**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені Ігоря Сікорського»**

**Факультет прикладної математики**

**Кафедра прикладної математики**

«До захисту допущено»

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_ О. Р. ЧЕРТОВ

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 р.

**Дипломна робота**

**на здобуття ступеня бакалавра**

**за освітньо-професійною програмою «Наука про дані та математичне моделювання»**

**спеціальності 113 «Прикладна математика»**

**на тему: «Математичне та програмне забезпечення системи прогнозування цін акцій»**

|  |  |  |
| --- | --- | --- |
| Виконав:  Студент IV курсу, групи КМ-71  Агафонов Дмитро Сергійович |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Керівник:  Старший викладач  Ладогубець Тетяна Сергіївна |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Консультант із нормоконтролю:  Старший викладач  Мальчиков Володимир Вікторович |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Рецензент:  Професор кафедри ММСА, д-р техн. наук, доцент  Зайченко Олена Юріївна |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Засвідчую, що в цій дипломній роботі немає запозичень із праць інших авторів без відповідних посилань.

Агафонов Д. С. \_\_\_\_\_\_\_\_\_\_\_\_

Київ — 2021

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет прикладної математики**

**Кафедра прикладної математики**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 113 «Прикладна математика»

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

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_ О. Р. ЧЕРТОВ

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 р.

**ЗАВДАННЯ**

**на дипломну роботу студенту**

Агафонову Дмитру Сергійовичу

1. Тема роботи: «Математичне та програмне забезпечення системи прогнозування цін акцій»,

керівник роботи Ладогубець Тетяна Сергіївна, старший викладач,

затверджені наказом по університету від «25» травня 2021 р. № 1331-С.

2. Термін подання студентом роботи: «9» червня 2021 р.

3. Вихідні дані до роботи: система повинна працювати з часовими рядами та текстом, результат роботи – прогнозований часовий ряд.

4. Зміст роботи: проаналізувати наявні можливості отримати прогноз ціни акцій на основі економічних, фінансових показників та даних, отриманих в результаті семантичного аналізу фінансових новин, технічного аналізу і математичних перетворень; дослідити методи прогнозування даних та обрати найкращий для даної задачі; виконати збір даних; побудувати модель, оцінити і візуалізувати прогноз, розробити інтерфейс.

5. Перелік ілюстративного матеріалу: архітектурні графи нейронних мереж, графіки часових рядів, ілюстрації роботи алгоритмів, блок-схема алгоритмів, презентація.

6. Дата видачі завдання: «01» лютого 2021 р.

Календарний план

| № з/п | Назва етапів виконання дипломної роботи | Термін виконання етапів роботи | Примітка |
| --- | --- | --- | --- |
| 1 | Огляд літератури за тематикою , визначення даних для прогнозування | 18.04.2021 |  |
| 2 | Проведення збору та обробки даних | 25.04.2021 |  |
| 3 | Проведення порівняльного аналізу можливих моделей прогнозування фінансових часових рядів | 28.04.2021 |  |
| 4 | Підготовка матеріалів першого розділу роботи | 07.05.2021 |  |
| 5 | Вивчення, дослідження та опис методів та алгоритмів, застосованих в ході обробки та прогнозування даних | 15.05.2021 |  |
| 6 | Підготовка матеріалів другого розділу роботи | 17.05.2021 |  |
| 7 | Програмна реалізація методів, обробка та тестування даних, побудова та оцінка моделі прогнозування | 25.05.2021 |  |
| 8 | Підготовка матеріалів третього розділу роботи | 27.05.2021 |  |
| 9 | Розробка веб-застосунку для візуалізації отриманого результату | 28.05.2021 |  |
| 10 | Оформлення пояснювальної записки | 31.05.2021 |  |

|  |  |  |
| --- | --- | --- |
| Студент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Дмитро АГАФОНОВ |
| Керівник роботи | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Тетяна ЛАДОГУБЕЦЬ | |

Анотація

Дипломну роботу виконано на 84 аркушах, вона містить 2 додатки та перелік посилань на використані джерела з 65 найменувань. У роботі наведено 33 рисунки та 3 таблиці.

Метою роботи є підвищення ефективності фінансової активності на фондовому ринку.

У рамках виконання дипломної роботи поставлено задачу на прогнозування ціни акцій компанії Goldman Sachs Group, виконано збір даних, які пов’язані чи впливають на прогнозовану ціну – корельованих активів, з провайдерів фінансової інформації, баз даних з історичною фінансовою інформацією. Проведено технічний аналіз для виявлення тенденцій у поведінці цін, в результаті якого отримано технічні індикатори. Виконано збір щоденних історичних фінансових новин, які класифіковано за настроєм з використанням NLP-моделі семантичного аналізу тексту BERT. Застосоване перетворення Фур’є для виділення глобальних та локальних трендів і для зменшення шуму в даних. Пониження розмірності даних та отримання високорівневих іноформативних ознак здійснено методом аналізу головних компонент (PCA).

В ході роботі розглянуто моделі прогнозування даних та обрано найкращу для даної задачі. Обрані дані очищено, проаналізовано і протестовано, виконані трансформації даних для застосування в обраній моделі. Модель реалізовано і обучено, отриманий прогноз оцінено метриками та візуалізовано. Розроблено інтерфейс для зручної демонстрації результату.

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 84 pages. It contains 2 appendixes and bibliography of 65 references. 33 figures and 3 tables are given in the thesis.

The purpose of the work is to increase the efficiency of financial activity in the stock market.

As part of the thesis, the task was to forecast the stock prices of Goldman Sachs Group company. Correlated assets, the data that is related to or affect the forecasted price is collected from financial information providers, databases with historical financial information. A technical analysis was conducted to identify trends in price behavior, which resulted in technical indicators. A collection of daily historical financial news, which is classified by mood using the NLP-model of semantic analysis of text BERT. Fourier transform is used to highlight global and local trends and to reduce data noise. The reduction of data dimensionality and obtaining high-level informative features was carried out by the method of principal components analysis (PCA).

In the course of the work the data forecasting models are considered and the best one for this task is chosen. The selected data were cleaned, analyzed and tested, data transformations were performed for use in the selected model. The model is implemented and trained, the obtained forecast is evaluated by metrics and visualized. An interface has been developed for convenient demonstration of the result.

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

ЗМІСТ

[Перелік умовних позначень, скорочень і термінів 9](#_Toc74748230)

[Вступ 10](#_Toc74748231)

[1 Постановка задачі 12](#_Toc74748232)

[2 Огляд існуючих математичних методів 13](#_Toc74748233)

[2.1 Передмова 13](#_Toc74748234)

[2.2 Розглянуті алгоритми 13](#_Toc74748235)

[2.2.1 Векторна авторегресія (Vector Autoregression) 13](#_Toc74748236)

[2.2.2 Багатовимірна лінійна регресія 15](#_Toc74748237)

[2.2.3 Модель ARIMA 18](#_Toc74748238)

[2.2.4 Прихована Марківська модель 20](#_Toc74748239)

[2.2.5 Штучні нейронні мережі ANN 21](#_Toc74748240)

[2.2.6 Порівняння розглянутих математичних методів 22](#_Toc74748241)

[2.3 Огляд існуючих комерційних програмних рішень 23](#_Toc74748242)

[2.3.1 StocksNeural 23](#_Toc74748243)

[2.3.2 Stocksight 23](#_Toc74748244)

[2.3.3 Deep Convolution Stock Technical Analysis 24](#_Toc74748245)

[2.3.4 Порівняння комерційних програмних рішень 24](#_Toc74748246)

[2.4 Висновки до розділу 25](#_Toc74748247)

[3 Математичне забезпечення 27](#_Toc74748248)

[3.1 Передмова 27](#_Toc74748249)

[3.2 Технічні індикатори 27](#_Toc74748250)

[3.2.1 Індикатори тенденції 28](#_Toc74748251)

[3.2.2 Осциляторні індикатори 32](#_Toc74748252)

[3.3 BERT для аналізу настроїв 35](#_Toc74748253)

[3.4 Перетворення Фур’є 38](#_Toc74748254)

[3.5 Метод головних компонент PCA 43](#_Toc74748255)

[3.6 Довга-короткострокова пам’ять LSTM 46](#_Toc74748256)

[3.6.1 Forget gate layer 51](#_Toc74748257)

[3.6.2 Input layer gate 52](#_Toc74748258)

[3.6.3 Оновлення стану ячейки 53](#_Toc74748259)

[3.6.4 Output gate layer 54](#_Toc74748260)

[3.7 Висновки до розділу 55](#_Toc74748261)

[4 ПРОГРАМНА РЕАЛІЗАЦІЯ 57](#_Toc74748262)

[4.1 Етапи виконання роботи 57](#_Toc74748263)

[4.2 Створення ознак 58](#_Toc74748264)

[4.2.1 Збір корельованих активів 59](#_Toc74748265)

[4.2.2 Створення технічних індикаторів 60](#_Toc74748266)

[4.2.3 Парсинг новин та класифікація finBERT 62](#_Toc74748267)

[4.2.4 Спектральний аналіз (перетворення Фур’є) 63](#_Toc74748268)

[4.3 Аналіз та підготовка даних 64](#_Toc74748269)

[4.4 Формування виборок 67](#_Toc74748270)

[4.5 Побудова LSTM та оцінка результату 71](#_Toc74748271)

[4.5.1 Алгоритм тренування моделі 73](#_Toc74748272)

[4.6 Графічний інтерфейс 74](#_Toc74748273)

[4.7 Висновки до розділу 76](#_Toc74748274)

[Висновки 77](#_Toc74748275)

[Перелік посилань 79](#_Toc74748276)

[Додаток А Лістинги програм 85](#_Toc74748277)

[Додаток Б Ілюстративний матеріал 107](#_Toc74748278)

Перелік умовних позначень, скорочень і термінів

ANN – штучна нейронна мережа, Artificial Neural Network.

API – програмний інтерфейс додатку.

AR – авторегресія.

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

BERT - Bidirectional Encoder Representations from Transformers.

EMA – експоненціальне ковзаюче середнє, exponential moving average

FinBERT – financial Bidirectional Encoder Representations from Transformers

GS – Goldman Sachs

HMM – прихована Марківська модель, Hidden Markov model

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

PCA – метод головних компонент, principal component analysis

RNN – рекурентна нейронна мережа, Recurrent Neural Network

RSI – індекс відносної сили, Relative strength index

ROC – швидкість зміни ціни, rate of change

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 [1]. 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 [2].

Існує багато точок зору щодо можливості ефективно прогнозувати вартість цінних паперів. Можна окремо виділити дві, які виражені в двох гіпотезах: гіпотезі випадкового блукання та гіпотезі ефективного ринку [3]. Перша стверджує, що зміна цін акцій є стохастичним процесом [4], тобто рух цін відбувається хаотично і майбутнє фондового ринку не може бути спрогнозовано за допомогою будь-яких наукових методів. Друга гіпотеза більш оптимістична, відповідно до неї будь-яка нова інформація, що потрапляє на ринок, практично моментально знаходить відображення у зміні цін. Виконуючи дану роботу, було припущено, що гіпотеза ефективного ринку – вірна, тому необхідно вказати декілька тверджень, що є формами ринкової ефективності [5]:

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.

На сьогоднішній день існую декілька варіантів отримати прогноз щодо поведінки акції на ринку: скористатись відповідним програмним забезпеченням, виконати аналіз самостійно або звернутись до фахівців. Проте існують деякі проблеми, пов’язані з цими варіантами відповідно: доступ до програм може бути досить дорогим або обмеженим, користувач може не володіти навичками технічного або фундаментального аналізу та високий рівень шахрайства, низький рівень компетентності або висока ціна за надану інформацію. Тому проблема прогнозування цін акції є актуальною.

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.

# Постановка задачі

Метою даної дипломної роботи є підвищення ефективності фінансової активності на фондовому ринку.

Для досягнення мети повинні бути виконані наступні етапи:

а) Обрати компанію, ціни акцій якої будуть прогнозуватись та знайомитися з економічними та фінансовими показниками та їх можливим впливом на ціну акцій обраної компанії.

б) Виконати збір даних, серед яких:

1. Корельовані активи [6];
2. Кількість щоденних запросів в Google Trends;
3. Щоденні історичні фінансові новини, що стосуються обраної компанії;

в) Створити додаткові дані:

1. Технічні індикатори;
2. Класифікувати фінансові новини за допомогою NLP;
3. Виконати перетворення Фур’є;

г) Виконати обробку отриманих даних:

1) Очистити, стандартизувати, стаціонаризувати та протестувати дані;

2) Для пониження розмірності та отримання високорівневих інформативних ознак застосувати метод аналізу головних компонент PCA

д) Проаналізувати декілька можливих методів машинного навчання і обрати найбільш підходящий для даної задачі. Обучити модель та оцінити її роботу.

е) Розробити інтерфейс для візуалізації результату.

# Огляд існуючих математичних методів

## Передмова

Існує багато методів, які можна застосувати для аналізу та прогнозування часових рядів. Але задачею даної дипломної роботи є прогнозування фінансових даних, зокрема поведінки цін акцій на ринку. Виходячи з цього, необхідно розглянути моделі, які гарно зарекомендували себе при роботі з багатовимірними фінансовими часовими рядами, головним пріорітетом у виборі є можливість методу знаходити нелінійні залежності, приховані патерни та глобальні і локальні тренди.

## Розглянуті алгоритми

При виборі моделі для прогнозування часового ряду цін акцій компанії було розглянуто наступні методи.

### Векторна авторегресія (Vector Autoregression)

Модель векторної авторегресії (VAR – Vector Autoregression) є досить популярним методом для прогнозування багатовимірних часових рядів. Модель векторної авторегресії була представлена Кристофером Сімсом [7] як більш оптимальна альтернатива системам економічних рівнянь, що визначають взаємозв’язок між економічними змінними – системам одночасних рівнянь [8]. Фактично, в VAR кожна змінна є лінійною функцією від самої себе та від попередніх значень всіх інших змінних. В векторно-матричній формі модель векторної авторегресії порядка p можна записати таким чином (формула 2.1):

де – вектор часових рядів;

– матриця коефціцєнтів;

– константа;

– залишкова похибка (residuals).

Формула (1.1) є замкнутою, тому що в якості пояснюючих (екзогенних) змінних виступають лаги пояснюваних (ендогенних) змінних. Модель, доповнена екзогенними змінними до порядку q в матричному вигляжі має формулу (2.2):

де – вектор часових рядів;

– матриця коефціцєнтів ендогенної змінної;

– константа;

– матриця коефціцєнтів екзогенної змінної;

– вектор екзогенних часових рядів;

– залишкова похибка (residuals).

Модель векторної авторегресії може бути застосована як до стаціонарних (тоді можна будувати звичайну VAR), так і до нестаціонарних, тобто інтегрованих першого порядку I(1) часових рядів (при побудові VAR вказується різниця відповідного порядку) [9].

Модель векторної авторегресії є статистичною, тобто функціональна залежність між майбутніми та фактичними значеннями часового ряду і зовнініми факторами задана аналітично (функцією).

### Багатовимірна лінійна регресія

Багатовимірна лінійна регресія в аналізі часових рядів є статистичним методом, що використовує декілька незалежних (екзогенних) змінних для прогнозування залежних (ендогенних) змінних. Метою використання багатовимірної лінійної регресії є знаходження лінійних залежностей між пояснюючими та пояснюваними змінними.

Формула багатовимірної лінійної регресії є лінійною функцією [10] (формула 2.3):

де – кількість елементів у вибірці;

– залежна (пояснювана) змінна;

– незалежна (пояснююча) змінна;

– вільний коефіцієнт (intercept);

– коефіцієнти пояснюючих змінних (slope);

– залишкова похибка (residuals).

Метою навчання моделі є мінімізація функції втрат (формула 2.4):

Для того, щоб додати змінну до вектора коефіцієнтів , необхідно:

а) Додати одну ознаку, де значення кожного елементу дорівнює одиниці ;

б) Встановити ;

в) Тоді .

В результаті, функцію втрат (2.4) можна записати як (формула 2.5):

Оскільки для обчислення на компьютері більш прийнятними є матричні обислення, то формулу багатовиміної лінійної регресії та функцію втрат можна представити в матричній формі.

Отже, значення пояснюючих змінних можна представити матрицею (2.6) розмірністю n x (p+1):

Значення пояснюваних змінних матрицею (2.7) розмірністю n x 1:

Функція втрат (2.4) може бути записана (формула 2.8):

де ;

Детальніше розписана формула 2.8 (формула 2.9):

Для мінімізації коефіцієнтів пояснювальних змінних виконуються операції з матрицями, отримана матриця має розмірність (p+1, n) (формула 2.10):

де першим у відповіді буде значення вільного коефіцієнту (intercept) з формул (2.3 – 2.4).

Для регресійної моделі діє ряд обмежень [11]:

* Між пояснюючими і пояснювальною змінною повинна бути лінійна залежність;
* Пояснюючи (незалежні) змінні не повині мати високу кореляцію одна з одною;
* Залишки моделі (залишкові похибки, residuals) мають бути нормально розподілені;
* Повинна виконуватись гомоскедастичність [12] залишків.

Зазвичай, багатовимірну лінійну регресію використовують для аналізу та тестування даних, виявлення лінійних залежностей між ними. Для задачі дипломної роботи необхідно, щоб модель могла виявляти нелінійні залежності в даних.

Модель багатовимірної лінійної регресії є статистичною, функціональна залежність між майбутніми та фактичними значеннями часового ряду, а також зовнініми факторами задана аналітично.

### Модель ARIMA

Модель авторегресійного інтегрованого ковзаючого середнього (autoregressive integrated moving average, далі – ARIMA) була запропонована Дж. Боксом і Г. Дженкінсом [13], тому також відома як модель Бокса-Дженкінса. Вона призначена для прогнозування нестаціонарних часових рядів , які мають наступні властивості [8]:

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

б) Ряд , отриманий в результаті застосування до початкового ряду процедури методу послідовних різниць, може бути описаний за допомогою моделі авторегресіного ковзаючого середнього ARMA. Іншими словами, ряд повинен мати властивість перетворення в стаціонарний в результаті послідовного взяття різниць [8].

ARIMA – це поєднання різницевої моделі авторегресії і моделі ковзаючого середного. Модель аналізованого процеса , де t = 1,2,..n може бути представлена формулою (2.11) [8]:

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

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

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

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

– момент часу;

можна представити як (2.12):

де – оператор різниці часового ряду порядку k;

– 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 застосування даної моделі є прогнозування руху часового ряду шляхом дослідження різниць між значеннями самого ряда. В ARIMA може бути додані окремі дані як незалежні змінні, з використанням яких виконується навчання моделі.

Модель дає гарний результат при використанні на даних, де попереднє значення впливає на наступне, але для задачі дипломної роботи підходить погано, оскільки необхідно отримати нелінійні та приховані залежності серед незалежних змінніих.

Модель багатовимірної лінійної регресії є статистичною, функціональна залежність задана аналітично.

### Прихована Марківська модель

Вперше прихована Марківська модель (Hidden Markov model – HMM) була описана в серії наукових статей Леонардом Баумом в 1960-х роках [14]. Це статистична модель, в якій система, що моделюється є марківським процесом [15] зі схованими станами. Передбачається, що є інший процес, поведінка якого залежить від першого. Метою є знаходження невідомих параметрів в залежності від спостережуваних. Тип навчання HMM – без вчителя (unsupervised).

HMM складається з кортежу (S, K, П, A, B), в якому:

а) – множина станів;

б) – вихідний алфавіт;

в) П – розподілення початкового стану;

г) A – розподілення ймовірностей станів переходу;

д) В – розподілення ймовірностей символу спостереження.

Приховані марківські мережі є гарним варіантом для прогнозування фінансових часових рядів, оскільки дані можуть бути згенеровані прихованим стохастичним процесом, з виявленням якого метод може впоратись. НММ використовується в різноманітних сферах діяльності: розпізнавання голосу, біоінформатика та аналіз різних часових рядів. Модель може бути застосована до багатовимірних часових рядів.

Як було зазначено в роботі [16], метод прихованих марківських мереж при роботі з багатовимірними даними та знаходженні прихованих патернів в даних має ряд фундаментальних проблем. Проблеми можна вирішити, застосовуючи відповідні алгоритми, але вони є складними та громіздкими для реалізації, тому було прийнято рішення від даного методу відмовитись.

Також приховані марківські моделі є чутливими до тривалості часових розривів у послідовностях, що також є недоліком.

### Штучні нейронні мережі ANN

Штучна нейронна мережа (Artificial Neural Network, ANN) [17] – це паралельно розподілений процесор, який має властивість до збереження та репрезентації отриманих в ході навчання знань. Мережа схожа з мозком в двох аспектах:

а) знання здобувається мережею в процесі навчання;

б) Для збереження знання використовуються сили міжнейронних з'єднань, звані також синаптическими вагами.

Модель ANN є структурною за типом. Перед застосуванням 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. Також нейронні мережі досить легко реалізувати.

Для дослідження та прогнозування часових рядів зазвичай використовуються наступні методи:

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 для вирішення задачі дипломної роботи було обрано модифікацію рекурентних нейронних мереж – довга короткострокова пам’ять (Long-Short Term Memory, LSTM), що має властивість таку, як довгострокова пам’ять, яка дуже корисна при аналізі та прогнозуванні часових рядів. Більш детально метод описаний в підрозділі 3.6.

### Порівняння розглянутих математичних методів

Порівняльна характеристика методів розв’язання задачі наведена в таблиці 2.1.

Таблиця 2.1 – Порівняння математичних методів

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Стаціонар-ність ряду | Тип навчання | Тип моделі | Нелінійні залежності в даних | Простота реалізації |
| VAR | Стаціонарний/  нестаціонарний | Supervised | Статична | Не виявляє | +/- |
| Лінійна регресія | Стаціонарний | Supervised | Статична | Не виявляє | + |
| ARIMA | Стаціонарний/  нестаціонарний | Supervised | Статична | Не виявляє | + |
| HMM | Стаціонарний/  нестаціонарний | Unsupervised | Структурна | Виявляє | +/- |
| ANN (RNN LSTM) | Стаціонарний/  нестаціонарний | Supervised | Структурна | Виявляє | + |

Необхідно відмітити, що стаціонаризувати часові ряди потрібно при застосуванні будь-яких моделей (підрозділ 4.4).

## Огляд існуючих комерційних програмних рішень

### StocksNeural

Онлайн-сервіс StocksNeural [18] розроблено для трейдерів, які не хочуть обтяжувати себе опануванням технічного та фундаментального аналізу і слідкувати за фінансовими показниками компаній. Додаток представляє собою веб-платформу, на якій можна отримати прогноз щодо рух цін на інструменти фондового ринку а також отримати рекомендації щодо покупки або продажу. Математичне забезпечення програми представляє собою поєднання інструментів технічного анаізу та глибокого машинного навчання. Програма використовує корельовані активи. Недоліками є те, що прогноз можна отримати на п’ять днів вперед і для доступу до платформи необхідно оформити платну підписку. Також мінусом є те, що розробники не розкривають інформацію про вхідні дані.

### Stocksight

Stocksight це програмне забезпечення для аналізу фондового ринку з відкритим вихідним кодом [19]. Для роботи використовує Elasticsearch, зберігає дані з Twitter і актуальні фінансові новини, що стосуються акції тієї або іншої компанії. Stocksight проводить аналіз настроїв в тексті, щоб встановити загальний фон навколо конкретної компанії з використанням NLP та бібліотек для аналізу настроїв. Доступ до коду є безкоштовним. Мінусом даного варіанту є те, що прогнозування здійснюється на маленький проміжок часу і програма не використовує дані технічного та функціонального аналізу, що може значно підвищити точність прогнозів. Також для роботи користувачу необхідно вміти користуватися засобами програмування, тому що запустити програму можна лише з персонального компьютера.

### Deep Convolution Stock Technical Analysis

Deep Convolution Stock Technical Analysis [20] є програмою з відкритим вихідним кодом. Для моделювання ринку акцій використовує глибокі згорткові нейронні мережі та технічні індикатори. Оскільки інструментами технічного аналізу є графіки, то за допомогою згорткових нейронних мереж можна проаналізувати візуальне представлення часового ряду та знайти необхідні для прогнозування патерни напрямку або зміни тренду. Доступ до коду є безкоштовним. Недоліком даного програмного рішення є те, що користувачу необхідно вміти користуватися засобами програмування, тому що запустити програму можна лише з персонального компьютера. Також при прогнозуванні не використовуються корельовані активи та аналіз настроїв новин та інвесторів, а тільки щоденні показники торгівлі на біржі (Open, High, Low, Close, Volume).

### Порівняння комерційних програмних рішень

Результати огляду існуючих комерційних програмних рішень наведені у таблиці 2.2.

Таблиця 2.2 – Огляд існуючих методів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Математична модель | Вхідні дані | Строк прогнозу | Платний доступ |
| StocksNeural | Нейронні мережі, технічний аналіз | Корельовані активи, рез.-ти технічного аналізу | 5 днів – 1 місяць | + |
| Stocksight | Нейронні мережі (NLP) | Текст новин | 1 день | - |
| Deep Convolution  Stock Technical  Analysis | Нейронні мережі (СNN) | Історичні дані прогнозованого ряду, рез.-ти технічного аналізу | 1 місяць | - |

## Висновки до розділу

В розділі 2 було обрано і порівняно п’ять методов аналізу та прогнозування, які можна застосувати для прогнозування фінансової інформації. Зокрема: векторна авторегресія, багатовимірна лінійна регресія, модель авторегресійного ковзаючого середнього, прихована марківська модель, штучні нейронні мережі. За сукупністю характеристик та властивостей було прийнято рішення застосувати рекурентні нейронні мережі, зокрема модифікацію RNN LSTM.

Серед програмних рішень було оглянуто три варіанти: онлайн-сервіс StocksNeural та програмне забезпечення з відкритим програмним кодом Stocksight та Deep Convolution Stock Technical Analysis. В ході огляду було визначено сильні та слабкі сторони представлених варіантів. Суттєвим недоліком є відсутність або комбінування вхідних даних, корисних для прогнозування або недостатня їх кількість. 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.

# Математичне забезпечення

## Передмова

Під час виконання роботи було застосовано декілька математичних моделей. Для проведення технічного аналізу використано алгоритми технічних індикаторів. Технічні індикатори поділяються на дві групи [21]: індикатори тенденції (підтверджують напрям тренду) та осциляторні (передбачають зміни тренду). Для зменшення шуму та виділення локальних та глобальних трендів в прогнозованих даних застосовано перетворення Фур’є. Класифікацію фінансових новин виконано за допомогою NLP моделі BERT. Було зібрано велику кількість ознак. Для зменшення їх розмірності застосовано аналіз головних компонент PCA. Для прогнозування даних побудована та навчена нейронна мережа RNN LSTM.

## Технічні індикатори

Індикатор – це результат математичних розрахунків на основі показників ціни та/або об’єму. Отримана величина використовується для прогнозування цінових змін.[22] Іншими словами, технічні індикатори – це алгоритми, що дозволяють отримати інформацію про майбутні ціни за допомогою котирувань за певний період. Розглянемо технічні індикатори, які використані в роботі.

### Індикатори тенденції

Один з типів технічних індикаторів – індикатори тенденції. Застосувавши дані алгоритми, можна отримати підтвердження напряму тренду. В роботі використані такі індикатори тенденції:

а) 7 та 21-денне просте ковзаюче середнє (SMA).

Ковзаюче середнє – один з найпоширеніших індикаторів для технічного аналізу фінансових даних. Значення індикатора – середня ціна цінного паперу за заданий час, який може бути встановлений за бажанням аналітика.

Для розрахунку ковзаючого середнього необхідно, щоб не було пропущених даних.

В методі ковзаючого середнього початкове значення елементів ряда замінюється на середню арифметичну всередині обраного інтервала. Отримане значення відноситься до середини обраного інтервалу, потім вікно здвигаться і відбувається розрахунок нового значення. Чим більше вікно, тим більш гладким є новий ряд.

Якщо розмір вікна непарне число, то формула (3.1) [23] розрахунку середнього:

При парному розмірі вікна формула (3.2):

де – значення і-го члена ряда;

– ціле додатнє число, означає інтервал згладжування.

б) експоненціальне ковзаюче середнє (EMA)

Експоненціальне ковзаюче середнє – це окремий випадок зваженого ковзаючого середнього, також поширений індикатор в технічному аналізі. Особливістю такого згладжування є розподілення ваг: більша вага надається останнім значенням ряду, менша – попереднім [23, 24]. Формула (3.3):

де - значення експоненційного ковзаючого середнього n-го порядку в точці t, тобто в останньому значенні ряда;

- значення експоненційного ковзаючого середнього n-го порядку в точці t-1, тобто в передостанньому значенні ряда;

- значення експоненційного ковзаючого середнього (n-1)-го порядку в точці t, тобто в останньому значенні ряда;

- коефіцієнт згладжування, що має бути в межах від 0 до 1. Також коефіцієнт може бути представлений як , де і має значення розміру вікна згладжування.

в) ціновий канал (Price Channel)

Ціновий канал – трендовий індикатор і складається з двох обмежуючих ліній, які рахуються як максимальне та мінімальне значення ціни за певний проміжок часу

г) полоси Боллінджера (Bollinger bands)

Популярний індикатор, розроблений Дж. Боллінджером, є багатовимірним індикатором. Дозволяє визначити, наскільки волатильними є економічні дані та циклічність волатильності. Волатильність – статистичний фінансовий показник, для розрахунку якого застосовується середньоквадратичне відхилення.[25] Межі полос Боллінджера розташовані на відстані, що дорівнює двом стандартним відхиленням від ковзаючого середнього. Лінія спротиву – на два стандартних відхилення вище, а лінія підтримки – на два стандартних відхилення нижче простої ковзаючої середньої з вікном 20 днів. За означенням:

Середня лінія: ;

Лінія спротиву: ;

Лінія підтримки: .

В період високої волатильності смуги розширюються, в період низької – звужуються, утримуючі ціни в своїх межах [26].

д) індикатор Ішимоку (Ichimoku indicator)

Індикатор Ішимоку розроблений для прогнозування руху фондового індексу Японії. Призначений для виявлення тренду, ліній підтримки і генерації сигналів до купівлі/продажу [27].

Для для побудови індикатора Ішимоку використовують три проміжки часу, рекомендовані розробником індикатора:

* короткий ;
* середній ;
* довгий .

Індикатор складається з ліній:

* лінія перевороту (conversion line) (формула 3.4):
* лінія стандарту (base line) (формула 3.5):
* спан А (span A) (формула 3.6):
* спан В (span B) (формула 3.7):

е) індекс маси (mass index)

Індекс маси призначений для виявлення розворотів тенденції на основі змін ширини діапазону між максимальною і мінімальной цінами. Якщо діапазон розширюється, індекс маси зростає, якщо звужується - індекс зменшується [22].

Для розрахунку індекса маси необхідно:

* розрахувати 9-ти денне EMA різниці між максимальними та мінімальними цінами;
* розрахувати 9-ти денне ЕМА за отриманим вище ЕМА, тобто розрахувати подвійне експоненційне згладжування;
* знайти суму для 25 періодів (формула 3.8):

ж) TRIX

TRIX це динамічний індикатор [22], який показує відносну швидкість зміни тричі експоненційно згладженого змінного середнього цін закриття. Потрійне експоненціальне згладжування використовується для відсіювання «незначних» циклів (тривалість яких коротше обраного числа періодів). Для розрахунку TRIX необхідно знайти тройне експоненційне згладжування та розрахувати відносну зміну значення за 1 день (формули 3.9 – 3.12).

де – експоненційне ковзаюче середнє;

– подвійне експоненційне ковзаюче середнє;

– потрійне експоненційне ковзаюче середнє;

- коефіцієнт згладжування, що має бути в межах від 0 до 1. Також коефіцієнт може бути представлений як , де і має значення розміру вікна згладжування.

### Осциляторні індикатори

Осциляторні індикатори – інший тип технічних індикаторів. Дає змогу передбачити момент зміни напрямку тренду. Застосовані такі індикатори:

а) стохастичний осциллятор (Stochastic Oscillator)

Стохастичний осциллятор – це індикатор технічного аналізу, який є відношенням актуальної ціни (close) до деякого діапазону цін за проміжок часу [22]. Представляє собоє дві лінії. Головна лінія має назву швидкий стохастик %K , друга лінія – повільний стохастик %D – ковзаюче середнє лінії %K за певний проміжок часу. Формула розрахунку (3.13):

де – актуальна ціна певного періоду;

– мінімальна ціна за останні n періодів;

– максимальна ціна за останні n періодів.

є ковзаючою середньою стосовно з невеликим періодом усереднення.

б) індекс відносної сили (Relative strength index)

Індекс відносної сили (RSI) - це індикатор імпульсу, який використовується в технічному аналізі. Він визначає силу тренда та ймовірність його зміни.

в) збіжність/розбіжність ковзаючих середніх (MACD)

Індикатор MACD – технічний індикатор, розроблений фінансистом  Джеральдом Апелем. Використовується для оцінки та прогнозування цін на валютних біржах, використовується для дослідження напряму тренду. Розраховується як різниця між 12-ти та 26-ти денним ковзаючим середнім. Отримана величина може бути вище або нижче нуля. Якщо МАСD вище нуля - значить, 12-ти денне ковзаюче середнє більше 26-ти денного. Якщо МАСD падає нижче нуля - значить 12-ти денне ковзаюче середнє менше 26-ти денного [22]. Визначається формулою (3.14):

Зазвичай сигналом до покупки активів вважають момент, коли лінія MACD перетинає знизу вгору сигнальну лінію (SMA 12-ти денного періоду від різниці двох SMA (12 та 26). Сигналом до позбуття активу вважають, коли лінія MACD перетинає сигнальну лінію зверху вниз.

г) швидкість зміни цін і моментум (ROC and Momentum)

Моментум обчислюється за формулою (3.15) як різниця між ціною в момент t та ціною n періодів назад.

Швидкість зміни цін обчислюється як порівняння поточної ціни з ціною попереднього періоду, віддаленого від поточного на n періодів і показує процентну зміну ціни від одного періоду до іншого [28].

д) середньозважена ціна за обсягом (VWAP)

У сфері фінансів середньозважена ціна за обсягом (VWAP) - це відношення вартості проданої угоди до загального обсягу торгів за певний часовий період (зазвичай один день) [29]. Це міра середньої ціни, по якій торгується акція протягом торгового періоду. Обчислюється за формулою (3.16):

де j – розмір вікна спостереження.

е) індекс грошового потоку (MFI)

Індекс грошового потоку (MFI) - це технічний осцилятор, який використовує дані про ціну і обсязі для визначення сигналів перекупленности або перепроданості активу. Його також можна використовувати для виявлення розбіжностей, що попереджають про зміну цінового тренда. Осцилятори переміщається між 0 і 100 [22].

## BERT для аналізу настроїв

Окрім історичних торгових даних показників схожих компаній, цін на сировинні товари, курсів іноземних валют, економічних індикаторів, цінних паперів з фіксованим доходом, зміни в ціні акцій певної компанії можуть відображати фінансові новини. Багато інвесторів слідкують за новинами і, почасти, приймають рішення про інвестиції базуючись в тому числі на новини, пов’язані з об’єктом інвестицій. Визначена компанія, ціни акцій якої будуть прогнозовані в ході виконання дипломного проекту – інвестиційний банк Goldman Sachs Group [30].

Модель BERT (Bidirectional Encoder Representations from Transformers), тобто двоспрямовані кодувальні представлення з Transformers – це техніка машинного навчання для предтренування обробки природної мови NLP, розроблена дослідниками Джейкобом Девлиним, Мін-Вэй Чангом, Кентоном Лі та Кристиною Тутановою [31].

BERT призначена для попереднього навчання на немаркованих текстових даних шляхом співставлення лівого та правого контексту. Завдяки цьому, при додаванні лише одного вихідного шару, модель може отримувати гарні результати для різноманітних задач, починаючи від прогнозування наступного речення та додавання пропущених слів в правильному контексті до відповідей на запитання та аналізу настроїв в тексті.

При реалізації моделі BERT існує два етапи: попередня навчання (pre-training) та налаштування (fine-tuning). Під час попереднього навчання модель тренується на нерозмічених даних у складі різних задач. Під час налаштування BERT ініціалізується з параметрами, отриманими в ході попереднього навчання, а потім параметри налаштовуються на розмічених даних. Великою перевагою архітектури моделі BERT є те, що вона уніфікована для розв’язання різних задач. Між наперед навченою архітектурою і фінальною є мінімальні відмінності. Наприклад, за виключенням вихідних шарів, одна й та сама архітектура використовується для переднавчання та для налаштування моделі. Як було зазначено вище, при переднавчанні модель отримує параметри, при налаштуванні параметри оптимізуються. Даний процес представлено на рисунку 3.1. [CLS] – це спеціальне позначення, яке додається на початку кожного вхідного масива, [SEP] – це позначення розділення (наприклад, двох речень, або питання та відповіді)

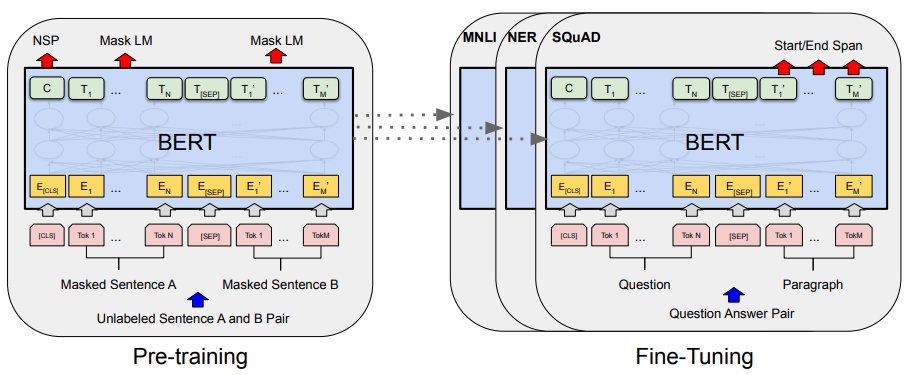


Рисунок 3.1 – Попереднє навчання та налаштування моделі BERT [31]

Архітектура моделі BERT є багатошаровий двоспрямований енкодер-трансформері (multi-layer bidirectional Transformer encoder), що, в свою чергу, є архітектурою для моделювання послідовної інформації (альтернатива RNN). Початкова реалізація архітектури трансформера описана в [32].

В моделі BERT використовується алгоритм токенізації підслів WordPiece, суть якого була викладена в роботі [34]. WordPiece спочатку ініціалізує словник, щоб включити кожен символ, присутній в навчальних даних, і поступово вивчає задану кількість правил злиття. WordPiece вибирає не найчастішу пару символів, а ту, яка максимізує ймовірність того, що навчальні дані будуть додані в словник.

Для даного токена вхідне представлення створюється шляхом суми відповідних вкладень токена, сегмента і позиції. Візуалізацію цієї конструкції можна побачити на рисунку 3.2.



Рисунок 3.2 – Вхідне представлення BERT [31]

Під час попереднього навчання двосторонній аналіз контекстів на всіх шарах створює деякі обмеження, тому для переднавчання моделі використовують масковану язикову модель (Masked Language Model MLM) [33]. Суть MLM полягає в тому, що замість прогнозування наступного слова на основі попереднього, як було б в язиковій моделі на основі LSTM або інших стандартних односпрямованих моделях, «маскується» 15% всіх токенів WordPiece в випадковому порядку, які потім прогнозуються. Це дозволяє отримати двоспрямовану наперед навчену модель.

BERT має дві версії:

* 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.

В роботі було застосовано версію BERT-base-uncased. Uncased означає, що текст повинен бути приведений до нижнього регістру перед токенізацією.

Для семантичного аналізу настрою тексту використана модель financial BERT (finBERT) [35]. FinBERT це преднавчена NLP модель для аналізу настрою в фінансовому тексті на основі BERT. Класифікація настроїв відбувається завдяки додаванню шару Dense (повнозв’язного нейрону з нелінійною функцією активації GELU [65]) після останнього схованого стану токена [CLS]. Це продемонстровано на рисунку 3.3. Після цього нейронна мережа навчається на розміченому датасеті.

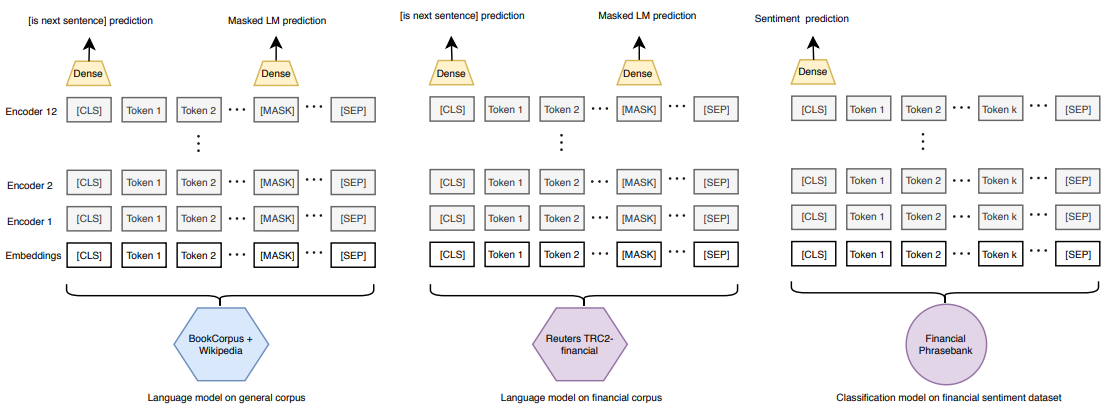


Рисунок 3.3 – Перший і другий етап переднавчання та налаштування моделі finBERT [35]

Для преднавчання моделі використано велика кількість фінансового тексту з Financial PhraseBank. Financial PhraseBank складається з великої кількості речень на англійській мові, які було випадково вибрано з фінансових новин бази даних LexisNexis [36] та розмічено експертами з досвідом роботи у фінансовій та бізнес-сфері.

## Перетворення Фур’є

Перетворення Фур'є застосовується при аналізі часових рядів з метою виділення локальних та глобальних трендів, а також зменшення шуму в даних. В даній роботі було проведено спектральний аналіз часового ряду прогнозованих даних для створення корисних для моделі ознак.

Коливання (або осциляції) широко розповсюджені в природі та технніці. Низькочастотні вільні коливання мають синусоїдальну форму, наприклад, рух маятника, гойдалок або конструкції мосту після збудження. Синусоїдальні коливання в загальному випадку описуються формулою (3.17):

де – амплітуда;

– частота – період;

– циклічна частота;

– фаза.

Деякі властивості [38] синусоїдального сигналу:

* синусоїдальний сигнал є стаціонарним, оскільки всі його параметри є сталими у часі;
* синусоїдальний сигнал є періодичним;
* визначений на інтервалі , тобто є теоретичною абстракцією;
* синусоїдальний сигнал є плавно змінюваною функцією, не має розривів в похідних будь-якого порядку, тому при будь-якій кінцевій кількості сигналів неможливо відтворити розриви.

Складні сигнали будь-якої форми можливо представити набором синусоїдальних коливань – гармонік. В сумі такий набір – спектр – представляє собою складні коливання.

Існує можливість гармонічного синтезу довільних періодичних залежностей, які задовольняють умовам Діріхле [39] на проміжку . Тригонометричний ряд [37] виду (формула 3.18):

є тригонометричним рядом Фур’є функції f, якщо для функції f мають сенс інтеграли, що знаходяться за формулами Ейлера-Фур’є (3.19) і (3.20):

Розкладання функції на гармонійні складові, тобто знаходження коефіцієнтів Фур’є, називається спектральним аналізом. Відтворення функції, представленої рядом Фур’є, називається спектральним синтезом.

Вищезазначений синтез (3.18) справедливий для періодичних функцій та сигналів, а ряд цін акцій має стохастичну природу, тобто неперіодичний. Існує два підходи для перетворення неперіодичних сигналів.

Перший полягає в тому, що якщо функція або сигнал лежать в скінченному проміжку часу, то можна вважати сигнали періодичними і застосувати описаний вище синтез. Виникає проблема розриву на початку та в кінці сигналу, що має негативні наслідки [38].

Інший підхід передбачає перехід до неперервного перетворення Фур’є в загальному вигляді для довільного сигнала. Пряме перетворення Фур’є в такому випадку дозволяє отримати функцію частоти від функції часу . Реалізується за формулою (3.21):

де – функція незалежної змінної часу t.

Перетворення (3.21) є абстракцією, оскільки не визначено межа інтегрування. Було введене поняття поточного частотного спектра, де верхня межа була замінена на значення в певний період часу. Також перейшли до функції , що означає спектральну щільність сигналу (3.22) [38]:

Вираз (4.6) можна представити в показниковій формі (3.23):

де модуль спектральної щільності (3.24):

і аргумент (фаза) (3.25):

Синусоїдна та косинусоїдна складові в (3.24) та (3.25) розраховуються як (3.26):

Якщо спектр визначений на кінцевому інтервалі часу t, то справедливі формули, які використовуються для періодичних сигналів. Отже, будь-який сигнал, який визначений на проміжку часу t, можна розкласти на кінцеве число гармонік, і чим більше це число, тим точніше розкладення.

Зворотнє перетворення Фур’є визначається формулою (3.27), що дозволяє за функцією знайти в аналітичному вигляді:

Були запропоновані методи швидкого перетворення Фур’є (Fast Fourier Transform FFT) для того, щоб збільшити швидкість і зменшити складність обчислень. Результат досягається завдяки спеціальній техніці комбінування функцій та врахуванню періодичності тригонометричних функцій. Таким чином, складнысть алгоритму FFT складає O(N logN) [38].

Для використання в обчислювальній техніці, для обробки цифрових сигналів, функція повинна бути визначена на дискретній множині замість неперервної області. В такому випадку використовується дискретне перетворення Фур’є (DFT), яке визначено формулою 3.28:

В результаті перетворення Фур’є отримаємо результат в комплексних числах, які найчастіше представляють у вигляді модуля та аргументу, де модуль (3.29):

та аргумент (фаза) (3.30):

Ці складові є осциляторними функціями, також модуль та аргумент мають зміст амплітуди і частоти.

Зворотнє перетворення Фур’є має вигляд (3.31):

## Метод головних компонент PCA

В ході виконання роботи було зібрано велику кількість даних. Для покращення якості і зменшення терміну навчання моделі необхідно понизити розмірність вхідних даних та прибрати серійну кореляцію між ознаками з мінімальною втратою інформації в даних. Виходячи з цього, було прийнято рішення застосувати метод головних компонент (Principal Component Analysis PCA) [41].

Формально задача переходу від початкового набору ознак до нового набору ознак меншої розмірності , де з мінімальними втратами корисної інформації можна описати наступним чином. Нехай – це вектор-функція початкових змінних розмірністю . Тоді – задана міра інформативності -мірної системи ознак . Вибір функціонала націлений на збереження якомога більшої інформації початкових даних [42].

Загальну оптимізаційну постановку задачі отримання інформативних даних меншої розмірності можна сформулювати як визначення такого набору ознак , знайдених в допустимих перетвореннях початкового набору , що (флормула 3.32):

Клас допустимих лінійних перетворень складається з будь-яких ортонормованих комбінацій початкових показників, тобто вектори довжиною 1 та ортогональні один до одного [42] (формула 3.33):

за умови ортонормованості векторів (3.34), (3.35):

де – математичне сподівання . Оскільки в роботі дані центровані, то .

В якості міри інформативності -мірної системи є вираз (3.36):

де – обчислення дисперсії відповідної випадкової величини.

Маючи початкові дані розмірністю та отримавши коваріаційну матрицію [43] розмірністю , можна знайти власні числа цієї матриці

Маючи загальну оптимізаційну постановку задачі отримання інформативних даних меншої розмірності (5.1), аналізовану ознаку з коваріаційною матрицею [42] розмірністю , визначимо критерій інформативності за допомогою (3.36) та клас допустимих лінійних перетворень за допомогою (3.33 – 3.35). При будь-якому фіксованому з 1…p, вектор головних компонент визначається як лінійна комбінація (3.37):

де L – матриця (3.38) і її строки задовольняють умові ортогональності.

Отримані і є головними компонентами вектора X.

Обчислення головних компонент відбувається шляхом знаходження положення власних векторів – елементів матриці L – в ортогональній проекції на які дисперсія буде максимальною. Знаходження власних чисел та власних векторів коваріаційної матриці описане в [40].

Характеристичне рівняння для матриці має вигляд (3.39):

де – одинична матриця.

За умови симетричності та невід’ємної визначеності коваріаційної матриці (будь-яка коваріаційна матриця має такі властивості) [42] відомо, що рівняння (5.8) має p невід’ємних кореней , які називаються характеристичні або власні корені матриці .

Для того, щоб забезпечити максимальне значення дисперсії, необхідно послідовно обирати з р власних векторів найбільші з залишених. Наприклад, підставляючи в систему рівнянь (3.39) і розв’язуючи її відносно першого рядка матриці L визначаємо компоненти вектора (перший рядок матриці L).

Таким чином, головна компонента отримується як лінійна комбінація (3.37) , де – власний вектор матриці , що відповідає k-му за величиною власному числу цієї матриці [42].

Критерій інформативності метода головних компонент може бути представлений у вигляді формули (3.40):

де - розмірність нового набору ознак меншої розмірності .

## Довга-короткострокова пам’ять LSTM

Як було зазначено у підрозділі 2.4, в якості моделі для побудови прогнозу фінансових числових рядів було обрано такий різновид рекурентних нейронних мереж RNN як довга короткострокова пам’ять LSTM.

Один з методів [47] реалізації рекурентної нейронної мережі – багато-до-багато, тобто many-to-many, що дозволяє подавати послідовності на вхід моделі та отримувати послідовності в результаті прогнозування (рис. 3.4). Такий варіант був використаний в ході виконання роботи.

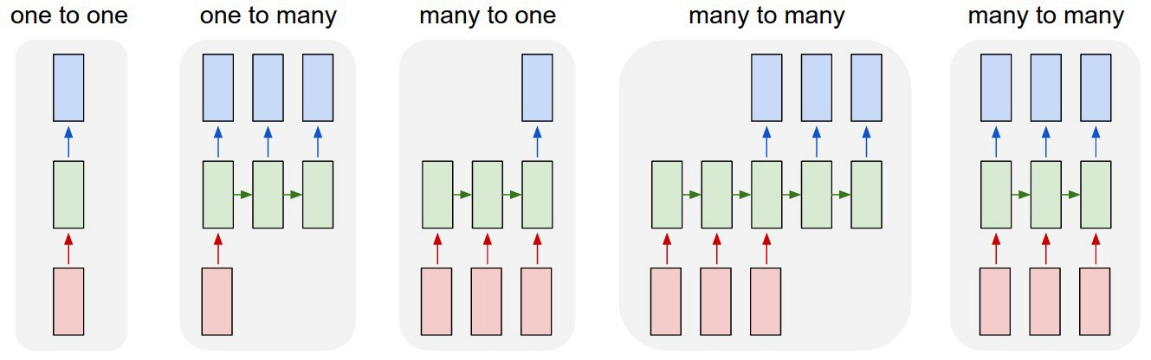


Рисунок 3.4 – Типи реалізації рекурентних нейронних мереж [45]

Архітектура довгої короткострокової пам’яті LSTM була розроблена для уникнення проблеми довгострокової залежності. Однією з найпривабливіших переваг RNN є можливість пов’язування попередньої інформації з поточною. Але існують певні обставини, коли нейронна мережа втрачає таку властивість. Якщо для виконання поточної задачі необхідна недавня інформація, дистанція між якою і місцем, де вона знадобилась, є невеликою, то класична рекурентна нейронна мережа може навчитися взаємодії з попередніми даними. Коли дистанція між необхідною інформацією та місцем її застосування зростає, то виникає проблема втрати довгострокового зв’язку (рис. 3.5).

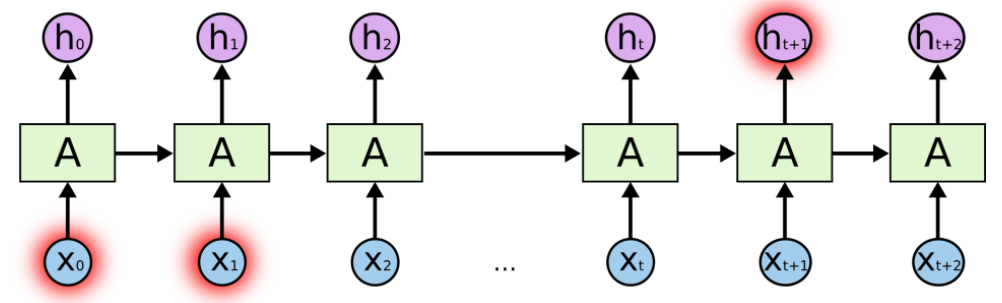


Рисунок 3.5 – Демонстрація проблеми довгострокової залежності [46]

В класичній архітектурі RNN проблему довгострокової залежності спричиняють такі явища, як проблема зникаючого [48] та зростаючого градієнту (Vanishing and Exploding gradient).

Інформація проходить 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 (на рисунку 3.6). Якщо часткова похідна похибки << 1, то при множенні на learning rate зміна градієнта буде незначною. Чим далі просування по мережі, тим нижче градієнт і складніше тренувати ваги (рисунок 3.5).

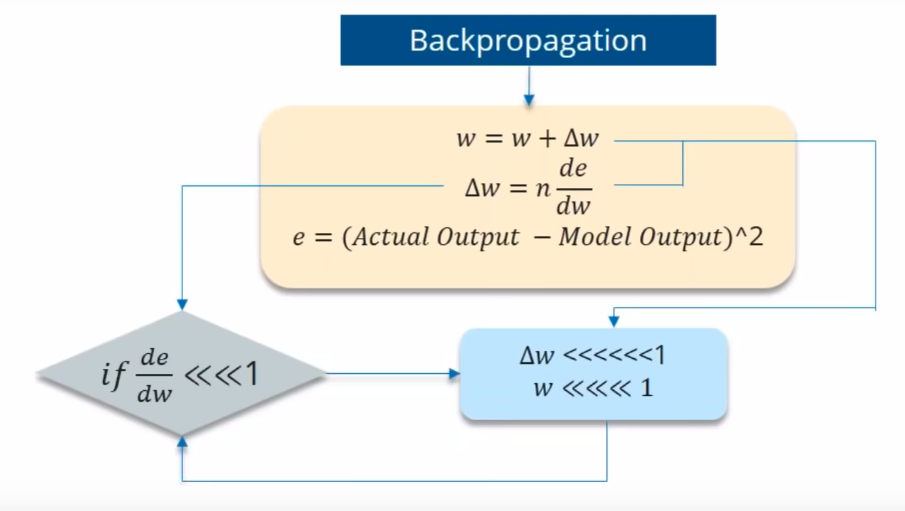


Рисунок 3.6 – Проблема зникаючого градієнту при оновленні вагів [45]

Проблема зростаючого градієнту наступає тоді, коли алгоритм надає вагам дуже великі значення. Великі градієнти помилок накопичуються і призводять до дуже великих оновлень вагів в ході тренування. Це призводить до нестабільності моделі і неможливості навчання на тренувальних даних (рисунок 3.7).

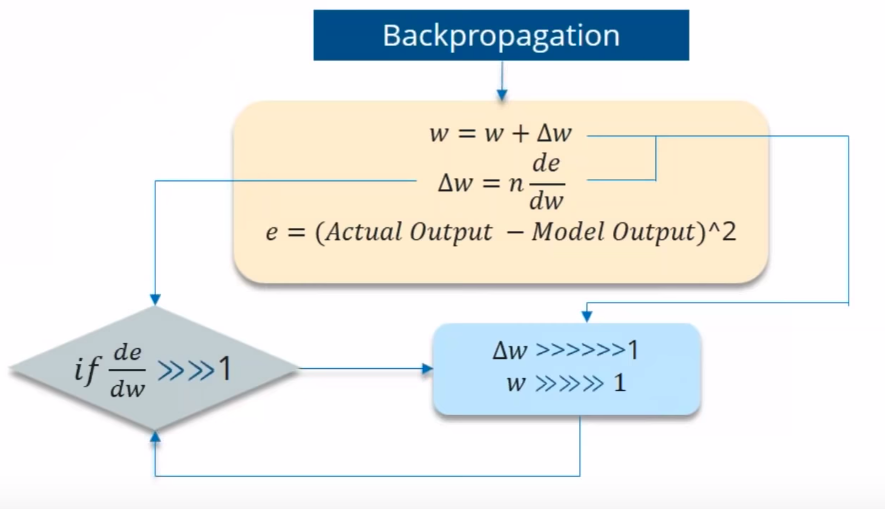


Рисунок 3.7 - Проблема зростаючого градієнту при оновленні вагів [45]

При прогнозуванні фінансової інформації довгострокові зв’язки грають важливу роль, тому однією з головних задач є забезпечення для моделі можливості їх встановлювати. Саме тому для використання в дипломній роботі було обрано модифікацію рекурентних нейронних мереж – LSTM. Дана архітектура була представлена дослідниками Зеппом Хохрайтером та Юргеном Шмідхубером в 1997 році [44] і в подальшому багато разів була модернізована. LSTM розроблені спеціально для стабільної роботи з довгостроковими залежностями. Мережа представляє собою ланцюг з модулей, кожен з яких має в собі чотири шари нейронної мережі, що взаємодіють один з одним особливим чином (рис. 3.8).

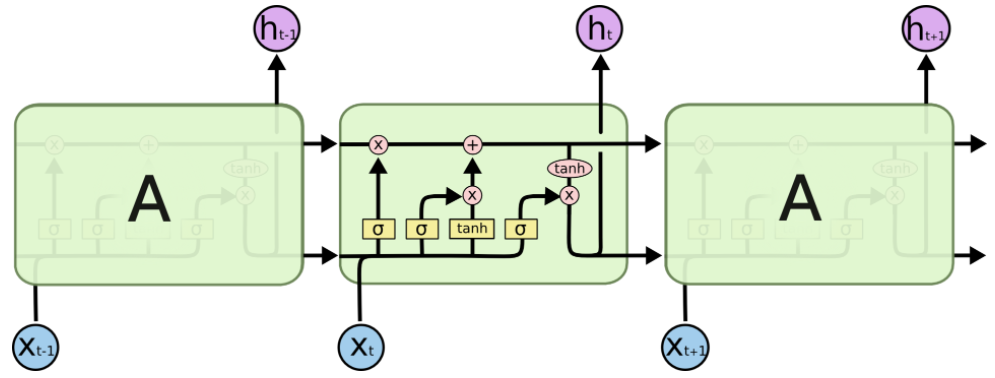


Рисунок 3.8 – Модуль, що повторюється, складається з 4 шарів [46]

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

та гіперболічний тангенс (формула (3.42):

Ключовим компонентом LSTM – це стан ячейки (cell state), що редставляє собою горизонтальну сполучну лінію, що проходить крізь весь ланцюг. Інформація переміщується по ньому всередині мережі (рис. 3.9).

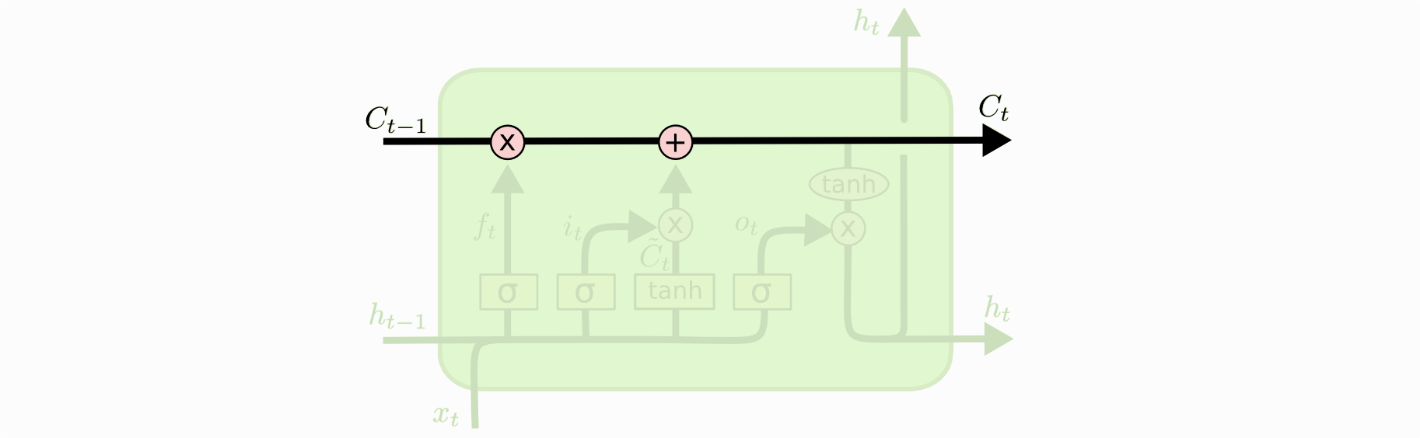


Рисунок 3.9 – Стан ячейки (cell state) – елемент модуля LSTM [46]

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.

### Forget gate layer

Forget gate layer – шар фільтру забування з сігмоїдальною функцією активації (3.41). Призначений для того, щоб визначити яку інформацію слід видалити зі стану ячейки [44, 49]. Він приймає на вхід актуальну інформацію та результат роботи попереднього модуля і повертає число в діапазоні [0, 1] для кожного числа зі стану ячейки (формула 3.43, рис. 3.10).

де – вихідне значення шару фільтру забування;

– актуальна інформація на вхід;

– результат роботи попереднього модуля;

– сигмоїдальна функція активації;

– вага входу шару фільтру забування;

– вага зміщення шару фільтру забування.

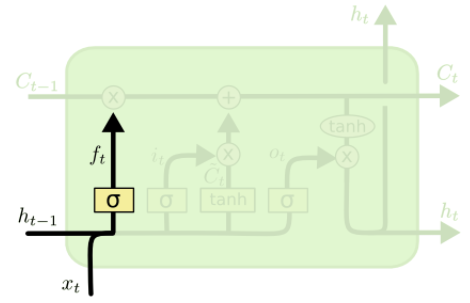


Рисунок 3.10 – Шар фільтру забування [46]

### Input layer gate

Input layer gate – шар фільтру входу. Складається з двох частин: сигмоїдального та tanh-шару [49]. Сигмоїдальний шар відповідає за значення, які необхідно оновити (формула 3.44), Tanh-шар будує вектор значень , які додаються до стану ячейки (формула 3.45). Принцип проілюстровано на рисунку 3.11.

де – вихідне значення сигмоїдального шару;

– вихідне значення Tanh-шару;

– сигмоїдальна функція активації;

– вага входу сигмоїдального шару фільтру входу;

– вага зміщення сигмоїдального шару фільтру входу;

– актуальна інформація на вхід;

– результат роботи попереднього модуля;

– вага входу Tanh-шару фільтру входу;

– вага зміщення Tanh-шару фільтру входу.

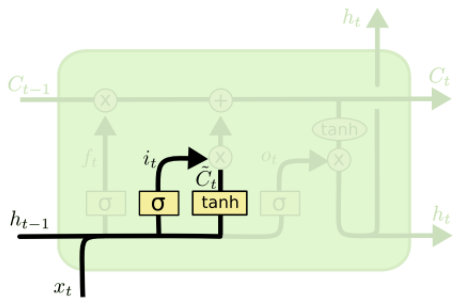


Рисунок 3.11 - Шар фільтру входу [46]

### Оновлення стану ячейки

При зміні старого стану ячейки на новий стан , відбувається послідовно множення старого значення на вихідне значення шару фільтру забування та додання виходу шару фільтру входу, яке обчислюється як (формула 3.46). Таким чином отримується нове значення стану ячейки [49]. Схематично процес показано на рис. 3.12.

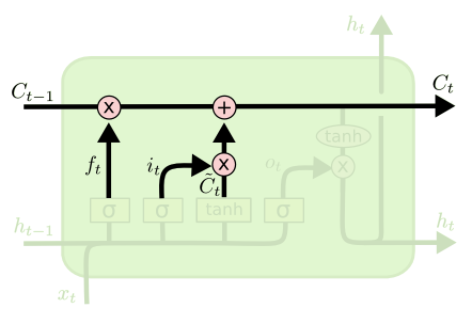


Рисунок 3.12 – Оновлення стану ячейки [46]

### Output gate layer

Шар фільтру виходу відповідає за значення, яке отримується в результаті роботи модуля нейронної мережі [49]. Вихідні дані базуються на додатку (формула 3.47) стану ячейки , обробленому функцією активації tanh з вихідними значеннями [-1; 1], та сигмоїдального шару (формула 3.48), який вирішує, яку інформацію виводити зі стану ячейки.

де - вихідне значення сигмоїдального шару;

- сигмоїдальна функція активації;

– функція активації гіперболічний тангенс;

- вага входу сигмоїдального шару фільтру виходу;

- вага зміщення сигмоїдального шару фільтру виходу;

– актуальна інформація на вхід;

– результат роботи попереднього модуля;

– результат роботи поточного модуля.

Принцип роботи шару виходу продемонстровано на рисунку 3.13.

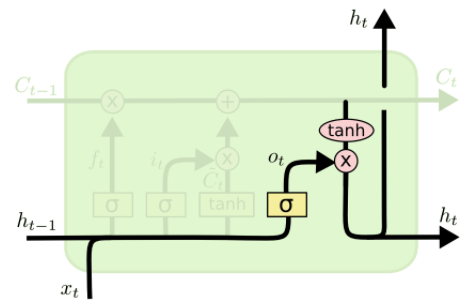


Рисунок 3.13 – Шар фільтру виходу [46]

## Висновки до розділу

В розділі 3 було описано математичне забезпечення, яке було використане в ході виконання роботи. Математичні алгоритми та моделі: методи технічного аналізу для створення тенденційних та осциляторних технічних індикаторів, модель BERT для класифікації настроїв фінансових новин, перетворення Фур’є для проведення спектрального аналізу та виділення локальних та глобальних трендів і зменшення шуму в даних, метод головних компонент PCA для зменшення розмірності даних, видалення серійної кореляції та виділення більш інформативних ознак, модифікацію рекурентних нейронних мереж модель довгої-короткострокової пам’яті LSTM для навчання на отриманих даних та для прогнозування фінансового часового ряду.

В наступному розділі розглянута практична програмна реалізація даних методів в рамках дипломної роботи.

# ПРОГРАМНА РЕАЛІЗАЦІЯ

## Етапи виконання роботи

В роботі була здійснена спроба спрогнозувати рух цін акцій компанії Goldman Sachs Group. Для цього було досліджено пошук і дослідження можливих корельованих активів, їх збір. За допомогою web-сервісу Google Trends [55] визначено щоденну частоту запитів користувачів щодо компанії. Створено технічні індикатори. Отримано щоденні фінансові новини, що стосуються компанії з ресурсу новин [58], які класифіковано за настроєм з використанням переднавченої NLP-моделі finBERT. За допомогою перетворень Фур’є отримано ознаки з характерним трендом та без шуму. Розмірність знижено методом головних компонент. Перед використанням в моделі дані досліджено, очищено та візуалізовано, також виконана трансформація для приведення даних у відповідність до вхідних критеріїв моделі. Побудована та навчена LSTM нейронна мережа, отриманий прогноз. Результат оцінений за допомогою метрик та візуалізований. Для покращення сприйняття користувачем розроблено web-застосунок. Послідовність виконаних дій представлено на схемі на рис. 4.1.

Дипломна робота була виконана в середовищі Jupyter Notebook із застосуванням мови програмування Python 3.9. Розробка веб-застосунку відбувалась в інтегрованій середі розробки PyCharm Community з використанням фреймворку для побудови веб-застосунків Dash.



Рисунок 4.1 – Етапи виконання роботи

## Створення ознак

Метою роботи є прогнозування руху ціни акцій компанії Goldman Sachs Group – GS. Необхідно встановити, чи буде ціна іди вгору або опускатися вниз. Для цього необхідно включити в модель якомога більше інформації, описуючи акцію з різних сторін. Використовуємо щоденні показники цін акцій з 01-01-2010 до 01-04-2021. Пізніше по ходу роботи дані будуть розділені не тренувальну, валідаційну та тестову вибірки. В якості ознак зібрані такі показники:

* корельовані активи;
* технічні індикатори;
* класифікований за настроєм текст фінансових новин;
* результати перетворення Фур’є.

Кореляція активів – це міра того, як інвестування рухаються одне по відношенню до іншого на окремому часовому проміжку. Можна виділити такі активи: показники схожих компаній, ціни на сировинні товари, курси іноземних валют, економічні індикатори, цінні папери з фіксованим доходом.

Технічні індикатори – це алгоритми, що дозволяють отримати інформацію про майбутні ціни за допомогою котирувань за певний період. Технічні індикатори, що використані в роботі, описані в підрозділі 3.2.1.

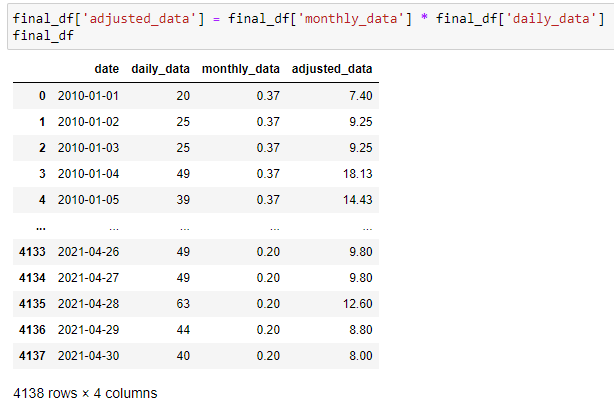
Економічні новини можуть бути індикатором зміни ціни акцій компанії. Аналізуючи текст новин за допомогою NLP можна класифікувати новини для прогнозування настрою ринку.

Використавши перетворення Фур’є, можна позбутися шуму в даних та виділити деякі тенденції.

### Збір корельованих активів

Фінансові дані компаній, курси іноземних валют, ціни на сировинні товари, економічні показники були зібрані з провайдерів фінансової інформації Yahoo! Finance [50], Investing.com [51] та з бази даних економічної інформації Федерального Резерву США [52].

Отримані дані щодо щоденних запитів через веб-сервіс Google Trends. Оскільки дані там представлені в форматі відносного об’єму запитів від 0 до 100, а також дані надаються в щоденному форматі з максимальним періодом в один місяць, то було здійснено запити на отримання щомісячних даних періодом на 10 років та щоденних даних помісячно. Дані зведені в єдиний датасет та створено зважений щоденний показник по запитам (рис. 4.2).



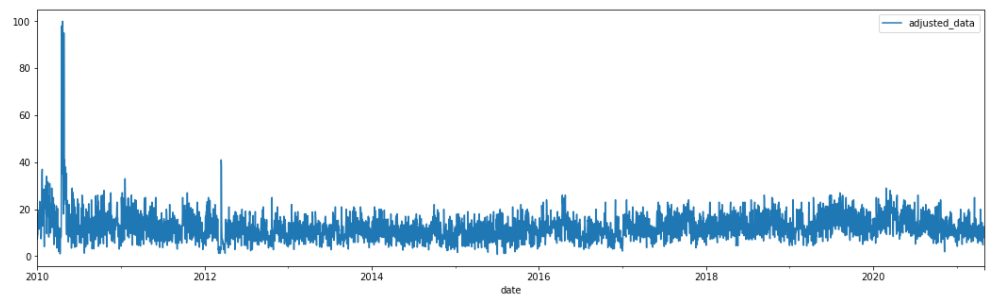


Рисунок 4.2 – Результат перетворень даних для отримання об’єму щоденних запитів з Google Trends.

Формування датасету здійснювалась за допомогою бібліотек мови програмування Numpy [53], Pandas [54] та API для запитів Google Trends – pytrends [56].

### Створення технічних індикаторів

Технічний аналіз був проведений з фінансовим часовим рядом – цінами акцій компанії GS. Отримані індикатори описані в підрозділі 3.2.1. Використані бібліотеки Python: Numpy [53], Pandas [54] та спеціалізована бібліотека для технічного аналізу ta [57]. З бібліотеки ta були імпортовані такі функції: IchimokuIndicator, MassIndex, TRIXIndicator, StochasticOscillator, rsi, VolumeWeightedAveragePrice, money\_flow\_index в силу того, що розробка алгоритму їх реалізації без використання спеціалізованої бібліотеки досить громіздка. Отримані індикатори візуалізовано: наприклад, індикатори тенденції – полоси Болліджера (рис. 4.3) та осциляторні індикатори – індекс маси, стохастичний осцилятор, індекс відносної сили (рис.4.4).

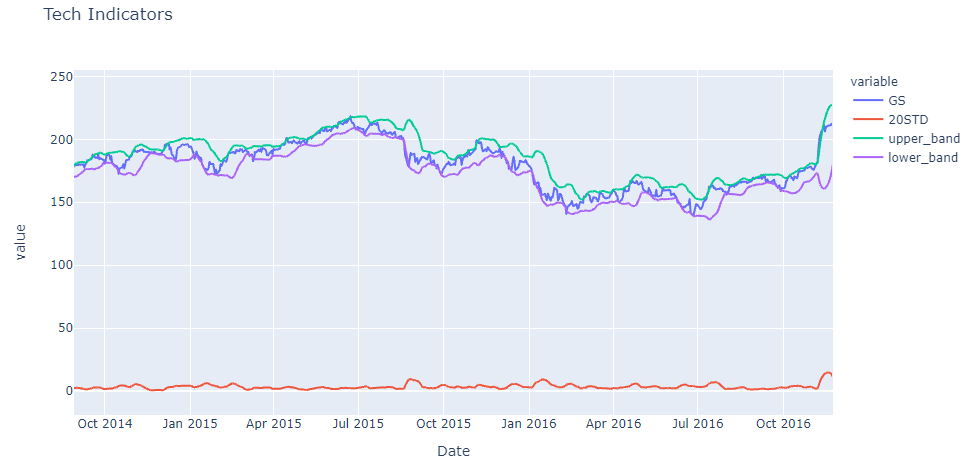


Рисунок 4.3 – Візуалізація індикаторів тенденції

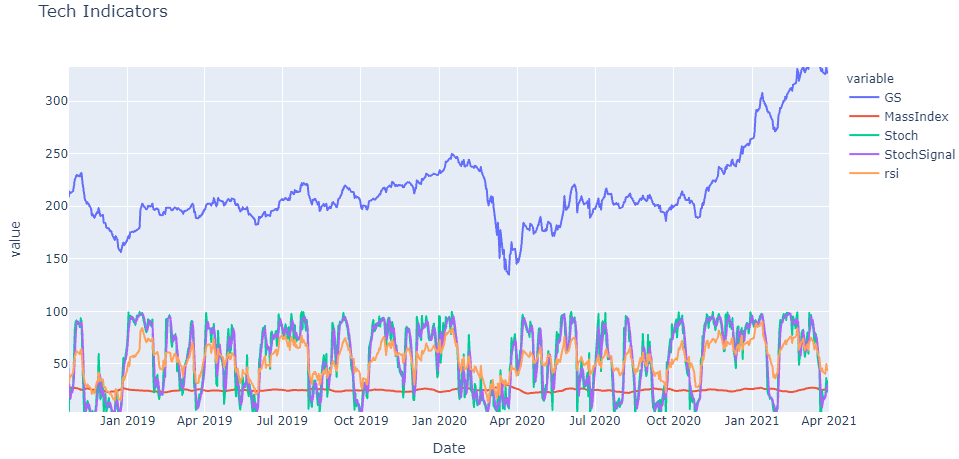


Рисунок 4.4 – Візуалізація осциляторних індикаторів

### Парсинг новин та класифікація finBERT

Серед можливих джерел історичних новин був обраний сайт міжнародної фінансово-аналітичної газети Financial Times [58]. Вибір був зроблений через те, що на даному ресурсі найбільш повне представлення архівних економічних новин. Необхідно було отримати історичні щоденні фінансові новини за період 10-ти років, пов’язані з вищезазначеною компанією. Запроси були розбиті на періоди по пів року через ліміт на самому сайті. В результаті сформовано датасет, який складається з дати, заголовку та короткого змісту новини (рис. 4.5).

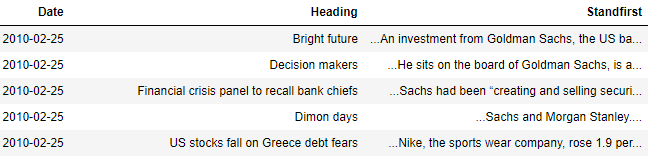


Рисунок 4.5 – Зібрані економічні новини

Для запитів на сайт та роботи з html-розміткою була використана бібліотека BeautifulSoup [59].

Проведена обробка та очистка тексту. Зокрема, за допомогою регулярних виразів видалені пробіли, переходи на нову строку та символи, що не є буквами та цифрами. Також літери приведені до нижнього регістру [60].

В ході застосування NLP (Natural Language Processing) текст новин необхідно семантично класифікувати за настроєм: позитивний, негативний та нейтральний. Класифікація відбулась з використанням ML фреймворку для глибокого навчання PyTorch, моделі BERT (Bidirectional Encoder Representations from Transformers) та преднавченої на фінансових документах NLP-моделі аналізу настроїв FinBERT.

Застосовано модель BERT-Base, Uncased**, що має 12 шарів, 768 прихованих нейронів 12 вихідних нейронів та 110 мільонів параметрів (підрозділ 3.3).**

Для семантичного аналізу настрою тексту використана модель finBERT. FinBERT це преднавчена NLP модель для аналізу настрою в фінансовому тексті.

При імплементації модель потребує встановлення ML фреймворку для глибокого навчання PyTorch [61] та NLP бібліотеку transformers [62], яка пропонує велику кількість переднавчених моделей. При створенні моделі необхідно вказати відповідні конфігурації.

Застосування переднавчених моделей є оптимальним рішенням при умові, що дані не розмічені. В результаті застосування моделі до кожної новини отримані класифіковані дані (3 класи) та макцимальні оцінки класифікації (рис 4.6).

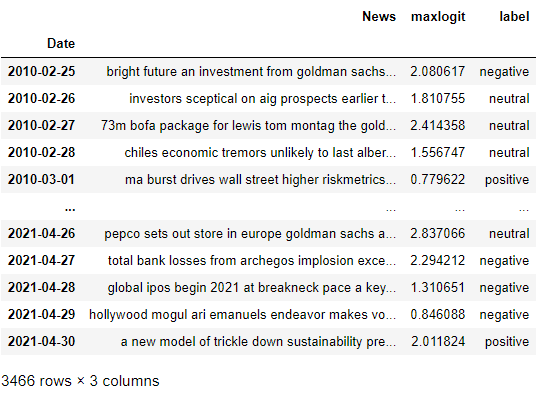


Рисунок 4.6 – Результат класифікації новин з finBERT

Процес класифікації зайняв 42 хвилини 4 секунди.

### Спектральний аналіз (перетворення Фур’є)

Перетворення Фур’є в аналізі часових рядів застосовується для виявлення локальних та глобальних трендів в даних та для позбавлення від шуму в даних. При реалізацій було використано швидке перетворення Фур’є, оскільки цей варіант має значні переваги при застосуванні на обчислювальній техніці (підрозділ 3.4). Для спектральної обробки в бібліотеці Numpy реалізована функція np.fft.fft() для швидкого перетворення і np.fft.ifft() для зворотного перетворення. При маніпуляціях зі спектром, видаляючи гармоніки з більш високою частотою та відновлюючи функцію, можна досягти згладжування даних за рахунок відсутності високочастотної складової. Реалізовано п’ять варіантів перетворення зі збереженням 3, 6, 9, 20, 100 низькочастотних гармонік. Графік отриманих результатів на рис. 4.7.

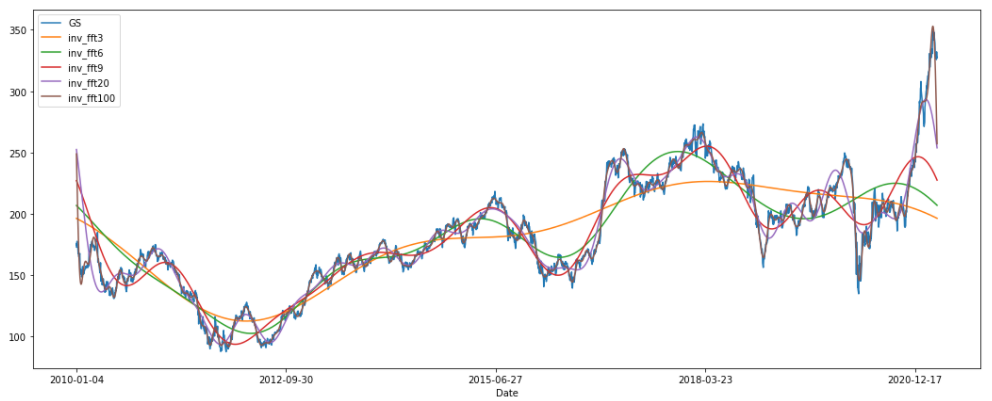


Рисунок 4.7 – Перетворення Фур’є

## Аналіз та підготовка даних

Проведено аналіз отриманих даних, отримано матрицю описових статистик. Розмірність 4106 х 83, пропущених значень нема. Побудовано стовбчасту діаграму, яка демонструє кількість класифікованих нейтрально, негативно та позитивно новин (рис. 4.8).

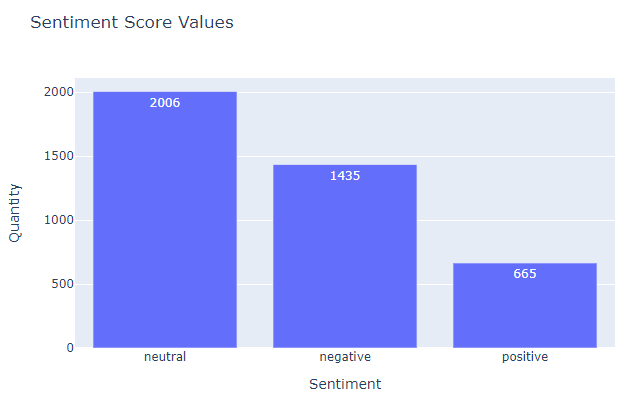


Рисунок 4.8 – Діаграма класифікованих новин

Побудовані кореляційні матриці, візуалізовані в якості heatmap. Оскільки даних багато, на рис. 4.9 приведена лише одна матриця. Можна зробити висновок, що в даних багато множинних кореляцій, які в подальшому буде видалено методом головних компонент.

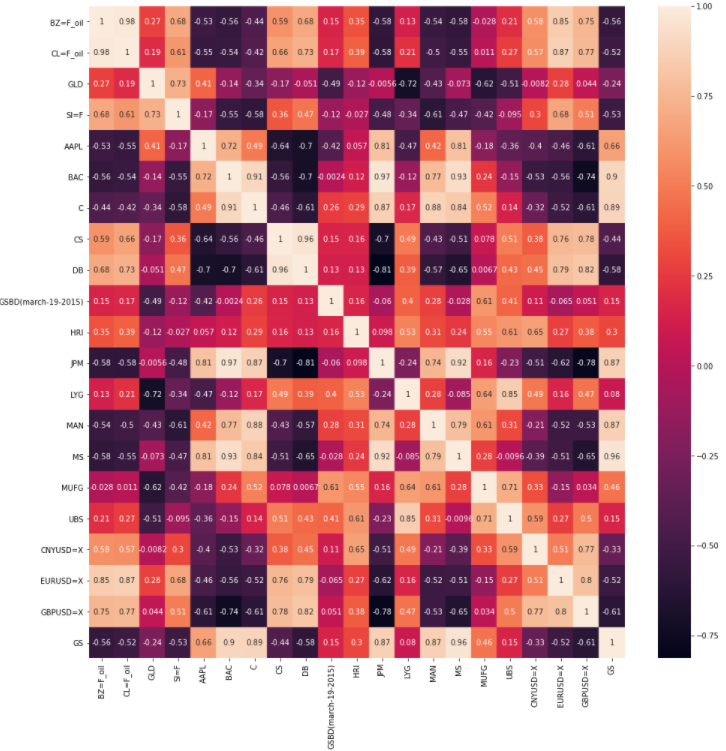


Рисунок 4.9 – Кореляційна матриця ознак (показники компаній)

Побудовані діаграми розмаху окремо для ознак: Природних ресурсів, акцій компаній, курсів валют, економічних індикаторів, індикаторів фінансових бірж (рисунок 4.10 – 4.11).

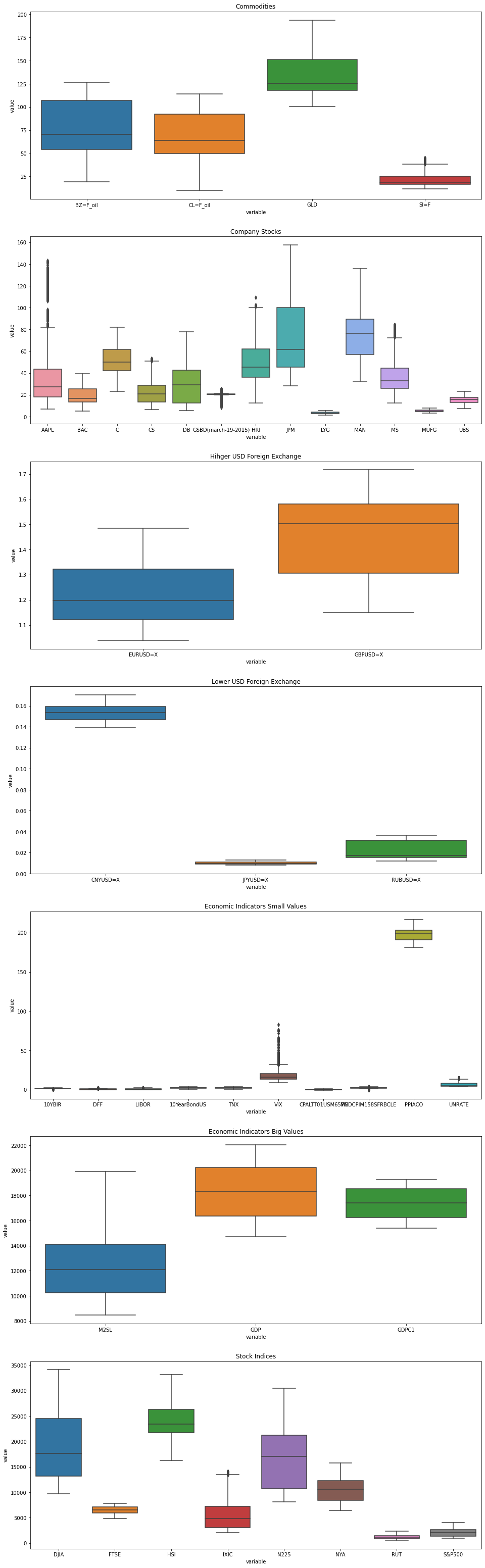


Рисунок 4.10 – Діаграми розмаху для природних ресурсів, акцій компаній, курсів валют

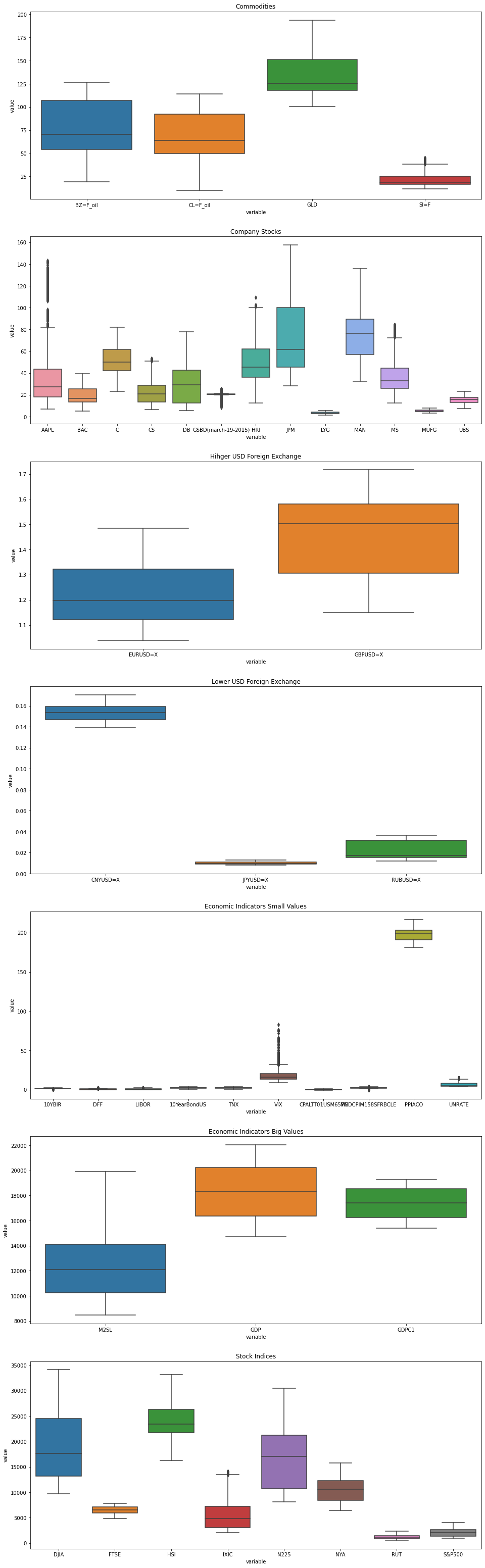


Рисунок 4.11 – Діаграми розмаху для економічних індикаторів, індикаторів фінансових бірж

## Формування виборок

Категоріальні змінні – настрої новин – були перетворені за допомогою метода Label Encoding. Виконана стандартизація (формула 4.1) окремо тренувальних / валідаційних і тестових даних для запобігання впливу міри середнього та показнику розсіювання на окремі вибірки.

де – елемент виборки;

– математичне сподівання;

– середньоквадратичне відхилення.

Оскільки дані стандартизовані і відцентровані відносно 0, можна застосувати метод головних компонент для зменшення розмірності. Крім того багато ознак мають високу кореляцію, якої також можна позбутися (рисунок 4.12).

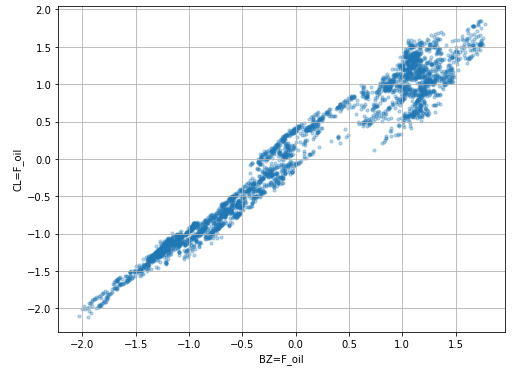


Рисунок 4.12 – Діаграма розсіювання стандартизованих ознак

Обчисливши сумарну пояснювану дисперсію для кожної з нових ознак, прийнято рішення понизити розмірність з 80 до 30 ознак, оскільки це є оптимальною вибіркою для максимального збереження початкових даних (рис. 4.13).

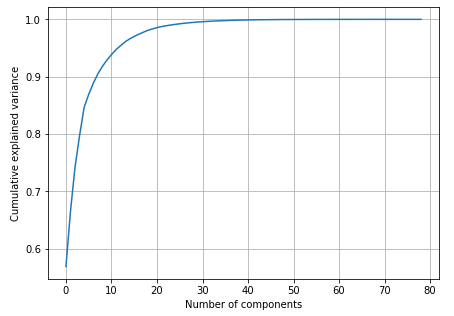


Рисунок 4.13 – Графік сумарної пояснювальної дисперсії

Для того, щоб прогнозування було можливе, необхідно, щоб часовий ряд мав постійні, не змінювані з часом:

* математичне сподівання;
* дисперсію;
* відсутність автокореляції.

Привести ряд до стаціонарного можна за допомогою взяття різниць (формула 4.2).

де n – розмірність початкового ряду.

Розмірність стаціонаризованого ряду буде . Стаціонарність ряду можна перевірити за допомогою статистичного тесту Дікі-Фуллера [63]. Ознаки досліджено на стаціонарність розширеним тестом Діки-Фуллера (рівень значимості 0.05). Всі ознаки, включаючи цільову – стаціонарні.

Оскільки LSTM може приймати на вхід та повертати послідовності, зробимо прогноз на 30 днів, базуючись на попередніх 60 днях. Для цього необхідно переупорядкувати дані. Незалежні змінні розміщені разом з наступними залежними змінними (приклад на рис. 4.14).

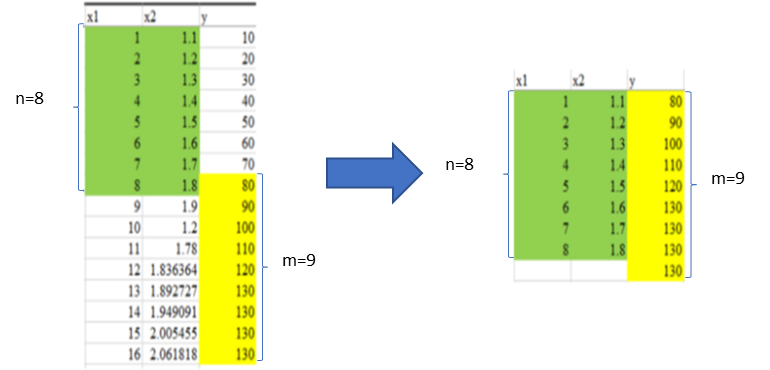


Рисунок 4.14 – Зсув даних у вибірці

З кожною ітерацією вікно здвигається на 1 значення, створюючи окремий елемент послідовності виборок. В результаті для подання в модель отримано дані формату (рис. 4.15)

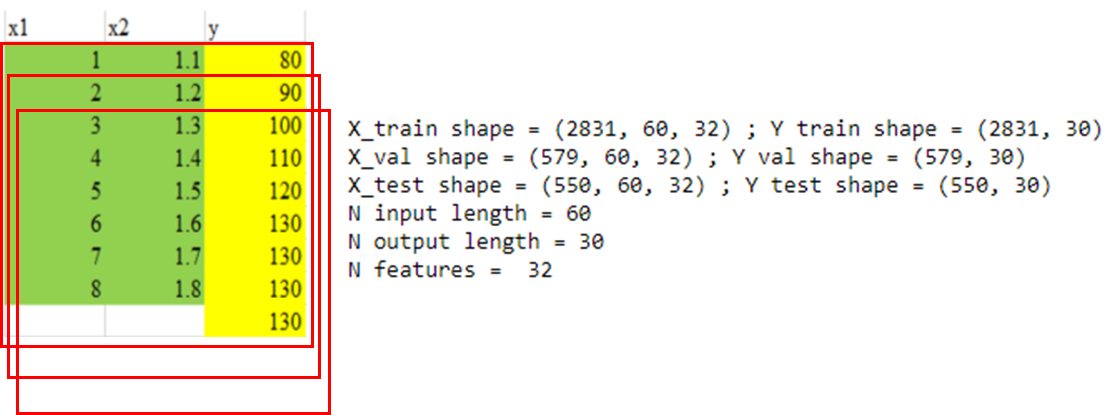


Рисунок 4.15 – Зсув вікна (зліва) та розмірність отриманих даних (зправа)

## Побудова LSTM та оцінка результату

Побудована нейронна мережа типу LSTM. Конфігурація нейронної мережі:

* два приховані шари по 120 вузлів;
* dropout-и 0.2;
* оптимізатор Adam з learning rate 0.0001;
* функція втрат MSE;
* 100 епох;
* batch size 32.

Підсумок моделі (рис. 4.16).

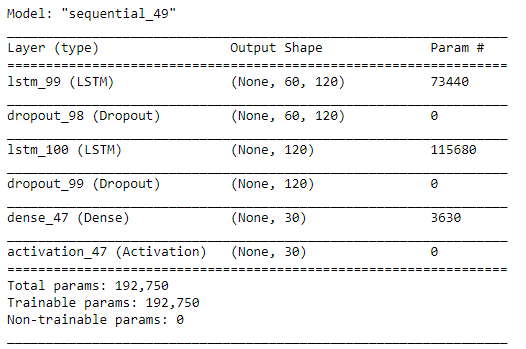


Рисунок 4.16 – Підсумок побудованої моделі

Після навчання та прогнозування отримані прогнозовані дані, які зворотно перетворені до часового ряду. Результати прогнозування візуалізовані (рис. 4.17).

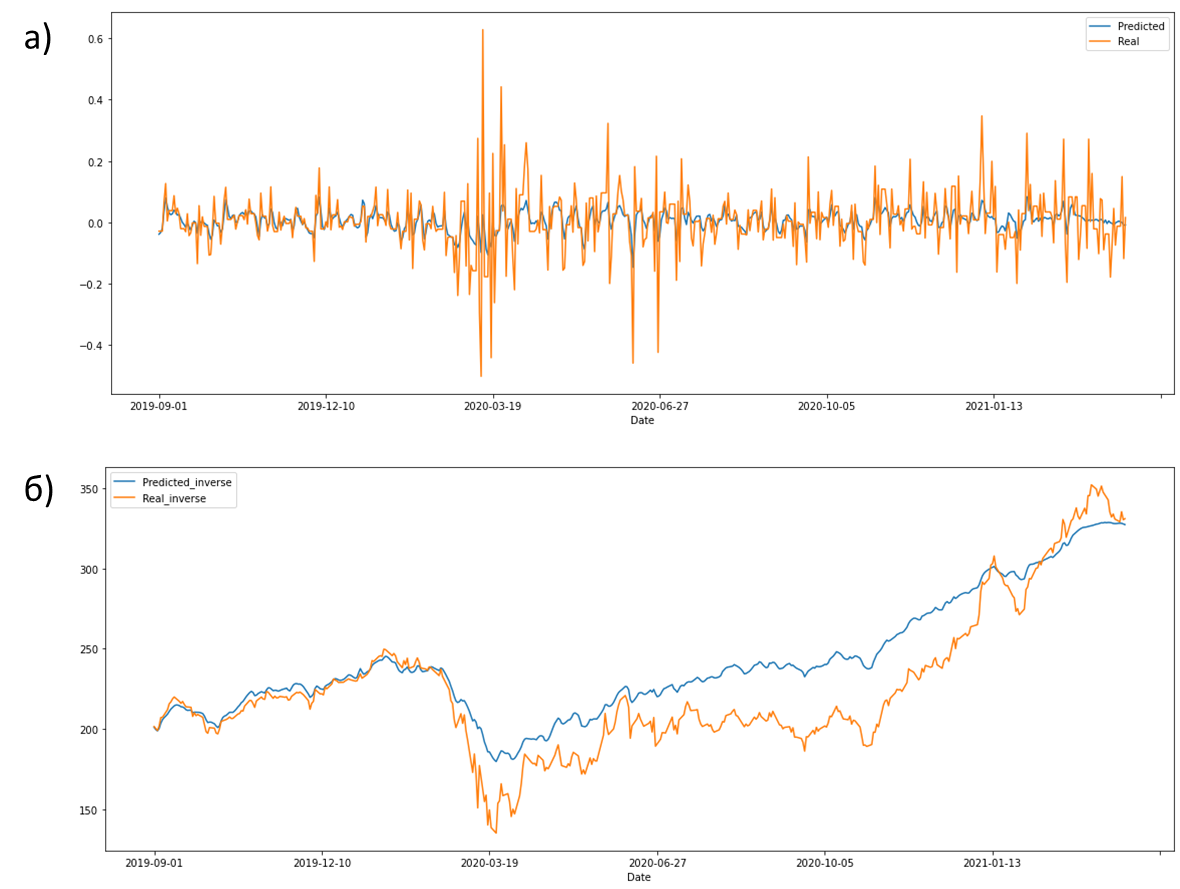


Рисунок 4.17 – Результати прогнозування: а) стаціонарний та стандартизований ряд; б) дані приведені до початкового стану

Підбор гіперпараметрів нейронної мережі проводився у вкладених циклах з послідовню заміною значень та навчанням різних моделей. Для оцінки якості моделей та в процесі підбору гіперпараметрів та оптимальної побудови використовувались такі метрики:

* середній квадрат похибки MSE;
* середня абсолютна похибка MAE;
* коефіцієнт детермінації R2;
* коваріація;
* коефіцієнт кореляції Пірсона.

Результати фінальної моделі приведені в таблиці 4.1.

Таблиця 4.1 - Метрики

|  |  |  |
| --- | --- | --- |
| Метрика | Оцінка отриманих даних | Оцінка зворотньо трансформованих даних |
| MSE | 0.005801 | 529.070971 |
| MAE | 0.046266 | 18.080647 |
| R2 | 0.354147 | 0.729061 |
| Коваріація | 0.002069 | 1506.239586 |
| Коеф. кореляції Пірсона | 0.707208 | 0.930567 |

Реалізація нейронної мережі виконувалась із застосуванням таких бібліотек Python, як Keras, Tensorflow, Statsmodels, Sklearn.

### Алгоритм тренування моделі

Алгоритм дій при побудові та тренуванні нейронної мережі продемонстровано на схемі на рис. 4.18.



Рисунок 4.18 – Спрощена схема алгоритму тренування нейронної мережі

## Графічний інтерфейс

Для спрощення візуальної оцінки користувачем та демонстрації отриманих даних був розроблений веб-інтерфейс. В ньому реалізована можливість використовувати технічні індикатори та їх комбінації та отримання прогнозу щодо поведінки акцій компанії Goldman Sachs. Для розробки був використаний фреймворк мови Python для розробки веб-додатків Dash [64]. Він побудований на основі Flask та Plotly. Dash має відкритий вихідний код і запускається в веб-браузері. Графіки інтерактивні, користувач може взаємодіяти з часовими періодами та масштабом (рис. 4.19, 4.20).

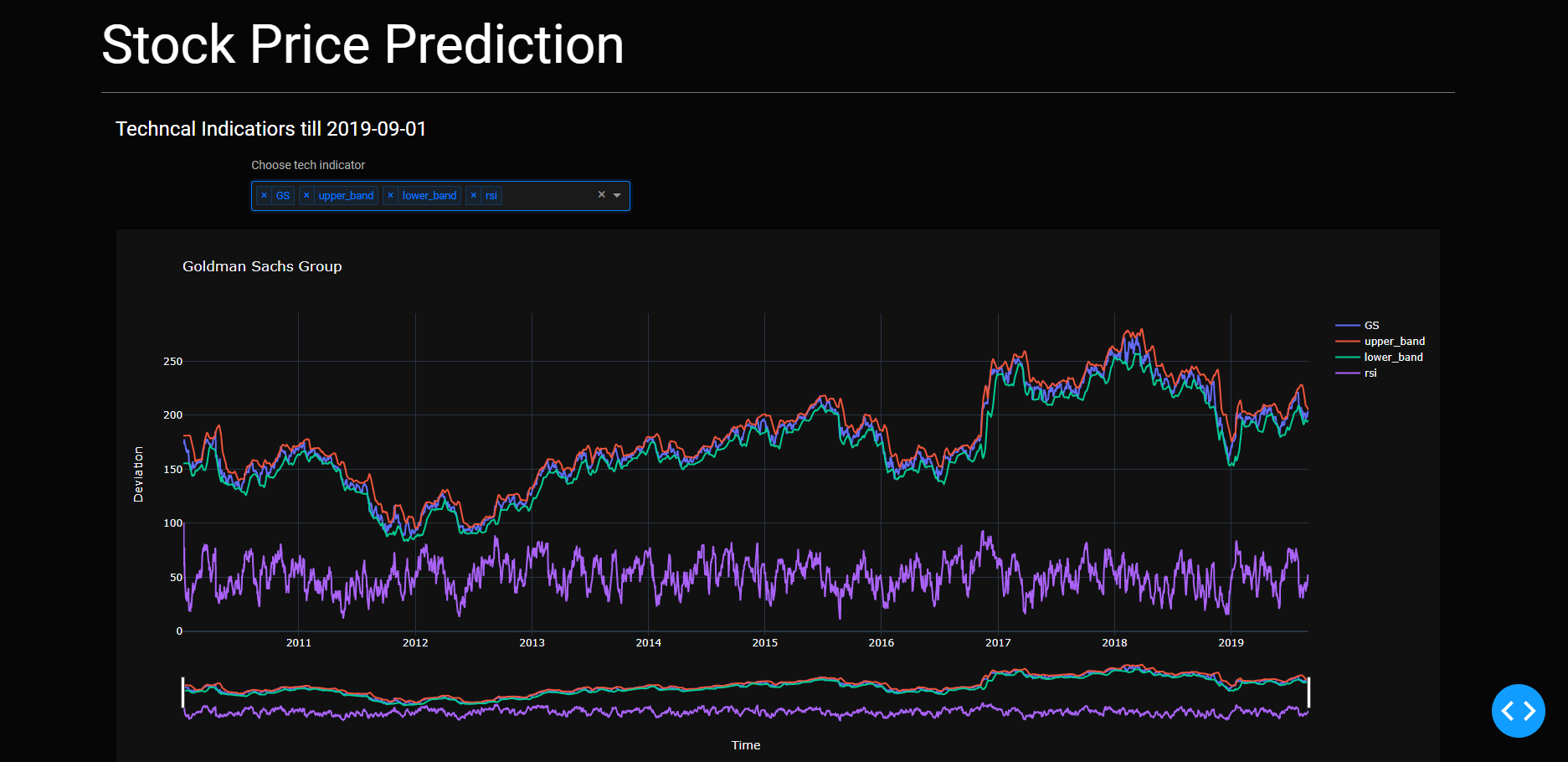


Рисунок 4.19 – Скріншот інтерфейсу програми (1)



Рисунок 4.20 – Скріншот інтерфейсу програми (2)

## Висновки до розділу

У розділі були детально описані етапи програмної реалізації: збір даних, створення технічних індикаторів, класифікація фінансових новин за настроєм, проведення спектрального аналізу, аналіз та візуалізація даних, підготовка даних для подання в модель, побудова і навчання нейронної мережі, її оцінка, створення веб-інтерфейсу. Стандартизація проводилася окремо на навчальних та тестових даних. Перед прогнозуванням всі часові ряди приведені до стаціонарних, що підтверджено статистичними тестами ознак. Результати прогнозу нейронної мережі візуалізовано та оцінено метриками. Побудовано веб-застосунок на Dash.

Висновки

В ході виконання дипломної роботи був реалізований підхід до прогнозу часового ряду цін акцій певної компанії, який базується на зборі якомога більшої кількості даних, які впливають або відображають рух на ринку акцій – корельованих активах.

Оглянуто найбільш популярні математичні методи, які застосовують для прогнозування багатовимірних часових рядів. Зокрема: векторна авторегресія, багатовимірна лінійна регресія, модель авторегресійного ковзаючого середнього, прихована марківська модель, штучні нейронні мережі. Обрано за сукупністю характеристик найбільш підходящий для вирішення поставленої мети – модифікація рекурентних нейронних мереж модель довгої короткострокової пам’яті LSTM. Оглянуто альтернативні програмні рішення, виявлено їх недоліки та обгрунтована актуальність виконання даної роботи.

Досліджено, описано та програмно реалізовано математичні методи та алгоритми, які реалізовані в даній дипломній роботі для обробки ознак та побудови моделі. Математичне забезпечення, яке було використане в ході виконання роботи: методи технічного аналізу для створення тенденційних та осциляторних технічних індикаторів, модель BERT для NLP обробки та класифікації настроїв фінансових новин, перетворення Фур’є для проведення спектрального аналізу та виділення локальних та глобальних трендів і зменшення шуму в даних, метод головних компонент PCA для зменшення розмірності даних, видалення серійної кореляції та виділення більш інформативних ознак, модифікацію рекурентних нейронних мереж модель довгої-короткострокової пам’яті LSTM для навчання на отриманих даних та для прогнозування фінансового часового ряду.

Було проведено дослідження та збір корельованих активів за період 10 років, створено додаткові ознаки з використанням технічного та спектрального аналізу. Програмно реалізовано описане вище математичне забезпечення. Результати роботи моделі LSTM проаналізовано: оцінено метриками та візуалізовано. Побудовано веб-застосунок на Dash, який є інтерактивним та дозволяє користувачу взаємодіяти з інформацією.

Дана дипломна робота в подальшому може бути покращена шляхом застосування нових методів і алгоритмів роботи з даними або знаходження нових варіантів їх комбінування. Наприклад, для прогнозування фінансового часового ряду можна застосувати генеративну змагальну мережу.

Перелік посилань

1. Энциклопедический словарь Брокгауза и Ефрона: Том IIIА (6). Бергер — Бисы / Ф. А. Брокгауз, И. А. Ефрон – Санкт-Петербург : Типо-Литография И. А. Ефрона, 1892. – 953 с.
2. Малюгин В. И. Рынок ценных бумаг: количественные методы анализа: Учеб. пособие. / Малюгин В. И. - Мн.: БГУ, 2001. - 318 с: ил.
3. Эдгар Петерс. Хаос и порядок на рынках капитала = Chaos and Order in the Capital Markets. A New View of Cycles, Prices, and Market Volatility: пер с англ. / Э. Петерс — М.: Мир, 2000. — 336 с.
4. Pearson, K. The Problem of the Random Walk / Pearson, K. – Nature. — 1905. — Vol. 72, no. 1865. — P. 294.
5. Уильям Бернстайн. Манифест инвестора: Готовимся к потрясениям, процветанию и всему остальному = The Investor’s Manifesto. Preparing For Prosperity, Armageddon, And Everything In Between: пер с англ. / Уильям Бернстайн. — М.: Альпина Паблишер, 2013. — 229 с.
6. Albanese, C.; D. Li; E. Lobachevskiy; G. Meissner. A Comparative Analysis or Correlation Approaches in Finance. / Albanese, C.; D. Li; E. Lobachevskiy; G. Meissner – 2010
7. Christopher A. Sims and Vector Autoregressions / Lawrence J. Christiano – Scand. J. of Economics 114(4), 1082–1104, 2012
8. Айвазян С.А. Прикладная статистика. Основы эконометрики. Том 2. / Айвазян С.А. — М.: Юнити-Дана, 2001. — 432 с.
9. Носко В.П. Эконометрика. Введение в регрессионный анализ временных рядов. / Носко В.П. — М., 2002. — 273 с.
10. Демиденко Е.З. Линейная и нелинейная регрессия. / Е.З. Демиденко — М.: Финансы и статистика, 1981. — 302 с.
11. Linear regression and the normality assumption / Amand F. Schmidt, Chris Finan – Faculty of Population Health, Institute of Cardiovascular Science, University College London, London WC1E 6BT, United Kingdom – December 2017
12. Breusch, T. S.; Pagan, A. R. A Simple Test for Heteroscedasticity and Random Coefficient Variation / Breusch, T. S.; Pagan, A. R. – 1979 – Econometrica. 47 (5): 1287–1294.
13. Box, George; Jenkins, Gwilym. Time Series Analysis: Forecasting and Control. / Box, George; Jenkins, Gwilym – 1970 – San Francisco: Holden-Day.
14. Baum, L. E.; Petrie, T. Statistical Inference for Probabilistic Functions of Finite State Markov Chains / Baum, L. E.; Petrie, T. – 1966 – The Annals of Mathematical Statistics. 37 (6): 1554–1563.
15. Марковский процесс / А. В. Прохоров // Большая российская энциклопедия: [в 35 т.] / гл. ред. Ю. С. Осипов. — М. : Большая российская энциклопедия, 2004—2017.
16. Prediction of Financial Time Series with Hidden Markov Models / Y Zhang – 2004 – Simon Fraseur University
17. 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. — 112 с.
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. Куликов А. А. Форекс для начинающих. Справочник биржевого спекулянта. 2-е изд. / Куликов А. А. — СПб.: Питер 2006—384 с.
22. Акелис, Стивен Б. Технический анализ от А до Я. Полный набор инструментов торговли… от «Абсолютного индекса ширины» до «Японских свечей» = Technical Analysis from A to Z: Covers Every Trading Tool… from the Absolute Breadth Index to the Zig Zag / Перевод с английского М. Волкова, А. Лебедева. — М.: Диаграмма, 1999. — 376 с. – стр. 26
23. Грешилов А. А., Стакун В. А., Стакун А. А. Математические методы построения прогнозов. / Грешилов А. А., Стакун В. А., Стакун А. А – М.: Радио и связь, 1997. - 112 с – стр 72
24. John J. Murphy. [Technical Analysis of the Financial Markets](https://books.google.com/books?id=ojTgDwAAQBAJ&pg=PA199" \l "v=onepage&q&f=false" \t "_blank) / John J. Murphy – Penguin, 1999 – Page 199.
25. Ивченко Г. И., Медведев Ю. И. Введение в математическую статистику. / Ивченко Г. И., Медведев Ю. И. — М. : Издательство ЛКИ, 2010. — §2.2. Выборочные моменты: точная и асимптотическая теория.
26. Колби Р.В., Мейерс Т.А. Энциклопедия технических индикаторов рынка / Колби Р.В., Мейерс Т.А.: Пер. с англ. – М.: Издательский дом "АЛЬПИНА" 2000. - 581 с.
27. Индикатор Ишимоку как основа торговой системы / НП «Форекс Клуб». — М.: Форекс Клуб, 2004. — 96 с.
28. ЛеБо Ч., Лукас Д. В. Компьютерный анализ фьючерсных рынков. / ЛеБо Ч., Лукас Д. В.  — М.: Издательский Дом «АЛЬПИНА», 1998. — 304 с.
29. Berkowitz, Stephen A.; Logue, Dennis E.; Noser, Eugene A. J. (). The Total Cost of Transactions on the NYSE. / Berkowitz, Stephen A.; Logue, Dennis E.; Noser, Eugene A. J. – Journal of Finance. American Finance Association. – March 1988 – 43 (1): 97–112.
30. Сайт компанії Goldman Sachs Group. [Електронний ресурс]. - Режим доступу: https://www.goldmansachs.com/
31. Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova . BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova – 2019 – Association for Computational Linguistics.
32. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems. / Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin – 2017 – p. 6000–6010
33. Wilson L Taylor. Cloze procedure: A new tool for measuring readability. / Wilson L Taylor – 1953 – Journalism Bulletin, 30(4):415–433.
34. Mike Shuster, Kaisuke Nakajima. Japanese and Korean Voice Search / Mike Shuster, Kaisuke Nakajima – Schuster et al. – 2012
35. Dogu Tan Araci. FinBERT: Financial Sentiment Analysis with Pre-trained Language Models / Dogu Tan Araci – 2019
36. Веб-сайт постачальника юридичних, державних, ділових і високотехнологічних джерел інформації. [Електронний ресурс]. - Режим доступу: https://www.lexisnexis.com/
37. Зорич В. А. Математический анализ. / Зорич В. А. — М.: Физматлит, 1984. — 544 с.
38. Афонский А. А., Дьяконов В. П. Цифровые анализаторы спектра, сигналов и логики / Под ред. проф. В. П. Дьяконова. — М.: СОЛОН-Пресс, 2009. — С. 248.
39. Фихтенгольц Г. М. Курс дифференциального и интегрального исчисленияю В 3 т. Т III. / Пред. и прим. А. А. Флоринского. — 8-е изд.. — М: ФИЗМАТЛИТ, 2008. — С. 492-493. — 728 с.
40. Гантмахер Ф. Р.. Теория матриц. / Гантмахер Ф. Р.. — М.: Наука, 1966. — 576 с. – cтр. 82
41. Pearson K. On lines and planes of closest fit to systems of points in space, Philosophical Magazine / Pearson K – 1901 – p. 559—572
42. Айвазян С. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д. Прикладная статистика. Классификация и снижение размерности. / Айвазян С. А., Бухштабер В. М., Енюков И. С., Мешалкин Л. Д. — М.: Финансы и статистика, 1989.— 607 с. – с. 639.
43. А. Н. Ширяев. Глава 2, §6. Случайные величины II // Вероятность. / А. Н. Ширяев — 3-е изд. — Cambridge, New York,...: МЦНМО, 2004. — Т. 1. — С. 301. — 520 с. – с. 331
44. Sepp Hochreiter; Jürgen Schmidhuber. Long short-term memory / Sepp Hochreiter; Jürgen Schmidhuber – Neural Computation : journal. — 1997. — Vol. 9, no. 8. — P. 1735—1780.
45. Google Stock Price Prediction using RNN – LSTM. [Електронний ресурс]. - Режим доступу: https://kgptalkie.com/google-stock-price-prediction-using-rnn-lstm/
46. Understanding LSTM Networks – 2015. [Електронний ресурс]. - Режим доступу: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
47. Denny Britz. Recurrent Neural Networks Tutorial / Denny Britz – September 17, 2015
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 Press, 2001
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. Анализ временных рядов / Канторович Г. Г. – НИУ ВШЭ – Россия, Москва, ул. Мясницкая, д.20 – 2002 – Т. 6. № 2. С. 251–273
64. Документація фреймворку dash. [Електронний ресурс]. - Режим доступу: https://dash.plotly.com/
65. Dan Hendrycks, Kevin Gimpel. Gaussian Error Linear Units (GELUs) / Dan Hendrycks, Kevin Gimpel – 2016
66. Лістинги програм

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

# імпорт бібліотек

# 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 кіькість 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])

# боксплоти

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 Н0 про нестаціонарність -- ряд стаціонарний

res.append('stationary')

else: # приймаємо Н0 -- 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)

1. Ілюстративний матеріал

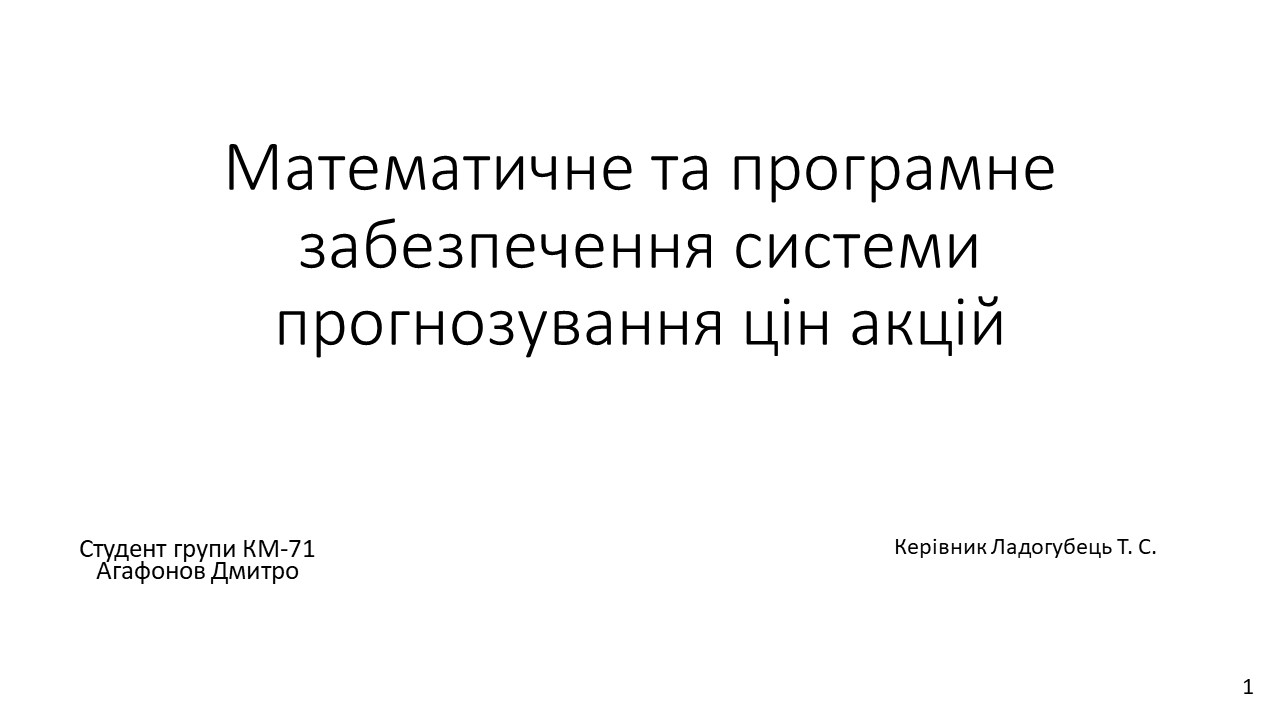


Рисунок Б.1 – Слайд 1

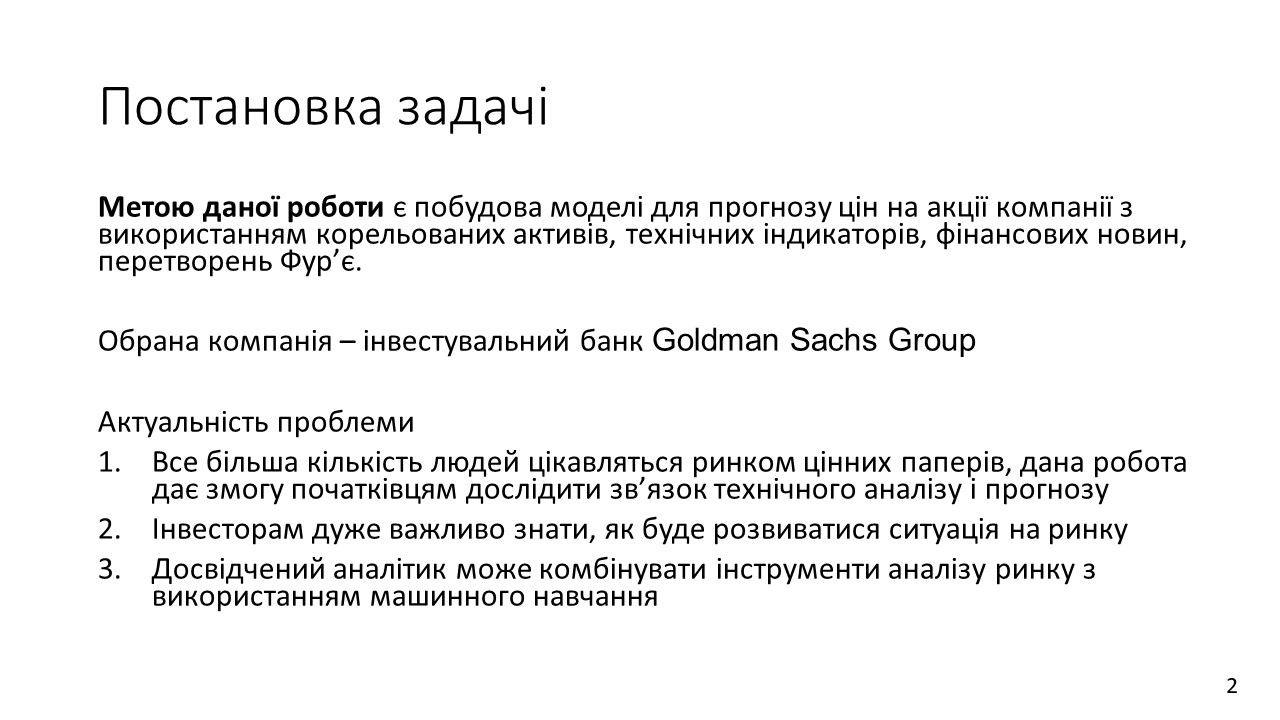


Рисунок Б.2 – Слайд 2

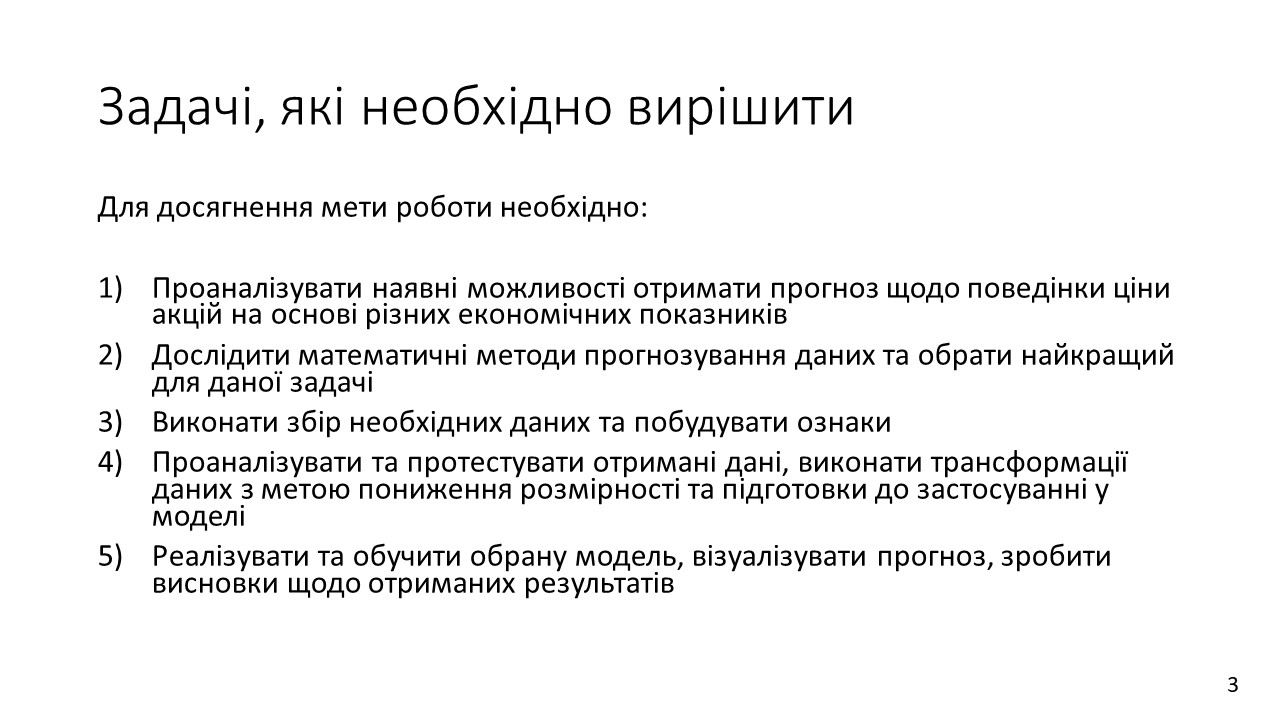


Рисунок Б.3 – Слайд 3

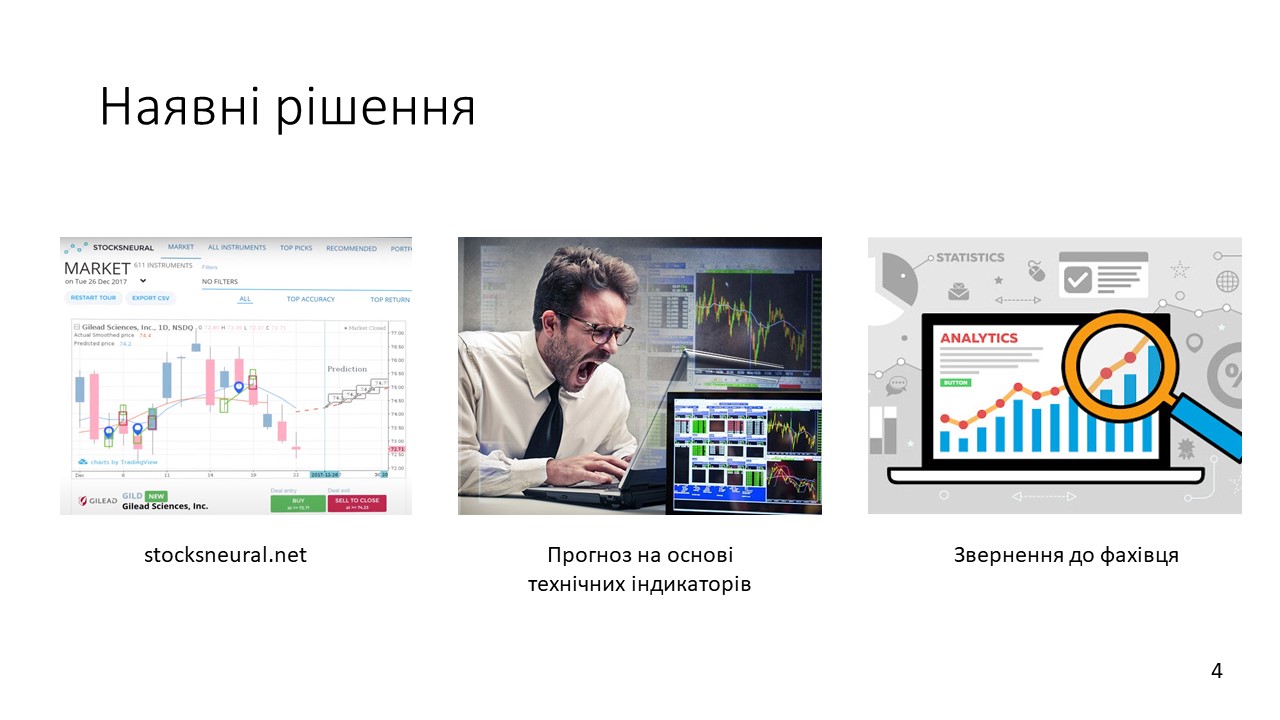


Рисунок Б.4 – Слайд 4

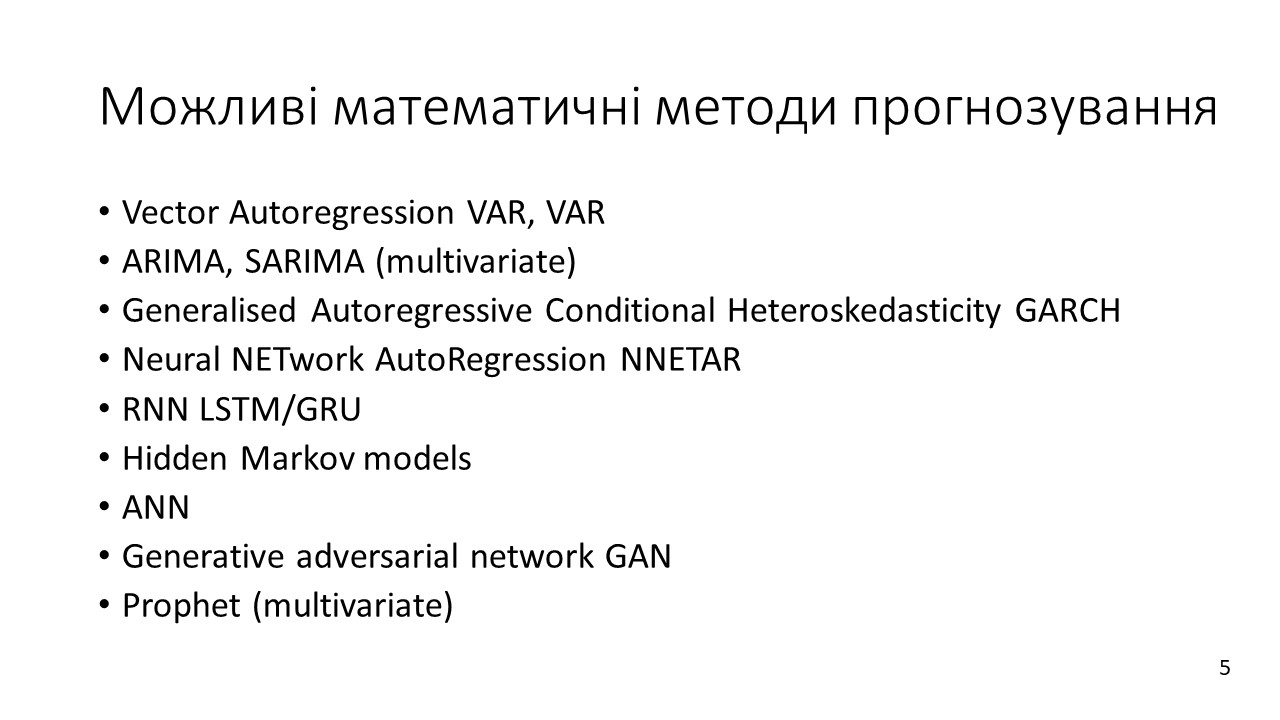


Рисунок Б.5 – Слайд 5

 Рисунок Б.6 – Слайд 6

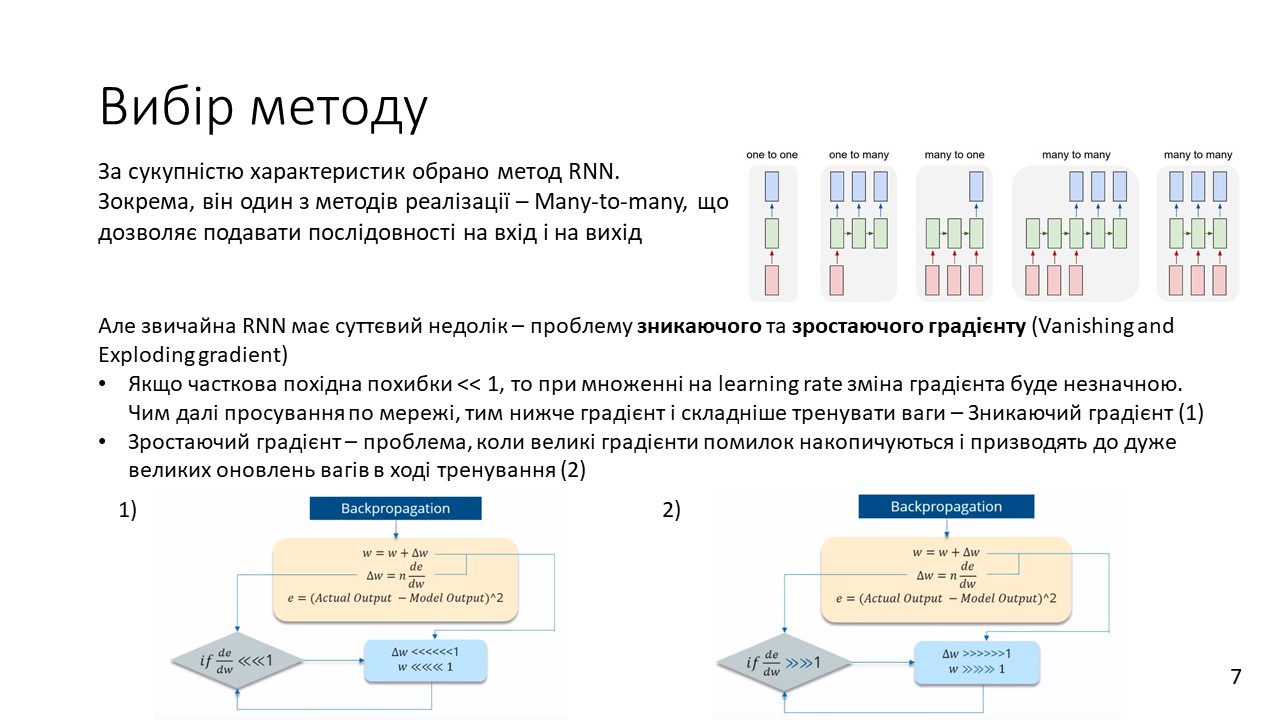


Рисунок Б.7 – Слайд 7

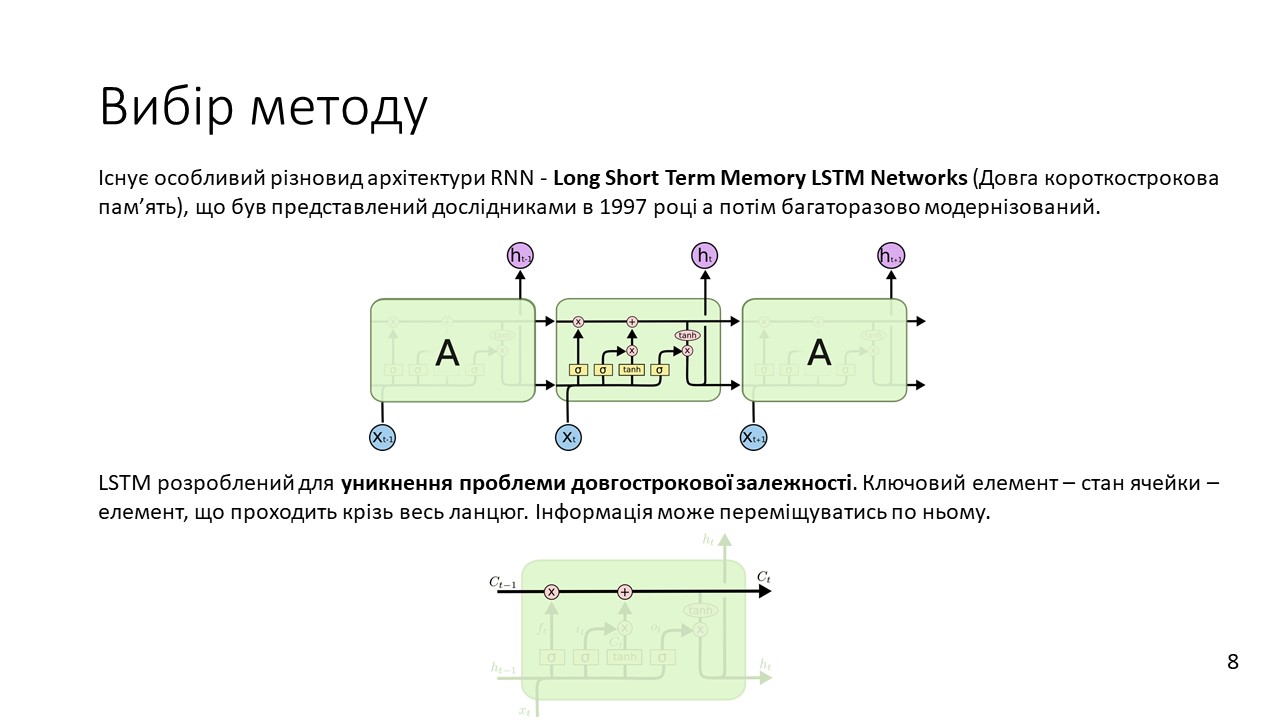


Рисунок Б.8 – Слайд 8

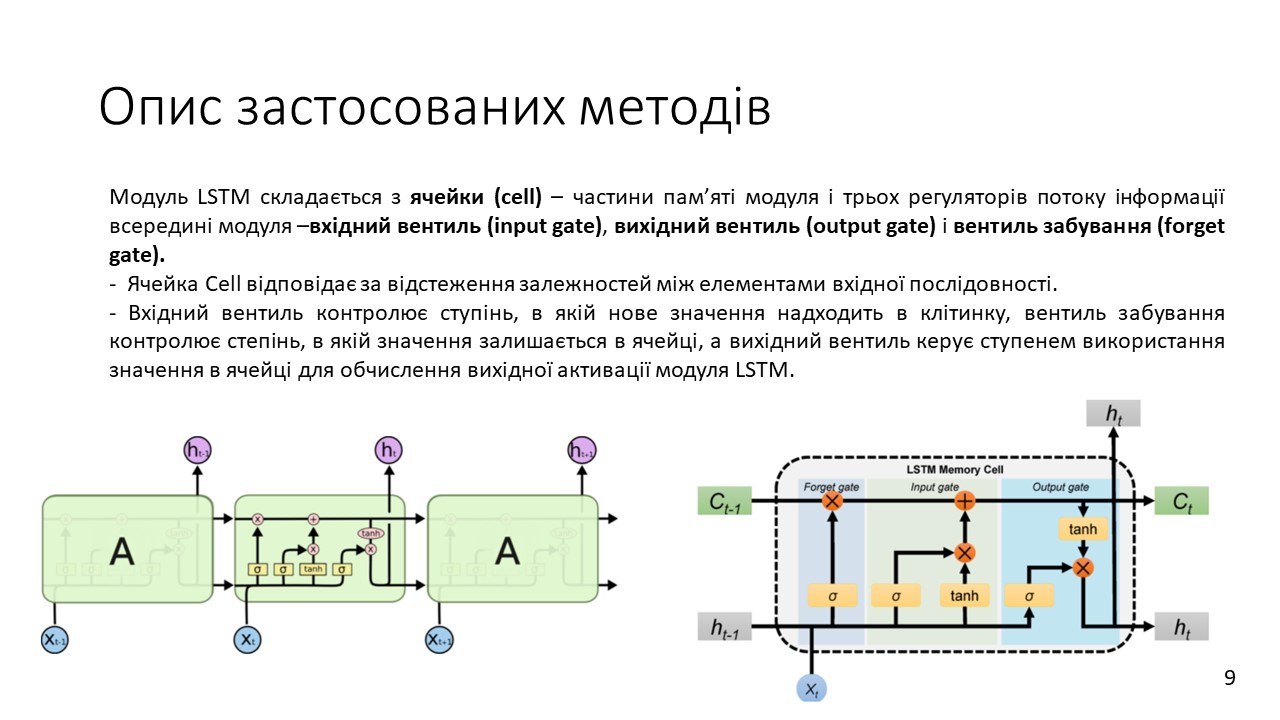


Рисунок Б.9 – Слайд 9

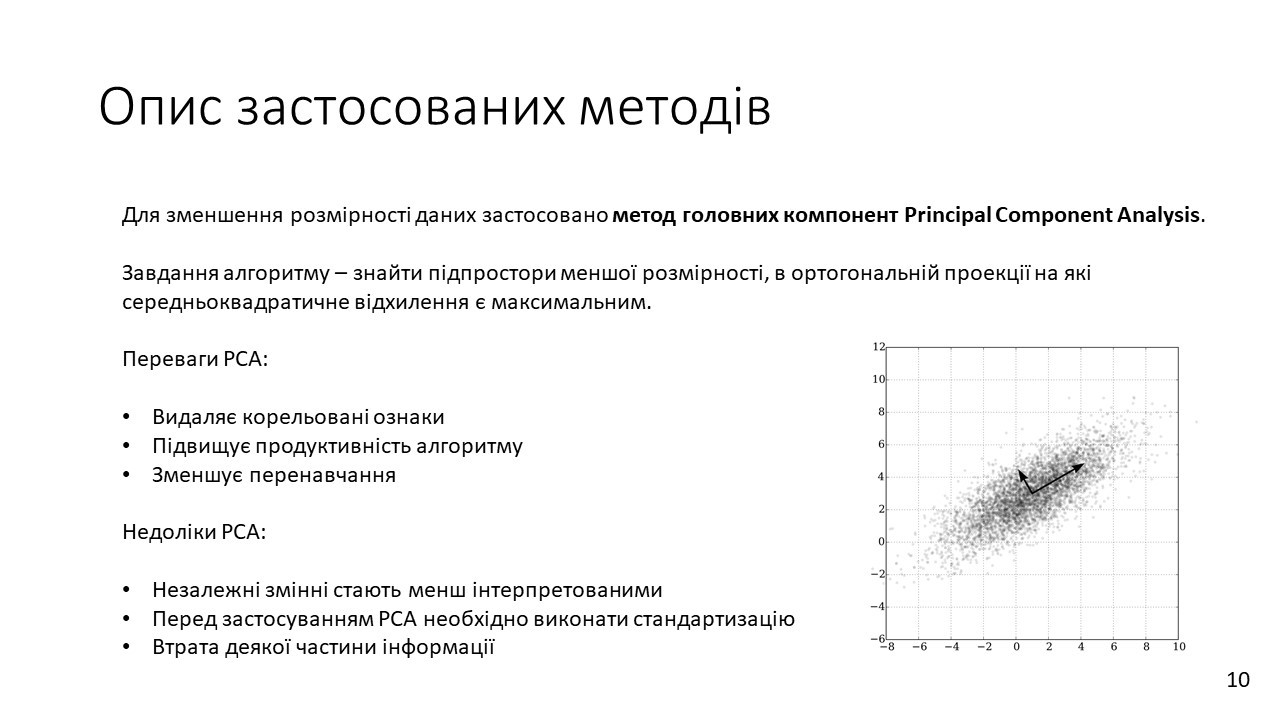


Рисунок Б.10 – Слайд 10



Рисунок Б.11 – Слайд 11

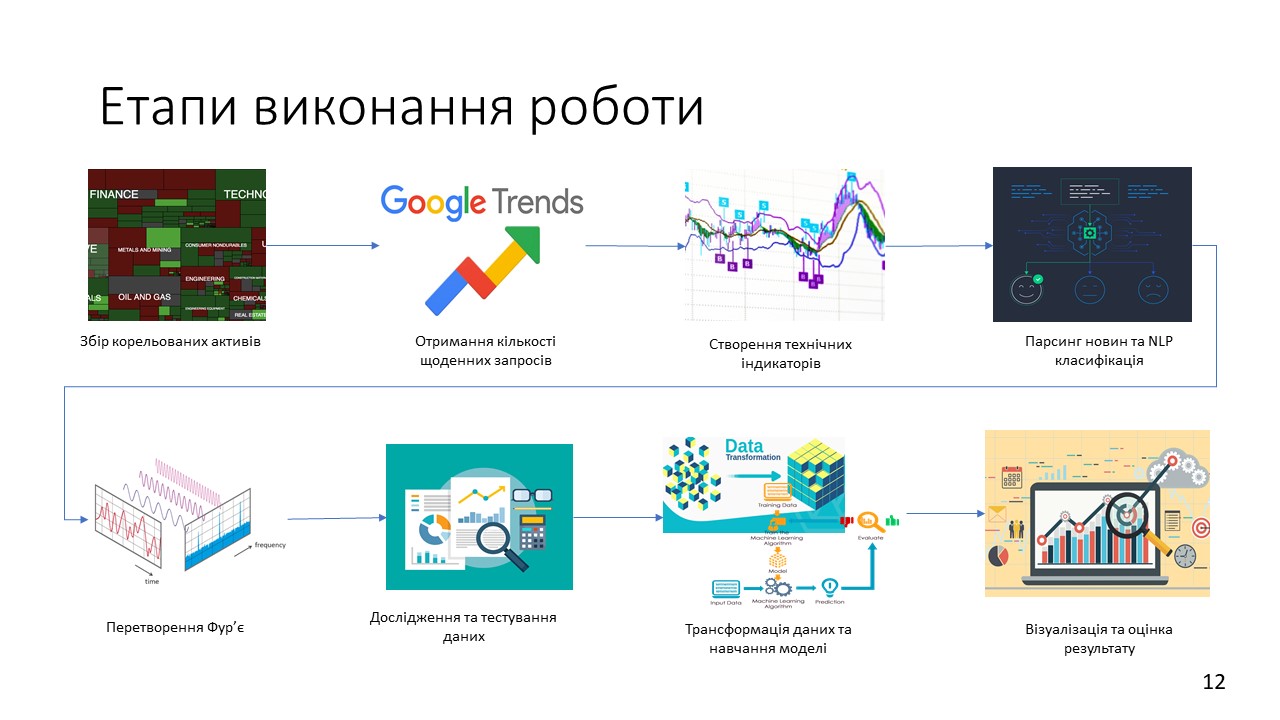


Рисунок Б.12 – Слайд 12



Рисунок Б.13 – Слайд 13



Рисунок Б.14 – Слайд 14



Рисунок Б.15 – Слайд 15



Рисунок Б.16 – Слайд 16



Рисунок Б.17 – Слайд 17

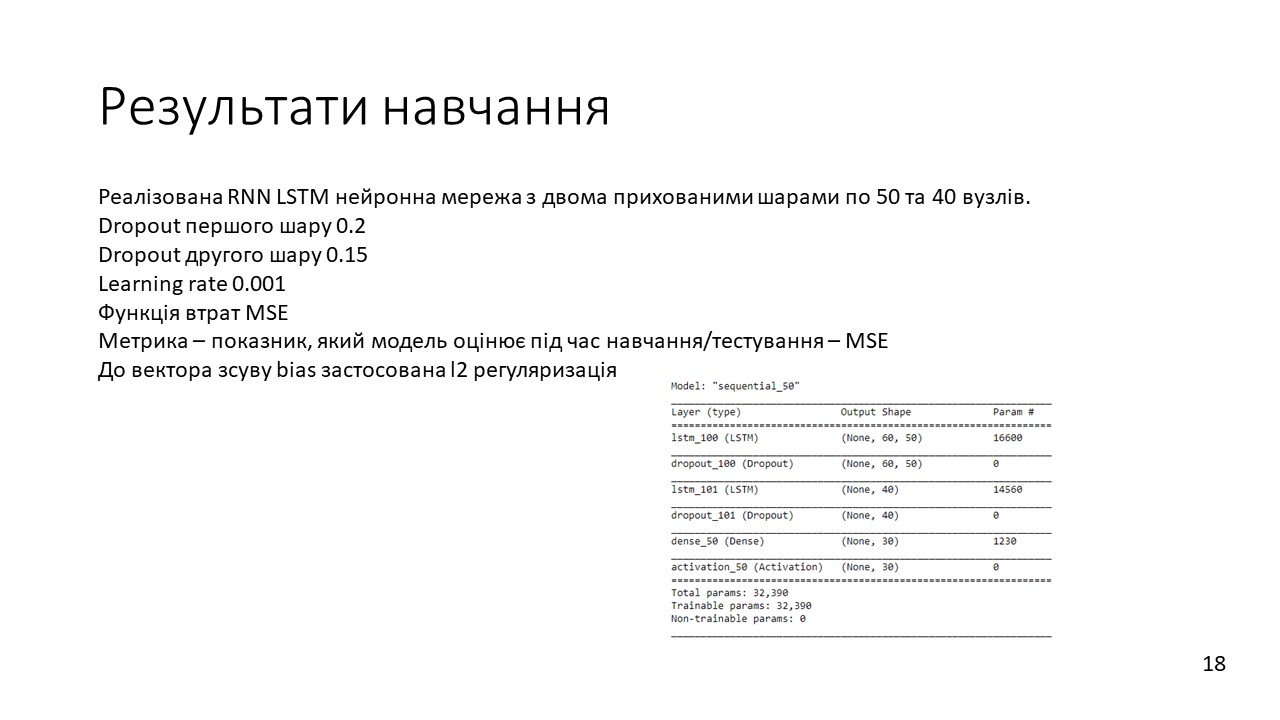


Рисунок Б.18 – Слайд 18

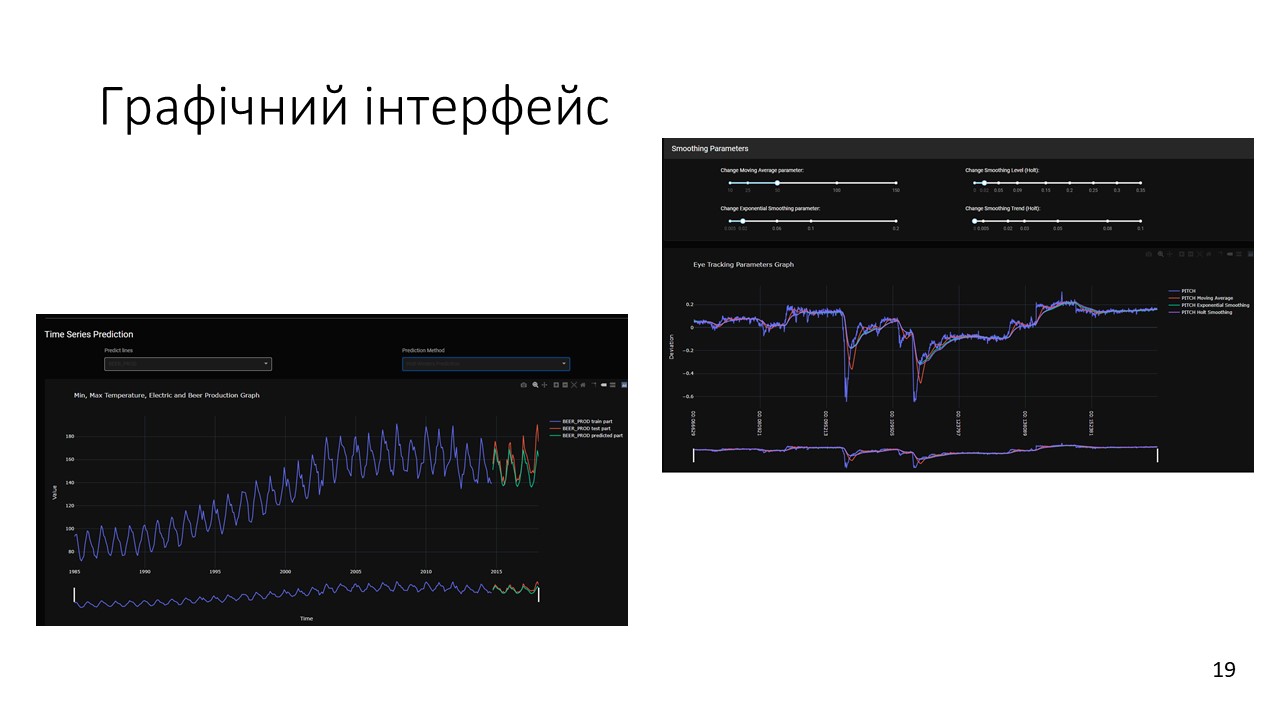


Рисунок Б.19 – Слайд 19

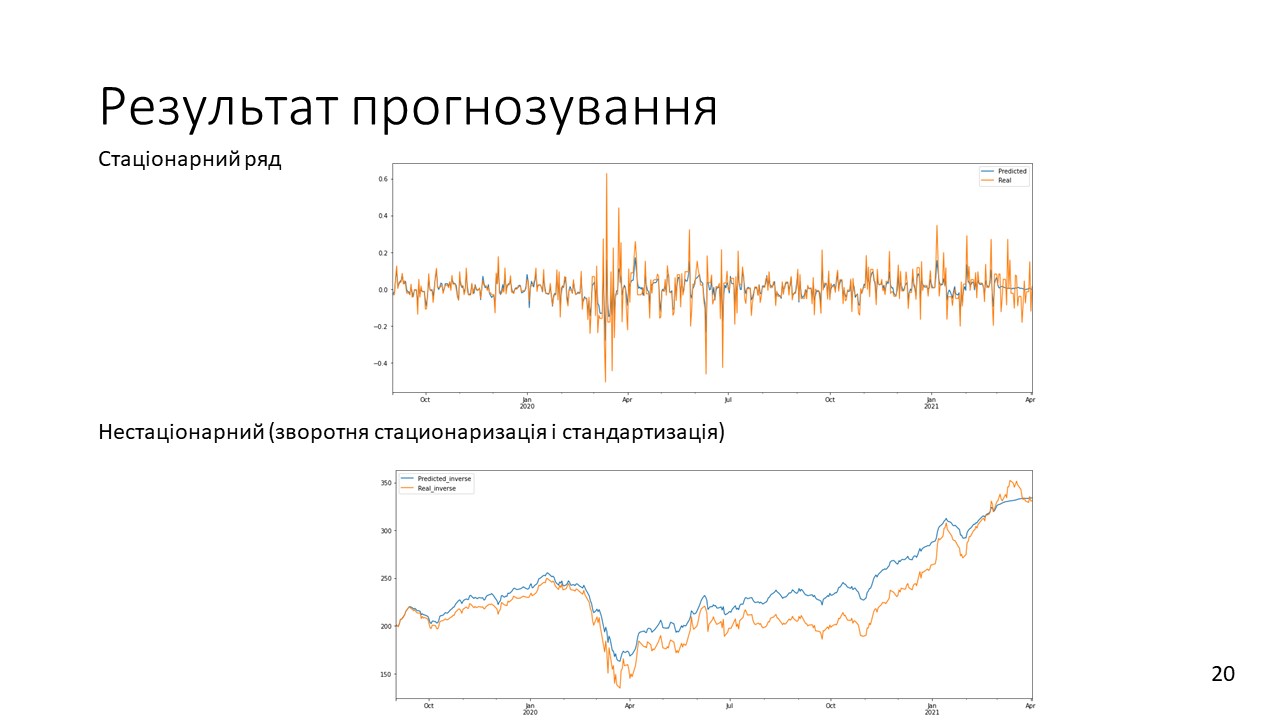


Рисунок Б.20 – Слайд 20

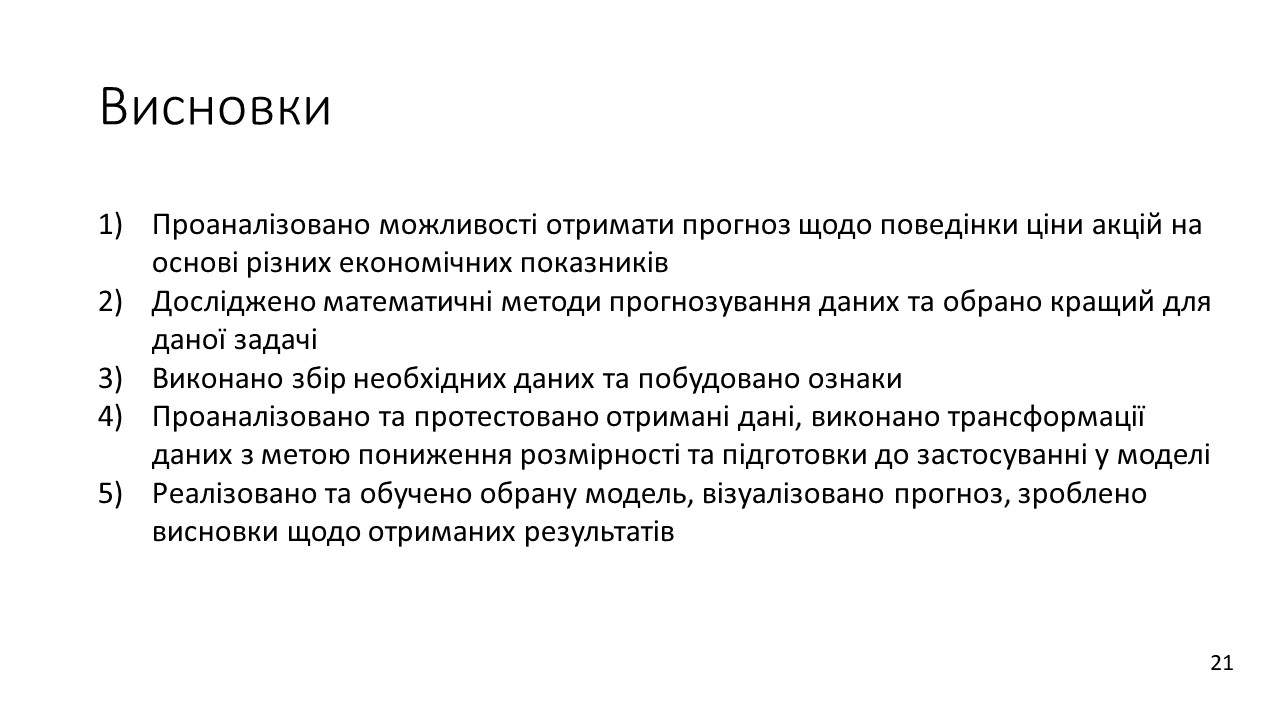


Рисунок Б.21 – Слайд 21



Рисунок Б.22 – Слайд 22