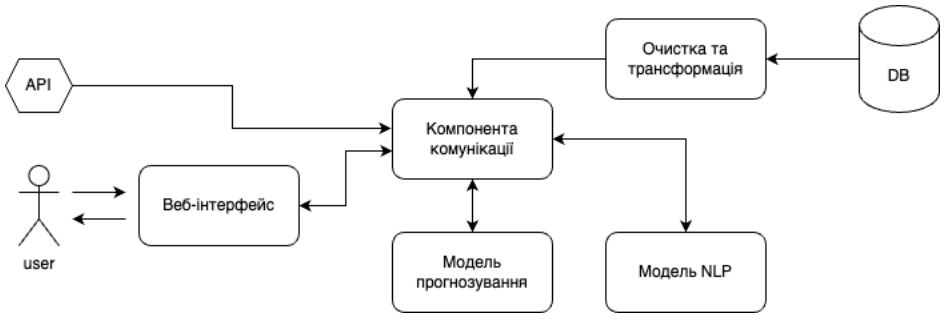


Рисунок 3.1 – Узагальнена модель системи прогнозування вартості фінансових активів

Розроблювана система буде мати два варіанти реалізації бекенду:

- Повний пайплайн – послідовне виконання кожного з етапів роботи програми для забезпечення повного циклу функціонування системи прогнозування вартості фінансових активів: отримання сирих даних з сховища даних, очистка даних, трансформація та генерація даних, аналіз настроїв фінансових

новин з мовної моделі, тренування основної моделі прогнозування, оцінювання моделі.

- Скорочений пайплайн – виконання частини етапів, призначено для використання вже навченої моделі для інференсу, застосовується, наприклад, в API.

Далі буде розглянуто структурне представлення системи у вигляді діаграм класів та діаграм компонентів кожної з реалізацій системи прогнозування вартості фінансових активів компанії, наведених вище.

3.2 Діаграма класів у нотації UML

В дисертаційній роботі розроблено систему, що складається з веб-інтерфейсу користувача, сховища даних та бекенду, які можуть комунікувати один між одним та з зовнишнім API.

Реалізовано два варіанти бекенду: повний та скорочений. Для кожного з кроків та етапів функціоналу створено окремий клас у відповідності до принципів SOLID об’єктно-орієнтованого програмування:

- Принцип єдиного обов’язку - кожен клас повинен виконувати лише один обов’язок. Це не означає, що в нього має бути тільки один метод. Це означає, що всі методи класу мають бути сфокусовані на виконання одного спільного завдання. Якщо є методи, які не відповідають меті існування класу, їх треба винести за його межі. [21]
- Принцип відкритості/закритості – класи мають бути відкриті до розширення, але закриті для змін. Якщо є клас, функціонал якого

передбачає чимало розгалужень або багато послідовних кроків, і є великий потенціал, що їх кількість буде збільшуватись, то потрібно спроєктувати клас таким чином, щоб нові розгалуження або кроки не призводили до його модифікації. [21]

- Принцип підстановки Лісков – якщо об’єкт базового класу замінити об’єктом його похідного класу, то програма має продовжувати працювати коректно. [21]
- Принцип розділення інтерфейсу – краще, коли є багато спеціалізованих інтерфейсів, ніж один загальний. Маючи один загальний інтерфейс, є ризик потрапити в ситуацію, коли похідний клас логічно не зможе успадкувати якийсь метод. [21]
- Принцип інверсії залежностей – складається з двох тверджень:
    - високорівневі модулі не повинні залежати від низькорівневих. І ті, і ті мають залежати від абстракцій;
    - абстракції не мають залежати від деталей реалізації. Деталі реалізації повинні залежати від абстракцій. [21]

3.2.1 Повний варіант реалізації

На приведеному рисунку 3.2 продемонстровано діаграму повної бекенд-частини розроблюваної системи.

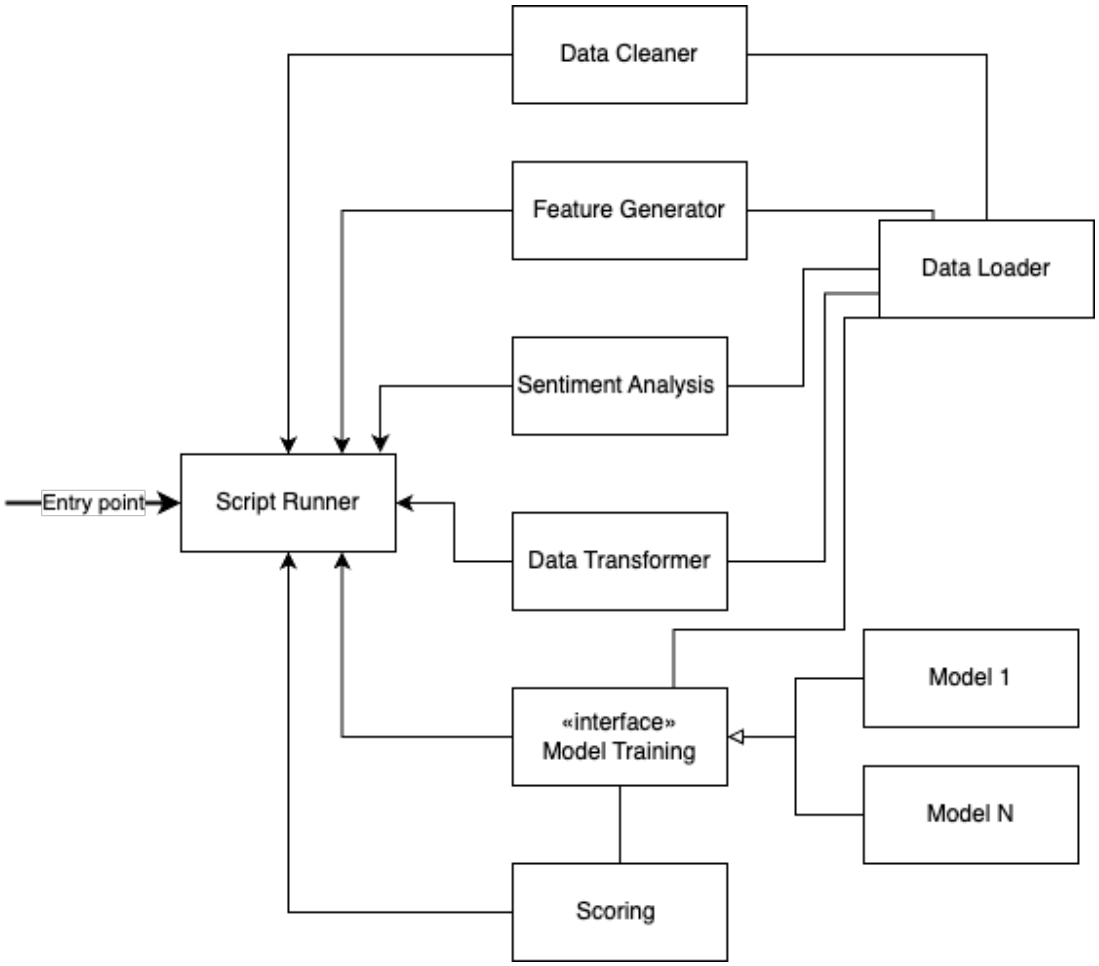


Рисунок 3.2 – Модель системи прогнозування вартості фінансового активу компанії – діаграма класів

Для реалізації цієї частини було визначено основні кроки, необхідні для отримання підготовлених даних, навчання моделі та валідації результатів. Для кожного з кроків та етапів функціоналу створено окремий клас.

До них входять:

- Клас Data Loader – представляє методи завантаження в програму даних деяких типів, необхідних для функціонування системи.
- Клас Data Cleaner – виконує очистку та певний препроцесинг вхідних сирих даних.

- Клас Feature Generator – клас відповідальний за генерацію нових інформативних та корисних для навчання моделі ознак
- Клас Sentiment Analysis – в даному класі виконується класифікація емоційного забарвлення текстів фінансових новин
- Клас Data Transformer – в класі виконується попередня трансформація даних, поєднання всіх очищених та згенерованих ознак в один датасет, розбиття датасету на тренувальну та тестову виборки з урахуванням часової складової.
- Клас Model Training – в класі відбувається тренування моделі на даних отриманих з класу Data Transformer. Виконуючи наслідування з даного класу класом-нащадком Model, можна розширити набір моделей, які застосовуються в системі
- Клас Scoring – в даному класі відбувається прогноз навченої моделі на тестових даних та оцінювання результатів роботи за допомогою метрик
- Клас Script Runner – клас відповідальний за запуск скрипта

3.2.2 Скорочений варіант реалізації

Скорочена реалізація системи необхідна для підтримки інференсу навченої моделі – забезпечення створення додаткових фічей, класифікації тексту новин, предикту навченої моделі на поданих для обробки та прогнозування даних. При скороченому варіанті реалізації, основні класи залишаються аналогічними повному варіанту, але оскілки при інференсі використовується вже натренована модель та відомі трансформації даних, то до класів скороченого варіанту системи входять:

- Клас Data Loader – представляє методи завантаження в програму даних деяких типів, необхідних для функціонування системи
- Клас Feature Generator – клас відповідальний за генерацію виявлених корисними в ході навчання моделі ознак
- Клас Sentiment Analysis – в даному класі виконується класифікація емоційного забарвлення текстів фінансових новин
- Клас Data Transformer – в класі виконується попередня трансформація даних, поєднання всіх очищених та згенерованих ознак в один датасет
- Клас Model – представляє навчену модель з атрибутами та методами для реалізації прогнозування, використовується в класі Scoring
- Клас Scoring – в даному класі відбувається прогноз навченої моделі на тестових даних та оцінювання результатів роботи за допомогою метрик
- Клас Script Runner – клас відповідальний за запуск скороченого скрипта

На рисунку 3.3 представлено діаграму скороченої бекенд-частини розроблюваної системи.

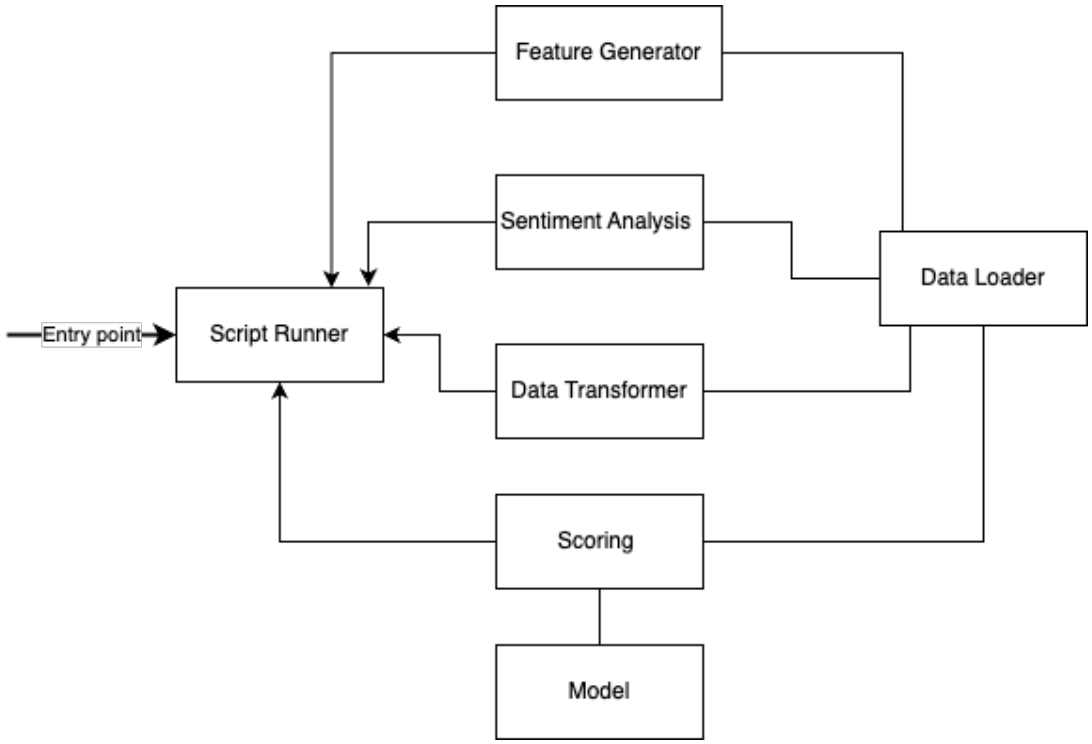


Рисунок 3.3 - Модель скорочена системи прогнозування вартості фінансового активу компанії – діаграма класів

3.3 Опис функціональних компонентів системи

Компонентна модель системи прогнозування цін фінансових активів складається з наступних частин:

- Веб-інтерфейсу користувача для взаємодії з системою
- Сховища даних для зберігання табулярних та текстових даних
- Компоненти очистки та обробки даних
- Компоненти генерації додаткових ознак
- Компоненти трансформування даних
- Моделі класифікації настроїв фінансових новин
- Моделі прогнозування

- Компоненти дослідження та аналізу даних
- Компоненти метричної оцінки прогнозування
- Компоненти візуалізації
- Комунікативних зв’язків між модулями

Опис внутрішньої структури та призначення компонентів

1. Компонент Сховища даних

Компонент являє собою модуль взаємодії зі сховищем даних, яке містить:

- необроблені сирі дані часових рядів, серед яких корельовані активи, біржові індекси, ціни на сировину, статистика пошукових запитів тощо
- текстові дані – фінансові новини про компанію, розмічені по датам

Також виконує функції збору зі сторонніх API, завантаження та зчитування даних в систему, передачу даних в наступний компонент Очистки та обробки даних.

2. Компонент Очистка та обробка даних

В даному компоненті відбувається очистка сирих даних, заповнення пропущених значень у даних. Забезпечується обробка набору даних пов’язаних з активами компанії, корельованими активами, індексами бірж, цінами на природні та господарські ресурси, статистикою по пошуковим запросам щодо розглянутої компанії. Також підлягають очищеню, препроцесингу текстові дані фінансових новин про компанію. Очищені та оброблені дані передаються в компоненти Генерації додаткових ознак, Класифікації емоційного забарвлення фінансових новин та Дослідження та аналізу даних.

3. Компонент Генерація додаткових ознак

В даному компоненті генеруються результати технічного аналізу застосованого до цільового часового ряду – вартості цінних паперів компанії. Обчислюються такі індикатори технічного аналізу [22] як:

- Ковзаюче середнє
- Стандартне відхилення
- Полоси Боліндже́ра
- Індикатор Ішимоку
- Кумулятивний індекс коливань
- Індекс маси
- Сходження-розходження ковзаючих середніх
- Експонентне згладжування
- Тройна експоненційно згладжена ковзаюча середня
- Тощо

Також формуються ознаки пов’язані з часовою складовою в різних варіаціях для досягнення найкращого результату в прогнозуванні – декомпозиція часових рядів, створення індексів днів, сезонні змінні, вихідні та свята. Дані передаються в компоненту Трансформер

4. Компонент Класифікація емоційного забарвлення

В даному компоненті виконується класифікація фінансових новин про компанію на позитивні, нейтральні та негативні. Очищені та оброблені дані подаються на модель finBERT, переднавчену на великому корпусі фінансових текстів. Дані передаються в компоненти Трансформер та Дослідження та аналіз даних.

5. Компонент Дослідження та аналіз даних

Отримує очищені та оброблені дані, також результати класифікації емоційного забарвлення фінансових текстів. В даному

компоненті виконується пошук кореляцій та інших можливих взаємозв’язків в даних для більш повного дослідження. Виконується дослідження зв’язку настрою щоденної фінансової новини та рухом вартості на цінні попери компанії. До даних застосовуються статистичні тести на стаціонарність, нормальний розподіл тощо.

6. Компонент Трансформер даних

В даному компоненті відбувається поєднання всіх наявних та згенерованих даних, отриманих з попередніх компонент, в один датасет. Проводиться кодування категоріальних ознак, нормалізація, масштабування числових даних та стаціонаризація даних у вигляді часових рядів. Отриманий датасет трансформується діленням на послідовності з урахуванням часу для навчання за умови використання послідовної моделі. Відбувається розбиття на тренувальну, валідаційну та тестову вибірки, або на фолди для крос валідації. Важливо при даних операціях враховувати залежність даних від часу, щоб уникнути «заглядання» в дані з майбутнього. Можуть бути застосовані алгоритми зниження розмірності даних, такі як UMAP чи t-SNE. Оброблені дані подаються на вхід тренування або прогнозування навченої моделі.

7. Компонент Навчання моделей

Даний компонент представляє собою базовий клас для навчання моделей машиного та глибокого навчання. Моделі можуть бути додані шляхом розширення інтерфейсів базового класу та наслідування базового класу класами-нащадками, що забезпечують функціонал нових моделей. В цьому класі реалізовано підбір гіперпараметрів за допомогою перехрестного навчання з урахуванням фактору часу та оптимізації функуії. Компонент повертає навчену модель та передає її в компонент Оцінки прогнозування.

8. Компонент Оцінка прогнозування

В компоненті проводиться прогнозування навченої в попередньому кроці моделі на тестових даних та оцінка цього прогнозу з використанням приведених метрик:

- R-Squared
- Mean Absolute Error
- Mean Squared Error
- Mean Absolute Percentage Error
- Symmetric Mean Absolute Percentage Error
- Тощо

Отримані результати прогнозування та оцінки результатів прогнозування передаються в модулі Візуалізації та Веб-інтерфейсу користувача.

9. Компонент Візуалізація

В даному компоненті візуалізуються результати прогнозування та оцінки результатів прогнозування моделі, а також результати дослідження та аналізу даних з попередніх компонент. Відбувається побудова різних типів графіків, необхідних для біль повного пояснення та аналізу роботи системи: гістограми, корелограми, графіки розсіяння, стовбчикові діаграми, графіки часових рядів. Результати роботи даного модуля можуть зберігатись в окремі файли та передаватись в компонент Веб-інтерфейс користувача.

10. Компонент Веб-інтерфейс користувача

Даний компонент забезпечує реалізацію функціоналу взаємодії користувача з представленою системою, що включає в себе можливість вивантажити результати прогнозу, виведення графіку цін акції компанії і можливість застосування індикаторів технічного аналізу, представлення результатів прогнозування цін в

візуалізованій формі. Також користувач може отримати діаграму зв’язку між фінансовими новинами та тенденції змін цін акцій. Також в даному компоненті реалізовано автоматична генерація звіту про процес навчання моделі для розробника.

3.4 Діаграма компонентів у нотації UML

Для динамічного представлення моделі системи прогнозування представлено діаграму компонентів, описаних в попередньому розділі. На рисунку 3.4 продемонстровано зв’язок компонентів та описана їх суть.

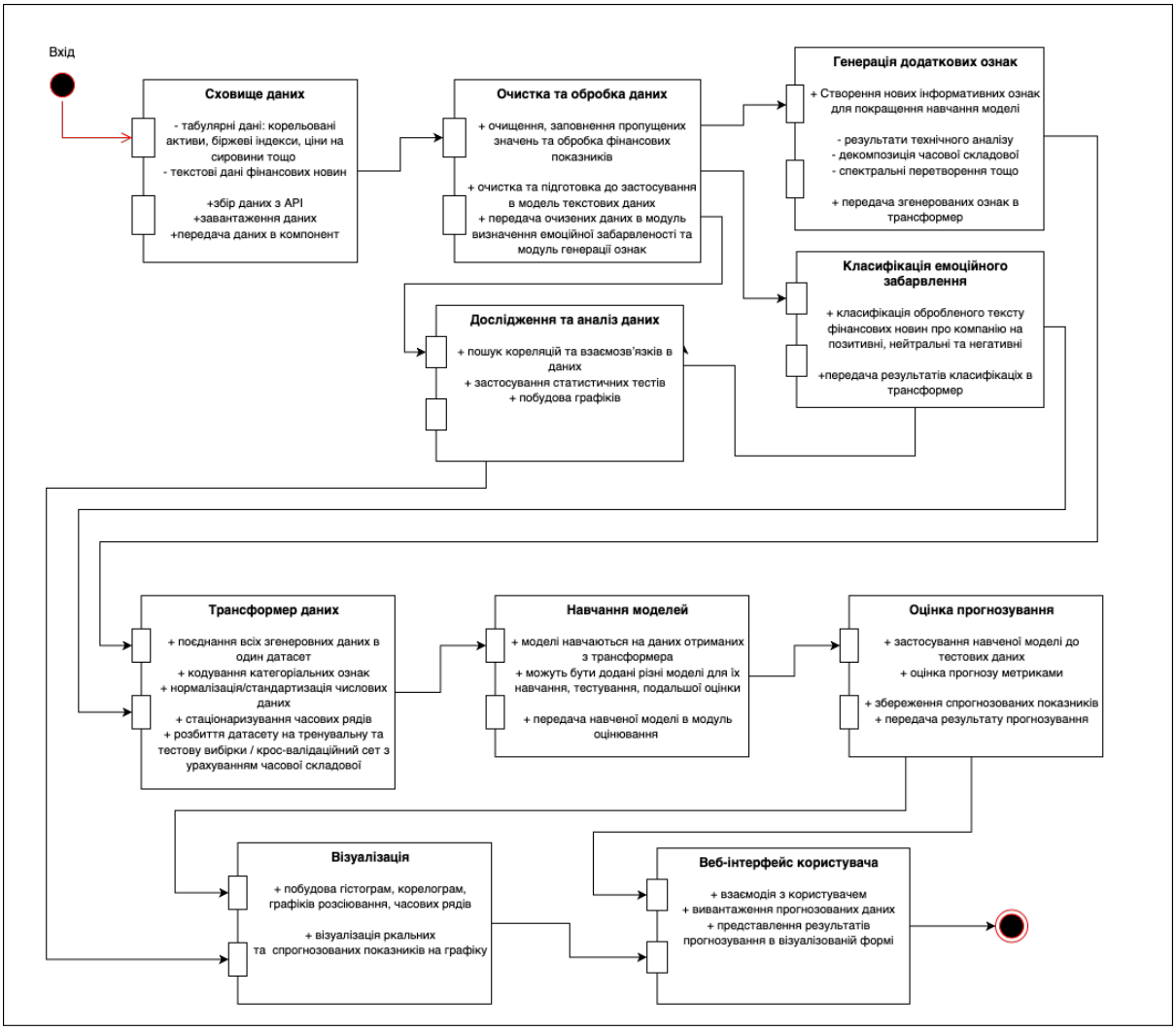


Рисунок 3.4 - Діаграма компонентів системи у нотації UML

3.5 Діаграма послідовності функціонування системи

Діаграма послідовності - це схема, що показує послідовність виконання операцій в системі. В контексті системи прогнозування цін акцій, діаграма послідовності може використовуватися для показу послідовності дій, що здійснюються системою під час виконання прогнозування вартості цінних паперів. Демонструє, які класи та об'єкти взаємодіють між собою та які методи викликаються під час процесу

прогнозування цін. Вона може також відображати послідовність отримання даних та їх обробки, наприклад, які дані використовуються для прогнозування цін та як вони перетворюються на вихідну інформацію. Діаграма послідовності представлена на рисунку 3.5.

Рисунок 3.5 - Діаграма послідовності функціонування системи прогнозування цін акцій

3.6 Висновки до розділу

В даному розділі магістерської дисертації було представлено та описано складові компоненти розроблюваної системи прогнозування вартості фінансових активів.

Розроблювана система має два варіанти реалізації бекенду:

- Повний пайплайн – послідовне виконання кожного з етапів роботи програми для забезпечення повного циклу функціонування системи прогнозування вартості фінансових активів: отримання сирих даних з сховища даних, очистка даних, трансформація та генерація даних, аналіз настроїв фінансових новин з мовної моделі, тренування основної моделі прогнозування, оцінювання моделі.
- Скорочений пайплайн – виконання частини етапів, призначено для використання вже навченої моделі для інференсу, застосовується, наприклад, в API.

Представлені діаграми класів для повного та скороченого варіантів реалізації, процес розробки структури класів проводився відповідно до принципів SOLID. Для кожного з етапів функціоналу розроблено окремий клас:

- Клас Data Loader
- Клас Data Cleaner
- Клас Feature
- Клас Sentiment Analysis
- Клас Data Transformer

- Клас Model
- Клас Scoring
- Клас Script Runner

Компонентна модель системи прогнозування цін фінансових активів складається з наступних частин:

- Компоненти Веб-інтерфейсу користувача для взаємодії з системою
- Компоненти взаємодії зі сховищем даних для зберігання табулярних та текстових даних
- Компоненти очистки та обробки даних
- Компоненти генерації додаткових ознак
- Компоненти трансформування даних
- Компоненти моделі класифікації настроїв фінансових новин
- Компоненти моделі прогнозування фінансового показника
- Компоненти дослідження та аналізу даних
- Компоненти метричної оцінки прогнозування
- Компоненти візуалізації

Проведено опис функціональних компонентів, побудована діаграма компонентів в нотації UML. Представлена діаграма послідовності функціонування системи.

4 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Змістовна постановка задачі

Прогнози фондового ринку допомагають інвесторам отримати вигоду на фінансових ринках. Різні статті пропонують різні методи прогнозування фондового ринку, але жодна модель не може дати точних прогнозів. У цьому дослідженні необхідно передбачити ціни на акції за допомогою моделі прогнозування, заснованої на методах ансамблювання. Попередньо проведена оцінка релевантності даних, які можуть бути корисні при навчанні моделі та можуть допомогти підвищити якість моделі що навчається. Зібраний набір даних обробляється, створюються та виділяються з даних корисні ознаки. За допомогою переднавченої моделі проводится класифікація тексту фінансових новин на позитивні, нейтральні та негативні класи. Дані конкатенуються в один датасет. Процес вибору моделі відбувається послідовним навчанням та оцінкою результатів прогнозування кожної з моделей чи технік ансамблювання. Найкраща навчена модель застосовується для функціонування в фінальній конфігурації системи для здійснення прогнозів. У цьому дослідженні ми використовуємо такі ознаки, як корельовані активи, історичні дані, дані схожих компаній, фінансові новини, дані фінансових бірж тощо для навчання нашої моделі. Результати експериментів повинні мати хорошу точність і низький рівень помилок, тому вони мають бути перспективним рішенням для роботи з динамічними цінами на акції.

4.2 Математична складова аналізу часових рядів

Аналіз часових рядів – це статистичний метод, який використовується для аналізу та моделювання даних часових рядів, тобто спостережень, що здійснюються через регулярні проміжки часу протягом певного часу. Математичні основи аналізу часових рядів включають кілька ключових понять, включаючи автокореляцію, стаціонарність і спектральний аналіз.

Автокореляція — це міра кореляції між спостереженнями в різні моменти часу. Якщо часовий ряд автокорельований, це означає, що значення ряду в один момент часу пов’язане зі значенням в інший момент часу. Автокореляцію можна визначити кількісно за допомогою функції автокореляції (ACF), яка вимірює кореляцію між часовим рядом і його версією з відставанням. [23]

Стаціонарність є ключовим поняттям в аналізі часових рядів, яке стосується статистичних властивостей часового ряду, що залишаються незмінними протягом часу. Стаціонарний часовий ряд має постійне середнє значення та дисперсію, а автокореляція між спостереженнями в різні моменти часу є постійною. [24] Стаціонарність важлива, оскільки багато моделей часових рядів припускають, що ряд є стаціонарним, щоб робити прогнози.

Спектральний аналіз — це метод, який використовується для аналізу частотних компонентів часового ряду. Функція спектральної щільності описує розподіл потужності на різних частотах у часовому ряду. [25] Спектральний аналіз може допомогти визначити циклічні закономірності в даних, наприклад сезонні коливання.

Інші ключові математичні поняття в аналізі часових рядів включають регресію часових рядів, яка передбачає використання моделі

лінійної регресії для прогнозування майбутніх значень часового ряду на основі минулих значень та інших коваріантів, а також прогнозування часових рядів, яке передбачає використання статистичних моделей для прогнозування майбутні значення часового ряду.

Підводячи підсумок, можна сказати, що аналіз часових рядів – це складна сфера, яка спирається на низку математичних концепцій і методів для аналізу та моделювання даних часових рядів. Автокореляція, стаціонарність і спектральний аналіз – лише деякі з ключових понять, які лежать в основі математичних основ аналізу часових рядів.

4.3 Ансамблювання моделей

Ансамблювання моделей — це техніка, яка використовується в машинному навчанні та статистичному моделюванні для покращення продуктивності прогнозних моделей. [26] Основна ідея ансамблювання полягає в тому, щоб поєднати кілька моделей, щоб отримати кращі прогнози, ніж будь-яка окрема модель. Об’єднання можна виконати кількома способами, наприклад, пакетуванням, посиленням і укладанням.

Об’єднання моделей може бути потужним інструментом для покращення продуктивності прогнозних моделей, особливо коли окремі моделі мають додаткові сильні та слабкі сторони. Однак об’єднання може бути дорогим з точки зору обчислень і вимагати більше ресурсів, ніж навчання окремої моделі. Тому важливо ретельно розглянути та вибрати найбільш відповідну техніку поєднання для конкретної проблеми та набору даних.

4.3.1 Bagging

Bagging (Bootstrap Aggregating) — це метод ансамблювання, який поєднує кілька базових учнів для покращення загальної ефективності прогнозування моделі. [26] Математика, що лежить в основі пакетування, передбачає використання початкової вибірки, усереднення по ансамблю та зменшення дисперсії.

Початкова вибірка — це статистичний метод, який передбачає випадковий вибір зразків із вихідних даних із заміною. Це означає, що кожна вибірка з однаковою ймовірністю буде відібрана кілька разів, у результаті чого утвориться новий набір даних, який має такий самий розмір, як і вихідний, але деякі екземпляри повторюються, а інші відсутні. Створюючи кілька початкових зразків, кожна модель навчається на дещо іншій версії навчальних даних. Це вносить різноманітність серед моделей, що допомагає зменшити надмір і підвищити загальну точність ансамблю.

Усереднення по ансамблю включає в себе середнє значення прогнозів, зроблених кожною окремою моделлю, для отримання остаточного прогнозу. З математичної точки зору, якщо у нас є набір базових учнів, h1, h2, ..., hn, і ми хочемо зробити прогноз для даного екземпляра x, ансамблевий прогноз y можна обчислити як (формула 4.1):

$$y = \frac{1}{n}\sum_{i=1}^{n} h_i(x) \ \#(4.1)$$

де h_i – передбачення і-го базового учня в екземплярі x.

Процес усереднення допомагає згладити будь-який шум або неузгодженість в окремих моделях, що призводить до більш надійного та точного прогнозу.

Зменшення дисперсії є однією з ключових переваг пакетування. Дисперсію прогнозу ансамблю можна розрахувати як (формула 4.2):

$$Var(y) = \frac{1}{n^2}\sum_{i=1}^{n} Var\big(h_i(x)\big) \#(4.2)$$

де $Var(h_i(x))$ представляє дисперсію і-го базового учня на екземплярі x.

Зі збільшенням кількості базових учнів в ансамблі дисперсія прогнозу ансамблю зменшується, що допомагає зменшити переобладнання та підвищити точність ансамблю.

Таким чином, математика, що лежить в основі беггінгу, передбачає використання початкової вибірки, усереднення по ансамблю та зменшення дисперсії. Створюючи кілька початкових зразків і поєднуючи прогнози кількох базових учнів, пакетування може підвищити точність і надійність моделей машинного навчання.

4.3.2 Boosting

Бустинг є одним з підходів до побудови ансамблю моделей у машинному навчанні, що дозволяє підвищити якість передбачень. Бустинг заснований на ідеї послідовного навчання слабких моделей, які під час навчання коригуються залежно від попередніх помилок. У процесі

бустингу після кожного навчання створюється нова модель, яка більш уважно підходить до об'єктів, на яких попередня модель помилялася.

В основі бустингу лежить використання ваг для об'єктів, що дозволяє зосередити увагу моделі на найбільш складних об'єктах. Ваги розподіляються на об'єкти в залежності від того, наскільки точно модель передбачила їх значення. При цьому вага об'єкта збільшується, якщо модель помиляється на ньому, і зменшується, якщо модель правильно передбачає значення.

Одним з найпопулярніших методів бустингу є градієнтний бустинг, який заснований на використанні градієнта функції втрат. Під час навчання кожна нова модель намагається скоригувати помилки попередніх моделей, використовуючи градієнт функції втрат відносно навчальної вибірки. Для цього використовуються ансамблі дерев рішень, де кожне дерево працює з вагами, що залежать від градієнта функції втрат.

Бустинг є ефективним підходом до побудови ансамблю моделей, що дозволяє досягти високої точності передбачень. Однак, він може бути вимогливим до обчислювальних ресурсів і часу навчання, оскільки процес предикту є ресурсозатратним.

Під час підвищення базові учні навчаються ітеративно, при цьому кожна наступна модель намагається виправити помилки, зроблені попередніми моделями. Ваги навчальних прикладів коригуються на кожній ітерації, щоб приділити більше уваги прикладам, які були неправильно класифіковані попередніми моделями. Це означає, що наступні моделі навчені більше зосереджуватися на складних семплах, які були неправильно класифіковані попередніми моделями.

З математичної точки зору, якщо є набір слабких базових учнів, h1, h2, ..., hn, і ми хочемо зробити прогноз для даного екземпляра x, ансамблевий прогноз у можна обчислити як (формула 4.3):

$$y = \text{sign}\left(\sum_{i=1}^{n} \alpha_i h_i(x)\right) \#(4.3)$$

де sign є функцією знака, яка відображає позитивні прогнози на +1, а негативні прогнози на -1,

α_i представляє вагу, призначену і-му базовому учню.

Вагові коефіцієнти оновлюються на кожній ітерації за допомогою алгоритму посилення, такого як AdaBoost, Gradient Boosting або XGBoost.

Вага, призначена і-му базовому учню, пропорційна його точності на попередній ітерації, причому вищі ваги надаються більш точним моделям. Вага розраховується за таким рівнянням (формула 4.4):

$$\alpha_i = \frac{1}{2}\log\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right) \#(4.4)$$

де ε_i – частота помилок і-го базового учня на навчальному наборі.

Логарифмічна функція гарантує, що вага пропорційна різниці між точністю та частотою помилок, причому більш точні моделі отримують більш високі ваги.

Алгоритм бустингу навчається ітеративно, причому кожна наступна модель навчається на зваженій версії навчального набору, який підкреслює приклади, неправильно класифіковані попередніми моделями. Цей ітераційний процес триває, доки не буде виконано критерій зупинки, наприклад досягнення максимальної кількості ітерацій або досягнення певного рівня точності.

Математика, що лежить в основі бустингу, передбачає використання зважених навчальних прикладів, навчання ітераційної моделі та виправлення помилок. Шляхом коригування ваги навчальних прикладів і ітеративного навчання базових учнів можна підвищити точність і

надійність моделей машинного навчання, особливо у випадку слабких моделей навчання.

4.3.3 Stacking

Стекінг (stacking) є ансамблем моделей машинного навчання, що використовується для комбінування результатів багатьох моделей, щоб отримати кращі передбачення. У стекінгу використовуються дві або більше різних моделей, називаємо їх базовими моделями, та одна модель, називаємо її мета-моделлю, яка використовує передбачення базових моделей як вхідні дані. [28]

Математична складова стекінгу включає в себе обчислення передбачень базових моделей та використання їх як вхідних факторів для мета-моделі. Якщо передбачення проводяться для класифікації, то мета-модель може бути побудована за допомогою алгоритму бінарної класифікації, такого як логістична регресія або SVM. Якщо ж передбачення проводяться для регресії, то мета-модель може бути побудована за допомогою лінійної регресії або SVM.

Для використання стекінгу, базові моделі можуть бути різних типів, таких як дерева рішень, нейронні мережі, метод опорних векторів тощо. Відповідно, передбачення можуть бути зроблені на основі різних алгоритмів, таких як різні ансамблі, глибокі нейронні мережі або методи регресії.

У стекінгу також існують різні підходи до обчислення ваг базових моделей. Один з найпоширеніших підходів - це використання мета-моделі для навчання ваг базових моделей на основі їхніх передбачень на

навчальних даних. Інший підхід полягає в тому, щоб використовувати фіксовані ваги для кожної базової моделі, які були

У стекуванні базові учні навчаються на одному навчальному наборі, а їхні прогнози використовуються як вхідні дані для метаучня. Метаучень — це модель вищого рівня, яка навчена комбінувати прогнози базових учнів для формування остаточного сукупного прогнозу. Метаучень можна навчити за допомогою різноманітних алгоритмів машинного навчання, таких як логістична регресія, лінійна регресія або нейронні мережі.

З математичної точки зору, є набір базових учнів, h1, h2, ..., hn, і мета-модель g, і необхідно зробити прогноз для даного семпла x, прогнозування ансамблю у можна обчислити як (формула 4.5):

$$y = g\big(h_1(x), h_2(x), \dots, h_n(x)\big) \#(4.5)$$

де $h_i(x)$ – передбачення і-го базового учня в екземплярі x.

Метаучень приймає прогнози базових учнів як вхідні дані та вчиться комбінувати їх, щоб сформувати остаточне прогнозування ансамблю.

Метаучень навчається за допомогою окремого набору перевірки, який не використовується для навчання базових учнів. Прогнози базових учнів у наборі перевірки використовуються як вхідні дані для мета-навчання, а вихідні дані мета-учня порівнюються з істинними мітками набору перевірки. Метаучень навчений мінімізувати різницю між його прогнозами та справжніми мітками, використовуючи функцію втрат, таку як середня квадратична помилка або втрата крос-ентропії.

Після того як метанавчається навчено, його можна використовувати, щоб робити прогнози на нових екземплярах, приймаючи прогнози базових учнів як вхідні дані та об’єднуючи їх для формування остаточного прогнозу ансамблю. Цей процес допомагає підвищити точність і

надійність моделей машинного навчання, особливо коли використовується кілька базових учнів різних типів.

Стекінг передбачає використання метанавчання для поєднання прогнозів кількох базових учнів різних типів. Навчаючи базових учнів на одному навчальному наборі та використовуючи окремий набір перевірки для навчання мета-учня, стекування може підвищити точність і надійність моделей машинного навчання, охоплюючи ширший діапазон функцій.

4.4 Класифікація настроїв фінансових текстів

4.4.1 Класифікація тексту з використанням переднавчених моделей

В даній магістерській дисертації розглядається аналіз настроїв, який класифікує текст як позитивний, негативний або нейтральний у певній області. Ця проблематика була досліджена в публікації [17]. Завданням моделювання мови є передбачення наступного слова в певному текстовому фрагменті. В обробці природної мови останнім часом одним з найбільш важливих досягнень є усвідомлення того, що моделі, навчені для моделювання мови, можуть бути успішно застосовані для більшості задач NLP з деякими модифікаціями. Такі моделі навчаються на дуже великих корпусах тексту, а потім налаштовуються на конкретні задачі шляхом додавання відповідних шарів, що налаштовуються на цільових даних [15]. Один з очевидних варіантів використання цього підходу - класифікація тексту.

ELMo (Embeddings from Language Models) є типом глибокого контекстуалізованого представлення слів, яке моделює як складні характеристики вживання слів, такі як синтаксис і семантику, так і те, як ці вживання змінюються в різних мовних контекстах. Цей підхід був одним з

перших успішних застосувань глибокого контекстуалізованого представлення слів. Для навчання ELMo використовуються глибокі двонаправлені мовні моделі, які попередньо тренуються на великих корпусах тексту. Для обчислення контекстуалізованих ембедінгів кожного слова використовуються приховані стани цих моделей. Ваги ELMo можуть бути використані для ініціалізації ембедінгів для наступних завдань, що дозволяє розрахувати контекстуалізовані ембедінги для будь-якого фрагменту тексту. Попередньо навчені ваги ELMo дозволяють покращити продуктивність більшості завдань порівняно зі статичними ембедінгами слів, такими як word2vec або GloVe.

ELMo використовує попередньо навчені мовні моделі для контекстуалізації уявлень, проте інформація, що була отримана з мовної моделі, присутня лише на першому шарі будь-якої моделі, яка використовує цю інформацію. ULMFit було першим рішенням, яке забезпечило справжнє трансферне навчання для NLP, використовуючи нові методи, такі як дискримінаційне тонке налаштування та похилі трикутні темпи навчання. Це рішення дало змогу ефективно налаштувати попередньо навчену мовну модель для класифікації тексту. Крім того, було введено подальше попереднє навчання мовної моделі на предметному корпусі, припускаючи, що дані окремого завдання з іншого розподілу, ніж загальний корпус, на якому навчалася початкова модель.

Основна ідея ULMFit полягає в ефективному тонкому налаштуванні попередньо навченої мовної моделі для вирішення завдань. Це досягнуто шляхом використання двоспрямованих кодувальних представлень з Transformers (BERT), що підвищило рівень трансферу моделі на нові завдання. [17]

4.4.2 BERT для вирішення фінансових задач: FinBERT

BERT (Bidirectional Encoder Representations from Transformers) - це техніка машинного навчання, яка використовує двоспрямовані кодувальні представлення з Transformers для переднавчання глибоких двонаправлених представлень на нерозмічених текстових даних. Ця техніка розроблена працівниками компанії Google [18] і дозволяє досягати гарних результатів для різних задач, таких як прогнозування наступного речення, додавання пропущених слів в правильному контексті, відповіді на запитання та аналіз настроїв в тексті.

Реалізація моделі BERT включає два етапи: попереднє навчання та налаштування. Під час попереднього навчання модель тренується на нерозмічених даних у складі різних задач, а під час налаштування BERT ініціалізується з параметрами, отриманими в ході попереднього навчання, і потім параметри налаштовуються на розмічених даних. Модель BERT є уніфікованою для розв'язання різних задач, і між наперед навченою архітектурою і фінальною майже немає відмінностей. Для переднавчання та налаштування моделі використовується одна й та сама архітектура, за винятком вихідних шарів, оскільки параметри оптимізуються під час налаштування. Даний процес представлено на рисунку 4.1. У вхідному масиві додавані спеціальні позначення [CLS] на початку та [SEP] як позначення розділення між реченнями чи питаннями та відповідями.

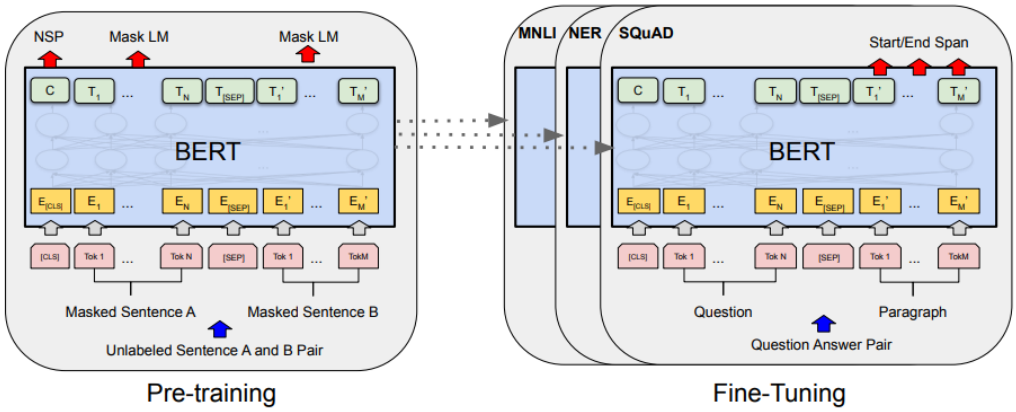


Рисунок 4.1 – Попереднє навчання та налаштування моделі BERT [18]

Архітектура моделі BERT є багатошаровий двоспрямований енкодер-трансформер (multi-layer bidirectional Transformer encoder), що, в свою чергу, є архітектурою для моделювання послідовної інформації (альтернатива RNN). Початкова реалізація архітектури трансформера описана в [19].

У моделі BERT використовується алгоритм токенізації WordPiece [20], який починається з ініціалізації словника, щоб включити кожен символ, що зустрічається в навчальних даних. Поступово WordPiece вивчає задану кількість правил злиття, вибираючи не найбільш часті пари символів, а ту, яка максимізує ймовірність додавання навчальних даних в словник.

Для кожного токена створюється вхідне представлення, що складається з суми відповідних вкладень токена, сегмента та позиції. Візуалізацію цього процесу можна побачити на рисунку 4.2.

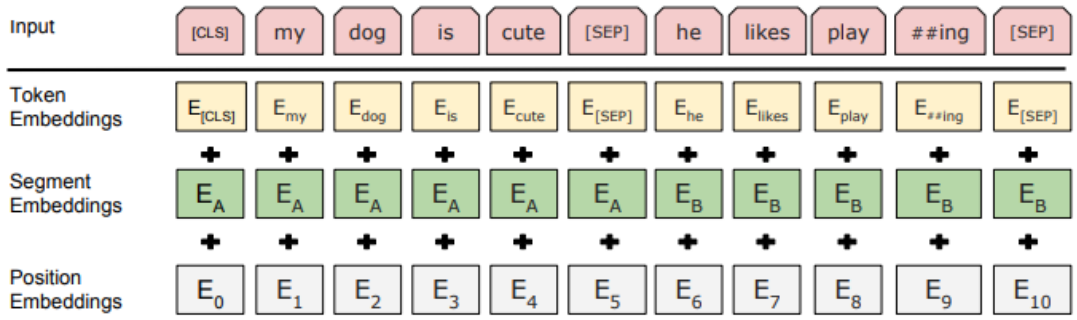


Рисунок 4.2 – Вхідне представлення BERT [18]

Для попереднього навчання моделі BERT з урахуванням обмежень, які створює двосторонній аналіз контекстів на всіх шарах, використовують масковану язикову модель (Masked Language Model MLM) [21]. MLM полягає в тому, що випадковим чином маскується 15% всіх токенів WordPiece, замість прогнозування наступного слова на основі попереднього, як це робиться в язиковій моделі на основі LSTM або інших стандартних односпрямованих моделях, і після цього прогнозується їх значення. Такий підхід дозволяє створювати двосторонню наперед навчену модель.

BERT існує в двох версіях:

- базова BERT-base, яка складається з 12-ти шарів енкодеру, 768-ми прихованих шарів, 12-ти голівок multi-head attention та 110-ти мільйонів параметрів;
- розширена BERT-large, яка складається з 24-х шарів енкодеру, 1024-х прихованих шарів, 16-ти голівок multi-head attention та 340-ка мільйонів параметрів.

Модель finBERT була використана для аналізу настрою тексту в області фінансів. finBERT є попередньо навченою моделлю глибокого навчання заснованою на BERT, яка використовується для аналізу настрою

тексту. Аналіз настрою в тексті здійснюється шляхом застосування шару Dense (повнозв'язний шар нейронів з функцією активації GELU [23]) до останнього прихованого стану токена [CLS]. Цей процес проілюстровано на рисунку 4.3. Далі модель навчається на розміченому датасеті.

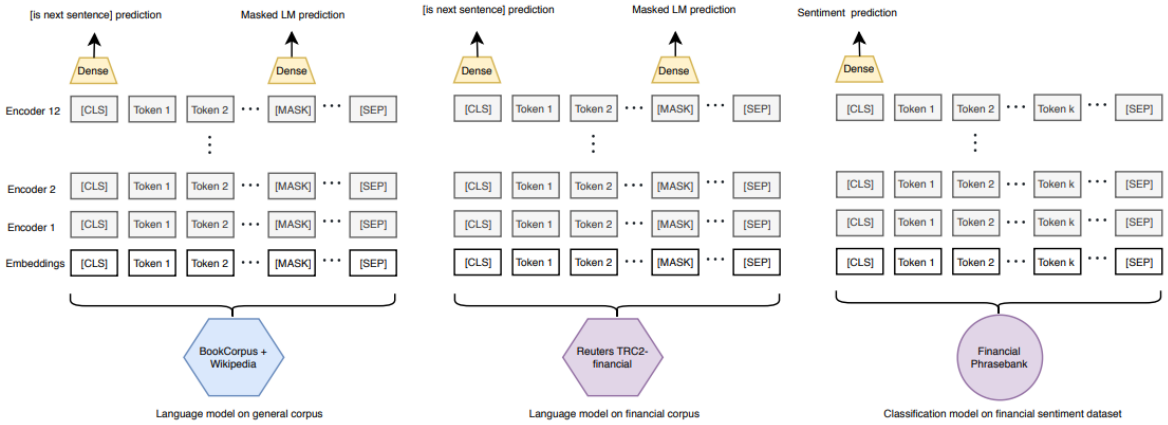


Рисунок 4.3 – Перший і другий етап переднавчання та налаштування моделі finBERT [22]

В ході переднавчання моделі використовувалася значна кількість фінансових текстів з Financial PhraseBank, який складається з багатьох речень на англійській мові. Ці тексти були випадково відібрані з бази даних LexisNexis [24], що містить фінансові новини, та були розмічені досвідченими експертами у фінансовій та бізнес-сферах. [17]

4.5 Генеративні змагальні мережі

GAN (Generative Adversarial Network) - це модель штучної нейронної мережі, яка використовується для генерації нових зображень, тексту або інших даних, що є схожими на реальні дані. [38] Вона складається з двох глибоких нейронних мереж: генератора та дискримінатора. Генератор

створює нові зразки, які намагаються імітувати реальні дані, тоді як дискримінатор намагається відрізнити ці створені зразки від реальних. Ці дві мережі навчаються взаємодіяти одна з одною в процесі тренування, покращуючи свої навички, що дозволяє генерувати більш реалістичні зображення та дані.

Етапи навчання GAN [38]:

- Генератор G, використовуючи випадкові дані (шум позначається z), намагається «генерувати» дані, які неможливо відрізнити від реальних даних або надзвичайно близькі до них. Його мета — генерувати розподіл, вкрай схожий на реальні дані.
- Реальні або згенеровані дані в випадковому порядку вписуються в Дискримінатор D, який працює як класифікатор, його мета встановити, чи надходять дані з Генератора, чи дані є реальними. D оцінює розподіл вхідної вибірки та порівнює з даними з реального набору.
- Помилки loss від G і D об'єднуються і поширюються методом зворотнього розповсюдження через генератор. Отже, втрати кожної моделі залежать як від генератора, так і від дискримінатора. Це крок, який допомагає генератору дізнатися про реальний розподіл даних. Якщо генератор не справляється добре зі створенням реалістичних даних (з однаковим розподілом), Дискримінатору буде легко відрізнити згенеровані від реальних наборів даних. Отже, losses Дискримінатора будуть дуже малими. Невеликі втрати дискримінатора призведуть до більших втрат генератора (див. рівняння нижче для L(D,G)). Це робить створення дискримінатора дещо складним, оскільки занадто хороший дискримінатор завжди призведе до величезних втрат генератора, через що генератор не зможе навчатися.

- Процес триває до тих пір, поки Дискримінатор більше не зможе відрізнити згенеровані дані від реальних.

При поєднанні Генератор та Дискримінатор починають працювати за принципом minimax, тобто Генератор намагається змусити Дискримінатор повертати хибний результат, тому збільшує ймовірність для хибних прикладів, або мінімізує $E_{z\sim p_z(z)}[\log(1-D(G(z)))]$. В свою чергу, мета Дискримінатора відрізнити дані, які надходять з Генератора D(G(z)), або виконати задачу максимізування $E_{x\sim p_r(x)}[\log D(x)]$. [39] Комбінована функція втрат має вигляд (4.6):

$$L(G,D) = E_{x\sim p_r(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-D(G(z)))] \quad (4.6)$$

Де G – генератор

D - дискримінатор

Генеративні змагальні мережі (GANs) є потужним інструментом в галузі глибинного навчання, який забезпечує здатність генерувати нові дані, що відповідають реальним даним. Основні переваги GAN-ів полягають у їх здатності збільшувати кількість наявних даних для тренування, створюючи нові приклади даних, які можуть бути використані для покращення точності моделей.

Крім того, GAN-и можуть генерувати нові дані з нульового рівня, що робить їх особливо корисними для задач генерації зображень, музики і т.д. У випадку задачі даної магістерської – генерації сигналів – фінансових показників.

Однак, наявні також і певні недоліки GAN-ів. Наприклад, вони можуть бути дуже складними для тренування і вимагати великої кількості даних та обчислювальних ресурсів для досягнення високої якості

генерованих даних. Крім того, якщо GAN-и будуть погано налаштовані, вони можуть генерувати шумові або неправдоподібні дані. [39]

4.5.1 Модифікація Metropolis-Hastings GAN

Metropolis-Hastings GAN (MH-GAN) є варіантом генеративної змагальної мережі (GAN), який може бути використаний для генерації та прогнозування часових рядів. Зазвичай GAN використовуються для генерації зображень, але MH-GAN використовується для прогнозування значень часових рядів на основі вхідних даних.

MH-GAN використовує метод Metropolis-Hastings для генерації нових значень часового ряду на основі попередніх значень, що були згенеровані GAN. Після генерації нових значень, вони перевіряються на відповідність реальному часовому ряду. Якщо нові значення краще відповідають реальному часовому ряду, вони приймаються як наступна ітерація генерації, інакше залишається попереднє значення. [41]

Перевагою MH-GAN є те, що він може генерувати часові ряди з високою точністю та стійкістю до шуму, в той час як недоліком є високі обчислювальні вимоги та складність архітектури при конфігуруванні моделі. Також потребує досить великої кількості даних для ефективної роботи.

4.5.2 Модифікація Wasserstein GAN

Навчання GAN є досить складним. Моделі можуть ніколи не сходитися, і може легко статися колапс роботи. Вихід – використовувати модифікацію GAN під назвою Wasserstein GAN — WGAN.

Wasserstein GAN (WGAN) - це вид генеративних змагальних мереж, який розв'язує проблему недостатньої якості генерованих зображень, яка часто виникає у стандартних GAN. WGAN використовує функцію відстані між розподілами, що називається відстанню Вассерштейна, для забезпечення більш стабільного та прогресивного навчання. [42]

Метрика відстань Вассерштейна між двома кумулятивними розподілами функцій F, G визначається як (формула 4.7):

$$d_p(F,G) = \inf \left|\left|U - V\right|\right|_p \#(4.7)$$

Де інфімум береться по кожним з пар випадкових величин (U, V) відповідно до кумулятивних розподілів F та G. $d_p(F,G)$ також записується як (формула 4.8)

$$W_p(\mu,\nu) = \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathbb{R}^d \times \mathbb{R}^d} ||x-y||^p d\gamma(x,y)\right)^{\frac{1}{p}} \#(4.8)$$

WGAN працює наступним чином. У генератора є набір параметрів, які можна навчити таким чином, щоб він генерував синтетичні зразки, що виглядають як реальні. У дискримінатора є також набір параметрів, які можна навчити розрізняти між реальними та синтетичними зразками. У WGAN ці параметри оптимізуються з метою зменшення відстані Вассерштейна між розподілом справжніх зразків і розподілом синтетичних зразків [43].

Для застосування WGAN до генерації та прогнозування часових рядів використовують відповідний набір даних, який складається з послідовностей чисел, що відображають значення змінної в різні моменти часу – у випадку даної магістерської роботи – часові ряди. Далі, генератор будує послідовності чисел, які відображають нові можливі реалізації цього часового ряду. Дискримінатор в свою чергу оцінює, наскільки синтетичний часовий ряд відрізняється від реального.

4.6 Архітектура мережі довгої короткострокової пам’яті

Однією з моделей для побудови прогнозу фінансових числових рядів обрано різновид рекурентних нейронних мереж RNN – мережа довгої короткострокової пам’яті LSTM.

У магістерській роботі було використано метод [44], який заснований на рекурентній нейронній мережі зі зв'язками "багато-до-багато" (many-to-many). Цей метод дозволяє моделі приймати послідовності на вхід і повертати послідовності у відповідь, як показано на рис. 4.4.

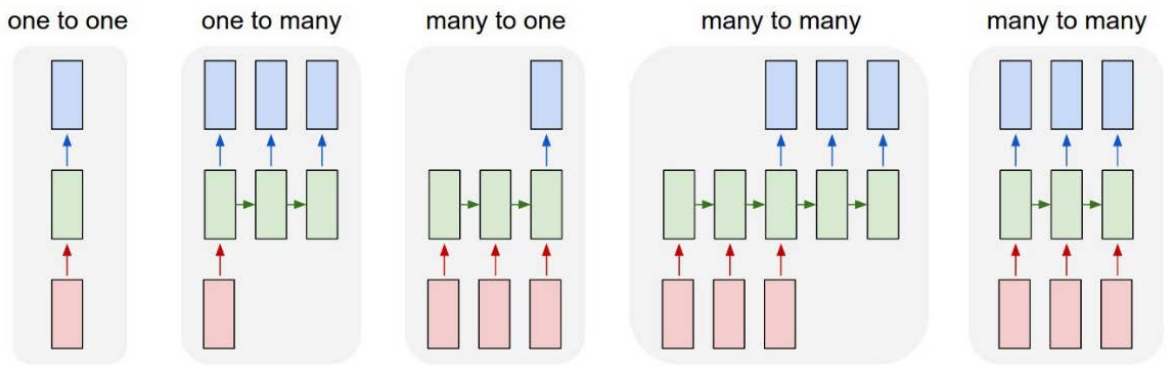


Рисунок 4.4 - Типи реалізації рекурентних нейронних мереж [45]

Архітектура LSTM була розроблена, щоб уникнути проблем довгострокової залежності в RNN. Однією з переваг RNN є здатність зв'язувати попередню інформацію з поточною, але в деяких випадках мережа втрачає цю властивість. Коли для виконання поточної задачі потрібна недавня інформація, дистанція між якою і місцем застосування є невеликою, класична RNN може ефективно працювати з попередніми даними. Але якщо дистанція між необхідною інформацією та її місцем застосування зростає, виникає проблема втрати довгострокового зв'язку (рис. 4.5).

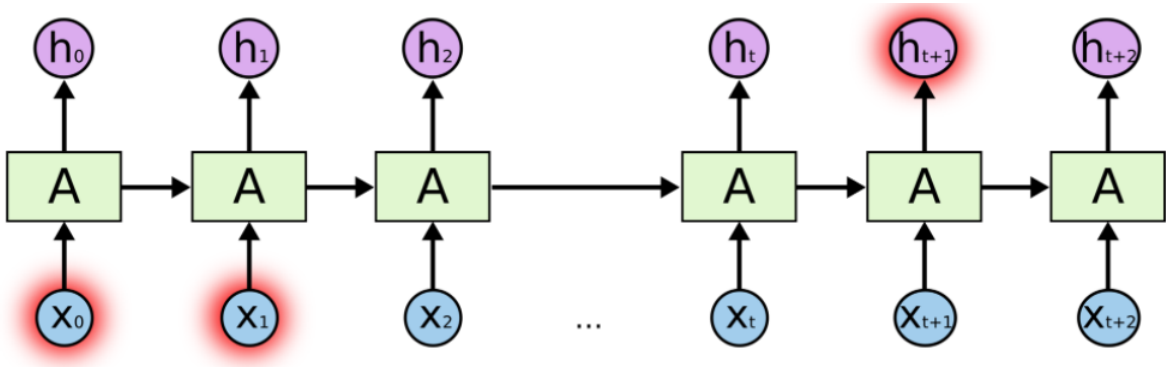


Рисунок 4.5 – Демонстрація проблеми довгострокової залежності [46]

У класичній архітектурі RNN виникають проблеми з довгостроковою залежністю, які спричинені явищами зникаючого та зростаючого градієнту (Vanishing and Exploding gradient) [48]. Під час проходження інформації через нейронну мережу від вхідних до вихідних нейронів, похибка обчислюється та розповсюджується в зворотньому напрямку для оновлення вагів. Під час навчання функція втрат порівнює отримані результати з бажаними (див. рисунок 3.6). Якщо часткова похідна похибки менша за 1, то при множенні на learning rate зміна

градієнта буде незначною. Чим далі просування по мережі, тим менше значення градієнта і складніше тренувати ваги (див. рисунок 4.5).

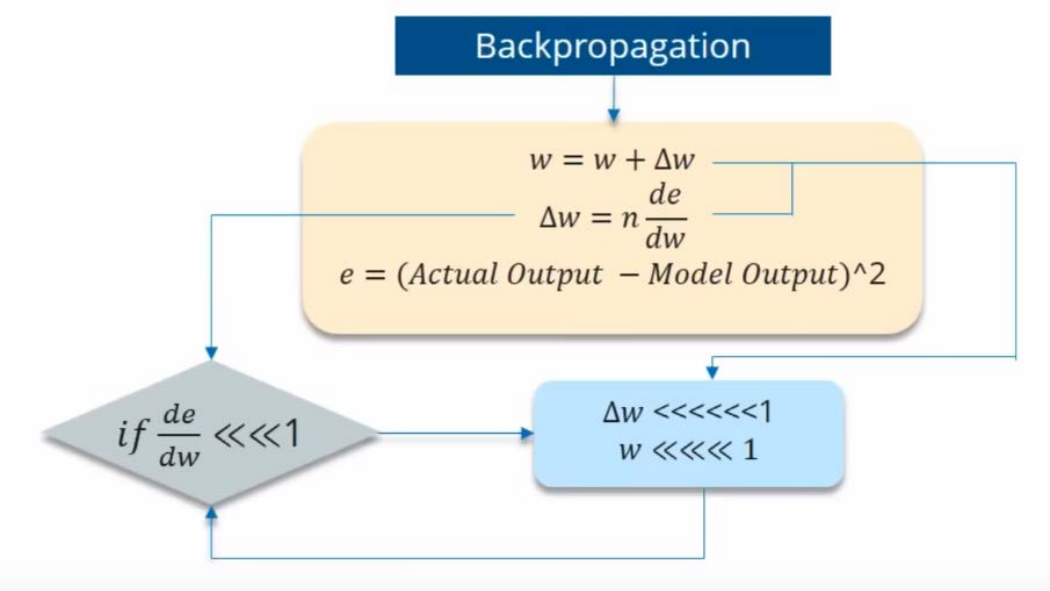


Рисунок 4.6 – Проблема зникаючого градієнту при оновленні вагів [45]

Зростаючий градієнт виникає, коли ваги мережі отримують дуже великі значення. Це призводить до зростання градієнта помилки і накопичення дуже великих значень, які ускладнюють процес тренування та призводять до нестабільності моделі (див. рис. 4.7).

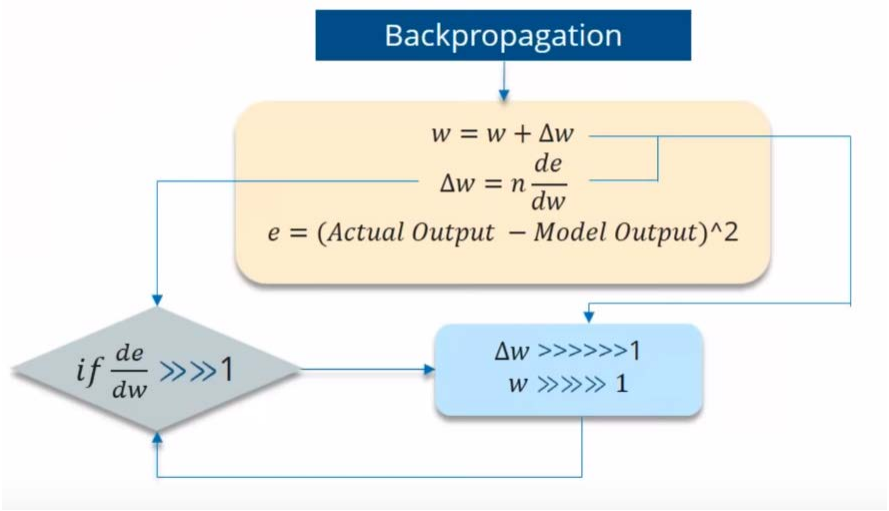


Рисунок 4.7 - Проблема зростаючого градієнту при оновленні вагів [45]

Задача забезпечення можливості встановлення довгострокових зв'язків важлива при прогнозуванні фінансової інформації, тому було обрано модифікацію рекурентних нейронних мереж - LSTM - для використання в дипломній роботі. LSTM були розроблені Хохрайтером та Шмідхубером в 1997 році [44] і з того часу були модернізовані багато разів. Ця архітектура була спеціально розроблена для стабільної роботи з довгостроковими залежностями. LSTM представляють собою ланцюг з модулів, кожен з яких містить чотири шари нейронних мереж, що взаємодіють особливим чином (див. рис. 4.8).

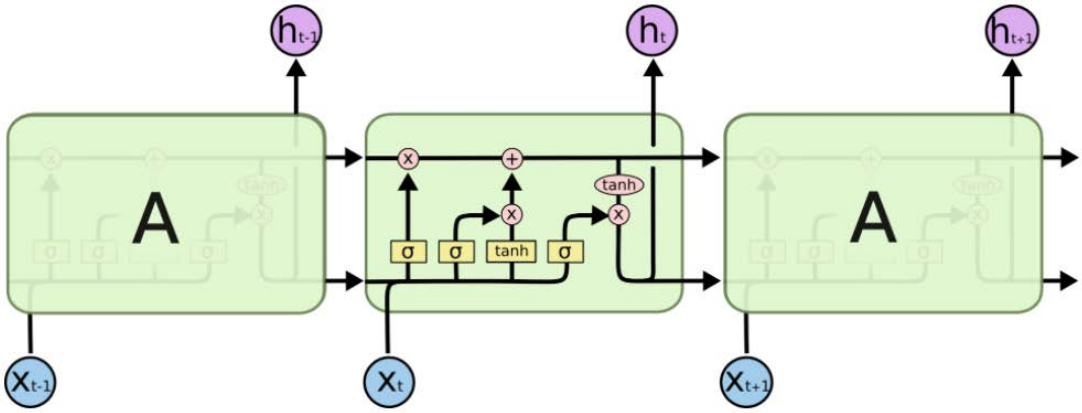


Рисунок 4.8 – Модуль, що повторюється, складається з 4 шарів [46]

В якості функцій активації в модулі LSTM застосовано [49] сигмоїду (формула 3.41):

$$\sigma(x) = \frac{1}{1 + e^{-x}} \ \#(3.41)$$

також гіперболічний тангенс (формула (3.42):

$$\tanh = \tanh(x) \ \#(3.42)$$

Один з ключових елементів в архітектурі LSTM - це стан ячейки (cell state), який є горизонтальною лінією, що пролягає через усі блоки мережі, дозволяючи інформації переміщуватись всередині мережі (див. рис. 4.9).

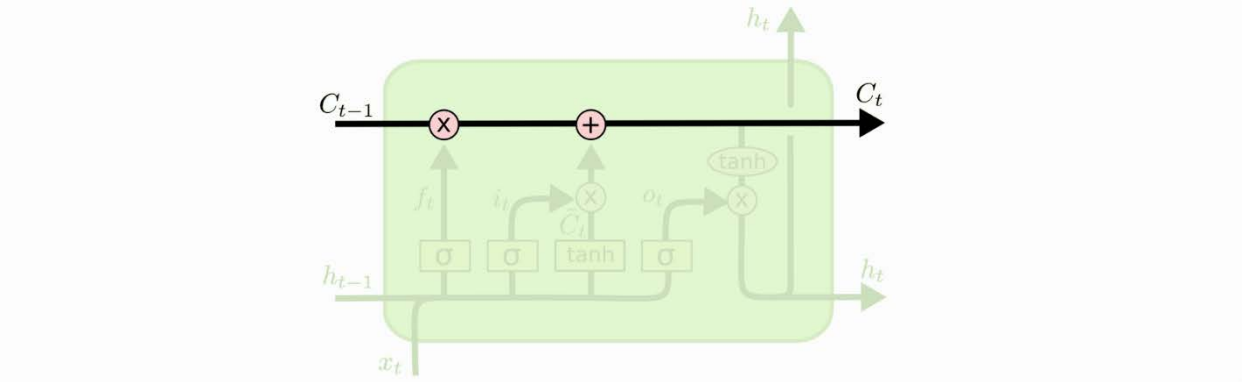


Рисунок 4.9 – Стан ячейки (cell state) – елемент модуля LSTM [46]

Модуль LSTM складається з ячейки (cell) – частини пам’яті модуля і трьох регуляторів потоку інформації всередині модуля –вхідний вентиль (input gate), вихідний вентиль (output gate) і вентиль забування (forget gate) [44, 49].

Розглянемо детальніше побудову та взаємодію всередині модуля.

4.6.1 Forget gate layer

Forget gate layer - це важливий компонент архітектури LSTM, який допомагає моделі визначати, яку інформацію необхідно забути з попереднього стану ячейки. Шар має сігмоїдальну функцію активації, яка забезпечує значення в діапазоні [0, 1]. Забування попереднього стану є

важливим для забезпечення стійкості до шуму та даних зі змінною якістю. Шар отримує на вхід поточну інформацію x_t та результат роботи попереднього модуля h_{t-1}. Цей процес дозволяє моделі зосередитися на нових даних та забезпечити оптимальну обробку даних в кожен момент часу. Функціонування forget gate layer проілюстровано на рисунку 4.10, а формули для розрахунку значень знаходяться в рівнянні 3.43.

$$f_t = \sigma\left(W_f * [h_{t-1}, x_t] + b_f\right) \#(3.43)$$

де f_t – вихідне значення шару фільтру забування;

x_t – актуальна інформація на вхід;

h_{t-1} – результат роботи попереднього модуля;

σ – сигмоїдальна функція активації;

W_f – вага входу шару фільтру забування;

b_f – вага зміщення шару фільтру забування.

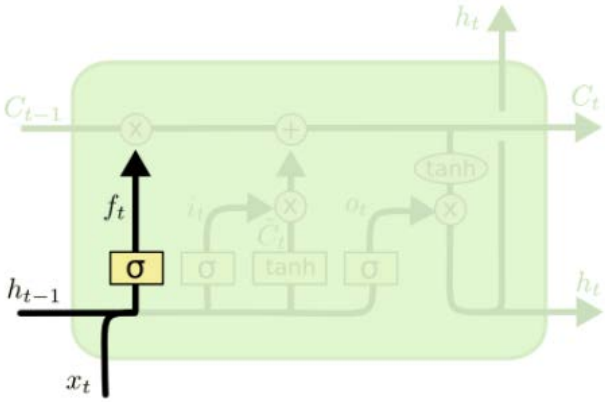


Рисунок 4.10 – Шар фільтру забування [46]

4.6.2 Input layer gate

Input layer gate є одним з ключових компонентів архітектури LSTM. Цей шар фільтрує вхідні дані та складається з двох пішарів: сигмоїдального та tanh-шару. Сигмоїдальний шар визначає, які значення необхідно оновити в стані ячейки на підставі попереднього стану ячейки та нового вхідного сигналу. За своєю природою сигмоїдальна функція активації дозволяє використовувати значення в діапазоні [0, 1], тому вона підходить для визначення того, яку частину інформації слід зберегти. Сигмоїдальний шар визначає, які значення вхідних даних потрібно оновити, використовуючи формулу 3.44. Тангенціальний шар будує вектор значень $\tilde{C}_t$, який додається до стану ячейки C_t, використовуючи формулу 3.45. Цей принцип проілюстровано на рисунку 4.11.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \#(3.44)$$

$$\tilde{C}_t = \tanh(W_C * [h_{t-1}, x_t] + b_C) \ \#(3.45)$$

де i_t – вихідне значення сигмоїдального шару;

$\tilde{C}_t$ – вихідне значення Tanh-шару;

σ – сигмоїдальна функція активації;

W_{i} – вага входу сигмоїдального шару фільтру входу;

b_i – вага зміщення сигмоїдального шару фільтру входу;

x_t – актуальна інформація на вхід;

h_{t-1} – результат роботи попереднього модуля;

W_C – вага входу Tanh-шару фільтру входу;

b_C – вага зміщення Tanh-шару фільтру входу.

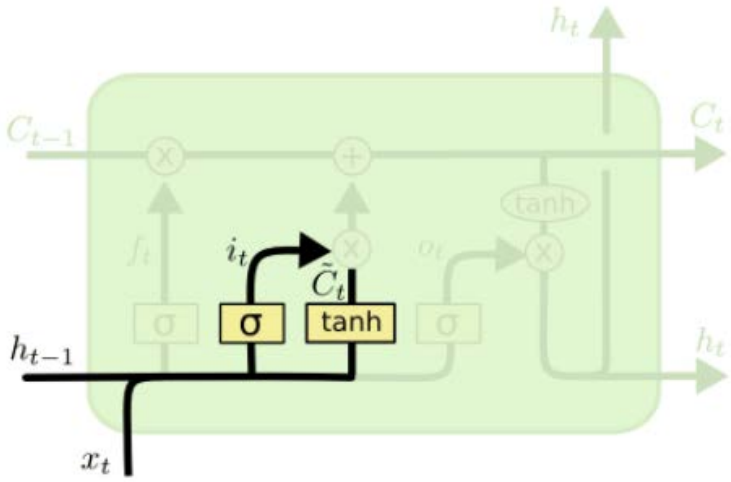


Рисунок 4.11 - Шар фільтру входу [46]

4.6.1 Оновлення стану ячейки

Під час оновлення стану ячейки LSTM, старе значення C_{t-1} множиться на вихідне значення шару фільтру забування f_t та додається виходу шару фільтру входу, яке обчислюється як добуток i_t та tanh-шару $\tilde{C}_t$ (формула 3.46). Це дозволяє зберегти корисну інформацію зі старого стану, яка не повинна бути вилучена, та оновити її за новими вхідними даними. Результатом цього процесу є нове значення стану ячейки C_t. [49] Схематично процес показано на рис. 4.12.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \ \#(3.46)$$

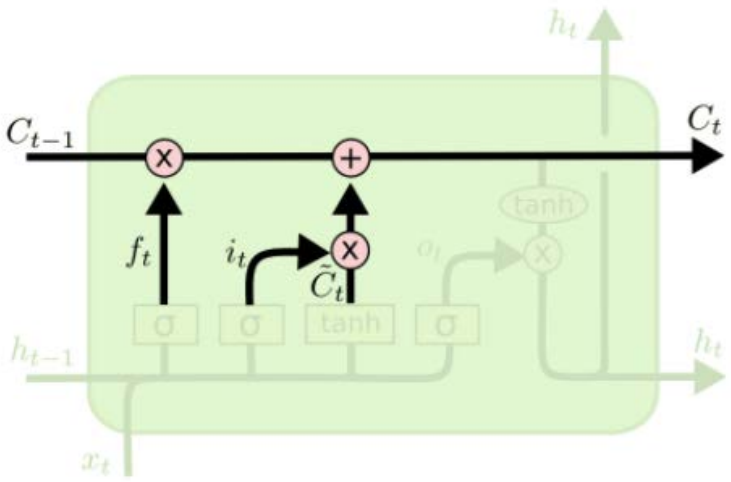


Рисунок 4.12 – Оновлення стану ячейки [46]

4.6.1 Output gate layer

Шар фільтру виходу відповідає за результат роботи нейронної мережі. Його вихідні значення обчислюються за допомогою сигмоїдального шару o_t (формула 3.48), який вирішує, яку інформацію виводити зі стану ячейки, та стану ячейки C_t, обробленого функцією активації tanh з вихідними значеннями в діапазоні [-1; 1] (формула 3.47). Результат роботи шару фільтру виходу показує, які значення мережа повинна виводити після обробки вхідних даних та врахування попереднього стану. [49]

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \#(3.47)$$

$$h_t = o_t * \tanh(C_t) \#(3.48)$$

де o_t - вихідне значення сигмоїдального шару;

σ - сигмоїдальна функція активації;

tanh – функція активації гіперболічний тангенс;

W_o - вага входу сигмоїдального шару фільтру виходу;

b_o - вага зміщення сигмоїдального шару фільтру виходу;

x_t – актуальна інформація на вхід;

h_{t-1} – результат роботи попереднього модуля;

h_t – результат роботи поточного модуля.

Принцип роботи шару виходу продемонстровано на рисунку 4.13.

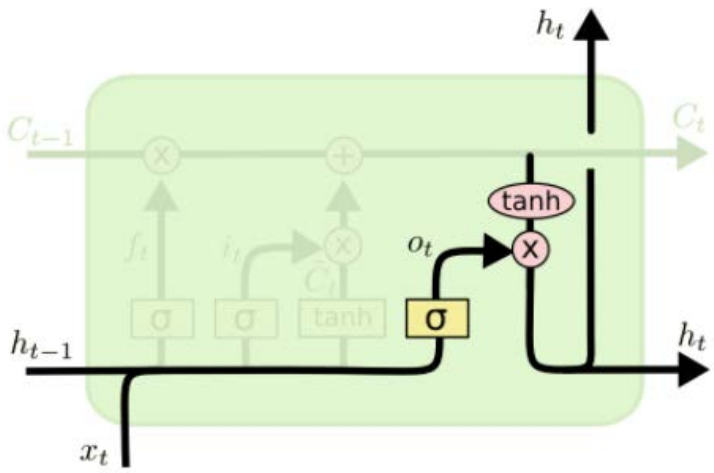


Рисунок 4.13 – Шар фільтру виходу [46]

4.7 Висновки до розділу

В розділі 3 було описано математичне забезпечення, яке було використане в ході проведення досліджень при виконанні магістерської роботи роботи. Математичні алгоритми та моделі: методи технічного аналізу для створення тенденційних та осциляторних технічних індикаторів, модель BERT для класифікації настроїв фінансових новин, декілька технік ансамблювання моделей машинного навчання: bagging,

boosting, stacking. Розглянуто модифікації генеративних нейронних мереж в контексті моделювання та прогнозування часових рядів, також модифікацію рекурентних нейронних мереж модель довгої-короткострокової пам’яті LSTM для навчання на отриманих даних та для прогнозування фінансового часового ряду.

5 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

5.1 Опис та алгоритм роботи програмного забезпечення

Для виконання усіх поставлених задач даного дисертаційного дослідження, необхідно програмно реалізувати наступні етапи функціоналу.

1. Виконати аналіз предметної області, з’ясувати, які дані можуть мати вплив на цільову змінну – дані вартості цінних паперів компанії.
2. Виконати збір даних з декількох джерел: сервісв та провайдерів з фінансовою інформацією Yahoo Finance [50], FRED [53], Investing.com [51], сервісу статистики по пошуковим запитам Google Trends [55], веб-сайту економічно-ділових новин Financial Times [58]
3. Здійснити очистку табулярних та текстових даних
4. Генерація додаткових корисних для навчання моделей ознак, зокрема проведення технічного аналізу, спектральні розкладання тощо
5. Класифікація тексту фінансових новин за допомогою переднавченої мовної моднлі finBERT
6. Провести об’єднання наявних даних, кодівання та масштабування, розбиття на виборки відповідно фактору часу
7. Провести тренування моделей машинного та глибокого навчання, декількох типів ансамблей моделей.
8. Виконати оцінку прогнозу отриманого при застосуванні навчерої моделі, спираючись на метрики, візуалізувати результат дослідження.

9. Забезпечити функціонування веб-інтерфейсу для взаємодії з користувачем

5.2 Програмне забезпечення компонентів системи

Програмне забезпечення написане на мові програмування python. В ході розробки використовувались наступні бібліотеки та пакети:

- NumPy (Numerical Python) - це бібліотека для мови програмування Python, яка дозволяє легко та швидко працювати з багатовимірними масивами та матрицями, включаючи ефективні функції для математичних операцій з цими об'єктами. Вона дозволяє легко створювати та маніпулювати масивами даних, здійснювати елементарні математичні операції над ними, виконувати лінійну алгебру, статистичний аналіз, обробку зображень та багато іншого.
- Pandas - це бібліотека мови програмування Python для роботи з даними. Вона надає інструменти для ефективного зберігання, обробки та аналізу структурованих даних, таких як таблиці, часові ряди та масиви даних. Pandas також має вбудовану підтримку для роботи зі зв'язками з базами даних та файловими форматами даних, такими як CSV, Excel та SQL.
- Scikit-learn - це бібліотека машинного навчання для Python, яка містить набір інструментів для класифікації, регресії, кластеризації та інших типів задач машинного навчання. Бібліотека містить реалізації багатьох алгоритмів навчання з урахуванням різних типів даних, що дозволяє легко побудувати

та навчити моделі машинного навчання. Scikit-learn також містить інструменти для підготовки та обробки даних, а також для оцінки та порівняння різних моделей машинного навчання.

- TensorFlow - це відкрите програмне забезпечення для машинного навчання та глибинного навчання, розроблене компанією Google. Він містить набір інструментів та бібліотек для побудови та тренування різних моделей нейронних мереж, а також забезпечує зручний інтерфейс для розподіленого обчислення.
- Keras - це вищорівнева бібліотека для машинного навчання, яка працює поверх TensorFlow (та інших фреймворків). Вона надає простий інтерфейс для створення та тренування різноманітних моделей нейронних мереж, включаючи звичайні нейронні мережі, рекурентні мережі та мережі глибинного навчання. Keras підтримує багато функцій, таких як візуалізація даних, робота зі зображеннями та текстами, розподілене обчислення та інші.
- Statsmodels - це бібліотека для статистичного моделювання та економетрики на мові Python. Вона надає ряд статистичних методів, які можна використовувати для аналізу даних та моделювання. Statsmodels включає ряд класів та функцій для проведення різноманітних статистичних тестів, побудови моделей регресії, аналізу часових рядів та багато іншого. Бібліотека містить також інструменти для візуалізації результатів та оцінки точності моделей.
- Plotly - це інтерактивна бібліотека для візуалізації дани. Вона дозволяє створювати високоякісні графіки, діаграми та інші візуалізації з можливістю інтерактивного взаємодії з користувачем.
- Streamlit - це відкрите програмне забезпечення для розробки веб-додатків з фокусом на машинному навчанні та аналізі даних. За

допомогою Streamlit можна створювати інтерактивні веб-додатки з використанням Python, де користувачі можуть взаємодіяти з даними та результатами аналізу у режимі реального часу.

5.3 Контейнерізація

Для забезпечення відповідності принципам інкапсуляції окремих компонентів системи з різним функціоналом, кожен з компонентів необхідно розмістити в окремому віртуальному середовищі з власними змінними середовища та залежностями.

Можна виділити три основних компонента, які мають сильно пов’язаний функціонал всередині себе та необхідність в більш слабкому зв’язку з іншими компонентами ззовні – це:

- Фронтенд – веб-інтерфейс для відображення результатів та взаємодії з користувачем
- Бекенд – компонент обробки, очистки, генерації даних, інференсу мовної та навченої для прогнозування моделі
- Сховище даних – база даних для зберігання табулярних та текстових даних

Docker контейнери - це легкі пакети програмного забезпечення, які містять у собі усі необхідні залежності та конфігурацію для виконання додатків. Вони використовують віртуалізацію на рівні операційної системи для забезпечення ізольованого середовища виконання додатків на будь-якому комп'ютері. Кожен контейнер містить в собі всі необхідні компоненти для роботи програмного забезпечення, такі як бібліотеки,

середовище виконання та конфігураційні файли, що дозволяє легко переносити додатки з одного середовища в інше, із запобіганням конфліктів між залежностями. Docker контейнери використовуються для створення різноманітних додатків, від веб-додатків до мікросервісів та аналітичних систем.

Для забезпечення безперебійної комунікації між контейнерами виконана оркестрація контейнерів, взаємодія та обмін інформацією між контейнерами виконується за допомогою інтерфейсу RestAPI (рис. 5.1).

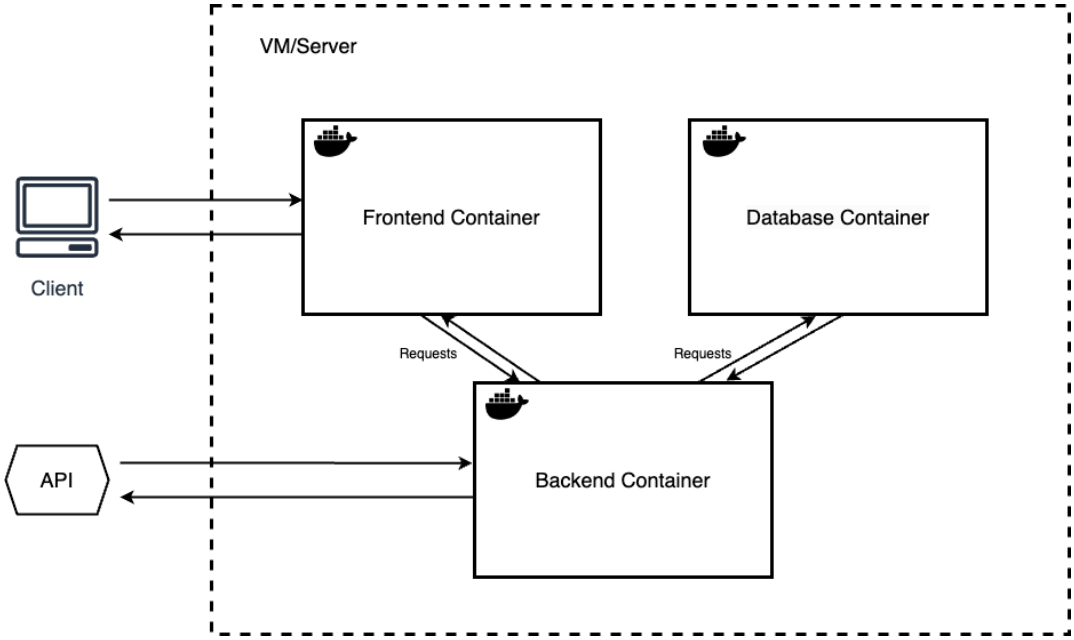


Рисунок 5.1 – Діаграма взаємодії компонентів архітектури

Frontend контейнер містить веб-сервер, який забезпечує інтерфейс для користувача, який взаємодіє з системою через веб-браузер. Користувач виконує запити на прогнозування цін акцій, які надсилаються до Backend контейнера.

Backend контейнер містить сервер, який оброблює запити на прогнозування цін акцій, отримує дані від Database контейнера та повертає результат користувачеві через Frontend контейнер. Backend контейнер

використовує моделі машинного навчання, навчені на історичних даних цін акцій, для прогнозування цін на майбутній період.

Database контейнер містить базу даних, в якій зберігаються історичні дані цін акцій, які використовуються для навчання статистичних моделей в Backend контейнері. Database контейнер забезпечує доступ до даних через відповідні запити в Backend контейнері.

5.4 Ансамбліювання контейнерів

Docker Compose - це інструмент для створення та управління багатоконтейнерними додатками з використанням Docker. Він дозволяє визначити всі компоненти додатку в конфігураційному файлі, такі як контейнери, мережі, об'ємні зберігання тощо, та запустити їх за допомогою однієї команди.

Конфігураційний файл Docker Compose містить інформацію про кожен контейнер додатку, таку як образ контейнера, встановлювані пакети, налаштування мереж та змінні середовища. За допомогою команди всі контейнери з конфігураційного файлу будуть запущені одночасно та будуть забезпечувати роботу додатку в цілому.

Docker Compose дозволяє також створювати та налаштовувати мережі між контейнерами, встановлювати порти для доступу до контейнерів та об'єднувати об'ємні зберігання між контейнерами. Це дозволяє забезпечувати зручне тестування, розгортання та масштабування багатоконтейнерних додатків.

У випадку системи прогнозування цін акцій, ми можемо використовувати Docker Compose для збирання та запуску трьох контейнерів: Frontend, Backend та Database.

Frontend контейнер може містити web-додаток, який дає користувачеві можливість взаємодіяти з системою прогнозування цін акцій через браузер. Backend контейнер може містити серверну частину, що обробляє запити від користувача та виконує прогнозування цін акцій. Database контейнер може містити базу даних, де зберігаються дані про акції, що потрібні для прогнозування.

Docker Compose дозволяє нам визначити всі ці контейнери в одному файлі-конфігурації, де ми можемо вказати, які образи контейнерів використовуються, як вони мають бути налаштовані та які мережі мають бути створені для взаємодії між ними.

Одним з переваг використання Docker Compose є те, що він дозволяє легко налаштовувати середовище розробки для багатоконтейнерних додатків та запускати їх на різних платформах, що робить його ідеальним інструментом для розгортання та управління багатоконтейнерними додатками.

5.5 Висновки до розділу

В даному розділі магістерської дисертації було описано алгоритм та етапи роботи розробленого програмного забезпечення, описані використані в програмному забезпеченні бібліотеки та пакети.

Наведені відомості про контейризацію окремих компонентів, описана взаємодія зазначених контейнерів між собою, продемонстрована діаграма взаємодії компонентів архітектури.

Описано процес створення багатоконтейнерного додатку з застосуванням оркестрації за допомогою Docker Compose.

6 ВЕРИФІКАЦІЯ ТА ВАЛІДАЦІЯ РЕЗУЛЬТАТІВ РОБОТИ СИСТЕМИ

6.1 Метрики оцінки роботи моделі

При розробці моделей та їх порівнянні застосовувались наступні метрики оцінки прогнозу часового ряду:

1. Середня абсолютна помилка (Mean Absolute Error - MAE): це середня абсолютна величина різниці між прогнозованою та фактичною значеннями часового ряду. Вона оцінює, наскільки віддаленими від істинного значення є прогнози.
2. Середньоквадратична помилка (Mean Squared Error - MSE): це середня квадратична величина різниці між прогнозованою та фактичною значеннями часового ряду. Вона зважує на відхилення більш великих прогнозних помилок, ніж MAE.
3. Корінь середньоквадратичної помилки (Root Mean Squared Error - RMSE): це квадратний корінь MSE, використовується для того, щоб метрика відображала ту ж саму одиницю вимірювання, що і самі дані.
4. Середньоквадратичне відхилення (Standard Deviation - SD): це міра розсіювання значень часового ряду. Використовується для оцінки точності прогнозів.
5. Коефіцієнт детермінації (Coefficient of Determination - R-squared): це міра того, наскільки добре прогнозовані значення відповідають фактичним значенням часового ряду. Значення R-squared знаходиться в діапазоні від 0 до 1, де 1 означає ідеальне прогнозування, а 0 - ніякої кореляції між прогнозом та фактичними значеннями.

6. Симетричне середнє абсолютне відхилення (Symmetric Mean Absolute Percentage Error - SMAPE): це метрика, яка враховує відносну помилку прогнозу. Вона визначається як середнє абсолютне відхилення прогнозу від фактичного значення, поділене на середнє арифметичне з фактичного та прогнозованого значень.

6.2 Оцінка коректності класифікації тексту фінансових новин

Оцінка коректності класифікації тексту фінансових новин за допомогою переднавченої моделі може бути здійснена на основі порівняння прогнозів моделі з реальними мітками класу, що вказані в датасеті. Для цього можна використовувати різні метрики оцінки класифікації, такі як точність (accuracy), чутливість (recall), специфічність (specificity) та F1-міра (F1-score).

Перед оцінкою коректності класифікації, необхідно підготувати дані для тестування. Зазвичай це включає в себе токенізацію, векторизацію та побудову матриці ознак для тестового набору даних. Для цього можна використовувати ті ж самі методи, що були використані для навчання моделі.

Далі, після підготовки тестових даних, можна викликати метод predict() на навченій моделі, щоб отримати прогнозований клас для кожного зразка в тестовому наборі даних. Після цього можна порівняти отримані прогнози з реальними мітками класу, щоб оцінити коректність класифікації.

Для оцінки коректності класифікації можна використовувати різні метрики, залежно від конкретної задачі та вимог до моделі. Наприклад,

точність (accuracy) визначається як відношення кількості правильно класифікованих зразків до загальної кількості зразків.

6.3 Крос-валідація на часових рядах

В часовому ряді часовий порядок є дуже важливим. Тому для крос-валідації на часових рядах ми можемо використовувати методи, такі як "Walk Forward Validation" (WFV) або "Rolling Window Validation".

У "Walk Forward Validation", ми вибираємо деяку початкову кількість даних як тренувальну вибірку, та після цього додаємо по одному новому спостереженню до тренувальної вибірки, навчаємо модель та тестуємо наступне спостереження як частину тестової вибірки. Цей процес повторюється далі до закінчення часового ряду. Кількість кроків може варіюватися, залежно від того, наскільки багато даних ми маємо та наскільки довгий часовий період ми хочемо передбачити.

У "Rolling Window Validation", ми також вибираємо деяку початкову кількість даних як тренувальну вибірку, але відмінність полягає в тому, що ми не додаємо нові спостереження до тренувальної вибірки, ми просто зсуваємо вибірку вправо на певну кількість спостережень та знову навчаємо та тестуємо модель

ВИСНОВКИ

При виконанні даної магістерської дисертації було проведено наступні дослідження та одержано практичні та теоретичні результати:

1. Проведено дослідження предметної області, вивчення напрямку та сфери діяльності розглянутої компанії для визначення найбільш впливових та корисних для прогнозування ознак. Виконано збір зазначених в дисертації даних. Проведено опис та порівняння основних моделей в контексті прогнозування фінансової інформації у вигляді часових рядів з довгостроковими та нелінійними залежностями. Визначення текстової фінансової інформації як джерела інсайдів про зміну показників компанії.
2. Здійснено огляд існуючих програмних рішень, також огляд застосунків та програмного забезпечення для прогнозування вартості фінансових активів активів; порівняння, виділення переваг та недоліків.
3. Виконано розробку та опис системи прогнозування вартості цінних паперів компанії, представленно класи та компоненти системи у вигляді діаграм та блок-схеми.
4. Виконано розробку та опис математичного забезпечення, яке використовується в процесі досліджень та реалізації системи. Огляд інструментів технічного аналізу фінансової інформації, мовних моделей для класифікації емоційного забарвлення тексту фінансових новин, архітектури застосованої нейронної мережі, технік ансамблювання моделей машинного навчання.
5. Виконано розробку та опис програмного забезпечення компонентів системи, реалізація інкапсуляції компонентів шляхом

контейнеризації. Забезпечення взаємодії між контейнеризованими компонентами з використанням оркестрації.

6. Проведено верифікацію та валідацію результатів шляхом застосування метрик до прогнозованих даних та їх порівняння. Здійснена оцінка коректності класифікації настроїв фінансових новин.
7. Імплементація та тестування розробленого програмного забезпечення на реальних даних.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Lo, A.W. and Mackinlay, A.C. A Non-Random Walk Down Wall Street 5th Ed. / Princeton University Press, — 2002 — P. 17-49.

2. Christoffersen, P.F. and F.X. Diebold. Financial asset returns, direction-of-change forecasting, and volatility dynamics / Management Science, — 2006. — p. 1273-1287

3. Hyndman, Rob J; Athanasopoulos, George. Vector Autoregressions / Forecasting: Principles and Practice, — 2018 — p. 333–335

4. Christopher A. Sims and Vector Autoregressions / Lawrence J. Christiano – Scand. J. of Economics — 2012 — p. 1082–1104

5. Айвазян С.А. Прикладная статистика. Основы эконометрики. Том 2. / Айвазян С.А. — М.: Юнити-Дана, 2001. — 432 с.

6. Носко В.П. Эконометрика. Введение в регрессионный анализ временных рядов. / Носко В.П. — М., 2002. — 273 с

7. Rencher, Alvin C. Christensen, William F. Chapter 10, Multivariate regression – Section 10.1, Introduction / Methods of Multivariate Analysis, Wiley Series in Probability and Statistics, John Wiley & Sons — 2012 — p. 19

8. Демиденко Е.З. Линейная и нелинейная регрессия. / Е.З. Демиденко — М.: Финансы и статистика, 1981. — 302 с.

9. Ho, Tin Kam. Random Decision Forests. / Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, — 14–16 August 1995. — pp. 278–282.

10. Breiman, Leo. Random forests. Machine learning, — 2001 — p. 5-32.

11. Lin Lin, Wang Fang, Xie Xiaolong, Zhong Shisheng. Random forests-based extreme learning machine ensemble for multi-regime time series

prediction. – Machine building, School of Mechatronics Engineering, Harbin Institute of Technology, Harbin, China / Expert Systems with Applications, Volume 83, — October 2017 — P 164-176

12. Вороновский Г. К., Махотило К. В., Петрашев С. Н., Сергеев С. А. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности. / Вороновский Г. К., Махотило К. В., Петрашев С. Н., Сергеев С. А. — Харьков: Основа, 1997. — 29 - 112 с.

13. Schmidhuber, Jürgen. Deep learning in neural networks: An overview. / Neural Networks. — p: 85–117.

14. Edgar Peters. Chaos and Order in the Capital Markets. A New View of Cycles, Prices, and Market Volatility — М.: Мир, 2000. — 336 с.

15. William Berdstein. The Investor’s Manifesto. Preparing For Prosperity, Armageddon, And Everything In Between. — М.: Альпина Паблишер, — 2013. — 229 с.

16. Li Guo, Feng Shi, and Jun Tu. Textual analysis and machine leaning: Crack unstructured data in finance and accounting. / The Journal of Finance and Data Science — 2016 — P. 153–170

17. Сирота С. В., Агафонов Д. С. Порівняльний аналіз підходів і методів оцінювання емоційного забарвлення фінансової інформації про поточний стан підприємства. Прикладна математика та комп’ютинг. ПМК-2022: п’ятнадцята науково-практична конференція магістрантів та аспірантів, Київ, 16-17 лист. 2022 р.: зб. Тез доп./ [редкол.: Дичка І. А. та ін.]. — К. : Просвіта, 2022. — С. 129-139.

18. Сервіс прогнозування курсу акцій. [Електронний ресурс]. – Режим доступу: https://stocksneural.net/

19. Репозиторій вихідного коду програми для класифікації поведінки акцій. [Електронний ресурс]. - Режим доступу: https://github.com/ shirosaidev/stocksight

20. Репозиторій вихідного коду програми для прогнозування поведінки акцій нейронними мережами. [Електронний ресурс]. - Режим доступу: https://github.com/philipxjm/Deep-Convolution-Stock-Technical-Analysis

21. Robert C. Martin. Design Principles and Design Patterns — 2000 – p 3-32

22. Stewen Akelis. Technical Analysis from A to Z: Covers Every Trading Tool… from the Absolute Breadth Index to the Zig Zag / M.: Диаграмма — 1999. — P. 376.

23. Gubner, John A. Probability and Random Processes for Electrical and Computer Engineers. / Cambridge University Press — 2006.

24. Priestley, M. B. Non-linear and Non-stationary Time Series Analysis. / Academic Press — 1988 — P. 345-360

25. Edward Brian Davies. Spectral Theory and Differential Operators / Volume 42 Cambridge Studies in Advanced Mathematics. // Cambridge University Press — 1996 — P. 212-230

26. Opitz, D.; Maclin, R. Popular ensemble methods: An empirical study. / Journal of Artificial Intelligence Research. — 1999 — p:169–198.

27. Breiman, L., Bagging Predictors / Machine Learning — 1996 — p.123-140

28. Clarke, B. Bayes model averaging and stacking when model approximation error cannot be ignored / Journal of Machine Learning Research — 2003 — pp 683-712

29. Bing Liu. 2012. Sentiment Analysis and Opinion Mining / Synthesis Lectures on Human Language Technologies — 2012 — P. 1–167.

30. Basant Agarwal and Namita Mittal. Machine Learning Approach for Sentiment Analysis. / Springer International Publishing — 2012 — P. 21–45.

31. Oscar Araque, Ignacio Corcuera-Platas, J. Fernando Sánchez-Rada, and Carlos A. Iglesias. Enhancing deep learning sentiment analysis with ensemble techniques in social applications / Expert Systems with Applications — 2017 — P. 236–246.

32. Xiaodong Li, Haoran Xie, Li Chen, Jianping Wang, and Xiaotie Deng. News impact on stock price return via sentiment analysis. / Knowledge-Based Systems — 2014 — P. 14–23.

33. Tim Loughran and Bill Mcdonald. Textual Analysis in Accounting and Finance: A Survey. / Journal of Accounting Research — 2016 — P. 1187–1230

34. Tim Loughran and Bill Mcdonald. When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks. / Journal of Finance — 2011 — P. 35–65.

35. Mathias Kraus and Stefan Feuerriegel. Decision support from financial disclosures with deep neural networks and transfer learning. / Decision Support Systems — 2017 — P. 38–48.

36. Bernhard Lutz, Nicolas Pröllochs, and Dirk Neumann. Sentence-Level Sentiment Analysis of Financial News Using Distributed Text Representations and Multi-Instance Learning. / Technical Report — 2018 — P. 13-25.

37. Babenko, Boris. Multiple instance learning: algorithms and applications. / View Article PubMed/NCBI Google Scholar — 2008 — P. 145-160.

38. Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua. Generative Adversarial Nets. / Proceedings of the International Conference on Neural Information Processing Systems — 2014 — pp. 2672–2680.

39. Salimans, Tim; Goodfellow, Ian; Zaremba, Wojciech; Cheung, Vicki; Radford, Alec; Chen, Xi. Improved Techniques for Training GANs — 2016 — p.1256-1287

40. Ho, Jonathon; Ermon, Stefano. Generative Adversarial Imitation Learning / Advances in Neural Information Processing Systems. — 2016 — pp: 4565–4573

41. RyanTurner, JaneHung, EricFrank, YunusSaatci, JasonYosinski. Metropolis-Hastings Generative Adversarial Networks — 2018 — p. 12-15

42. Ian Goodfellow. Comments on Wasserstein GAN / MachineLearning — 2017.

43. Arjovsky Martin; Chintala Soumith; Bottou Léon. Wasserstein Generative Adversarial Networks. / International Conference on Machine Learning — 2017 — pp: 214–223.

44. Denny Britz. Recurrent Neural Networks Tutorial / Denny Britz — 2015 — p. 219-237.

45. Preeti Agarwal, Mansaf Alam. A Lightweight Deep Learning Model for Human Activity Recognition on Edge Devices / Procedia Computer Science — 2020 — P. 2368

46. Benjamin Lindemann, Timo Müller, Hannes Vietz, Nasser Jazdi, Michael Weyrich. A survey on long short-term memory networks for time series prediction — 2021 — p. 37-48.

47. Denny Britz. Recurrent Neural Networks Tutorial / Denny Britz — 2015 — p. 425-456

48. S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. / Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber – In S. C. Kremer and J. F. Kolen, editors, A Field Guide to Dynamical Recurrent Neural Networks. IEEE — 2001 — p. 893-924

49. Klaus Greff; Rupesh Kumar Srivastava; Jan Koutník; Bas R. Steunebrink & Jürgen Schmidhuber. LSTM: A Search Space Odyssey / Klaus Greff; Rupesh Kumar Srivastava; Jan Koutník; Bas R. Steunebrink & Jürgen Schmidhuber — 2015

50. Провайдер фін. інформації Yahoo! Finance. [Електронний ресурс]. - Режим доступу: https://finance.yahoo.com/

51. Провайдер фін. інформації Investing.com. [Електронний ресурс]. - Режим доступу: https://ru.investing.com/

52. Сховище економічних даних FRED. [Електронний ресурс]. - Режим доступу: https://fred.stlouisfed.org/

53. Документація бібліотеки Numpy. [Електронний ресурс]. - Режим доступу: https://numpy.org/

54. Документація бібліотеки Pandas. [Електронний ресурс]. - Режим доступу: https://pandas.pydata.org/

55. Сервіс Google Trends. [Електронний ресурс]. - Режим доступу: https://trends.google.com/

56. Документація бібліотеки PyTrends. [Електронний ресурс]. - Режим доступу: https://pypi.org/project/pytrends/

57. Документація бібліотеки для технічного аналізу TI. [Електронний ресурс]. - Режим доступу: https://technical-analysis-library-in-python.readthedocs.io/

58. Веб-сайт економічно-ділових новин Financial Times. [Електронний ресурс]. - Режим доступу: https://www.ft.com/

59. Документація бібліотеки BeautifulSoup. [Електронний ресурс]. - Режим доступу: https://www.crummy.com/software/BeautifulSoup/

60. Steven Bird, Ewan Klein, and Edward Loper. Natural Language Processing with Python / Steven Bird, Ewan Klein, and Edward Loper – O'Reilly Media, – 2009

61. Документація бібліотеки pytorch. [Електронний ресурс]. - Режим доступу: https://pytorch.org/

62. Документація бібліотеки transformers. [Електронний ресурс]. - Режим доступу: https://huggingface.co/transformers/

63. Документація фреймворку dash. [Електронний ресурс]. - Режим доступу: https://dash.plotly.com/

64. Dan Hendrycks, Kevin Gimpel. Gaussian Error Linear Units (GELUs) / Dan Hendrycks, Kevin Gimpel – 2016

Додаток А Лістинги програми

Лістинг файлу 2GoogleTrendsConstruct.ipynb – отримання даних про кількість запитів з сервісу Google Trends

```
# імпорт бібліотек
import numpy as np
import pandas as pd

from calendar import monthrange
import datetime
from pytrends.request import TrendReq
pytrends = TrendReq(hl='en-US', tz=360) # відключення до Гугл
### Create daily data
last_day_months= pd.date_range(start = '2010-01-01', end= '2021-05-01', freq='M').to_numpy().astype('datetime64[D]').astype('str')
last_day_months
first_day_months = np.arange('2010-01-01', '2021-05-01', dtype = 'datetime64[M]').astype('datetime64[D]').astype('str')
first_day_months
z = list(zip(first_day_months, last_day_months))
z
#створення щоденних даних
def create_df(periods):
    kw_list = ["goldman sachs"]

    final_df = pd.DataFrame()
    for i in periods:
        pytrends.build_payload(kw_list, timeframe=str(i[0]) +' '+ str(i[1]), geo='US') #запит даних за 1 місяць
        df = pytrends.interest_over_time()
        df.drop(['isPartial'], axis=1, inplace=True)

        final_df = pd.concat([final_df, df]) #додання в датасет

    return final_df
%%time
daily_df = create_df(z)
Wall time: 2min 16s
daily_df #датасет зі щоденними даними
daily_df.to_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\google_trends\GSGoogleTrends_daily.csv')
daily_df.plot(figsize = (18,5)); #візуалізація
### Create month data
#створення запиту на щомісячні дані за 10 років
kw_list = ["goldman sachs"]
pytrends.build_payload(kw_list, timeframe='2010-01-1 2021-05-01', geo='US')
month_df = pytrends.interest_over_time()
month_df.drop(['isPartial'], axis=1, inplace=True)
month_df
month_df.to_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\google_trends\GSGoogleTrends_monthly.csv')
month_df.plot(figsize = (18,5)); #візуалізація
### Stack dfs together
daily_df.reset_index()
final_df = pd.DataFrame()
final_df = daily_df.copy().reset_index()
final_df = final_df.rename({'goldman sachs':'daily_data'}, axis=1)

#зведення даних  в єдиний датасет відповідно до дат
final_df['monthly_data'] = final_df['date'].map(lambda x: x.replace(day=1))\
                                .map(month_df['goldman sachs'])
final_df['monthly_data'] = final_df['monthly_data'] / 100
final_df
#отримання фінальних зважених даних
final_df['adjusted_data'] = final_df['monthly_data'] * final_df['daily_data']
final_df
final_df.set_index('date', inplace=True)
final_df
final_df[['adjusted_data']].plot(figsize = (18,5)); #візуалізація
```

```
#збереження даних
final_df.to_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\google_trends\GSGoogleTrends_final.csv')
```

Лістинг файлу 3create_dataset.ipynb – зведення даних в один датасет

```
# імпорт бібліотек
import glob, os
from pathlib import Path
import pandas as pd
import numpy as np
# import datetime
# шлях до директорії
PATH = os.getcwd()

start = '2010-01-04'
end = '2021-04-01'
# основа для формування датасета
# встановлення часового проміжку
def create_base_df(start, end):
    df= pd.DataFrame()
    df['Date'] = pd.date_range(start= start, end=end, freq='D')
    df.set_index('Date', inplace=True)
    return df

# отримання назв файлів з папки
def get_files(path):
    os.chdir(path)

    csv_files = [f for f in glob.glob("*.csv")]
    return csv_files

# з'єднання даних в один датасет
def merge_dfs(df, csv_files, path):
    for f in csv_files:
        print(f)
        df_com = pd.read_csv((path + f).replace(' ', ''))
        df_com.set_index('Date', inplace=True)
        df_com = df_com[['Close']]
        df_com.rename({'Close': f[:-4]}, axis=1, inplace=True)
        #     print(df_com.head(5))
        df = pd.merge(df, df_com, how='left', left_index=True, right_index=True)
    return df
# відображення датасетів
def display_dfs(csv_files, path):
    for f in csv_files:
        df = pd.read_csv((path + f).replace(' ', ''))
        print(f)
        print(df.head(5))
        print('-------------------------------------------------')

# заповнення щомісячних даних в щоденний формат без пустих значень
def append_till_date(period_data, freq):
    dfs = []
    for f in period_data:
        df_ = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\indicators\{}'.format(f) + '.csv', index_col='DATE')
        df_.index=pd.to_datetime(df_.index)# зміна типу даних індекса
        empty_df = pd.DataFrame()
        empty_df['DATE'] = pd.date_range(start=str(df_.index[::-1][0]), end='2021-04-01', freq=freq) # часовий проміжок
        empty_df = empty_df[ 1:]
        empty_df.set_index("DATE", inplace=True)
        df_ = df_.append(empty_df, ignore_index=False)
        dfs.append(df_)
        print(df_)
    period_dfs = dict(zip(period_data, dfs))

    return period_dfs
### Total timeline
df = create_base_df(start, end)
```

```
df
### Commodities
# назви файлів з сировинними товарами
path_commodities = PATH + '\data\commodities\ '
csv_files = get_files(path_commodities)
csv_files
['BZ=F_oil.csv', 'CL=F_oil.csv', 'GLD.csv', 'SI=F.csv']
# з'єднання в єдиний датафрейм
df_commodities = merge_dfs(df, csv_files, path_commodities)
df_commodities
# filled_df_commodities = filling_nan(df_commodities)
# візуалізація
df_commodities.plot(figsize= (18, 5));
### Companies
path_companies = PATH + r'\data\companies\ '
# назви файів з показниками компаній
csv_files = get_files(path_companies)
csv_files
# відображення датасетів
display_dfs(csv_files, path_companies)
# додання до загального датасету
df_companies = merge_dfs(df_commodities, csv_files, path_companies)
df_companies
# mask = df_companies['GSBD(march-19-2015)'].index <= '2015-03-18'
# df_companies['GSBD(march-19-2015)'] = df_companies['GSBD(march-19-2015)'].mask(mask, df_companies['GSBD(march-19-2015)'] \
#
# візуалізація
df_companies[df_companies.columns[4:]].plot(figsize=(18,5));
### FX
path_fx = PATH + r'\data\FX\ '
# назви файлів з даними закордонних валют
csv_files = get_files(path_fx)
csv_files
# відображення відповідних датафреймів
display_dfs(csv_files, path_fx)
# додання до загального дф
df_fx = merge_dfs(df_companies, csv_files, path_fx)
df_fx
# візуалізація
df_fx[df_fx.columns[17:]].plot(figsize=(18,5));
### GoogleTrends
# отримання сформованих даних гугл трендів
google_trends_df = pd.read_csv(PATH + r'\data\google_trends\GSGoogleTrends_final.csv', \
                               index_col='date')
google_trends_df.head(5)
# візуалізація
google_trends_df['adjusted_data'].plot(figsize=(18,5))
# додання гугл-трендів в єдиний датасет
df_gt = pd.merge(df_fx, google_trends_df['adjusted_data'], how='left', left_index=True, right_index=True)
df_gt.rename( {'adjusted_data':'google_trends_GS'}, axis =1, inplace=True) # зміна назви ознаки
df_gt
### Indicators
path_indicators = PATH + r'\data\indicators\ '
# назви файлів економічних індикатоів
csv_files = get_files(path_indicators)
csv_files
# відображення датафреймів
display_dfs(csv_files, path_indicators)
df_gt.shape
#### Daily dfs

csv_files
# показник  10YBIR
df_ = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\indicators\10YBIR.csv')
df_.set_index('Date', inplace=True)
df_.rename( {'Value':'10YBIR'}, axis=1, inplace=True) # зміна назви ознаки

df_
# додання в загальний датасет
df_gt = pd.merge(df_gt, df_, how='left', left_index=True, right_index=True)

# показник DFF
df_ = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\indicators\DFF.csv')
df_['Date'] = pd.to_datetime(df_['DATE'])
df_.drop('DATE', axis=1, inplace=True)
```

```
df_.set_index('Date', inplace=True)
# df_ = df_[['Close']]

df_
# додання в загальний датасет
df_gt = pd.merge(df_gt, df_, how='left', left_index=True, right_index=True)

# показник LIBOR
df_ = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\indicators\LIBOR.csv')
df_['Date'] = pd.to_datetime(df_['DATE'])
df_.drop('DATE', axis=1, inplace=True)
df_.set_index('Date', inplace=True)
df_=df_.rename({'USD1MTD156N':'LIBOR'}, axis=1)
df_
# додання в загальний датасет
df_gt = pd.merge(df_gt, df_, how='left', left_index=True, right_index=True)

# показник US10-YearBondYield
df_ = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\indicators\US10-YearBondYield.csv')
df_['Date'] = pd.to_datetime(df_['Date'])
df_.set_index('Date', inplace=True)
df_.sort_values(by='Date', inplace=True)
df_ = df_[['Price']]
df_ =df_.rename({'Price':'10YearBondUS'},axis =1)

df_
df_gt = pd.merge(df_gt, df_, how='left', left_index=True, right_index=True)

# показник TNX
df_ = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\indicators\^TNX.csv')
df_['Date'] = pd.to_datetime(df_['Date'])
df_.set_index('Date', inplace=True)
df_.sort_values(by='Date', inplace=True)

df_ = df_[['Close']]
df_ =df_.rename({'Close':'TNX'},axis =1)

df_
# додання в загальний датасет
df_gt = pd.merge(df_gt, df_, how='left', left_index=True, right_index=True)

# показник VIX
df_ = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\indicators\^VIX.csv')
df_['Date'] = pd.to_datetime(df_['Date'])
df_.set_index('Date', inplace=True)
df_.sort_values(by='Date', inplace=True)

df_ = df_[['Close']]
df_ =df_.rename({'Close':'VIX'},axis =1)

df_
# додання в загальний датасет
df_gt = pd.merge(df_gt, df_, how='left', left_index=True, right_index=True)
# розмірність отриманоого датасету
df_gt.shape
(4106, 29)
# df_gt.to_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\pre_ready_data\df_indicators_daily.csv')

#### Monthly dfs

csv_files
monthly_data = ['CPALTT01USM657N(monthly)', 'M2SL(monthly)', 'MEDCPIM158SFRBCLE(monthly)', 'PPIACO(monthly)',
'UNRATE(monthly)']
monthly_cols = ['CPALTT01USM657N', 'M2SL', 'MEDCPIM158SFRBCLE', 'PPIACO', 'UNRATE']
# заповнення щомісячних даних в щоденний датасет
monthly_dfs = append_till_date(monthly_data,'MS')
monthly_dfs[monthly_data[0]]
# df_gt = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\pre_ready_data\df_indicators_daily.csv',\
#                     index_col='Date')

# перейменування ознак (replace('(monthly)')
df_gt.index = pd.to_datetime(df_gt.index)
for f in monthly_data:
    mothly_df = monthly_dfs[f]
```

```
    df_gt[f.replace('(monthly)', '')] = df_gt.index.map(lambda x: x.replace(day=1)).map(mothly_df[f.replace('(monthly)', '')])

df_gt
# df_gt = fill_periods(df_gt, monthly_data, monthly_dfs, 'monthly')
# візуалізація щомісячних даних
dfa = df_gt[monthly_cols]   #.interpolate(method='linear')
dfa = (dfa-dfa.min())/(dfa.max()-dfa.min())
dfa.plot(figsize=(18,5));
#### Quarterly dfs
csv_files
# щоквартальні дані
quart_data = ['GDP(quarterly)','GDPC1(quarterly)']
quart_cols = ['GDP','GDPC1']
quart_dfs = append_till_date(quart_data, 'QS')
df_gt
# заповнення пустих значень
df_gt = fill_periods(df_gt, quart_data, quart_dfs, 'quarterly')
df_gt
# візуалвзація
dfa = df_gt[quart_cols]   #.interpolate(method='linear')
#dfa = (dfa-dfa.min())/(dfa.max()-dfa.min())
dfa.plot(figsize=(18,5));
### Market indices
path_market_indices = PATH + r'\data\market_indices\ '
# дані показників круних бірж
csv_files = get_files(path_market_indices)
csv_files
# відображення даних
display_dfs(csv_files, path_market_indices)
dfs = []
for f in csv_files:
    df = pd.read_csv(PATH + r'\data\market_indices\{}'.format(f))

    date_set = set(['Дата', 'Date'])
    date_intersection = ' '.join(set(df.columns).intersection(date_set))# визначення колонок з датою

    df.rename({date_intersection: 'Date'}, axis=1, inplace=True)# перейменування


    stock_set = set(['Цена', 'Close'])
    stock_intersection = ' '.join(set(df.columns).intersection(stock_set))# визначення колонок некоректною назвою

    df.rename({stock_intersection: f[:-4]}, axis=1, inplace=True)# перейменуваня

    df = df[['Date', f[:-4]]]
    df['Date'] = pd.to_datetime(df['Date'])# зміна типу даних дати
    df.set_index('Date', inplace=True)

    df.sort_index(inplace=True)# впорядкування за датою
    dfs.append(df)
    market_indices_cols = [i[:-4] for i in csv_files]
market_indices_cols
['DJIA', 'FTSE', 'HSI', 'IXIC', 'N225', 'NYA', 'RUT', 'S&P500']
market_indices_dfs = dict(zip(market_indices_cols, dfs))
market_indices_dfs['DJIA']
q = market_indices_dfs['DJIA'] = market_indices_dfs['DJIA']
q = q[q.index <= '2021-05-01']
market_indices_dfs['DJIA'] = q
market_indices_dfs['DJIA']
# заповнення щоденних даних для індексів бірж
def append_till_date_market_indices(period_data, freq):
    dfs = []
    for f in period_data:
        df_ = market_indices_dfs[f]
        empty_df = pd.DataFrame()
        empty_df['Date'] = pd.date_range(start=str(df_.index[::-1][0]), end='2021-04-01', freq=freq) # необхідний період часу
        empty_df = empty_df[ 1:]
        empty_df.set_index("Date", inplace=True)
        df_ = df_.append(empty_df, ignore_index=False)
        dfs.append(df_)
        print(df_)
    period_dfs = dict(zip(period_data, dfs))

    return period_dfs
complete_market_indices_dfs = append_till_date_market_indices(market_indices_cols, 'D')
```

```
for i, df in complete_market_indices_dfs.items():
    df_gt = pd.merge(df_gt, df, how='left', left_index=True, right_index=True)
df_gt
df_gt.shape
(4106, 44)
df_gt[df_gt.columns[36:]].plot(figsize=(18,5));# візуалізація
# кідбкість пропущених значенб
nan_vals_quan = df_gt.isna().sum()
nan_vals_quan.plot.bar(figsize=(18,5));
nan_vals_quan
df_gt.info()
# df_gt.to_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\pre_ready_data\ready_df_with_nan.csv')
```

Лістинг файлу 4create_technical_indicators.ipynb – створення технічних індикаторів

```
# імпорт бібліотек

import numpy as np
import pandas as pd
import plotly.express as px

from ta.trend import IchimokuIndicator, MassIndex, TRIXIndicator
from ta.momentum import StochasticOscillator, rsi
from ta.volume import VolumeWeightedAveragePrice, money_flow_index

import os
!pip install ta
PATH = os.getcwd()

start = '2010-01-04'
end = '2021-04-01'
### Merde GS data to daily dates
# імпорт даних
df = pd.read_csv(PATH + '\data\GSraw.csv', index_col='Date')
df
# обрання часового проміжку
#df = df[['Close']]
df = df[(df.index >= start) & (df.index <= end)]
df
# кіль-ть пропущених даних
df.isna().sum()
# формування датасету
df_daily = pd.DataFrame()
df_daily['Date'] = pd.date_range(start = start, end = end)
df_daily.set_index('Date', inplace=True)
df_daily
# додання до датасету
df_gs = pd.merge(df_daily, df, how='left', left_index=True, right_index=True)
df_gs.rename({'Close':'GS'},axis=1,inplace=True)
df_gs
# кількість пропущених занченб
df_gs.isna().sum()
# інфо
df_gs.info()
# візуалізація
df_gs[['Open', 'High', 'Low', 'GS', 'Adj Close']].plot(figsize=(18,5));
### Fill nans
# заповнення пропущених занчень
df_gs = df_gs.interpolate(method='linear')# методом інтерполяуії
df_gs.plot(figsize=(18,5));
df_gs.isna().sum()
# df_gs.to_csv(PATH + '\data\pre_ready_data\GSready.csv')
df_gs = pd.read_csv(PATH + '\data\pre_ready_data\GSready.csv', index_col='Date')
df_gs.index = pd.to_datetime(df_gs.index)
df_gs
### Create TI
def get_technical_indicators(df):
```

```
#Simple Movind Average SMA 7 and 21 days
df['MA7'] = df['GS'].rolling(window=7).mean()
df['MA21'] = df['GS'].rolling(window=21).mean()

#Exponential Movind Average EMA
df['EMA'] = df['GS'].ewm(alpha = 0.3).mean()

#Price Channel
df['PCh_up'] = df['GS'].rolling(window=20).max()
df['PCh_dn'] = df['GS'].rolling(window=20).min()
df['PCc_mid'] = (df['PCh_up'] + df['PCh_dn']) / 2

#Bollinger bands (Линии Боллинджера)
ma20 = df['GS'].rolling(window=20).mean().to_numpy()
df['20STD'] = df['GS'].rolling(window=20).std()

df['upper_band'] = ma20 + 2*df['20STD']
df['lower_band'] = ma20 - 2*df['20STD']

#Ichimoku
IchimokuIndicatorData = IchimokuIndicator(df['High'], df['Low'], window1 = 9, window2 = 26, window3 = 52, fillna=True)
df['LeadingSpanA'] = IchimokuIndicatorData.ichimoku_a()
df['LeadingSpanB'] = IchimokuIndicatorData.ichimoku_b()
df['BaseLine'] = IchimokuIndicatorData.ichimoku_base_line()
df['ConversionLine'] = IchimokuIndicatorData.ichimoku_conversion_line()

# Mass Index
MassIndexData = MassIndex(high = df['High'], low = df['Low'], window_fast = 9, window_slow = 25, fillna=True)
df['MassIndex'] = MassIndexData.mass_index()

# TRIX Indicator
TRIXIndicatorData = TRIXIndicator(close= df['GS'], window = 15, fillna=True)




#Stochastic Oscillator
StochasticOscillatorData = StochasticOscillator(high = df['High'], low = df['Low'], close = df['GS'], \
                                    window = 14, smooth_window = 3, fillna = True)
df['Stoch'] = StochasticOscillatorData.stoch()
df['StochSignal'] = StochasticOscillatorData.stoch_signal()

#Relative Strength Index
df['rsi'] = rsi(df['GS'], window = 14, fillna=True)

#Moving Average Convergence/Divergence MACD
df['EMA12']= df['GS'].ewm(span = 12).mean()
df['EMA26']= df['GS'].ewm(span = 26).mean()

df['MACD'] = df['EMA12'] - df['EMA26']

# Previous day price
df['GSshift1'] = df['GS'].shift(1, fill_value = df['GS'].iloc[0])

#Rate of Change ROC
df['ROC7'] = (df['GS'].diff(7) / df['GS'].shift(7)) *100

#Momentum Oscillator
df['momentum'] = (df['GS'] / df['GS'].shift(7)) * 100




# Volume Weighted Average Price
VolumeWeightedAveragePriceData = VolumeWeightedAveragePrice(high = df['High'], low = df['Low'], close = df['GS'], \
                                        volume = df['Volume'], window = 14, fillna = True)
df['VWAP'] = VolumeWeightedAveragePriceData.volume_weighted_average_price()


# Money Flow Index (MFI)
df['MFI'] = money_flow_index(high = df['High'], low = df['Low'], close = df['GS'], \
                                        volume = df['Volume'], window = 14, fillna = True)
```

```
    return df


# застосування функції
df_ti = get_technical_indicators(df_gs)
df_ti.head(10)

# кількість пропущених значень
df_ti.isna().sum()
# візуалізація тех індикаторів
def plot_ti(df):

    fig = px.line(df, x=df.index, y = df.columns, title='Tech Indicators')
    fig.show()

plot_ti(df_ti[['GS','MA7', 'MA21','EMA']])
# візуалізація тех індикаторів
plot_ti(df_ti[['GS','PCh_up', 'PCh_dn', 'PCc_mid']])
# візуалізація тех індикаторів
plot_ti(df_ti[['GS','20STD', 'upper_band',
       'lower_band']])
# візуалізація тех індикаторів
plot_ti(df_ti[['GS','LeadingSpanA', 'LeadingSpanB', 'BaseLine','ConversionLine']])
# візуалізація тех індикаторів
plot_ti(df_ti[['GS','MassIndex', 'Stoch', 'StochSignal', 'rsi']])
# візуалізація тех індикаторів
plot_ti(df_ti[['GS','EMA12','EMA26', 'MACD']])
# візуалізація тех індикаторів
plot_ti(df_ti[['GS','ROC7', 'momentum', 'VWAP', 'MFI']])
### Fill Nan
# df_ti[df_ti.index<='2010-03-01'].plot(figsize=(10,5));
df_ti.isna().sum()
df_ti.info()
# колонки з пустими значеннями
cols_with_na = df_ti.isna().sum()[df_ti.isna().sum() != 0].index.tolist()
cols_with_na
# колонки з пустими значеннями
cols_with_na = df_ti.isna().sum()[df_ti.isna().sum() != 0].index.tolist()
cols_with_na
# візуалізація
filled_df[cols_with_na][filled_df.index<='2010-03-01'].plot(figsize=(10,5));
# збереження
filled_df.to_csv(PATH + '\data\pre_ready_data\\tech_info.csv')
```

Лістинг файлу 5parse_news.ipynb – парсинг новин з сайту

```
# імпорт бібліотек

import requests
from bs4 import BeautifulSoup

import pprint
import pandas as pd
import os

import time
# шлях до файлу
PATH = os.getcwd()
# !pip install beautifulsoup4
# функція що отримує контент за тегом з html розмітки

def get_content(soup, tag, class_):
    page_content = soup.find_all(tag, class_= class_) # пошук даних за тегом
    print('Page content quantity: ', len(page_content))

    dates = []
    headers = []
    for block in page_content:
```

```
        date = block.find('time').text # отримання часу новини
        dates.append(date)

        header = block.find('h4').text # отримання заголовку
        headers.append(header)


    print('Dates quantity: ', len(dates)) # кількість дат

    print('Headers quantity',len(headers)) # кількість заголовків



    return dates, headers

### Create archive news df
# запрос на веб-сторінку та отримання коду сторінки
source_archive_news = requests.get('https://www.goldmansachs.com/media-relations/in-the-news/archive/index.html').text
soup_archive_news = BeautifulSoup(source_archive_news, 'lxml') # ініціалізація парсера
# дати та заголовки новин
news_dates, news_headers = get_content(soup_archive_news, 'div', "page-feed__content")
Page content quantity:  302
Dates quantity:  302
Headers quantity 302
# сворення датафрейму
news_archive_df = pd.DataFrame({'Date': news_dates, 'News': news_headers})
news_archive_df['Date'] = pd.to_datetime(news_archive_df['Date'])
news_archive_df.set_index('Date', inplace=True)
news_archive_df
# кіль-ть пропущених значень
news_archive_df.isna().sum()
News    0
dtype: int64
news_archive_df.info()
### Create current news df
# запрос на веб-сторінку та отримання коду сторінки
source_current_news = requests.get('https://www.goldmansachs.com/media-relations/in-the-news/current/index.html').text
soup_current_news = BeautifulSoup(source_current_news, 'lxml')
# дати та заголовки новин
current_news_dates, current_news_headers = get_content(soup_current_news, 'div', "page-feed__content")
Page content quantity:  40
Dates quantity:  40
Headers quantity 40
# сворення датафрейму
news_current_df = pd.DataFrame({'Date': current_news_dates, 'News': current_news_headers})
news_current_df['Date'] = pd.to_datetime(news_current_df['Date'])
news_current_df.set_index('Date', inplace=True)
news_current_df
# поєднання архівних та актуальних новин
news_df = pd.concat([news_current_df, news_archive_df])
news_df
final_news_df = news_df[news_df.index >= '2010-01-01']
final_news_df
### Create archive press-release df
# запрос на веб-сторінку та отримання коду сторінки
source_press_rel = requests.get('https://www.goldmansachs.com/media-relations/press-releases-and-comments/current/index.html').text
soup_press_rel = BeautifulSoup(source_press_rel, 'lxml')
# дати та заголовки прес-релізів
press_rel_dates, press_rel_headers = get_content(soup, 'section', \
                                    "default-component page-feed__content article-content-page line-article-no-image")
Page content quantity:  656
Dates quantity:  656
Headers quantity 656
# створення датафрейму
press_rel_archive_df = pd.DataFrame({'Date': press_rel_dates, 'Press_rel': press_rel_headers})
press_rel_archive_df['Date'] = pd.to_datetime(press_rel_archive_df['Date'])
press_rel_archive_df.set_index('Date', inplace=True)
press_rel_archive_df
press_rel_archive_df.isna().sum()
final_press_rel_df = press_rel_archive_df[press_rel_archive_df.index >= '2010-01-01']
final_press_rel_df
### Financial Times News
# створення часових проміжків інтервалом в пів року
d1 = pd.date_range(start='2010-01-01', end='2021-05-01', freq='6MS').to_numpy().astype('datetime64[D]').astype('str').tolist()
```

```
d2 = pd.date_range(start='2010-06-30', end='2021-05-01', freq='6M').to_numpy().astype('datetime64[D]').astype('str').tolist()
d2.append('2021-05-01')

d_lst = list(zip(d1, d2))
d_lst
# функція яка виконує збір дати, заголовку та контексту новини

def parse_ft1(date_period):
    time_array = []
    header_array = []
    standfirst_array = []

    for period in date_period: # піврічний період
        print('Period: ', period)
        for number in range(1, 51): # максимаьна кількість новинна сторінці 50
            #print('Page number: ', number)
            # запит на сайт з новинами та ініціадізація парсера
            source = requests.get('https://www.ft.com/search?q=Goldman%20Sachs%20Group%20Inc&page={}&contentType=article&dateTo={}&dateFrom={}&sort=date&expandRefinements=true'.format(number, period[1], period[0])).text
            soup = BeautifulSoup(source, 'lxml')

            time.sleep(2)

            page_content = soup.find_all('div', class_= 'o-teaser__content') # пошук контента в html коді за класом

            for content in page_content:
                date = content.find('div', class_ = 'o-teaser__timestamp').text # час новини
                heading = content.find('div', class_ = 'o-teaser__heading').text # заголовок новини
                standfirst = content.find('p', class_ = 'o-teaser__standfirst') # контекст новини
                if standfirst == None:
                    standfirst = ' '
                else:
                    standfirst = standfirst.text

                time_array.append(date), header_array.append(heading), standfirst_array.append(standfirst)


            if number%10 == 0:
                print('Page number: ', number)

                print('Length of time array: ', len(time_array))
                print('Length of header array: ', len(header_array))
                print('Length of news array: ', len(standfirst_array))
            else:
                continue

    return time_array, header_array, standfirst_array

time_array, header_array, standfirst_array = parse_ft1(d_lst)
# формування датасету
ft_df = pd.DataFrame({'Date': time_array, 'Heading': header_array, 'Standfirst': standfirst_array})
ft_df
# приведення дати в формат datetime
ft_df['Date'] = pd.to_datetime(ft_df['Date'])
ft_df.sort_values(by='Date', inplace=True)
ft_df
# збереження
# ft_df.to_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\pre_ready_data\GSnewsFT.csv', index=False)
```

Лістинг файлу 6BERT_classification.ipynb – класифікація новин finBERT

```
# імпорт бібліотек

from transformers import BertTokenizer, BertForSequenceClassification
import torch

import pandas as pd
```

```
### Prepare grouped data
# завантаження даних
df_news = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\pre_ready_data\GSnewsFT.csv')
# кількість пропущених даних
df_news.isna().sum()
df_news
#### Clean text
df_proc = df_news.copy()
cols = ['Heading', 'Standfirst']
# очищення даних

for col in cols:
    df_proc[col].replace('[^\w\s]', '', regex=True, inplace=True) # видалення всіх символів окрім букв та чисел
    df_proc[col] = df_proc[col].str.lower() # переведення в нижній регістр
    df_proc[col] = df_proc[col].str.split() # видалення пробілів
    df_proc['Standfirst'][df_proc['Standfirst'].str.len() == 1] = ' ' # видалення одиничних слів

    df_proc[col] = df_proc[col].str.join(' ')
df_proc
# конкатенація заголовку та основного змісту новини
df_proc['News'] = df_proc['Heading'] + ' ' + df_proc['Standfirst']
df_proc
#### Group text by date
df_proc_group = df_proc.copy()
df_proc_group = df_proc_group.groupby(['Date'])['News'].apply(' '.join).reset_index()
df_proc_group
### finBERT classification
# токенізатор base-uncased
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
# модель для класифікації
model = BertForSequenceClassification.from_pretrained(r'C:\Users\adima\Desktop\thesis\models\sentiment\finbert\pytorch_model.bin', \
                                          config=r'C:\Users\adima\Desktop\thesis\models\sentiment\finbert\config.json', \
                                          num_labels = 3)
label_list=['positive','negative','neutral']

# класифікація новин
def get_sentiment(new):

    inputs = tokenizer(new, return_tensors="pt", truncation = True, max_length =512)
    outputs = model(**inputs)

    maxlogit = float(torch.max(outputs[0]))
    label = label_list[torch.argmax(outputs[0])]


    return maxlogit, label
%%time
sentiment_tuple = df_proc_group['News'].apply(get_sentiment)
Wall time: 42min 4s
sent_df = pd.DataFrame(sentiment_tuple.tolist(), columns=['maxlogit', 'label'])
sent_df
df_sent_classified = pd.concat([df_proc_group, sent_df], axis=1)
df_sent_classified.set_index('Date', inplace=True)
df_sent_classified
# df_sent_classified.to_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\pre_ready_data\sentiment_classified_news.csv')
```

Лістинг файлу 7FFTransform_and_ARIMA.ipynb – спектральний аналіз

```
# імпорт бібліотек

import pandas as pd
import numpy as np

from sklearn.metrics import mean_squared_error

import plotly.express as px
```

```
import warnings
warnings.filterwarnings('ignore')
# завантаження даних
df = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\pre_ready_data\GSready.csv')
df = df[['Date', 'GS']]
df
### Fourier Transformation
# функція перетворень Фур'є

def fourier_transform(df):
    fft_arr = np.fft.fft(np.array(df['GS'].tolist())) # швидке перетворення Фур'є
    fft_df = pd.DataFrame()
    fft_df['fft'] = fft_arr
    fft_df['amplitude'] = np.abs(fft_arr) # амплітуда
    fft_df['phase'] = np.angle(fft_arr) # фаза
    for n in [3, 6, 9, 20, 100]:
        fft_smth = fft_arr.copy()
        fft_smth[n : -n] = 0  #limit freq
        fft_df['inv_fft' + str(n)] = np.fft.ifft(fft_smth) # зворотнє перетворення
        fft_df['fft'+ str(n)+'amplitude'] = np.abs(fft_smth) # амплітуда
        fft_df['fft'+ str(n)+'phase'] = np.angle(fft_smth) # фаза

    return fft_df

fft_df = fourier_transform(df)
fft_df
# зведення в один датасет
final_fft_df = pd.concat([df, fft_df], axis=1)
final_fft_df.set_index('Date', inplace=True)
final_fft_df
# візуалізація
final_fft_df[['GS']+['inv_fft3']+['inv_fft6']+['inv_fft9']+['inv_fft20']+['inv_fft100']].plot(figsize=(18,7));
# final_fft_df.to_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\pre_ready_data\fft_data.csv')
```

Лістинг файлу 8Merge_dataframes.ipynb – зведення всіх даних в один датасет

```
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
df= pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\pre_ready_data\ready_df_with_nan.csv')
df
df.isna().sum()
### Fill NaN with interpolation and remove outliers
df_ = df.copy()
df_.columns
def remove_outliers(df):
    Q1 = df.quantile(0.2)
    Q3 = df.quantile(0.8)
    IQR = Q3 - Q1
    return df[(df >= (Q1 - 1.5 * IQR)) & (df  <= (Q3 + 1.5 * IQR))]
#### Commodities and companies
cols1= ['BZ=F_oil', 'CL=F_oil', 'GLD', 'SI=F', 'AAPL', 'BAC', 'C', 'CS', 'DB', 'HRI', 'JPM', 'LYG', 'MAN', 'MS', 'MUFG', 'UBS']
comm_cols = ['BZ=F_oil', 'CL=F_oil', 'GLD', 'SI=F']
comp_cols = ['AAPL', 'BAC', 'C', 'CS', 'DB', 'HRI', 'JPM', 'LYG', 'MAN', 'MS', 'MUFG', 'UBS']
df_[comm_cols].plot(figsize=(18,7));
df_[comm_cols] = remove_outliers(df_[comm_cols])
df_[comm_cols] = df_[comm_cols].interpolate(method='linear')

df_[comm_cols].plot(figsize=(18,7));
df_[comp_cols].plot(figsize=(18,7));
```

```
#df_[comp_cols] = remove_outliers(df_[comp_cols])
df_[comp_cols] = df_[comp_cols].interpolate(method='linear')

df_[comp_cols].plot(figsize=(18,7));
df_.info()
df_['GSBD(march-19-2015)'][(df_['Date']<'2015-03-25') & (df_['Date']>'2015-03-10')]
df_['GSBD(march-19-2015)'][1900:] = df_['GSBD(march-19-2015)'][1900:].interpolate(method='linear')
df_['GSBD(march-19-2015)'][1900:]
df_['GSBD(march-19-2015)'].fillna(df_['GSBD(march-19-2015)'].mean(), inplace=True)
df_['GSBD(march-19-2015)'].plot(figsize=(18,7));
#### FX
fx_cols = ['CNYUSD=X', 'EURUSD=X', 'GBPUSD=X', 'JPYUSD=X', 'RUBUSD=X']
df_[fx_cols].plot(figsize=(18,7));
df_[fx_cols] = remove_outliers(df_[fx_cols])
df_[fx_cols] = df_[fx_cols].interpolate(method='linear')

df_[fx_cols].plot(figsize=(18,7));
#### Indicators
indicator_cols = ['10YBIR', 'DFF', 'LIBOR', '10YearBondUS', 'TNX',
                  'VIX', 'CPALTT01USM657N', 'M2SL', 'MEDCPIM158SFRBCLE', 'PPIACO',
                  'UNRATE', 'GDP', 'GDPC1']
daily_indicator_cols = ['10YBIR', 'DFF', 'TNX', 'VIX', 'LIBOR', '10YearBondUS']
monthly_indicator_cols = ['CPALTT01USM657N', 'M2SL','MEDCPIM158SFRBCLE','PPIACO', 'UNRATE' ]
quarterly_indicator_cols = ['GDP', 'GDPC1']

df_[indicator_cols].info()
df_['LIBOR'] = pd.to_numeric(df_['LIBOR'],errors='coerce')
df_[quarterly_indicator_cols].plot(figsize=(18,7));
df_[quarterly_indicator_cols] = df_[quarterly_indicator_cols].fillna(method='ffill')
df_[quarterly_indicator_cols].plot(figsize=(18,7));
#### Indicies
index_cols = ['DJIA', 'FTSE', 'HSI', 'IXIC', 'N225', 'NYA',
      'RUT', 'S&P500']
df_[index_cols].info()
df_['DJIA'] = pd.to_numeric(df_['DJIA'].str.replace('.','').str.replace(',','.'), errors='coerce')
df_['S&P500'] = pd.to_numeric(df_['S&P500'].str.replace('.','').str.replace(',','.'), errors='coerce')
df_[index_cols].info()
df_[index_cols].plot(figsize=(18,7));
df_[index_cols] = remove_outliers(df_[index_cols])
df_[index_cols] = df_[index_cols].interpolate(method='linear')

df_[index_cols].plot(figsize=(18,7));
# df_.to_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\pre_ready_data\filled_df19-05.csv',index=False)
df_['Date'] = pd.to_datetime(df_['Date'])
df_.set_index('Date', inplace=True)
df_
### Merge dataframes
#### Technical Indicators
tech_ind_df = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\pre_ready_data\tech_info.csv', index_col='Date')
tech_ind_df
tech_ind_df.drop('GS', axis=1, inplace=True)
df_merged_ind = pd.merge(df_, tech_ind_df, how='left', left_index=True, right_index=True)
df_merged_ind
#### Google Trends
google_trends_df = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\google_trends\GSGoogleTrends_final.csv', \
                        index_col='date')
google_trends_df
google_trends_df['adjusted_data'].plot(figsize=(18,7));
remove_outliers(google_trends_df['adjusted_data']).isna().sum()
0
google_trends_df['adjusted_data'] = remove_outliers(google_trends_df['adjusted_data'])
df_merged_gt = pd.merge(df_merged_ind, google_trends_df[['adjusted_data']], how='left', left_index=True, right_index=True)
df_merged_gt.rename({'adjusted_data':'google_trends'}, axis=1, inplace=True)
df_merged_gt
df_merged_gt['google_trends'].isna().sum()
25
df_merged_gt['google_trends'] = df_merged_gt['google_trends'].interpolate(method = 'linear')
df_merged_gt['google_trends'].isna().sum()
0
df_merged_gt['google_trends'].iloc[:500].plot(figsize=(18,7))
#### Fourier Transform
fft_df =  pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\pre_ready_data\fft_data.csv', \
                        index_col='Date')
fft_df
fft_df.info()
```

```
fft_df.columns
Index(['GS', 'fft', 'amplitude', 'phase', 'inv_fft3', 'fft3amplitude',
       'fft3phase', 'inv_fft6', 'fft6amplitude', 'fft6phase', 'inv_fft9',
       'fft9amplitude', 'fft9phase', 'inv_fft20', 'fft20amplitude',
       'fft20phase', 'inv_fft100', 'fft100amplitude', 'fft100phase'],
      dtype='object')
fft_df.drop(['fft', 'GS', 'amplitude', 'phase',
           'fft3amplitude', 'fft3phase', 'fft6amplitude', 'fft6phase',
       'fft9amplitude', 'fft9phase','fft20amplitude',
       'fft20phase', 'fft100amplitude', 'fft100phase'], axis=1, inplace=True)
compl_obj_cols = fft_df.select_dtypes('object').columns
fft_df[compl_obj_cols]= fft_df.select_dtypes('object').astype(np.complex).apply(lambda x: np.real(x))
fft_df
fft_df[['inv_fft3']+['inv_fft6']+['inv_fft9']+['inv_fft20']+['inv_fft100']].plot(figsize=(18,7));
df_merged_fft = pd.merge(df_merged_gt, fft_df, how='left', left_index=True, right_index=True)
df_merged_fft
#### Sentiment classified news
bert_df = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\pre_ready_data\sentiment_classified_news.csv', \
                      index_col='Date')
bert_df
df_merged_bert = pd.merge(df_merged_fft, bert_df, how='left', left_index=True, right_index=True)
df_merged_bert
df_merged_bert.drop('News', axis=1,inplace=True)
df_merged_bert['maxlogit'].fillna(0, inplace=True)
df_merged_bert['label'].fillna('neutral', inplace=True)
df_merged_bert
df_merged_bert.rename({'label':'sentiment'}, axis=1, inplace=True)
ready_df = df_merged_bert.copy()
# ready_df.to_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\ready_data.csv')

gs = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\GSready.csv', index_col='Date')
gs
ready_df = pd.merge(ready_df, gs, how='left', left_index=True, right_index=True)

ready_df['google_trends'].iloc[:500].plot(figsize=(18,7));
```

Лістинг файлу 9Statistical_check.ipynb – аналіз даних

```
# імпорт бібліотек
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objects as go

%matplotlib inline

from sklearn.preprocessing import LabelEncoder

import statsmodels.formula.api as smf
from statsmodels.tsa.stattools import adfuller
from scipy.stats import shapiro

import warnings
warnings.filterwarnings('ignore')
# завантаження даних
df = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\ready_data.csv', index_col='Date')
# описательные статистики
df.describe()
# кореляція
corr = df.corr()
corr['GS']
### EDA
# розмірність
df.shape
(4106, 83)
# кількість пропущених значень
```

```
df.isna().sum().sum()
0
cat_vals = df['sentiment'].value_counts()
cat_vals

# barchart кількість нейтральних, негативних та позитивних новин
x = cat_vals.index
y = cat_vals.values
fig = go.Figure(data=[go.Bar(
            x=x, y=y,
            text=y,
            textposition='auto',
        )])

fig.update_layout(
    title="Sentiment Score Values",
    xaxis_title="Sentiment",
    yaxis_title="Quantity",
    autosize=False)
fig.show()
# кореляційна матриця
# дані компаній
corr_cols = df.columns.tolist()
plt.figure(figsize = (16,16))
sns.heatmap(df[corr_cols[0:20] + ['GS']].corr(), annot=True); # дані валют та економічних індексів
plt.figure(figsize = (16,16))
sns.heatmap(df[corr_cols[20:36] + ['GS']].corr(), annot=True);
# дати біржевих індексів та ТА
plt.figure(figsize = (16,16))
sns.heatmap(df[corr_cols[36:57] + ['GS']].corr(), annot=True);
# дані ТА та ШПФ
plt.figure(figsize = (16,16))
sns.heatmap(df[corr_cols[57:]].corr(), annot=True);
# колонки за типом
comm_cols = ['BZ=F_oil', 'CL=F_oil', 'GLD', 'SI=F']
comp_cols = ['AAPL', 'BAC', 'C', 'CS', 'DB', 'GSBD(march-19-2015)', 'HRI', 'JPM', 'LYG', 'MAN', 'MS', 'MUFG', 'UBS']
higher_fx_cols = ['EURUSD=X', 'GBPUSD=X']
lower_fx_cols = ['CNYUSD=X', 'JPYUSD=X', 'RUBUSD=X']
economic_ind_cols_small = ['10YBIR', 'DFF', 'LIBOR', '10YearBondUS', 'TNX',
      'VIX', 'CPALTT01USM657N',  'MEDCPIM158SFRBCLE', 'PPIACO',
      'UNRATE']
economic_ind_cols_big = ['M2SL','GDP', 'GDPC1']
stock_ind_cols = ['DJIA', 'FTSE', 'HSI', 'IXIC', 'N225', 'NYA',
      'RUT', 'S&P500']

data_list = [comm_cols, comp_cols, higher_fx_cols, lower_fx_cols, economic_ind_cols_small,
           economic_ind_cols_big, stock_ind_cols]
correlation_assets = ['Commodities', 'Company Stocks', 'Hihger USD Foreign Exchange','Lower USD Foreign Exchange',
                  'Economic Indicators Small Values','Economic Indicators Big Values', 'Stock Indices']
correlation_assets_dict = dict(zip(correlation_assets, data_list))
# функція boxplots
def disp_boxplots(dict_):
    n = len(dict_)
    fig, axes = plt.subplots(n, 1, figsize=(16, 8*n))
    for i in range(n):
        sns.boxplot(ax = axes[i], x="variable", y="value", \
                data=pd.melt(df[dict_[list(dict_.keys())[i]]]))
        axes[i].set_title(list(dict_.keys())[i])
# бокcплоти
disp_boxplots(correlation_assets_dict)
```

Лістинг файлу 2pca_lstm.ipynb – підготовка даних та навчання моделі

```
# імпорт бібліотек

import pandas as pd
import numpy as np
```

```
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.decomposition import PCA

from sklearn.metrics import mean_absolute_error , mean_squared_error, r2_score
from scipy.stats.stats import pearsonr

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Activation, Dropout
from tensorflow.keras.optimizers import Adam, SGD
#from tensorflow.keras.regularizers import l2
#from tensorflow.keras.models import save_model
#from tensorflow.keras import regularizers

from statsmodels.tsa.stattools import adfuller
from scipy.stats import shapiro

import matplotlib.pyplot as plt

from tensorflow.keras.models import load_model

# перевірка версії Numpy
np.__version__
'1.19.5'
# завантаження даних
df = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\ready_data.csv', index_col = 'Date')
df.head()
df.drop('Adj Close', axis = 1, inplace=True)
df.columns
### Label Encoding
# кодування настроїв новин від 0 до 2
label_encoder = LabelEncoder()
df['sentiment'] = label_encoder.fit_transform(df['sentiment'])
for ind, x in enumerate(label_encoder.classes_):
    print(ind, x)
### Train test split
# дати поділу вибірки
first_timestamp = '2018-01-01' # тренувальні - валідаційні
second_timestamp = '2019-09-01'  # валідаційні - тестові
# поділ датасету на тренувальні+валідаційні та тестові
train_val_dataset = df[df.index < '2019-09-01']
test_dataset = df[df.index >= '2019-09-01']

train_val_dataset.shape, test_dataset.shape
((3527, 82), (579, 82))
# довжина вчідного вектора та передбачуваного вектора
n_steps_in, n_steps_out = 60 , 30
# подовження тестової вибірки
test_dataset_extended = train_val_dataset.tail(n_steps_in-1)
test_dataset_extended = test_dataset_extended.append(test_dataset, ignore_index = False)

train_val_dataset.shape, test_dataset_extended.shape
((3527, 82), (638, 82))
test_dataset_extended.head(2)
### Standartization
# всі колонки крім категоріальної та сили емоції
cols_to_transform = np.asarray(list(filter(lambda x: x !='maxlogit' and x != 'sentiment', np.asarray(df.columns))))
train_val_dataset_scale = train_val_dataset.copy()
test_dataset_scale = test_dataset_extended.copy()

# стандартизація
standard_scaler = StandardScaler()


# тренуваьна та тестова вибірки окремо
train_val_dataset_scale[cols_to_transform] = standard_scaler.fit_transform(train_val_dataset_scale[cols_to_transform])
test_dataset_scale[cols_to_transform] = standard_scaler.transform(test_dataset_scale[cols_to_transform])

train_val_dataset_scale.head(2)
### PCA
# функція побудови діаграми розсіювання
def plot_scatter(df, x, y):
    plt.figure(figsize=(8,6))
    plt.scatter(df[x], df[y], alpha=0.3, s=10)
    plt.xlabel(x)
    plt.ylabel(y)
```

```
    plt.grid()
    plt.show()
# діаграма розсіювання
plot_scatter(train_val_dataset_scale, 'BZ=F_oil', 'CL=F_oil')
# метод головних компонент
pca = PCA()
# підбор оптимальної розмірності
X_pca = pca.fit(train_val_dataset_scale[train_val_dataset_scale.columns[:-3]])
# графік підбору оптимальної розмірності
plt.figure(figsize=(7, 5))
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of components')
plt.ylabel('Cumulative explained variance')
plt.grid()
plt.show()
# обрана кількість ознак
num_components = 30
# метод гол. комп.
pca = PCA(num_components)
# тренуальна та тестова вибірки окремо
X_train_val_pca = pca.fit_transform(train_val_dataset_scale[train_val_dataset_scale.columns[:-3]])
X_test_pca = pca.transform(test_dataset_scale[test_dataset_scale.columns[:-3]])
# початкова розмірнвсть
X_train_val_pca.shape, X_test_pca.shape
((3527, 30), (638, 30))
n_pcs= pca.n_components_ # get number of component
# get the index of the most important feature on EACH component
most_important = [np.abs(pca.components_[i]).argmax() for i in range(n_pcs)]
initial_feature_names = df.columns
# get the most important feature names
most_important_names = [initial_feature_names[most_important[i]] for i in range(n_pcs)]
np.array(most_important_names)
# отримана розмірніть
pca_train_val_df, pca_test_df = pd.DataFrame(X_train_val_pca), pd.DataFrame(X_test_pca)

pca_train_val_df.set_index(train_val_dataset_scale.index, inplace=True)
pca_test_df.set_index(test_dataset_scale.index, inplace=True)

# з'єднання датасетів
pca_train_val_df = pd.merge(pca_train_val_df, train_val_dataset_scale[train_val_dataset_scale.columns[-3:]], \
                        how= 'right', left_index=True, right_index=True)
pca_test_df = pd.merge(pca_test_df, test_dataset_scale[test_dataset_scale.columns[-3:]], \
                    how= 'right', left_index=True, right_index=True)

pca_train_val_df.shape, pca_test_df.shape

### Stationarization
# колонки для стаціонаризації
cols_to_stationarize = list(filter(lambda x: x !='maxlogit' and x != 'sentiment', np.asarray(pca_train_val_df.columns)))
stationary_train_val_df = pca_train_val_df.copy()
stationary_test_df = pca_train_val_df.tail(1).append(pca_test_df, ignore_index=False)

# навчальна та тестова вибірки окремо
stationary_train_val_df[cols_to_stationarize] = stationary_train_val_df[cols_to_stationarize].diff()
stationary_train_val_df.fillna(method='bfill', inplace=True)

stationary_test_df[cols_to_stationarize] = stationary_test_df[cols_to_stationarize].diff()
stationary_test_df = stationary_test_df[1:]

# отримана розмірність
stationary_train_val_df.shape, stationary_test_df.shape
((3527, 33), (638, 33))
# кількітсть пропущених значень
stationary_train_val_df.isna().sum().sum(), stationary_test_df.isna().sum().sum()
(0, 0)
# візуалізація нової ознаки
stationary_train_val_df[1].plot(figsize=(17,6));
# візуалізація нової ознаки
stationary_test_df[1].plot(figsize=(17,6));
### Testing data
# тест Дікі-Фуллера для кожної ознаки на стаціонарність
def get_adfuller(df, check_stat_cols):
    res = []
    for col in check_stat_cols:
        dftest = adfuller(df[col])
```

```
        pval = dftest[1]
        if pval <= 0.05: # не приймаємо H0 про нестаціонарність -- ряд стаціонарний
            res.append('stationary')
        else:           # приймаємо H0 -- ряд нестаціонарний
            res.append('non-stationary')
    adfuller_result_df = pd.DataFrame({'Columns': check_stat_cols, 'Result': res})

    return adfuller_result_df
# результат тестування по кожній ознаці
adfuller_result_df = get_adfuller(stationary_train_val_df, cols_to_stationarize)
adfuller_result_df
# кількість стаціонарних/нестаціонарних
adfuller_result_df.groupby(['Result']).count()
###  Create data for NN
first_timestamp, second_timestamp
('2018-01-01', '2019-09-01')
# поділ на:
train_dataset  = stationary_train_val_df[stationary_train_val_df.index < '2018-01-01'] # тренувальні
val_dataset = stationary_train_val_df[(stationary_train_val_df.index >= '2018-01-01')] # валідаційні
test_dataset_extended = stationary_test_df # тестові
# розмірність
train_dataset.shape, val_dataset.shape, test_dataset_extended.shape
((2919, 33), (608, 33), (638, 33))
# довжина (кількість) пояснюючих і пояснбваних змінних
n_steps_in, n_steps_out
(60, 30)
# розширення валідаційного датасету
val_dataset_extended = train_dataset.tail(n_steps_in-1)
val_dataset_extended = val_dataset_extended.append(val_dataset, ignore_index = False)
# розмірність
train_dataset.shape, val_dataset_extended.shape, test_dataset_extended.shape
((2919, 33), (667, 33), (638, 33))
# перетворення на масиви numpy
train_dataset_arr = train_dataset.to_numpy()
val_dataset_ext_arr = val_dataset_extended.to_numpy()
test_dataset_ext_arr = test_dataset_extended.to_numpy()
# зсув для формування вхідних та прогнозованих даних
def split_sequences(sequences, n_steps_in, n_steps_out):
    X, y = list(), list()
    n=1
    for i in range(len(sequences)):
        # знаходження кінця послідовності
        end_ix = i + n_steps_in
        out_end_ix = end_ix + n_steps_out-1

        # перевірка, чи не вийшов алг за межі датасета
        if out_end_ix > len(sequences):
            break
        # збір вхідних та вихідних даних
        seq_x, seq_y = sequences[i:end_ix, :-1], sequences[end_ix-1:out_end_ix, -1]

        X.append(seq_x)
        y.append(seq_y)

        n+=1

    print('Len ready arr', n)
    print('Len sequance', len(sequences))
    print('----------')


    return np.array(X), np.array(y)
# приведення даних до нового виду
X_train, y_train = split_sequences(train_dataset_arr, n_steps_in, n_steps_out) # тренувальні
X_val, y_val = split_sequences(val_dataset_ext_arr, n_steps_in, n_steps_out)# валідаційні
X_test, y_test = split_sequences(test_dataset_ext_arr, n_steps_in, n_steps_out)# тестові
# розмірність даних
print('X_train shape =', X_train.shape, '; Y train shape =', y_train.shape)
print('X_val shape =', X_val.shape, '; Y val shape =', y_val.shape)
print('X_test shape =', X_test.shape, '; Y test shape =', y_test.shape)
n_features = X_train.shape[2]
# довжини послідовностей
print('N input length =', n_steps_in)
print('N output length =', n_steps_out)
print('N features = ', n_features)
```

```
#### Save data
# шлях зберігання
outfile = r'C:\Users\adima\Desktop\thesis\data_overview\data\train_val_test.npz'
# зберігання
np.savez(outfile, X_train = X_train, y_train = y_train, \
                 X_val = X_val, y_val = y_val, \
                  X_test = X_test, y_test = y_test)
# завантаження
npzfile = np.load(outfile)
npzfile.files
### LSTM Building
#optimizer

opt = Adam(learning_rate = 0.0001)

#model

model = Sequential()

model.add(LSTM(120, return_sequences=True, input_shape=(n_steps_in, n_features)))#перший шар
model.add(Dropout(0.2))# дропаут від перенавчання

model.add(LSTM(120))# другий шар
model.add(Dropout(0.2))# дропаут від перенавчання

model.add(Dense(n_steps_out))
model.add(Activation('linear')) # для рішення проблеми регресії - ф акт лінійна

model.compile(loss='mse' , optimizer=opt , metrics=['mse']) # компіляція моделі
# опис
model.summary()
# навчання нейромережі
   history = model.fit(X_train , y_train , epochs=100, batch_size=32, \
                steps_per_epoch=25 , verbose=1 , validation_data=(X_val, y_val) , shuffle=False)
history.history.keys()
def plot_error(train_err, val_error):
   plt.plot(history.history[train_err])
   plt.plot(history.history[val_error])
   plt.title('model ' + train_err)
   plt.ylabel(train_err)
   plt.xlabel('epoch')
   plt.legend(['train', 'val'], loc='upper right')
   plt.grid()
   plt.show()
# прогнозування
y_pred = model_.predict(X_test)
# позмірність вихідних даних
y_pred.shape
(550, 30)
# last_train_val
last_train_val = pca_test_df['GS'].loc['2019-08-31']

# функція зворотної стандартизації
def invTransform(scaler, data):
   data = data.to_numpy()
   dummy = pd.DataFrame(np.zeros((len(data), scaler.n_features_in_)))
   dummy[dummy.columns[-1]] = data

   return scaler.inverse_transform(dummy)[:, -1]

# формування результатів роботи
def stack_output_test(y_pred, y_test):
   arr_pred = [block[0] for block in y_pred] # пониження розмірності, перехід до часового ряду
   arr_test = [block[0] for block in y_test]
   arr_pred, arr_test = np.array(arr_pred), np.array(arr_test)
   arr_pred, arr_test = np.append(arr_pred, y_pred[-1]), np.append(arr_test, y_test[-1])

   if len(arr_pred) == len(arr_test):
      print('Ok')
   else:
      print('Doesnt match')

   pred_df = pd.DataFrame(data = {'Predicted':arr_pred , 'Real': arr_test}, \
                     index = pd.to_datetime(np.arange(second_timestamp, len(arr_test), dtype = 'datetime64[D]')))
```

```
    # зворотня стаціонаризація
    pred_df['Predicted_non_stationary'] = np.r_[last_train_val, pred_df['Predicted']].cumsum()[1:]
    pred_df['Real_non_stationary'] = np.r_[last_train_val, pred_df['Real']].cumsum()[1:]

    # зворотня стандартизація
    pred_df['Predicted_inverse'] = invTransform(standard_scaler, pred_df['Predicted_non_stationary'])
    pred_df['Real_inverse'] = invTransform(standard_scaler, pred_df['Real_non_stationary'])


    return pred_df
# результуючий датасет
pred_df = stack_output_test(y_pred, y_test)
pred_df
pred_df[['Predicted', 'Real']].plot(figsize=(19,7));
pred_df[['Predicted_non_stationary', 'Real_non_stationary']].plot(figsize=(19,7));

pred_df[['Predicted_inverse', 'Real_inverse']].plot(figsize=(19,7));
# зберігання моделі
model.save('model120120.h5')
model_ = load_model('model120120.h5')
# зберігання отриманих даних
a = pred_df.copy()
a = a.reset_index()
a.rename({'index':'Date'}, axis=1, inplace=True)
a.to_csv(r'C:\Users\adima\Desktop\thesis\project\data\predictions.csv', index=False)
# метрики
res_df = pd.DataFrame()
res_df['Metrics'] = ['MSE', 'MAE', 'R2', 'Cov', 'Rearson_corr']
def score_metr(real, pred):
    mse = mean_squared_error(real, pred) # mean_squared_error
    mae = mean_absolute_error(real, pred) # mean_absolute_error
    r2 = r2_score(real, pred) # r2_score
    cov = np.cov(real, pred)[0][1] # клваріація
    corr = pearsonr(real, pred)[0] # кореляція Пірсона

    return [mse, mae, r2, cov, corr]
res_df['Stat_Stand_Score'] = score_metr(pred_df['Real'], pred_df['Predicted'])
res_df['Final_Score'] = score_metr(pred_df['Real_inverse'], pred_df['Predicted_inverse'])

res_df
```

Лістинг файлу app.py – розроблений веб-застосунок

```
import dash
import dash_core_components as dcc
import dash_html_components as html
import dash_bootstrap_components as dbc
from dash.dependencies import Input, Output
import pandas as pd
import numpy as np

import plotly.graph_objs as go
import plotly.express as px

import warnings
warnings.filterwarnings('ignore')

# ------------------------------------------------------

BS = dbc.themes.CYBORG

app = dash.Dash(__name__, external_stylesheets=[BS])


# ------------import and set dataframes-------------------


last_train_val = float(0.6671207107686448)
```

```
first_timestamp = '2018-01-01'
second_timestamp = '2019-09-01'
# outfile = r'C:\Users\adima\Desktop\thesis\data_overview\data\train_val_test.npz'
#
# npzfile = np.load(outfile)
# print(npzfile.files)
#
# X_train = npzfile['X_train']
# y_train = npzfile['y_train']
# X_val = npzfile['X_val']
# y_val = npzfile['y_val']
# X_test = npzfile['X_test']
# y_test = npzfile['y_test']

ti_df = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\pre_ready_data\tech_info.csv', index_col='Date')
ti_df = ti_df[ti_df.index<=second_timestamp]


ind_list = ['GS', 'MA7', 'MA21',
      'EMA', 'PCh_up', 'PCh_dn','20STD', 'upper_band',
      'lower_band', 'LeadingSpanA', 'LeadingSpanB', 'BaseLine',
      'ConversionLine', 'MassIndex', 'Stoch', 'StochSignal', 'rsi', 'EMA12',
      'EMA26', 'MACD','ROC7', 'momentum', 'VWAP', 'MFI']

pred_df = pd.read_csv(r'C:\Users\adima\Desktop\thesis\project\data\predictions.csv', index_col='Date')
stat_train = pd.read_csv(r'C:\Users\adima\Desktop\thesis\project\data\stationaty_train.csv', index_col='Date')
df = pd.read_csv(r'C:\Users\adima\Desktop\thesis\data_overview\data\ready_data.csv', index_col = 'Date')

gs_train_stat = stat_train[['GS']]
gs_real = df[['GS']]

# print(pred_df)
# print(gs_train_stat)
# print(gs_real)



# ----------------------------------------------------------

def plt3():
   trace0 = go.Scatter(
      x=gs_real.index, y=gs_real['GS'], name= 'Real', marker ={'color': 'white'} ##1f77b4
         )

   trace1 = go.Scatter(
         x=pred_df.index, y= pred_df['Predicted_inverse'], name='Predicted', marker ={'color': '#099632'} #ff7f0e
      )

   # trace2 = go.Scatter(
   #       x=df2_test_.index, y=df2_test_['BEER_PROD'], name= 'Beer test part', marker ={'color': 'white'}
   #    )



   data= [trace0, trace1]
      # data =[]
   layout = go.Layout(
         title="Stock prediction Graph from "+second_timestamp,
         height=700,
      )

   fig = go.Figure(data=data, layout=layout)



   fig.add_shape(type="line",
               # xref="x",
               # yref="paper",
               x0=pred_df.index[0],
               y0=-15,
               x1=pred_df.index[0],
               y1=360,
               line=dict(color="red", width=1, ),
               # fillcolor=fillcolor,
               # layer=layer
```

```
            )

    fig.update_xaxes(rangeslider_visible=True)

    fig.update_layout(
            xaxis_title="Time",
            yaxis_title="Value",
            template='plotly_dark'
    )
    return fig

# ---------------------------------------------------------


app.layout = html.Div([

    html.H1("Stock Price Prediction", style={'margin': '1% 0 1% 7.5%'}),
    html.Hr(style={'background-color': 'grey', 'size': '1px', 'width': '85%'}),

        html.H5("Techncal Indicatiors till "+second_timestamp, style={'margin': '1.5% 0 1% 8.4%'}),

            dbc.Row([
                dbc.Col(dbc.Label('Choose tech indicator'),
                        width={'size': 3, 'offset': 2})

                # dbc.Col(dbc.Label('Smooth type'),
                #         width={'size': 3, 'offset': 2}),
            ]),

            dbc.Row([
                dbc.Col(

                    dcc.Dropdown(id='feature_drop1',
                                 options=[
                                     {'label': i, 'value': i} for i in ind_list
                                 ],
                                 value=['GS'],
                                 style = {"background-color": "#1A1A1A", 'margin': '0 0 5% 0', 'color': 'black'},
                                 multi=True

                                 ),
                    width={'size': 3, 'offset': 2}
                )

            ]),

        dbc.Row([

            dbc.Col(dcc.Graph(id='graph1'),
                # width=10, lg={'size': 10, 'offset': 1}
                width={'size': 10, 'offset': 1}
                    )

        ]),

    dbc.Row([

        dbc.Col(dbc.Table.from_dataframe(ti_df[['Open', 'High', 'Low', 'GS', 'Adj Close', 'Volume']].head(5), striped=True, bordered=True,
hover=True),
                width={'size': 10, 'offset': 1}
                # width=10, lg={'size': 10, 'offset': 1}
                )

    ]),

    html.Hr(style={'background-color': 'grey', 'size': '1px', 'width': '83%'}),
    html.H5("Goldman Sachs Stock Prediction", style={'margin': '1.5% 0 1% 8.4%'}),

    dbc.Row([

            dbc.Col(dcc.Graph(id='graph3', figure=plt3()),
                # width=10, lg={'size': 10, 'offset': 1}
                width={'size': 10, 'offset': 1}
                    )
```

```
        ])
])


@app.callback(
    Output('graph1', 'figure'),
    [Input('feature_drop1', 'value')])
def plt1(feature_drop1):
    #cols = feature_drop1

    data = []

    for i in feature_drop1:
        data.append(go.Scatter(
            x=ti_df.index, y=ti_df[i], name=i
        ))

    layout = go.Layout(
        title="Goldman Sachs Group",
        height=670,
    )

    fig = go.Figure(data=data, layout=layout)

    fig.update_xaxes(rangeslider_visible=True)

    fig.update_layout(
        xaxis_title="Time",
        yaxis_title="Deviation",
        template='plotly_dark')


    return fig


if __name__ == '__main__':
    app.run_server(debug=True)
```

Додаток Б Ілюстративний матеріал

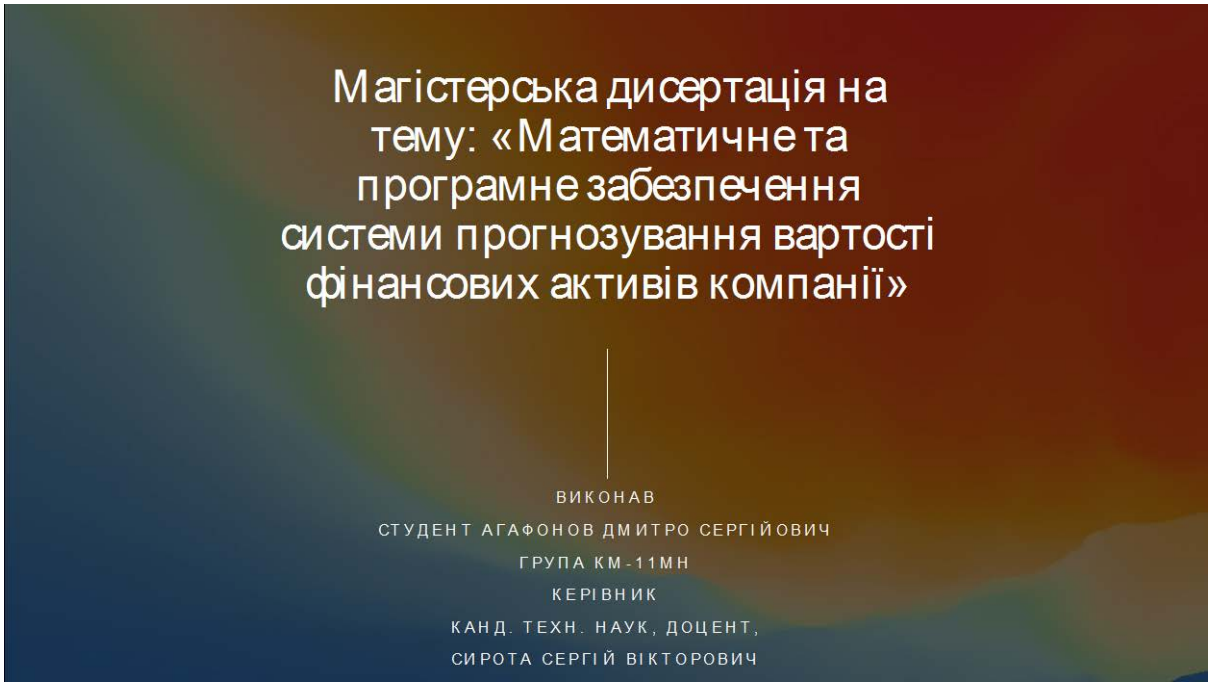


Рисунок Б.1 – Слайд 1

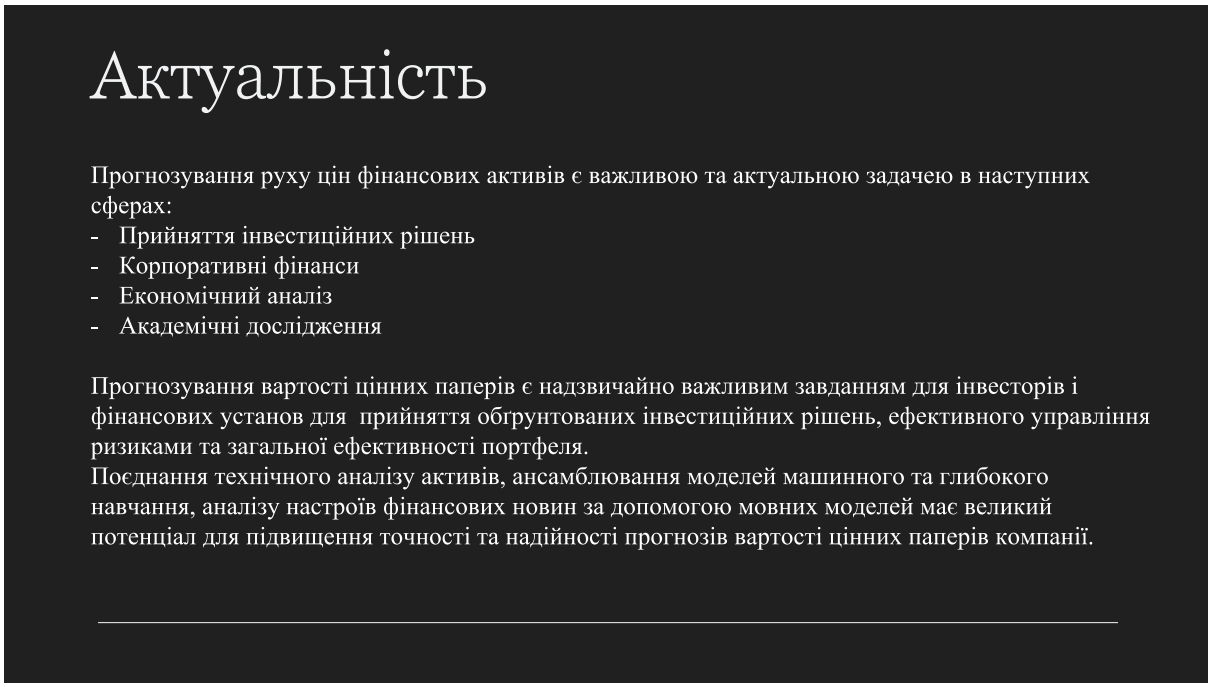


Рисунок Б.2 – Слайд 2

Постановка задачі

- ***Метою дисертаційної роботи є*** розробка математичного та програмного забезпечення для покращення прогнозу вартості фінансових активів компанії (порівняно з традиційними методами прогнозування часових рядів) із застосуванням технічного аналізу, семантичного аналізу фінансового тексту, технік ансамблювання моделей глибокого та машинного навчання з метою підвищення конкурентоспроможності компанії на ринку, покращення оптимізації та управління її ресурсами, забезпечення інвесторів та клієнтів фінансовим прогнозом.
- ***Предметом дослідження*** є техніки ансамблювання моделей машинного навчання, вплив на вартість фінансових активів компанії таких показників, як корельовані активи – показники залежних, схожих за економічною діяльністю або конкуруючих компаній; біржеві товари – енергетична сировина, кольорові та дорогоцінні метали, промислова сировина тощо; курси валют; фондові індекси; статистика по запитам в пошуковій системі; фінансові новини. Можливості технічного аналізу для прогнозування ймовірних змін вартості фінансових показників. Створення інформативних високорівневих ознак із застосуванням нелінійних методів зменшення розмірності даних, методів відбору ознак, автоенкодерів. Порівняльний аналіз базових моделей прогнозування, об’єднання їх в ансамбль, застосовуючи техніки bagging, stacking, boosting.

Рисунок Б.3 – Слайд 3

Завдання на магістерську дисертацію

1. Дослідження предметної області. Вивчення напрямку та сфери діяльності розглянутої компанії для визначення найбільш впливових та корисних для прогнозування ознак. Збір зазначених даних. Опис та порівняння основних моделей в контексті прогнозування фінансової інформації у вигляді часових рядів з довгостроковими та нелінійними залежностями. Визначення текстової фінансової інформації як джерела інсайдів про зміну показників компанії.
2. Огляд існуючих програмних рішень. Огляд застосунків та програмного забезпечення для прогнозування вартості фін. активів, порівняння, виділення переваг та недоліків.
3. Розробка та опис системи прогнозування вартості цінних паперів компанії, представлення класів та компонентів системи у вигляді діаграм.

Рисунок Б.4 – Слайд 4

Завдання на магістерську дисертацію

5. Розробка та опис математичного забезпечення, яке використовується в процесі досліджень та реалізації системи. Огляд інструментів технічного аналізу фінансової інформації, мовних моделей для класифікації емоційного забарвлення тексту фінансових новин, архітектури застосованої нейронної мережі, технік ансамблювання моделей машинного навчання.
6. Розробка та опис програмного забезпечення компонентів системи, реалізація інкапсуляції компонентів шляхом контейнеризації. Забезпечення взаємодії між контейнеризованими компонентами з використанням оркестрації.
7. Проведення верифікації та валідації результатів шляхом застосування метрик до прогнозованих даних та їх порівняння. Оцінка коректності класифікації настроїв фінансових новин.
8. Імплементація та тестування розробленого програмного забезпечення на реальних даних

Рисунок Б.5 – Слайд 5

Компанія та її діяльність

Для виконання магістерської дисертації була обрана компанія – великий американський банківський холдинг Морган Стенлі

атним капіталом
- Інвестиційний менеджмент

Рисунок Б.6 – Слайд 6

Рисунок Б.7 – Слайд 7

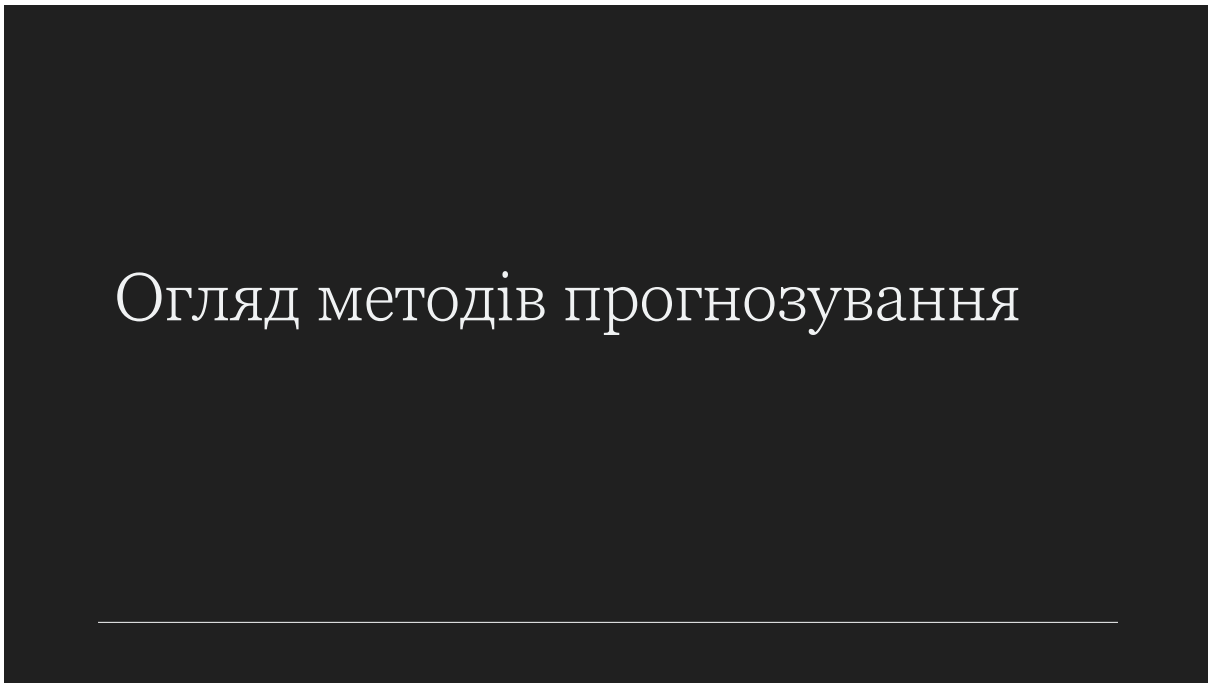


Рисунок Б.8 – Слайд 8

Векторна авторегресія

Vector Autoregression є популярним методом прогнозування багатовимірних часових рядів, зокрема у фінансовому та економічному аналізі. Модель VAR описує залежність між змінними часового ряду шляхом визначення кожної змінної як лінійної комбінації її попередніх значень та попередніх значень всіх інших змінних

$$y_t = a_0 + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + \varepsilon_t = a_0 + \sum_{m}^{p} A_m y_{t-m} + \varepsilon_t,$$

де $y_t = y_t^1, y_t^2, \ldots, y_t^k$ – вектор часових рядів;

A_i – матриця коефіцієнтів;

a_0 – константа;

ε_t – залишкова похибка (residuals).

Функціональна залежність моделі векторної авторегресії визначається аналітично між майбутніми та фактичними значеннями часового ряду та зовнішніми факторами.
Недоліки: алгоритм погано враховує довгострокові та нелінійні залежності в даних

Рисунок Б.9 – Слайд 9

ARIMA

Модель авторегресійного інтегрованого ковзаючого середнього ARIMA поєднує в собі трьохкомпонентну модель, що включає авторегресійну, інтегровану та ковзаючу середню складові. Модель ARIMA може бути записана як ARIMA(p, d, q), де p - порядок авторегресії, d - порядок інтегрування, а q - порядок ковзної середньої.

$$x_k(t) = \alpha_1 x_k(t-1) + \alpha_2 x_k(t-2) + \cdots + \alpha_p x_k(t-p) + \delta(t) - \\ -\theta_1 \delta(t-1) - \cdots - \theta_q \delta(t-q),$$

де α_i, θ_i – коефіцієнти моделі, що змінюються під час навчання;
p – кількість зсунутих точок, порядок зсуву (для авторегресійної частини);
k – кількість застосувань різниць до початкового ряду, поки ряд не стане стаціонарним;
q – розмір вікна ковзаючого середнього або порядок ковзаючого середнього (для частини ковзаючого середнього);
t – момент часу;

Метою використання моделі ARIMA (Autoregressive Integrated Moving Average) є прогнозування динаміки часового ряду шляхом аналізу різниць між його значеннями. У модель ARIMA можуть бути включені додаткові змінні як незалежні, що використовуються для навчання моделі.
Недоліки: дає гарні результати лише коли поточне значення даних в часовому ряді залежить від попередніх значень, алгоритм погано враховує довгострокові та нелінійні залежності в даних

10

Рисунок Б.10 – Слайд 10

Моделі на основі дерев прийняття рішень

Модель випадкового лісу (Random Forest) - це ансамбль рішень, що використовуються для задач класифікації, регресії та інших завдань машинного навчання. Вона використовує дерева рішень як базові моделі, які об'єднуються в ліс за допомогою процесу багатократного вибіру підвибірки даних та підвибірки ознак. Формула передбачення моделі лісу:

$$\hat{y}(t) = \frac{1}{K}\sum_{k=1}^{K} f_k(X(t), X(t-1), \dots, X(t-p), \mathbf{z}) \quad (2.13)$$

$\hat{y}(t)$ – передбачене значення часового ряду в час t
$X(t), X(t-1), \dots, X(t-p)$ – останні p значень часового ряду
$\mathbf{z}$ – вектор додаткових ознак
f_k – функція прогнозування на k-ом дереві
K – кількість дерев в лісі

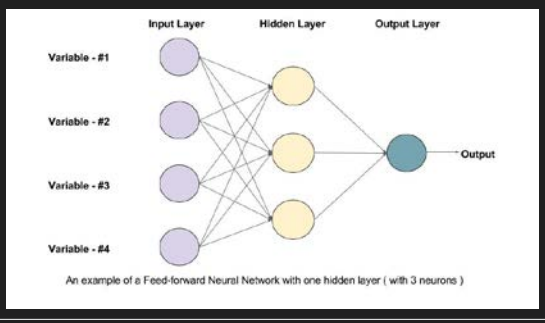
Переваги дерев'яних моделей: при прогнозуванні часових можуть обробляти нелінійні залежності між вхідними змінними та цільовою змінною, також відносно швидкі для навчання і можуть бути застосовані до багатовимірних дани. Також стійкі до викидів і відсутніх значень.
Недоліки: можуть не врахувати важливі залежності та кореляції між минулими та майбутніми значеннями в часі, інтерпретація моделі лісу може бути складною, оскільки внесок кожної функції в прогноз поширюється на всі дерева в лісі.

11

Рисунок Б.11 – Слайд 11

Штучні нейронні мережі

м чином, щоб вона мінімізувала помилку на виході.



12

Рисунок Б.12 – Слайд 12

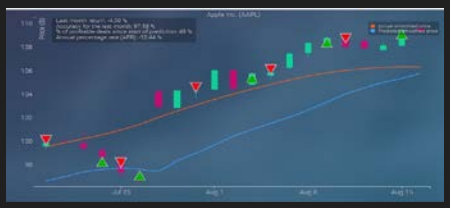
Існуючі програмні рішення

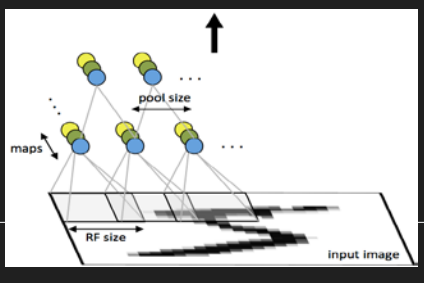
13

Рисунок Б.13 – Слайд 13

Існуючі програмні рішення

StocksNeural





14

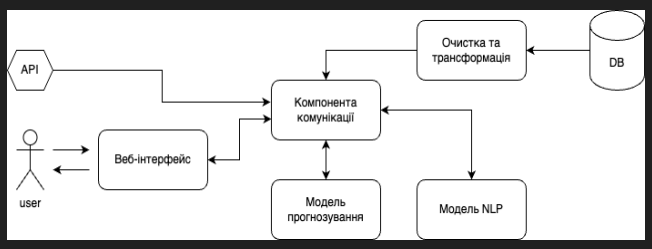
Рисунок Б.14 – Слайд 14

Модель системи

15

Рисунок Б.15 – Слайд 15

Узагальнена модель системи



н, моделі прогнозування та компоненти взаємодії.

16

Рисунок Б.16 – Слайд 16

Компонентна модель системи

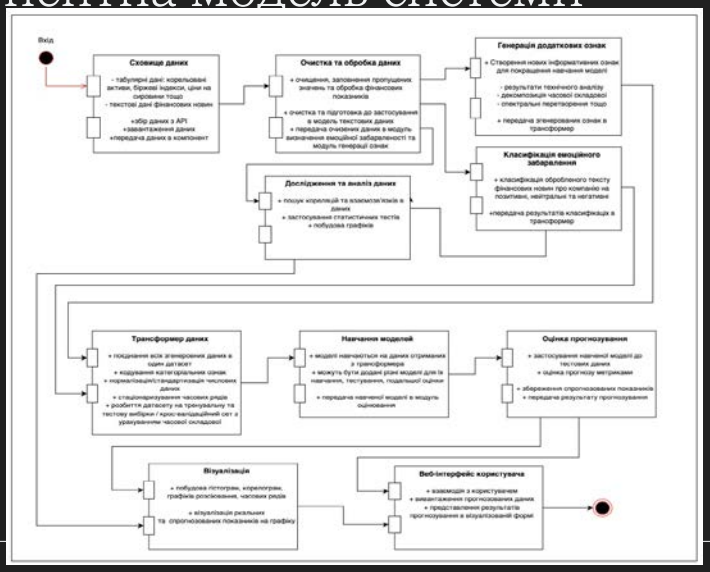
Компонентна модель системи прогнозування цін фінансових активів складається з наступних частин:

- Веб-інтерфейсу користувача для взаємодії з системою
- Сховища даних для зберігання табулярних та текстових даних
- Компоненти очистки та обробки даних
- Компоненти генерації додаткових ознак
- Компоненти трансформування даних
- Моделі класифікації настроїв фінансових новин
- Моделі прогнозування
- Компоненти дослідження та аналізу даних
- Компоненти метричної оцінки прогнозування
- Компоненти візуалізації

17

Рисунок Б.17 – Слайд 17

Компонентна модель системи

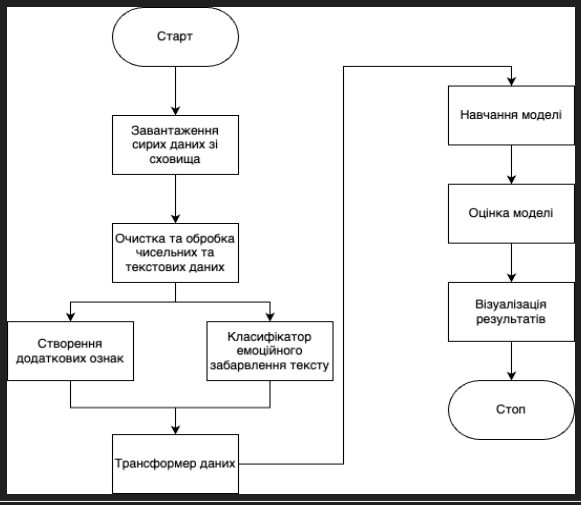


стеми

18

Рисунок Б.18 – Слайд 18

Блок-схема послідовності функціонування



19

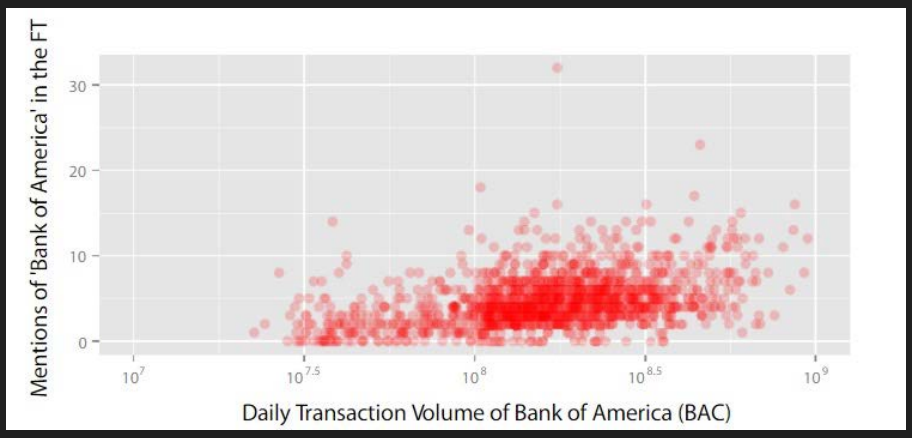
Рисунок Б.19 – Слайд 19

Математичне забезпечення

20

Рисунок Б.20 – Слайд 20

Зв’язок фінансових індикаторів та новин

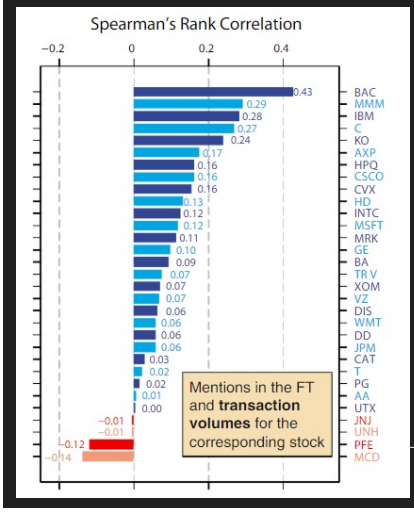


21 et al., 2013)

Рисунок Б.21 – Слайд 21

Зв’язок фінансових індикаторів та новин

lanyali et al., 2013)

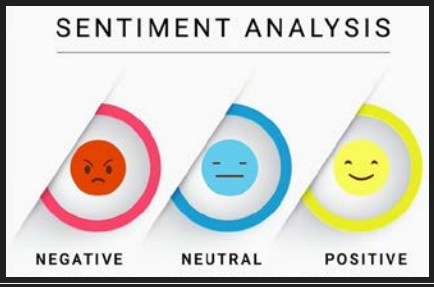


22

Рисунок Б.22 – Слайд 22

Класифікація новин за настроєм (finBERT

зитивні



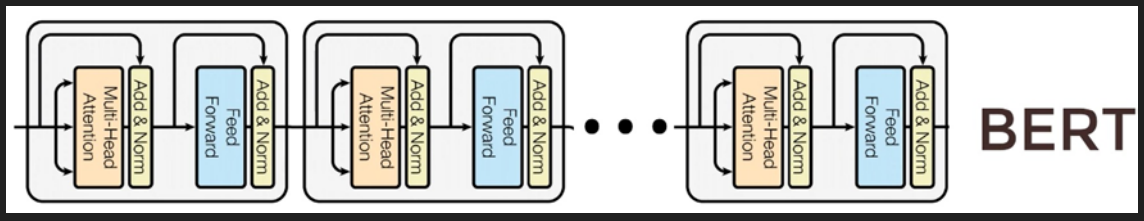
23

Рисунок Б.23 – Слайд 23

BERT (1)

ння

- Фактично *двоспрямована*



24

Рисунок Б.24 – Слайд 24

BERT (2)

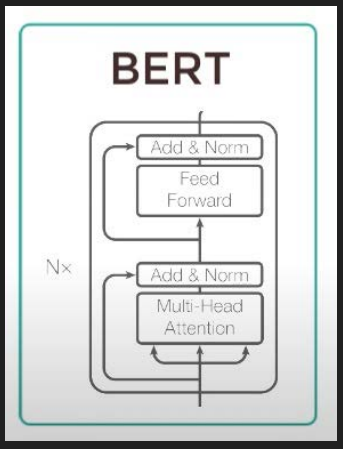
Застосується для задач:

- Нейронний машинний переклад
- Відповіді на запитання
- Сумаризація тексту
- Аналіз настроїв у тексті

«Розуміння» природньої мови

Проблема ***«розуміння» природньої мови*** вирішується:

- Попереднім навчанням стека енкодерів із застосуванням підходу маскованої мовної моделі Masked Language Model MLM, що випадково маскує 15% слів у вхідних даних.
- Модель навчається прогнозуючи пропущені слова, в результаті модель вивчає залежності в тексті та обумовлена як лівим, і правим контекстом.



Енкодер

25

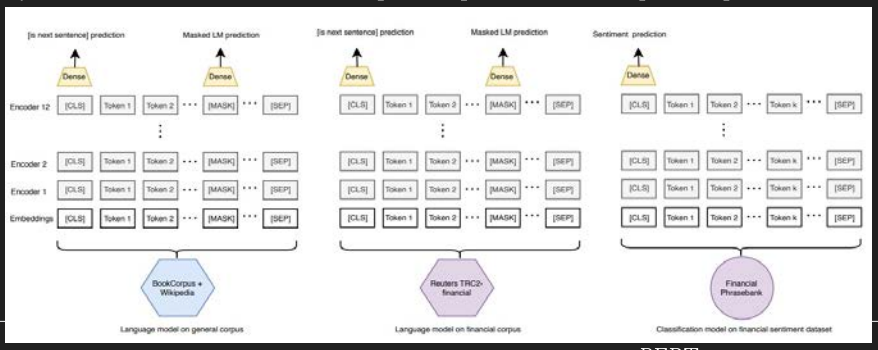
Рисунок Б.25 – Слайд 25

BERT (3)

Для використання мовної моделі в специфічній сфері фінансів, застосовується підхід Transfer Learning.

Навчання та налаштування BERT складається з трьох етапів:

1) Попереднє навчання на загальному корпусі English Wikipedia and BooksCorpus
2) Подальше попереднє навчання на фінансових корпусах
3) Налаштування моделі для задачі класифікації фінансових настроїв на розмічених даних



та налаштування BERT

26

Рисунок Б.26 – Слайд 26

finBERT

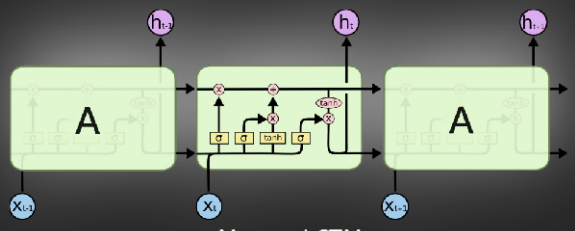
FinBERT — це переднавчена модель NLP для аналізу настрою фінансового тексту. Вона будується шляхом: 1) подальшого навчання мовної моделі BERT на корпусі фінансових текстів; 2) налаштування моделі на виконанні задачі класифікації на розмічених даних.

Pre-training: тренування на фінансових даних	Fine-tuning: задача класифікації
• English Wikipedia and BooksCorpus (базові дані для попереднього навчання BERT) П'ять фінансових корпусів на англ. різних направленостей та розмірів: • FinancialWeb –13 million financial news • YahooFinance – financial articles (published in the last four years) from Yahoo Finance • RedditFinanceQA – a corpus that contains automatically collected question-answer pairs about financial issues • Reuters TRC2-financial • Financial PhraseBank	QA) • *Financial Sentiment Analysis* (SA)

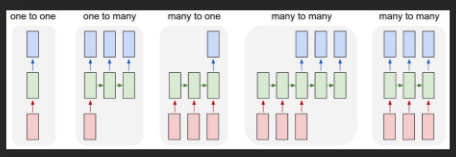
27

Рисунок Б.27 – Слайд 27

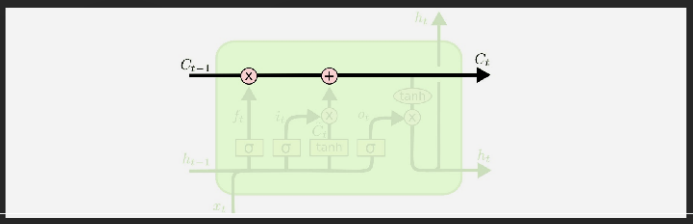
Модель прогнозування (LSTM)



Модуль LSTM



Many-to-many дозволяє подавати послідовності на вхід і отримувати на виході



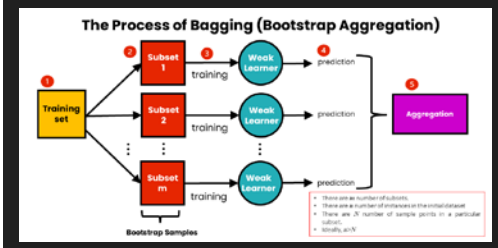
Стан ячейки

28

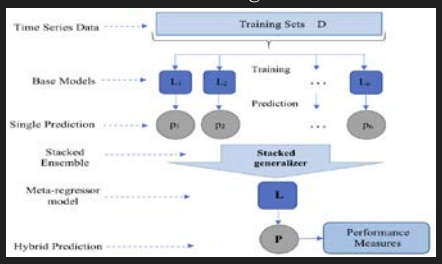
Рисунок Б.28 – Слайд 28

Техніки ансамблювання

Bagging



Stacking



Boosting

29

Рисунок Б.29 – Слайд 29

Програмне забезпечення

30

Рисунок Б.30 – Слайд 30

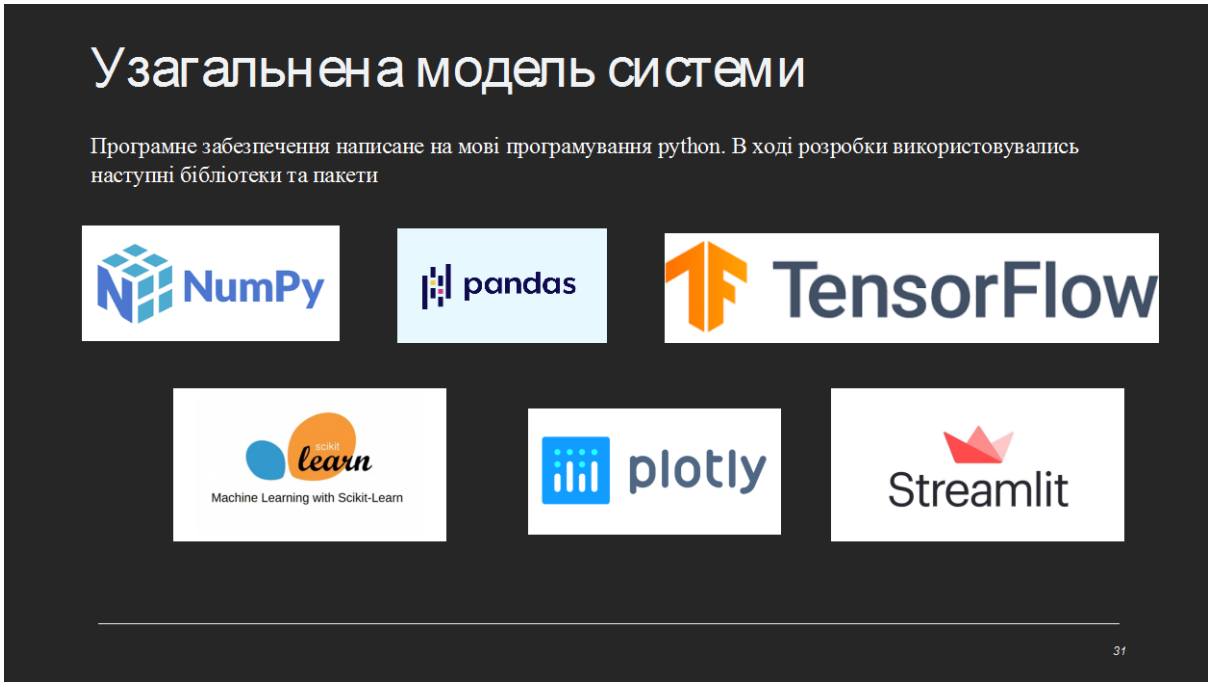


Рисунок Б.31 – Слайд 31

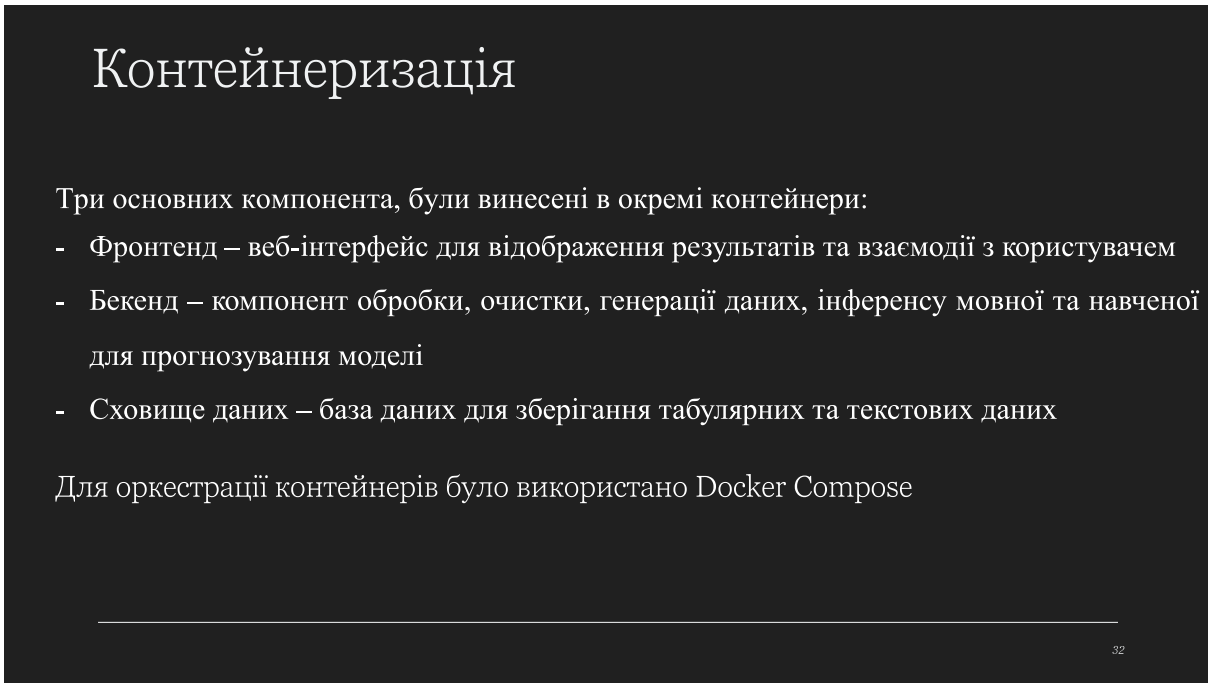


Рисунок Б.32 – Слайд 32

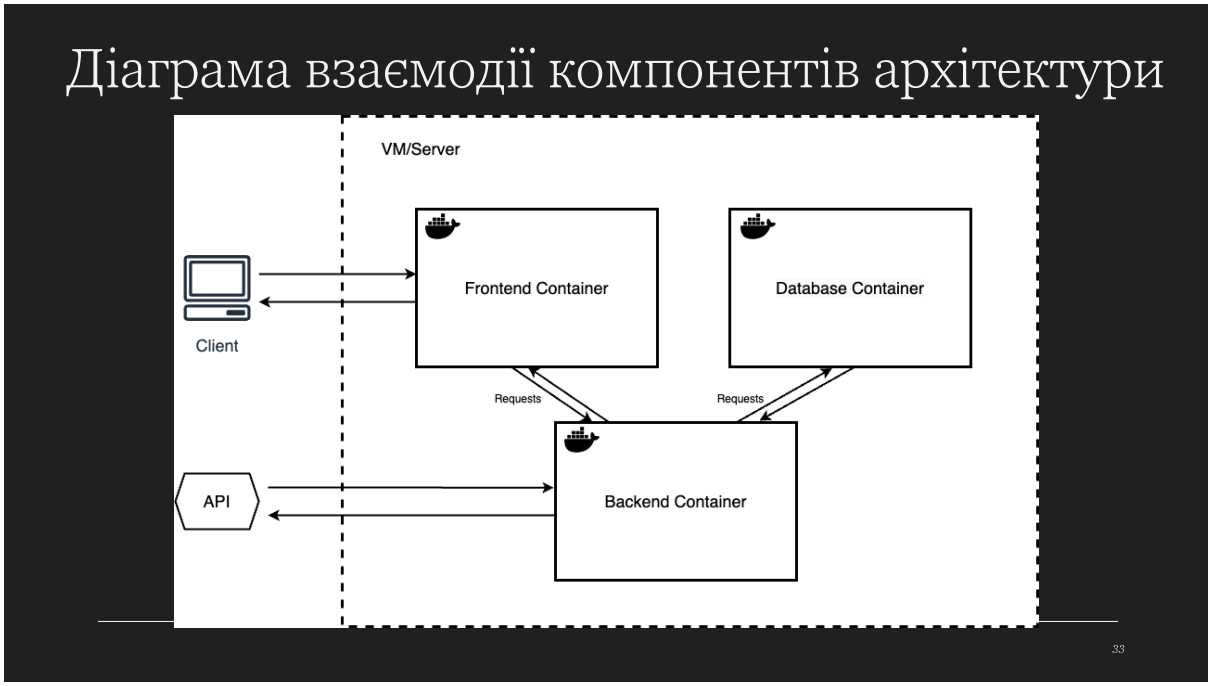


Рисунок Б.33 – Слайд 33

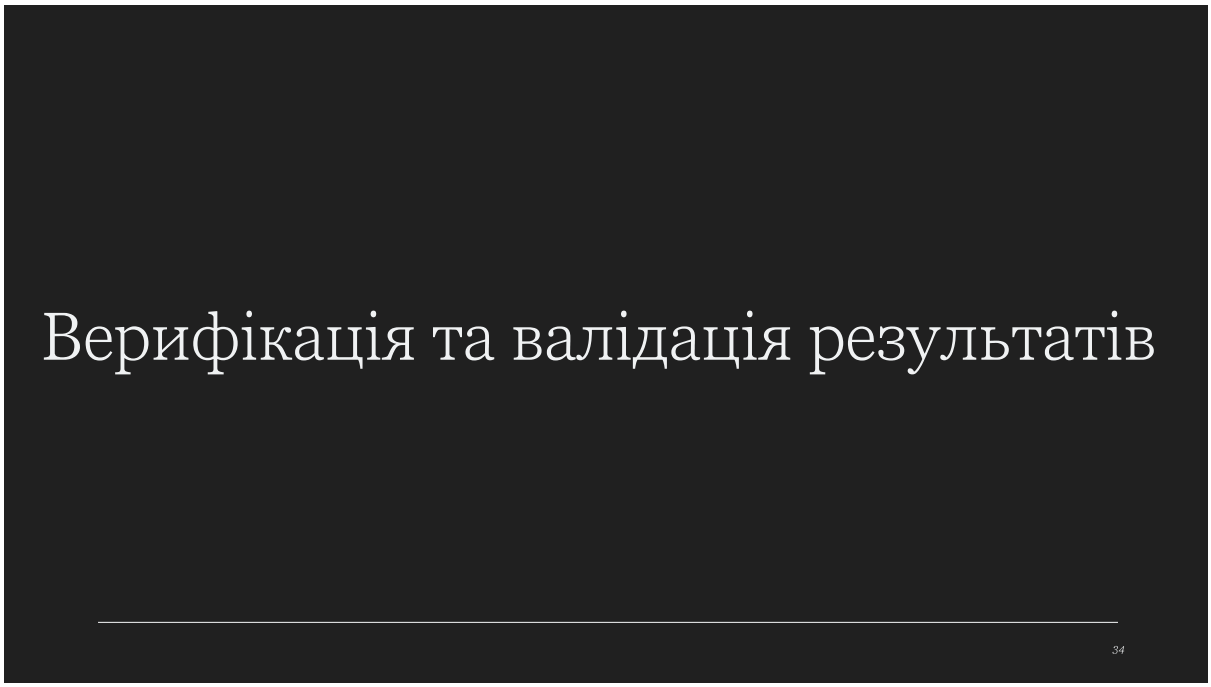


Рисунок Б.34 – Слайд 34

Рисунок Б.35 – Слайд 35

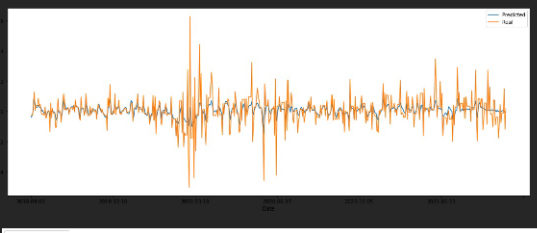
Рисунок Б.36 – Слайд 36

Реалізація xgboost regression

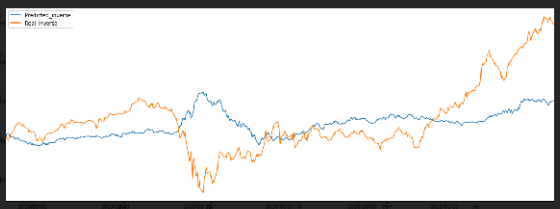
	Metrics	Stat_Stand_Score	Final_Score
0	MSE	0.005801	529.070971
1	MAE	0.046266	18.080647
2	R2	0.354147	0.729061
3	Cov	0.002069	1506.239586
4	Rearson_corr	0.707208	0.930567

Оцінка метриками

а)



б)



Візуалізація прогнозу

37

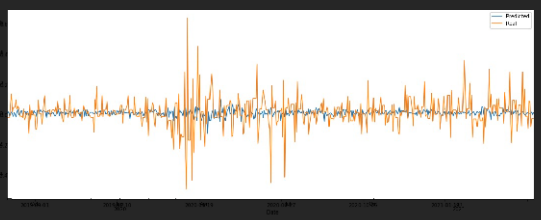
Рисунок Б.37 – Слайд 37

Реалізація bagging DecisionTree+RNN

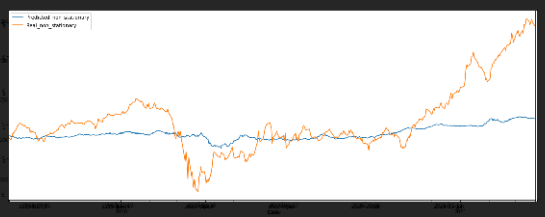
	Metrics	Stat_Stand_Score	Final_Score
0	MSE	0.010590	2909.356662
1	MAE	0.069814	44.501218
2	R2	-0.179106	-0.677538
3	Cov	-0.000055	1215.419148
4	Rearson_corr	-0.014985	0.640400

Оцінка метриками

а)



б)



Візуалізація прогнозу

38

Рисунок Б.38 – Слайд 38

Реалізація LSTM та оцінка результату

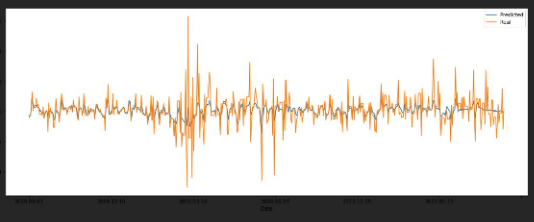
```
Model: "sequential_49"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_99 (LSTM)               (None, 60, 120)           73440
_________________________________________________________________
dropout_98 (Dropout)         (None, 60, 120)           0
_________________________________________________________________
lstm_100 (LSTM)              (None, 120)               115680
_________________________________________________________________
dropout_99 (Dropout)         (None, 120)               0
_________________________________________________________________
dense_47 (Dense)             (None, 30)                3630
_________________________________________________________________
activation_47 (Activation)   (None, 30)                0
=================================================================
Total params: 192,750
Trainable params: 192,750
Non-trainable params: 0
_________________________________________________________________
```

Побудова моделі

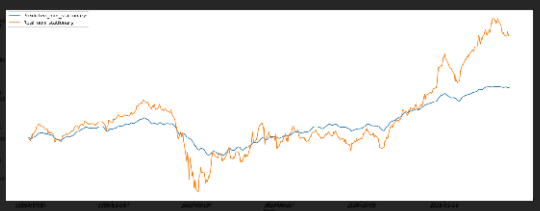
	Metrics	Stat_Stand_Score	Final_Score
0	MSE	0.007665	1081.957294
1	MAE	0.055221	24.237500
2	R2	0.146698	0.445926
3	Cov	0.000784	651.015381
4	Rearson_corr	0.536974	0.843824

Оцінка метриками

а)



б)



Візуалізація прогнозу

39

Рисунок Б.39 – Слайд 39

Висновки

В ході виконання магістерської дисертації розроблена система для покращення прогнозу вартості фінансових активів компанії із застосуванням технічного аналізу, семантичного аналізу фінансового тексту, технік ансамблювання моделей глибокого та машинного навчання з метою підвищення конкурентоспроможності компанії на ринку, покращення оптимізації та управління її ресурсами, забезпечення інвесторів та клієнтів фінансовим прогнозом.

- Досліджено напрямок сфери діяльності розглянутої компанії для визначення найбільш впливових та корисних для прогнозування ознак. Виконано збір зазначених даних. Проведено порівняння основних моделей в контексті прогнозування фінансової інформації у вигляді часових рядів з довгостроковими та нелінійними залежностями. Визначено текстову фінансову інформацію як джерела інсайдів про зміну показників компанії.
- Проведено огляд та порівняння застосунків та програмного забезпечення для прогнозування вартості фін. активів, порівняння, виділення переваг та недоліків.
- Виконана розробка та опис системи прогнозування вартості цінних паперів компанії, представлення класів та компонентів системи у вигляді діаграм.

40

Рисунок Б.40 – Слайд 40

Висновки

- Виконано огляд інструментів технічного аналізу фінансової інформації, мовних моделей для класифікації емоційного забарвлення тексту фінансових новин, архітектури застосованої нейронної мережі, технік ансамблювання моделей машинного навчання
- Виконана розробка та опис програмного забезпечення компонентів системи, реалізація інкапсуляції компонентів шляхом контейнеризації. Забезпечення взаємодії між контейнеризованими компонентами з використанням оркестрації.
- Проведено верифікаця та валідація результатів шляхом застосування метрик до прогнозованих даних та їх порівняння. Оцінка коректності класифікації настроїв фінансових новин.
- Виконана імплементація та тестування розробленого програмного забезпечення на реальних даних

41

Рисунок Б.41 – Слайд 41

Дякую за увагу!

42

Рисунок Б.42 – Слайд 42