НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

На правах рукопису
УДК 004.852

До захисту допущено
В.о. завідувача кафедри ШІ
_____О.І. Чумаченко
«___»_____2022 р.

Магістерська дисертація

на здобуття ступеня магістра

за спеціальністю 122 «Комп'ютерні науки»

на тему: «Оптимізація параметрів та структури гібридних мереж глибокого навчання та їх застосування в задачах прогнозування»

Виконав:
студент ІІ курсу, групи КІ-з11мп
Кузьменко Олексій Віталійович _____

Керівник:
Професор кафедри ММСА, д.т.н., професор,
Зайченко Юрій Петрович _____

Рецензент:
Завідувачка кафедри ПЗКС ФПМ, д.т.н., доцент,
Сулема Євгенія Станіславівна _____

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

Студент _____

Київ
2022

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

Рівень вищої освіти – другий (магістерський)

Спеціальність – 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ШІ
_____ О.І. Чумаченко
«___» _____ 2022 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Кузьменку Олексію Віталійовичу

1. Тема дисертації: «Оптимізація параметрів та структури гібридних мереж глибокого навчання та їх застосування в задачах прогнозування», науковий керівник дисертації Зайченко Юрій Петрович, професор кафедри ММСА, д.т.н., професор, затверджено наказом по університету від «02» листопада 2022 р. № 4040-с
2. Термін подання студентом дисертації: «12» грудня 2022 р.
3. Об'єкт дослідження: фінансові процеси на ринках цінних паперів та методи їх прогнозування.
4. Предмет дослідження: гібридні мережі глибокого навчання на основі самоорганізації та оцінка їх ефективності в задачах прогнозування на фінансових ринках.
5. Перелік завдань, які потрібно виконати:
    1) провести аналіз предметної області;
    2) визначити актуальність теми дослідження;

3) проаналізувати існуючі підходи для прогнозування часових рядів, розглянути їх переваги та недоліки, визначити найбільш ефективні інструменти моделювання;

4) здійснити оптимізацію параметрів та структури гібридних мереж глибокого навчання на основі самоорганізації;

5) розробити алгоритм та програмне забезпечення для проведення експериментальних досліджень гібридних мереж глибокого навчання на основі самоорганізації;

6) провести обчислювальні експерименти з оптимізації параметрів гібридних мереж глибокого навчання для вирішення задачі прогнозування та порівняти їх з результатами нейронної мережі LSTM;

7) розробити стартап-проєкт для виходу програмного продукту на комерційний ринок;

8) зробити висновки по роботі та оформити пояснювальну записку.

6. Перелік ілюстративного матеріалу: алгоритм роботи програмного забезпечення, схема параметрів гібридної мережі глибокого навчання, процес синтезу структури гібридної мережі глибокого навчання.

7. Дата видачі завдання: «01» вересня 2022 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів роботи	Примітка
1.	Отримання завдання на магістерську дисертацію	01.09.2022 – 04.09.2022	Виконано
2.	Опрацювання технічної літератури за темою магістерської дисертації	05.09.2022 – 17.09.2022	Виконано
3.	Аналіз предметної області, огляд існуючих підходів для вирішення аналогічних задач, дослідження актуальності теми дисертації	18.09.2022 – 25.09.2022	Виконано
4.	Математичне моделювання гібридних мереж глибокого навчання	26.09.2022 – 03.10.2022	Виконано
5.	Розробка та тестування	04.10.2022 – 16.10.2022	Виконано

	програмного забезпечення для проведення експериментальних досліджень		
6.	Виконання та аналіз обчислювальних експериментів	17.10.2022 – 22.10.2022	Виконано
7.	Висновки за результатами проведених досліджень, розробка рекомендацій щодо подальшого розвитку	23.10.2022 – 24.10.2022	Виконано
8.	Розробка стартап-проєкту для виходу програмного продукту на комерційний ринок	25.10.2022 – 04.11.2022	Виконано
9.	Підготовка ілюстративного матеріалу	05.11.2022 – 10.11.2022	Виконано
10.	Оформлення пояснювальної записки	11.11.2022 – 17.11.2022	Виконано

Студент _____ О.В. Кузьменко

Науковий керівник _____ Ю.П. Зайченко

РЕФЕРАТ

Магістерська дисертація: 102 с., 28 табл., 19 рис., 27 джерел, 3 додатки.

ГІБРИДНА МЕРЕЖА ГЛИБОКОГО НАВЧАННЯ, МГУА-НЕО-ФАЗІ, ОПТИМІЗАЦІЯ, СИНТЕЗ СТРУКТУРИ, LSTM, КОРОТКОСТРОКОВЕ ПРОГНОЗУВАННЯ, СЕРЕДНЬОСТРОКОВЕ ПРОГНОЗУВАННЯ, ФІНАНСОВІ РИНКИ.

Об’єктом дослідження є фінансові процеси на ринках цінних паперів та методи їх прогнозування.

Предмет дослідження – гібридні мережі глибокого навчання на основі самоорганізації та оцінка їх ефективності в задачах прогнозування на фінансових ринках.

Мета дослідження полягає в оцінці точності гібридних мереж глибокого навчання у задачі прогнозування на фінансових ринках, порівнянні їх ефективності на різних інтервалах з нейронною мережею LSTM та визначенні класів задач прогнозування, для яких застосування відповідних обчислювальних інтелектуальних технологій є найбільш перспективним.

Запропоновано та розроблено систему для оптимізації параметрів і синтезу структури гібридних мереж глибокого навчання, а також візуалізації результатів навчання та прогнозування. Експериментальним шляхом доведено ефективність запропонованої системи.

ABSTRACT

Master's thesis: 102 p., 28 tabl., 19 fig., 27 ref., 3 appendices.

HYBRID DEEP LEARNING NETWORK, GMDH-NEO-FUZZY, OPTIMIZATION, STRUCTURE SYNTHESIS, LSTM, SHORT-TERM FORECASTING, MIDDLE-TERM FORECASTING, FINANCIAL MARKETS.

The object of the research is financial processes in securities markets and methods of their forecasting.

The subject of the research is hybrid deep learning networks based on self-organization and evaluation of their effectiveness in forecasting tasks in financial markets.

The purpose of the study is to assess the accuracy of hybrid deep learning networks in the task of forecasting financial markets, compare their effectiveness at different intervals with the LSTM neural network, and determine the classes of forecasting tasks for which the application of appropriate computational intelligence technologies is the most promising.

A system for parameter optimization and synthesis of the structure of hybrid deep learning networks, as well as visualization of learning and forecasting results, is proposed and developed. The effectiveness of the proposed system has been proven experimentally.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ШНМ – штучна нейронна мережа;

ШІ – штучний інтелект;

DL – Deep Learning – глибоке навчання;

МГУА – метод групового урахування аргументів;

ФН – функція належності;

SB – Selection Block –блок селекції;

NFN – Neo-Fuzzy Neuron – нео-фазі нейрон;

NS – Nonlinear Synapse – нелінійний синапс;

LSTM – Long Short-Term Memory – довга короткочасна пам’ять;

EMBRI – Emerging Markets Bond Total Return Index – Індекс загальної прибутковості облігацій ринків, що розвиваються;

MAPE – Mean Absolute Percentage Error – середня абсолютна похибка у відсотках;

MSE – Mean Squared Error – середньоквадратична похибка;

ВСТУП

Розробка та впровадження ШНМ на основі прогресивних технологій є одним із пріоритетних напрямів розвитку галузей науки і техніки в усіх промислово розвинутих країнах. При розв’язуванні прикладних задач з метою підвищення точності та зменшення складності обчислень виникають проблеми пошуку оптимальної топології мережі і, відповідно, структурної (визначення кількості прихованих шарів і нейронів у них, міжнейронних зв’язків окремих нейронних мереж) та параметричної (встановлення вагових коефіцієнтів) оптимізації. Основні обмеження відомих методів і технологій, що використовуються на даний момент, пов’язані з недостатньою ефективністю вирішення задач навчання ШНМ, налаштування та адаптації до проблемної області, обробки неповної та неточної вихідної інформації, інтерпретації даних та накопичення експертних знань, представлення інформації з різні джерела тощо.

Побудова гібридних нейронних мереж, що складаються з різних типів, кожна з яких навчається за певним алгоритмом пошарово, у багатьох випадках може значно підвищити ефективність ШНМ. Дослідження принципів гібридизації ШНМ, нечіткої логіки та генетичних алгоритмів дозволяє створювати нові типи моделей, які мають більш високу якість прогнозування при зниженні обчислювальних витрат на навчання.

Основні обмеження відомих методів і технологій ШІ зумовлені недостатньою ефективністю їх навчання, складністю налаштування та адаптації до проблемної зони в умовах неповної та неточної вихідної інформації, складністю накопичення експертних знань та ін. Таким чином, однією з актуальних проблем у розробці сучасних систем ШІ є розробка інтегрованих, гібридних систем на основі глибокого навчання.

На жаль, сьогодні відсутня методологія проектування топологій гібридних нейронних мереж та гібридних технологій їх структурно-параметричного синтезу з використанням глибокого навчання. Основним фактором, що сприяє розвитку таких систем, є розширення використання нейронних мереж (НМ) для вирішення різноманітних задач. Використання інших технологій для вирішення такого класу задач призводить до важких символьних обчислень або до великих обчислювальних проблем [1].

В роботі досліджується вплив параметрів гібридних нейронних мереж глибокого навчання та їх структури на точність короткострокового і середньострокового прогнозування. Розроблено систему для вибору оптимальних параметрів та структури гібридних мереж глибокого навчання. Її ефективність підтверджено шляхом порівняльного аналізу з рекурентною нейронною мережею LSTM на основі проведених експериментів.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Прогнозування часових рядів здійснюється на різні інтервали. Тому можна виділити такі типи прогнозів: короткостроковий (до 7 днів), середньостроковий (від 7 днів до 45 днів) та довгостроковий (декілька місяців). Вдосконалені методи прогнозування зазвичай повинні мінімізувати похибку прогнозування або зменшувати обчислювальні витрати.

1.1 Актуальність теми дослідження

Проблемам прогнозування курсів акцій і ринкових індексів на фондових біржах приділяється велика увага інвесторів і різних грошових фондів. Для їх вирішення були розроблені та досліджені потужні інтелектуальні методи та технології, серед яких нейронні мережі та системи нечіткої логіки.

Ефективним інструментом моделювання для короткострокового та середньострокового прогнозування нестаціонарних часових рядів є мережі LSTM. Вони давно розроблені та успішно застосовуються для прогнозування на біржах.

В якості альтернативного підходу для прогнозування у фінансах є застосування різних типів нейронних мереж: MLP, нечітких нейронних мереж, нео-нечітких мереж і мереж глибокого навчання. Новим трендом для мереж глибокого навчання є новий клас мереж NN – гібридні мережі глибокого навчання на основі методу МГУА. Прикладна самоорганізація в цих мережах дозволяє навчати не тільки ваги нейронів, але й будувати оптимальну структуру мережі. Завдяки методу навчання в цих мережах ваги коригуються не одночасно, а шар за шаром. Це запобігає феномену

зникнення або вибуху градієнта, що дуже важливо для мереж з багатьма рівнями.

У перших роботах у цій галузі використовувалися у якості вузлів гібридні мережі Ванга-Менделя. Але недоліком таких нейронів є вимога навчати не лише нейронні ваги, але й параметри нечітких наборів у антецедентах правил. Це потребує великих розрахункових витрат і збільшує час навчання. Тому пізніше були розроблені нео-фазі-мережі глибокого навчання, в яких у якості вузлів використовуються нео-фазі-нейрони Ямакави. Основна властивість таких нейронів полягає в тому, що необхідно навчати лише ваги нейронів, а не нечіткі набори. Це вимагає менше обчислень порівняно з нейронами Ванга-Менделя і значно скорочує час навчання в цілому.

Тому виникає проблема порівняння ефективності гібридних мереж DL та LSTM в задачах прогнозування у фінансовій сфері.

Метою даної роботи є дослідження точності гібридних мереж глибокого навчання та LSTM у задачі прогнозування ринкових індексів на біржі, порівняння їх ефективності на різних інтервалах та визначення класів задач прогнозування, для яких застосування відповідних обчислювальних інтелектуальних технологій є найбільш перспективними [2].

Актуальність даної роботи полягає в оптимізації параметрів та структури гібридних мереж DL, що дозволяє підвищити точність прогнозування та зменшити обчислювальні витрати.

1.2 Огляд публікацій за темою дослідження

Останнім часом було проведено багато досліджень для отримання нейронних мережевих алгоритмів за рахунок поєднання різних парадигм

м’яких обчислень, що призвело до гібридних підходів з перевагами, властивим різним формальним парадигмам.

Робота [3] присвячена дослідженню та застосуванню методу нечіткого індуктивного моделювання групового методу обробки даних в задачах прогнозування у фінансовій сфері. Розглянутий метод відноситься до методів самоорганізації і дозволяє виявити внутрішні приховані закони у відповідній області об’єкта. Перевагою алгоритму є можливість побудови оптимальних моделей. На випадок невизначеності описано новий нечіткий метод, який дозволяє практично автоматично будувати нечіткі моделі. Розглянуто розширення нечіткого алгоритму для різних частинних описів – ортогональних поліномів Чебишева та тригонометричних поліномів Фур’є. Також приділяється увага проблемі адаптації нечітких моделей. Експериментальні дослідження в задачі прогнозування макроекономічних показників України та порівняння з ARIMA і мережею зворотного розповсюдженням підтвердили високу точність нечіткого методу в задачах прогнозування в макроекономіці.

У роботі [4] розглянуто нечіткий МГУА з гаусівською та дзвоноподібною функціями належності та показано їх подібність до трикутної. Розглянуто нечіткі МГУА з різними частковими описами ортогональних поліномів Чебишева та Фур’є, а також розширення та узагальнення нечіткого МГУА на випадок нечітких входів та проаналізовано його властивості. Проведені експериментальні дослідження МГУА і нечіткого МГУА в задачах прогнозування курсів акцій на фондових ринках та здійснений порівняльний аналіз результатів підтвердили високу точність нечіткого МГУА в задачах прогнозування у фінансовій сфері.

У статті [5] розглядається еволюційна каскадна система з нейро-фаззі вузлами в основі, а також алгоритми її навчання. В процесі навчання запропонована система налаштовує свою архітектуру в online режимі, а не лише параметри. Вузлами розглянутої еволюційної каскадної системи є

нейро-фаззі системи. Налаштування параметрів функцій належності відбувається за алгоритмом, який використовує градієнтну процедуру мінімізації критерію навчання. Також підтверджено ефективність розглянутої еволюційної каскадної нейро-фаззі мережі та процедур її навчання. Підтвердження базується на експериментах, що проводились при вирішенні задачі прогнозування нестаціонарних сигналів.

Надзвичайно актуальним питанням є оптимізація швидкості навчання глибоких нейронних мереж. На даний час здебільшого використовуються нейронні мережі на базі персептрону Розенблата. Але результати такого підходу не задовольняють індустріальні та наукові вимоги в контексті швидкодії навчання нейронних мереж. Крім того, даний підхід не вирішує проблему вибухаючого та зникаючого градієнта. Вирішення цієї проблеми розглядається в статті [6], в якій запропоновано використовувати нео-фаззі нейрон, оскільки його властивості засновані на F-перетворенні. У роботі розглядаються переваги застосування у якості основного компонента нейромережі нео-фаззі нейрона. Розглядається архітектура глибокої нео-фаззі нейронної мережі та алгоритм навчання на основі зворотного поширення похибки. Ця архітектура містить нео-фаззі нейрони з трикутною функцією належності. Також перелічені основні переваги, які з’являються при застосуванні нео-фаззі нейрона у нейронній мережі в якості головного компонента. Результати навчання порівнювалися з метою оцінити якість апроксимації багатошарового персептрону, який використовує сигмоїдальні та ReLU-функції активації та нео-фаззі мережею, яка була запропонована. Моделі містили по чотири приховані шари, на кожному з яких знаходились по 50 нейронів. За результатами експериментів було встановлено, що нео-фаззі мережа за меншу кількість епох навчання досягає такої ж якості, як і багатошаровий персептрон.

За допомогою нео-фазі системи було отримано високі показники точності у роботі [7] для розпізнавання зображень (на прикладі розпізнавання

емоцій). Вирішувалася задача в умовах короткого набору даних і перекриття класів. Відмінною рисою системи є її гібридне навчання, що включає контрольоване навчання з викладачем, ледаче навчання за принципом «нейронів у точках даних» та самонавчання за Т. Кохоненом. Крім того, є можливість налаштовувати як синаптичні ваги, так і функції належності спеціальної форми з метою забезпечення покращених можливостей апроксимації. Розглянута система має високу швидкість навчання та забезпечує хорошу якість розпізнавання, що підтверджено результатами обчислювального експерименту.

Довга короткочасна пам’ять (LSTM) є однією з найпопулярніших моделей глибокого навчання, які використовуються сьогодні для прогнозування часових рядів, що є особливо важкою проблемою для вирішення через наявність довгострокового тренду, сезонних і циклічних коливань і випадкового шуму. Продуктивність LSTM сильно залежить від вибору кількох гіперпараметрів, які потрібно вибирати дуже ретельно, щоб отримати хороші результати. Будучи відносно новою моделлю, немає встановлених інструкцій щодо налаштування LSTM. У статті [8] було розглянуто цю прогалину в дослідженні. Набір даних було створено з індійського фондового ринку та розроблено для нього модель LSTM. Потім його було оптимізовано шляхом порівняння моделей без стану та моделей із збереженням стану та налаштування кількості прихованих шарів. Методи глибокого навчання потребують великої кількості обчислень. Якщо доступні спеціальні обчислювальні ресурси, можна проводити більше експериментів з більшою кількістю даних і вищими епохами. Це може надати більше ясності щодо налаштування гіперпараметрів. Розглянута робота обмежена налаштуванням базової архітектури LSTM. Аналогічним чином можна налаштувати різні типи моделей LSTM.

Для подолання труднощів існуючих моделей у роботі з нестаціонарними та нелінійними характеристиками високочастотних

фінансових часових рядів даних, особливо з їх слабкою здатністю до узагальнення, у статті [9] пропонується метод ансамблю, заснований на методах усунення шуму в даних, включаючи вейвлет-перетворення (WT) і аналіз сингулярного спектру (SSA), а також нейронна мережа довгострокової короткочасної пам’яті (LSTM) для побудови моделі прогнозування даних. Фінансовий часовий ряд розкладається та реконструюється WT і SSA для усунення шумів. За умови знешумлення реконструюється гладка послідовність з ефективною інформацією. Послідовність згладжування вводиться в LSTM і отримується прогнозоване значення. З індексом Dow Jones Industrial Middle (DJIA) як об’єктом дослідження ціна закриття DJIA кожні п’ять хвилин ділиться на короткострокову (1 година), середньострокову (3 години) і довгострокову (6 годин). відповідно. На основі середньоквадратичної помилки (RMSE), середньої абсолютної помилки (MAE), середньої абсолютної процентної помилки (MAPE) і стандартного відхилення абсолютної процентної помилки (SDAPE), експериментальні результати показують, що в короткостроковій, середньостроковій і довгостроковій перспективах усунення шуму в даних може значно підвищити точність і стабільність прогнозу, а також може ефективно покращити здатність узагальнення моделі прогнозування LSTM. Оскільки WT і SSA можуть отримувати корисну інформацію з оригінальної послідовності та уникати переобладнання, гібридна модель може краще зрозуміти структуру послідовності ціни закриття DJIA. А модель WT-LSTM краща, ніж еталонна модель LSTM і SSA-LSTM.

1.3 Аналіз часових рядів

Аналіз експериментальних даних, які спостерігалися в різні моменти часу, призводить до нових унікальних проблем у статистичному моделюванні та висновках. Очевидна кореляція, створена вибіркою суміжних моментів у часі, може серйозно обмежити застосовність багатьох звичайних статистичних методів, які традиційно залежать від припущення, що ці суміжні спостереження є незалежними та однаково розподіленими. Систематичний підхід, за допомогою якого хтось шукає відповіді на математичні та статистичні питання, поставлені цими часовими кореляціями, зазвичай називають аналізом часових рядів.

Вплив аналізу часових рядів на наукові програми можна частково задокументувати шляхом створення скороченого переліку різноманітних сфер, у яких можуть виникнути важливі проблеми часових рядів. Наприклад, багато знайомих часових рядів трапляються в галузі економіки, де ми постійно стикаємося з щоденними котируваннями фондових ринків.

Перший крок у будь-якому дослідженні часових рядів завжди передбачає ретельний аналіз записаних даних, нанесених на графік у часі. Цей аналіз часто пропонує метод аналізу, а також статистику, яка буде корисною для узагальнення інформації в даних. Перш ніж детальніше розглянути конкретні статистичні методи, доцільно згадати, що існують два окремі, але не обов’язково взаємовиключні підходи до аналізу часових рядів, які зазвичай визначають як підхід у часовій області та підхід у частотній області. Підхід у часовій області розглядає дослідження зв’язків із запізненням як найважливіше (наприклад, як те, що сталося сьогодні, впливає на те, що станеться завтра), тоді як предметний підхід розглядає дослідження циклів як найважливіше (наприклад, що економічний цикл через періоди зростання та спаду) [10].

1.3.1 Міри залежності

Повний опис часового ряду, який спостерігається як сукупність n випадкових змінних у довільні моменти часу $t_1, t_2, \dots, t_n$ для будь-якого натурального числа n, забезпечується спільною функцією розподілу, яка оцінюється як ймовірність того, що значення рядів разом менші за n констант $c_1, c_2, \dots, c_n$ тобто

$$F_{t_1,t_2,\dots,t_n}(c_1, c_2, \dots, c_n) = Pr(x_{t_1} \leq c_1, x_{t_2} \leq c_2, \dots, x_{t_n} \leq c_n).$$

На жаль, ці багатовимірні функції розподілу зазвичай не можуть бути легко записані, якщо випадкові величини не є спільно нормальними. Хоча функція спільного розподілу повністю описує дані, це громіздкий інструмент для відображення та аналізу даних часових рядів. Функцію розподілу необхідно обчислювати як функцію n аргументів, тому будь-яка побудова графіків відповідних багатовимірних функцій щільності практично неможлива. Граничні функції розподілу

$$F_t(x) = P\{x_t \leq x\}$$

або відповідні функції граничної густини

$$f_t(x) = \frac{\partial F_t(x)}{\partial x},$$

коли вони існують, часто є інформативними для дослідження граничної поведінки ряду. Іншою описовою мірою є середня функція значення.

Середня функція визначається як

$$\mu_{xt} = E(x_t) = \int_{-\infty}^{\infty} x f_t(x) dx,$$

якщо вона існує, де E позначає звичайний оператор очікуваного значення. Якщо немає плутанини щодо того, який часовий ряд розглядається, то нижній індекс можна опустити і записати μ_{xt} як μ_t.

Функція автоковаріації визначається як добуток другого моменту

$$\gamma_x(s,t) = cov(x_s, x_t) = E[(x_s - \mu_s)(x_t - \mu_t)],$$

для всіх s і t. Якщо немає плутанини щодо того, який часовий ряд мається на увазі, то нижній індекс можна опустити і записати $\gamma_x(s,t)$ як $\gamma(s,t)$. Варто звернути увагу, що $\gamma_x(s,t) = \gamma_x(t,s)$ для всіх моментів часу s і t.

Автоковаріація вимірює лінійну залежність між двома точками в одній серії, що спостерігаються в різний час. Дуже плавні ряди демонструють автоковаріаційні функції, які залишаються великими, навіть коли t і s знаходяться далеко одна від одної, тоді як уривчасті ряди, як правило, мають автоковаріаційні функції, які майже дорівнюють нулю для великих розривів. Якщо $\gamma_x(s,t) = 0$, а x_s та x_t не є лінійно пов'язаними, то між ними може існувати певна залежність. Проте, якщо x_s та x_t є двовимірними нормальними, $\gamma_x(s,t) = 0$ забезпечує їх незалежність. Зрозуміло, що для $s = t$ автоковаріація зводиться до (передбачуваної скінченної) дисперсії, оскільки

$$\gamma_x(t,t) = E[(x_t - \mu_t)^2] = var(x_t).$$

Автокореляційна функція (АКФ) визначається як

$$\rho(s,t) = \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\gamma(t,t)}}.$$

АКФ вимірює лінійну передбачуваність ряду в момент часу t, скажімо x_t, використовуючи лише значення x_s. Використовуючи нерівність Коші-Шварца, можна легко помітити, що $-1 \leq \rho(s,t) \leq 1$. Якщо можливо точно передбачити x_t на основі x_s за допомогою лінійного співвідношення $x_t = \beta_0 + \beta_1 x_s$, то кореляція буде $+1$ (при $\beta_1 > 0$) та -1 (при $\beta_1 < 0$). Таким чином, можна отримати приблизну міру здатності спрогнозувати ряд в момент часу t на основі значення в момент часу s.

Часто виникає необхідність виміряти передбачуваність іншого ряду y_t, використовуючи ряд x_s. Припускаючи, що обидва ряди мають кінцеву дисперсію, можна використати перехресну коваріаційну функцію між двома рядами:

$$\gamma_{xy}(s,t) = cov(x_s, y_t) = E\big[(x_s - \mu_{xs})\big(y_t - \mu_{yt}\big)\big].$$

Взаємна кореляційна функція записується у вигляді

$$\rho_{xy}(s,t) = \frac{\gamma_{xy}(s,t)}{\sqrt{\gamma_x(s,s)\gamma_y(t,t)}}.$$

Наведені вище ідеї можна легко поширити на випадок, коли розглядається більше, ніж два ряди. Наприклад, $x_{t1,}x_{t2}, \dots, x_{tr}$; тобто багатовимірний часовий ряд із r компонентами. Отже, у такому випадку

$$\gamma_{jk}(s,t) = E\big[\big(x_{sj} - \mu_{sj}\big)(x_{tk} - \mu_{tk})\big]; j,k = 1,2,\dots,r.$$

Розглянуті функції автоковаріації та крос-коваріації можуть змінюватися під час руху вздовж ряду, оскільки значення залежать як від s, так і від t, розташування моментів у часі. Наприклад, автоковаріація залежить від поділу x_s та x_t, скажімо, $h = |s - t|$, а не від того, де точки розташовані в часі. Поки точки розділені одиницями h, розташування двох точок не має значення. Це поняття називається слабкою стаціонарністю, коли середнє є постійним, і дозволяє аналізувати вибіркові дані часового ряду.

1.3.2 Нестаціонарні ряди

У поведінці часових рядів може спостерігатись певна закономірність. Для її розуміння вводиться поняття регулярності, використовуючи концепцію, яка називається стаціонарністю.

Суворо стаціонарний часовий ряд – це той, для якого ймовірнісна поведінка кожного набору значень $\{x_{t_1}, x_{t_2}, \dots, x_{t_k}\}$ є ідентичною набору, зміщеного у часі $\{x_{t_1+h}, x_{t_2+h}, \dots, x_{t_k+h}\}$. Це говорить про те, що

$$Pr\left\{x_{t_1} \le c_1, \dots, x_{t_k} \le c_k\right\} = Pr\{x_{t_1+h} \le c_1, \dots, x_{t_k+h} \le c_k\}$$

для всіх $k = 1, 2, \dots,$ усіх моментів часу $t_1, t_2, \dots, t_k$, усіх чисел $c_1, c_2, \dots, c_k$ і всіх часових зрушень $\mathrm{h} = 0, \pm 1, \pm 2$, ...

Якщо часовий ряд є суворо стаціонарним, то всі багатовимірні функції розподілу для підмножин змінних повинні узгоджуватися зі своїми відповідними функціями у зміщеному наборі для всіх значень параметра зсуву h. Наприклад, при $k = 1$

$$Pr\{x_s \le c\} = Pr\{x_t \le c\}$$

для будь-яких моментів часу s і t.

Таким чином, якщо функція дисперсії процесу існує, то функція автоковаріації ряду x_t задовольняє $\gamma(s,t) = \gamma(s+h, t+h)$ для всіх s, t і h. Цей результат можна інтерпретувати, сказавши, що функція автоковаріації процесу залежить лише від різниці в часі між s і t, а не від фактичного часу.

Слабко стаціонарний часовий ряд x_t є процесом скінченної дисперсії таким, що функція середнього значення μ_t, є постійною і не залежить від часу t, а функція автоковаріації $\gamma(s,t)$ залежить від s і t лише через їх різницю$|s-t|$.

Стаціонарність вимагає регулярності в середній та автокореляційній функціях, щоб ці величини (принаймні) можна було оцінити шляхом усереднення.

Оскільки середня функція $E(x_t) = \mu_t$ стаціонарного часового ряду не залежить від часу t, то можна записати $\mu_t = \mu$. Крім того, оскільки функція автоковаріації $\gamma(s,t)$ стаціонарного часового ряду x_t залежить від s і t лише через їх різницю$|s-t|$, то можна спростити позначення. Нехай $s = t + h$, де h представляє часовий зсув або затримку. Тоді

$$\gamma(t+h, t) = cov(x_{t+h}, x_t) = cov(x_h, x_0) = \gamma(h, 0).$$

Різниця в часі між $t+h$ і t така ж сама, як різниця між h і 0, то функція автоковаріації стаціонарного часового ряду не залежить від часового аргументу t.

Функцію автоковаріації стаціонарного часового ряду можна записати у такому вигляді:

$$\gamma(h) = cov(x_{t+h}, x_t) = E[(x_{t+h} - \mu)(x_t - \mu)].$$

Автокореляційна функція стаціонарного часового ряду:

$$\rho(h) = \frac{\gamma(t+h,t)}{\sqrt{\gamma(t+h,t+h)\gamma(t,t)}} = \frac{\gamma(h)}{\gamma(0)}.$$

Два часові ряди, скажімо, x_t і y_t, називаються спільно стаціонарними, якщо кожен з них є стаціонарним, і функція крос-коваріації

$$\gamma_{xy}(h) = cov(x_{t+h}, y_t) = E[(x_{t+h} - \mu_x)(y_t - \mu_y)]$$

є функцією лише зсуву h.

Функція крос-кореляції спільно стаціонарних часових рядів x_t та y_t визначається як

$$\rho_{xy}(h) = \frac{\gamma_{xy}(h)}{\sqrt{\gamma_x(0)\gamma_y(0)}}.$$

Лінійний процес x_t визначається лінійною комбінацією білого шуму w_t, що змінюється, і описується у такому вигляді:

$$x_t = \mu + \sum_{j=-\infty}^{\infty} \psi_j w_{t-j}, \quad \sum_{j=-\infty}^{\infty} |\psi_j| < \infty.$$

Для лінійного процесу функція автоковаріації задається виразом

$$\gamma_x(h) = \sigma_w^2 \sum_{j=-\infty}^{\infty} \psi_{j+h}\psi_j$$

для $h \geq 0$; а $\gamma_x(-h) = y_x(h)$. Цей метод демонструє функцію автоковаріації процесу в термінах відстаючих добутків коефіцієнтів. Необхідно лише виконання умови $\sum_{j=-\infty}^{\infty}|\psi_j| < \infty$, щоб процес мав кінцеву дисперсію.

Процес $\{x_t\}$ називається процесом Гауса, якщо n-вимірні вектори $x = (x_{t_1}, x_{t_2}, \dots, x_{t_n})'$ для кожного набору різних моментів часу $t_1, t_2, \dots, t_n$ і кожне натуральне число n мають багатовимірний нормальний розподіл.

1.3.3 Оцінка кореляції

Хоча теоретичні функції автокореляції та крос-кореляції корисні для опису властивостей певних гіпотетичних моделей, більшість аналізів необхідно виконувати з використанням вибіркових даних. Це обмеження означає, що лише вибіркові точки $x_1, x_2, \dots, x_n$ доступні для оцінки середньої, автоковаріаційної та автокореляційної функцій.

Вибіркова функція автоковаріації визначається як

$$\hat{\gamma}(h) = n^{-1} \sum_{t=1}^{n-h} (x_{t+h} - \overline{x})(x_t - \overline{x}),$$

де $\hat{\gamma}(-h) = \hat{\gamma}(h)$ для $h = 0, 1, \dots, n-1$.

Вибіркова автокореляційна функція визначається наступним чином:

$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)}.$$

Функція вибіркової автокореляції характеризує розподіл вибірки, що дозволяє оцінити, чи походять дані з абсолютно випадкового чи білого ряду, чи кореляції є статистично значущими з деякими затримками.

Вибіркова крос-коваріаційна функція $\hat{\gamma}_{xy}(h)$ записується у вигляді:

$$\hat{\gamma}_{xy}(h) = n^{-1} \sum_{t=1}^{n-h} (x_{t+h} - \bar{x})(y_t - \bar{y}),$$

де $\hat{\gamma}_{xy}(-h) = \hat{\gamma}_{yx}(h)$ визначає функцію для негативних лагів.

Вибіркова взаємну кореляційна функція має вигляд:

$$\hat{\rho}_{xy}(h) = \frac{\hat{\gamma}_{xy}(h)}{\sqrt{\hat{\gamma}_x(0)\hat{\gamma}_y(0)}}.$$

Вибіркову взаємну кореляційну функцію можна також розглядати графічно як функцію $lag\, h$ для пошуку випереджаючих або відстаючих зв'язків у даних. Оскільки $-1 \leq \hat{\rho}_{xy}(h) \leq 1$, практичну важливість піків можна оцінити, порівнюючи їх величини з теоретичними максимальними значеннями.

1.4 Прогнозування часових рядів

Для прогнозування часових рядів використовуються численні інструменти, які відрізняються один від одного, що дозволяє застосовувати їх при вирішенні відповідних класів задач. Одним із сучасних та ефективних інструментів прогнозування часових рядів є глибокі мережі. В даний час теорія і практика машинного навчання перебувають у стані «глибокої революції», до чого призвело успішне застосуванням мереж глибокого навчання, які представляють собою третє покоління нейронних мереж. Алгоритми глибокого навчання перевершили у точності найкращі альтернативні підходи, а в деяких випадках продемонстрували розуміння

сенсу вхідної інформації. Але серйозним недоліком мереж глибокого навчання є проблема визначення їх правильної структури, як підібрати достатню кількість їх шарів.

В рамках даної роботи будуть розглядатися гібридні мережі глибокого навчання на основі самоорганізації та рекурентні нейронні мережі.

1.4.1 Гібридні мережі глибокого навчання на основі самоорганізації

Доведено, що нейронні мережі є універсальними апроксиматорами і мають деякі чудові властивості, такі як паралельна обробка інформації, здатність працювати з неповними зашумленими вхідними даними та можливості навчання для досягнення бажаної реакції (виходу).

З іншого боку, МГУА використовує принцип самоорганізації, що дозволяє побудувати оптимальну структуру моделі прогнозування в процесі роботи алгоритму. Дуже перспективним є поєднання переваг цих обох підходів для вирішення проблеми – побудови ефективної моделі прогнозування фінансових ринків.

Розглянемо гібридну нейронну мережу глибокого навчання (рис. 1.1).

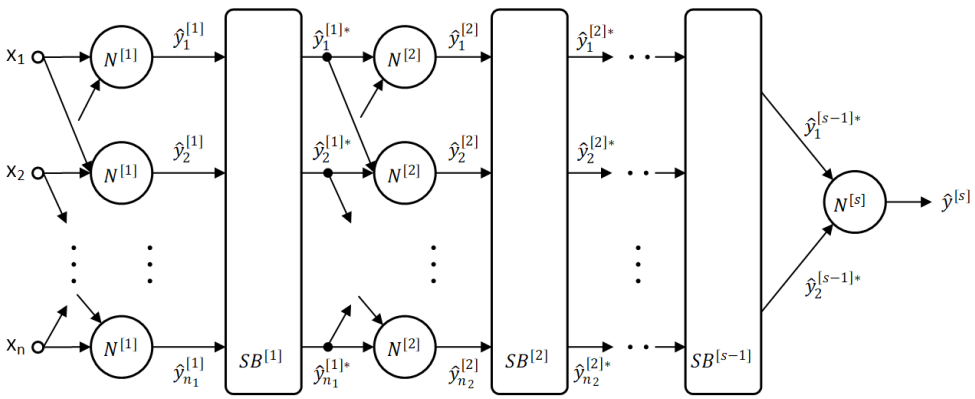

Рисунок 1.1 – Гібридна мережа глибокого навчання МГУА-нео-фазі

Дана гібридна мережа має багаторівневу архітектуру прямого зв’язку, а її вузлами є нео-фазі нейрони з двома входами та одним виходом. Найважливішими перевагами вузлів є висока швидкість навчання, обчислювальна простота, можливість знаходження глобального мінімуму критерію навчання в реальному часі, а також те, що вони характеризується нечіткими лінгвістичними правилами.

Для послідовного синтезу нейронних шарів використано ідею алгоритму МГУА, тобто шари синтезуються до тих пір, поки зовнішній критерій не почне зростати.

Алгоритм синтезу описується такими кроками:

1. Формуються пари з виходів нео-фазі нейронів поточного шару (на першій ітерації використовується набір вхідних сигналів). Кожна пара подається на відповідний нео-фазі нейрон.
2. Регулюються синаптичні вагові коефіцієнти кожного нео-фазі нейрона за допомогою навчальної вибірки.
3. Використовуючи тестову підвибірку, розраховується значення зовнішнього критерію для кожного нео-фазі нейрона:

$$\varepsilon_p^{[s]} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} (y(i) - \hat{y}_p^{[s]}(i))^2$$

де N_{test} – розмір тестової підвибірки, s – номер шару, p – номер нейрона в поточному шарі $p = \overline{1, n_s}$, $\hat{y}_p^{[s]}(i)$ – сигнал відповіді p-го нейрона s-го рівня для i-го вхідного вектора.

4. Розраховується мінімальне значення зовнішнього критерію для всіх нео-фазі нейронів поточного шару:

$$\varepsilon^{[s]} \geq \min_{p} \varepsilon_p^{[s]}.$$

Перевіряється умова $\varepsilon^{[s]} \geq \varepsilon^{[s-1]}$, де $\varepsilon^{[s]}, \varepsilon^{[s-1]}$ – критеріальні значення найкращих нейронів s-го та $(s-1)$-го шарів відповідно. Якщо умова виконується, то на попередньому рівні обирається нейрон, який має мінімальне значення критерію. В іншому випадку відбираються F найкращих нейронів відповідно до значення критерію та здійснюється перехід до кроку 1 і цикл повторюється.

5. Визначається остаточна структуру мережі. Рухаючись назад від найкращого нейрона передостаннього шару вздовж вхідних зв’язків і проходячи послідовно всі шари нейронів та зберігаючи у кінцевій структурі лише ті нейрони, які використовуються в наступному шарі.

Після зупинки роботи алгоритму МГУА можна сказати, що синтезована остаточна оптимальна структура МГУА-нео-фазі мережі. Вона є не тільки оптимальною, але й навченою та готовою до обробки нових даних. Однією з найважливіших переваг такого підходу є можливість використовувати прості, але дуже швидкі процедури навчання для коригування вагових коефіцієнтів [11].

1.4.2 Рекурентні нейронні мережі

Рекурентні нейронні мережі є гнучкими моделями, оскільки вони мають здатність кодувати часовий контекст у своїх зв’язках для фіксації змінної в часі динаміку основної системи.

Ефективною архітектурою вважається LSTM, яка широко використовується в даний час завдяки її чудовій продуктивності в точному

моделюванні як короткострокових, так і довгострокових залежностей у даних. LSTM намагається вирішити проблему зникаючого градієнта, не накладаючи жодних упереджень на нещодавні спостереження. Це відбувається за рахунок складного внутрішній процесора, який називається коміркою.

Внутрішній стан блоків LSTM змінюється лише через лінійні взаємодії. Це дозволяє інформації плавно поширюватися в часі з подальшим збільшенням ємності пам’яті комірки. LSTM захищає та контролює інформацію в комірці через три вентилі, які реалізуються сигмоїдним та точковим мультиплікатором. Щоб контролювати поведінку кожного вентиля, набір параметрів навчається за допомогою методу градієнтного спуску, щоб мінімізувати цільову функцію.

Схема комірки LSTM зображена на рис. 1.2.

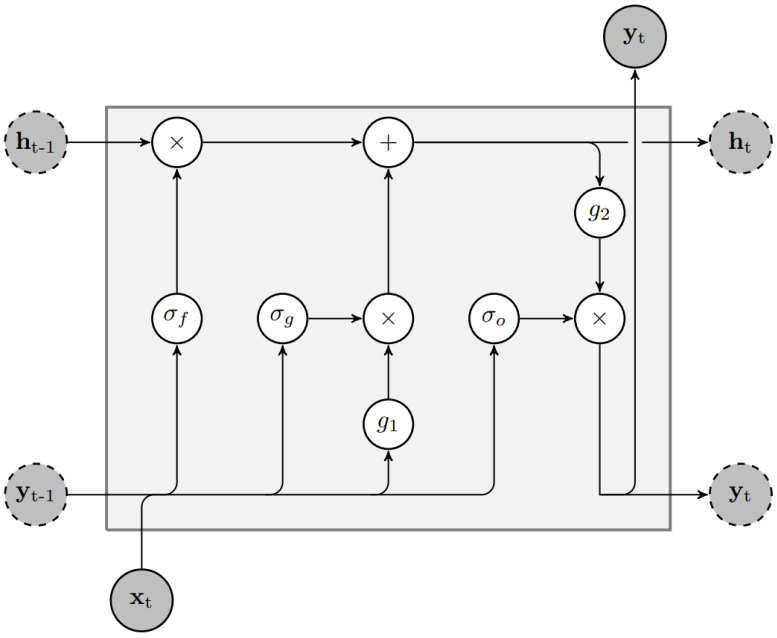

Рисунок 1.2 – Схема комірки в архітектурі LSTM

Темно-сірі кола з суцільною лінією – це змінні, вміст яких обмінюється з входом і виходом комірки. Темно-сірі кола з пунктирною лінією представляють внутрішні змінні стану, вмістом яких обмінюються комірки

прихованого шару. Оператори g_1 та g_2 є нелінійним перетворенням, зазвичай реалізованим як гіперболічний тангенс. Білі кружечки з $+$ та $\times$ представляють лінійні операції, тоді як σ_f, σ_u і σ_o є сигмоїдами, що використовуються у вентилях забуття, оновлення та виведення відповідно.

Прямий перехід для оновлення стану комірки та обчислення вихідних даних можна описати у такому порядку:

1) вентиль забуття: $\sigma_f[t] = \sigma\big(W_f x[t] + R_f y[t-1] + b_f\big)$,
2) стан кандидатів: $\tilde{h}[t] = g_1(W_h x[t] + R_h y[t-1] + b_h)$,
3) вентиль оновлення: $\sigma_u[t] = \sigma(W_u x[t] + R_u y[t-1] + b_u)$,
4) стан комірки: $h[t] = \sigma_u[t] \odot \tilde{h}[t] + \sigma_f[t] \odot h[t-1]$,
5) вентиль виходу: : $\sigma_o[t] = \sigma(W_o x[t] + R_o y[t-1] + b_o)$,
6) вихід: $y[t] = \sigma_o[t] \odot g_2(h[t])$,

де $x[t]$ – вхідний вектор у момент часу t. W_f, W_h, W_u і W_o – вагові матриці, які застосовуються до входу комірки LSTM. R_f, R_h, R_u і R_o – матриці, які визначають ваги рекурентних зв'язків, а b_f, b_h, b_u і b_o – вектори зміщення. Функція σ є сигмоїдою , тоді як g_1 і g_2 – нелінійними функціями активації, зазвичай реалізованими як гіперболічні тангенси, які транслюють значення на інтервал $[-1, 1]$. Символом $\odot$ позначено множення між двома векторами (добуток Адамара).

Кожен вентиль в комірці виконує специфічну та унікальну функцію. Вентиль забуття σ_f вирішує, яку інформацію слід відкинути з попереднього стану комірки $h[t-1]$. Вхідний вентиль σ_u працює зі станом $h[t-1]$ після модифікації вентилем забуття і вирішує, наскільки новий стан $h[t]$ має бути оновлений новим кандидатом $\tilde{h}[t]$. Щоб отримати результат $y[t]$, спочатку комірка фільтрує свій поточний стан із нелінійністю g_2. Потім вихідний вентиль σ_o вибирає частину стану, яка буде повернута на вихід. Кожен вентиль залежить від поточного зовнішнього входу $x[t]$ і попереднього виходу комірок $y[t-1]$.

Якщо $\sigma_f = 1$ і $\sigma_u = 0$, то поточний стан комірки переноситься на наступний інтервал часу точно таким, яким він є. В LSTM проблема зникнення градієнта не виникає через відсутність нелінійних функцій передачі. На практиці вентилі оновлення та забуття ніколи не бувають повністю відкритими або закритими через функціональну форму сигмоїди [12].

1.5 Вимоги до експериментальних даних

Точність моделей залежить від навчальної, перевірочної та тестової вибірок даних. Вони повинні відповідати деяким вимогам, щоб бути ефективно застосованими у процесі навчання. Розглянемо деякі основні критерії, необхідні для підбору експериментальних даних [13].

1.5.1 Неперервність та синхронність

Вимірювання та реєстрація експериментальних даних повинні відбуватися за однакові періоди дискретизації T_s. Це правило є обов'язковим для різних типів процесів. Якщо його порушити, то виникає ризик змінити спектральний склад досліджуваного сигналу. Таких моментів варто уникати, оскільки це призводить до зниження якості сигналу, тобто втрати деякої інформації. Також важливо проводити синхронні записи сигналів на вході та виході. Несинхронні вхідні та вихідні сигнали не можна використовувати для побудови передаточних функцій, тому що зв'язки між вхідними та вихідними

даними порушуються. Це особливо стосується систем, які працюють в режимі реального часу.

1.5.2 Повнота вибірки

Дані слід обирати на достатньо тривалому часовому відрізку з метою надання моделі описових параметрів різних станів досліджуваного процесу. Стани процесів можна розділити на два типи – перехідні та сталі. Перший стан передбачає перехід процесу з деякого вихідного режиму в цільовий. А якщо система тривалий час не змінює свій стан, то вважається, що на даному проміжку часу вона знаходиться у сталому стані. Такі моменти необхідно обов’язково враховувати, оскільки вони є притаманними більшості процесів.

1.5.3 Інформативність

Дотримання вимоги інформативності даних можна перевірити числом похідних досліджуваного сигналу. Вважається, що більше число похідних свідчить про більшу інформативність. Іншим підходом для визначення інформативності може бути оцінка величини дисперсії – більша дисперсія свідчить про вищу інформативність. Якщо частотна смуга сигналу перекриває амплітудно-часову характеристику деякого процесу, то такий процес можна назвати достатньо збудженим і це свідчить про достатню інформативність.

1.6 Критерії якості моделей

Оцінювання моделей здійснюють на основі помилки прогнозування або точності прогнозу. Для цього використовуються різні критерії, які набули широкого використання [14]. Розглянемо деякі з них та виділимо особливості кожного окремо.

1.6.1 MAE

Середня абсолютна похибка (Mean Absolute Error, MAE) визначається за формулою:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|e_i| = \underset{i=1,N}{\text{mean}}|e_i|.$$

MAE характеризує відхилення прогнозних даних від реальних в цілому і залежить від вимірювальної шкали та трансформації даних на виході. Варто відмітити деякі недоліки даного критерію:

1) складність інтерпретації;
2) залежність даних часового ряду.

Другий недолік є особливо незручним, оскільки для практичного застосування часто виникає необхідність порівнювати моделі на різних рядах.

1.6.2 MSE

Середня квадратична похибка (Mean Squared Error, MSE) описується формулою:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2,$$

де n – розмірність набору; Y_i – значення прогнозу; $\hat{Y}_i$ – цільове значення.

В основі даного критерію лежать квадрати помилок і він залежить від масштабу даних. Така залежність впливає на представлення значення у визначених одиницях вимірювання.

Для MSE компенсація негативних та позитивних значень не спостерігається, оскільки відбувається піднесення до квадрату. У такому разі помилки можуть знаходитись на всій множині невід'ємних чисел. Менше значення помилки свідчить про кращий результат прогнозування.

1.6.3 MAPE

Середня абсолютна похибка у відсотках (Mean Absolute Percentage Error, MAPE) розраховується за формулою:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{\hat{Y}_i}\right|,$$

де n – розмірність набору; Y_i – значення прогнозу; $\hat{Y}_i$ – цільове значення.

MAPE дозволяє оцінити прогнозну похибку у відсотках, а тому даний критерій зручно застосовувати для оцінки якості прогнозу. Даний критерій використовується досить часто, оскільки дозволяє інтерпретувати похибки незалежно від масштабу даних чи одиниць вимірювання, а отримані значення похибок є інтуїтивно зрозумілими.

До недоліків MAPE можна віднести ризик отримання екстремальних значень у випадку, коли проводиться робота з малими даними, тобто даний критерій чутливий до масштабу.

1.6.4 MPE

Середня похибка у відсотках (Mean Percentage Error, MPE) визначається за формулою:

$$MPE = \frac{1}{N}\sum_{i=1}^{N}\frac{e_i}{y_i} \times 100.$$

MPE та MAPE схожі, але є одна суттєва відмінність – значення у чисельнику без модуля. MPE відображає зміщення прогнозу у відсотках. Додатні значення похибки свідчать про занижений прогноз, а від’ємні – про завищений.

MPE зручно використовувати для оцінки загальної похибки прогнозування, але недоліком є те, що даний критерій важко сприймається інтуїтивно.

1.6.5 SMAPE

Симетричне середня абсолютна похибка у відсотках (Symmetric Mean Absolute Percentage Error, SMAPE) розраховується на основі значень середнього арифметичного прогнозного значення і цільового. Визначається за формулою:

$$SMAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{2|e_i|}{|y_i| + |\hat{y}_i|}.$$

Критерій SMAPE характеризує зсув прогнозу у напрямку завищення. Тобто, менше значення даного критерію свідчить про завищення прогнозу.

1.7 Висновки до розділу 1

У розділі було сформульовано актуальність теми дослідження. Розглянуті праці вітчизняних і закордонних вчених свідчать про ефективність нейронних мереж глибокого навчання при вирішенні задачі прогнозування часових рядів у фінансовій сфері. Особливу увагу привертають гібридні мережі глибокого навчання на основі самоорганізації, оскільки при їх застосуванні можна вирішити проблему малої вибірки навчальних даних, вибухового градієнту, а також вони потребують менше часу для навчання. Цього можна досягнути за рахунок оптимізації параметрів та структури мережі. Також розглянуто особливості аналізу та прогнозування часових рядів, сформульовано вимоги до експериментальних даних, зроблено огляд критеріїв якості моделей.

Отже, проблема дослідження є актуальною. Для її вирішення запропоновано дослідити гібридну мережу глибокого навчання МГУА-нео-фазі, а також порівняти ефективність її застосування в задачах прогнозування у фінансовій сфері з поширеною нейронною мережею LSTM.

РОЗДІЛ 2. МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ГІБРИДНИХ МЕРЕЖ ГЛИБОКОГО НАВЧАННЯ НА ОСНОВІ САМООРГАНІЗАЦІЇ

Нейронні мережі глибокого навчання, нечіткі набори та еволюційні обчислення надзвичайно розширили та збагатили поле моделювання, але вони також породили низку нових методологічних проблем. Багато з них пов’язані зі зростанням обчислювальних витрат по мірі росту розмірності моделі (скажімо, коли збільшується кількість змінних).

Найуспішніші підходи до гібридизації систем із навчанням та адаптацією були розроблені у вигляді нейронних нечітких систем на основі самоорганізації, а саме завдяки алгоритму МГУА. Цей метод запропонував академік О.Г. Івахненко (Інститут кібернетики НАН України) наприкінці 60-х років XX ст. [15]. Пізніше, українські вчені – професор Ю.П. Зайченко (НТУУ КПІ ім. Ігоря Сікорського) та професор Є.В. Бодянський (Харківський національний університет радіоелектроніки) – запропонували гібридну мережу глибокого навчання на основі МГУА, яка отримала назву МГУА-нео-фазі [16].

При моделюванні гібридних мереж глибокого навчання на основі МГУА варто звертати увагу на число вхідних сигналів, кількість нечітких наборів у вузлах мережі, а також якісну та кількісну характеристики навчальної та перевірочної вибірок даних.

2.1 Ідеї та основні принципи МГУА

МГУА належить до сімейства індуктивних алгоритмів, які використовуються при математичному моделюванні мультипараметричних даних. В основу методу покладені гібридизація та селекція, за рахунок чого

відбувається побудова складніших моделей. При такому підході кожен наступний крок дозволяє отримати вищу точність моделювання, оскільки модель ускладнюється.

Припустимо, що вибірка даних складається з N спостережень деяких векторів $X_{(i)}$ (вхід) та $Y_{(i)}$ (вихід):

$$\{X(1)\ \ Y(1)\}$$
$$\{X(2)\ \ Y(2)\}$$
$$\ldots.$$
$$\{X(N)\ \ Y(N)\}$$

Маючи такі вихідні дані, потрібно знайти $F_{(x)}$ (рис. 2.1), причому структура моделі невідома.

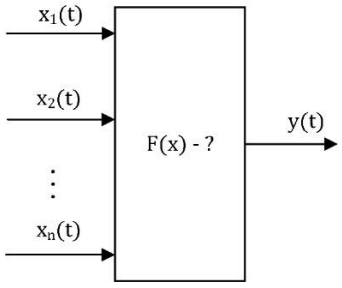

Рисунок 2.1 – Модель, яка потребує ідентифікації

Відмінними рисами даного завдання є:

1) вид функціональної залежності невідомий, а визначено лише клас моделей, наприклад, поліномів довільного ступеня нелінійності чи гармонійний ряд (Фур'є);

2) коротка вибірка даних;

3) тимчасові ряди $x_i(t)$, у випадку, нестаціонарні.

У разі застосування класичних методів статистичного аналізу, наприклад регресійного чи дисперсійного, неможливо, і потрібно

використовувати «нестандартні» методи, засновані, наприклад, на застосуванні ідей штучного інтелекту. До них належить метод групового обліку аргументів, запропонований О.Г. Івахненко та розвинений у численних роботах О.Г. Івахненко та його учнів. МГУА є способом індуктивного моделювання складних систем.

Наведемо основні принципи МГУА, які справедливі й у нечіткого його варіанта [17].

Повна залежність між входами та виходами у класі поліноміальних моделей може бути представлена за допомогою узагальненого полінома Колмогорова-Габора:

$$Y = a_0 + \sum_{i=1}^{N} a_i x_i + \sum_{j=1}^{N} \sum_{i \le j} a_{ij} x_i x_j + \sum_{i=1}^{N} \sum_{j \le i} \sum_{k \le j} a_{ijk} x_i x_j x_k + \cdots,$$

де всі коефіцієнти a_0, a_i, a_{ij} невідомі.

При визначенні значень коефіцієнтів (побудові моделі) у якості критерія використовується критерій регулярності (точності):

$$\overline{\varepsilon^2} = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i))^2 \to min,$$

де N – обсяг вибірки спостережень.

Необхідно забезпечити мінімум $\overline{\varepsilon^2}$.

Принцип множинності моделей. На даній вибірці існує множина моделей, що забезпечує нульову помилку. Для цього досить підвищувати рівень полінома моделі. Якщо є N вузлів інтерполяції, то можна побудувати ціле сімейство моделей, з яких кожна буде давати нульову помилку при проходженні через експериментальні точки:

$$\overline{\varepsilon^2} = 0.$$

При цьому варто враховувати, що ступінь нелінійності не повинен перевищувати $n - 1$ для вибірки точок у кількості n.

Принцип самоорганізації. Позначимо S – складність моделі, яка визначається числом членів полінома Колмогорова-Габора. Значення помилки $\overline{\varepsilon^2}$ залежить від складності моделі. Причому в міру підвищення складності вона спочатку падає, досягає мінімуму, а потім починає зростати. Потрібно вибрати таку оптимальну складність, при якій помилка буде мінімальна. Крім того, якщо враховувати дію перешкод, можна виділити наступні моменти:

1. При різному рівні завад залежність $\overline{\varepsilon^2}$ від складності S буде змінюватися, зберігаючи при цьому загальну спрямованість (мається на увазі, що з підвищенням складності вона спочатку зменшуватиметься, а потім зростатиме).
2. У разі збільшення рівня перешкод величина $\min_S \overline{\varepsilon^2}$ зростатиме.
3. Зі збільшенням рівня перешкод оптимальна складність $S_0 = \arg\min \overline{\varepsilon^2}$ буде зменшуватися (оптимальне значення складності зміщуватиметься вліво). Причому $\overline{\varepsilon^2}(S_0) > 0$, якщо рівень перешкод не нульовий.

Теорема про неповноту Геделя. У будь-якій формальній логічній системі є ряд тверджень і теорем, які не можна ні спростувати, ні довести, залишаючись у межах цієї системи аксіом.

У разі ця теорема означає, що вибірка завжди неповна.

Один із способів подолання неповноти вибірки, яка є наслідком теореми про неповноту Геделя, — це принцип зовнішнього доповнення. Він полягає у наступному. Вся вибірка розбивається на дві частини: навчальну та перевірочну. При цьому перевірна вибірка - це така вибірка, точки якої не

використовувалися при навчанні системи (тобто при пошуку оцінних значень коефіцієнтів полінома Колмогорова-Габора).

Пошук найкращої моделі здійснюється таким чином:

- вся вибірка ділиться на навчальну та перевірочну: $N_{\text{виб}} = N_{\text{навч}} + N_{\text{перев}}$;
- на навчальній вибірці $N_{\text{навч}}$ визначаються оцінки коефіцієнтів a_0, a_i, a_{ij};
- на перевірочній вибірці $N_{\text{перев}}$ відбираються найкращі моделі.

Принцип свободи вибору (неостаточність проміжного рішення):

1. Для кожної пари x_i і x_j будуємо часткові описи (всього C_n^2) виду
   - або $y^{(s)} = \varphi(x_i, x_j) = a_0 + a_i x_i + a_j x_j, s = 1 \dots C_N^2$ (лінійні);
   - або $y^{(s)} = \varphi(x_i, x_j) = a_0 + x_i + a_j x_j + a_{ii} x_i^2 + a_{ij} x_i x_j + a_{jj} x_j^2, s = 1 \dots C_N^2$ (квадратичні).

2. Визначаємо коефіцієнти цих моделей за МНК, використовуючи навчальну вибірку, тобто знаходимо оцінки $\hat{a}_0, \hat{a}_1, \dots, \hat{a}_j, \dots, \hat{a}_N, \hat{a}_{11}, \dots, \hat{a}_{ij}, \dots, \hat{a}_{NN}$.

3. Далі на перевірочній вибірці для кожної з цих моделей шукаємо оцінку за критерієм регулярності:

$$\overline{\delta_s^2} = \frac{1}{N_{\text{перев.}}} \sum_{i=1}^{N_{\text{перев.}}} [Y(k) - \bar{Y}_S(k)]^2,$$

де $Y(k)$ – дійсне значення вихідний змінної в k-й точці перевірочної вибірки; $\bar{Y}_S(k)$ – вихідне значення у k-й точці перевірочної вибірки відповідно до s-ї моделі; $N_{\text{перев.}}$ – число точок перевірочної вибірки. Або за критерієм незміщеності:

$$N_{\text{см}} = \frac{1}{N_1 + N_2} \sum_{k=1}^{N_{\text{пров.}}} (y_k^* - y_k^{**})^2,$$

де вся вибірка поділяється на дві частини N_1 і N_2; y_k^* – виходи моделі, побудованої на вибірці N_1; y_k^{**} – виходи моделі, побудованої на вибірці N_2.

4. Визначаємо F найкращих моделей (рис. 2.2) за одним із вищевказаних критеріїв. Вибрані моделі y_i подаємо на другий ряд. Шукаємо

$$z_I = \varphi^{(2)}(y_i, y_j) = a_0^{(2)} + a_1^{(2)} y_i + a_2^{(2)} y_j + a_3^{(2)} {y_i}^2 + a_4^{(2)} y_i y_j + a_5^{(2)} {y_i}^2.$$

Оцінка тут така сама, як у першому ряді. Відбір найкращих здійснюється знову так само, але $F_2 < F_1$. Процес конструювання рядів повторюється до того часу, поки середній квадрат помилки зменшуватиметься. Коли на шарі m отримуємо збільшення помилки $\overline{\varepsilon^2}$, процес синтезу моделі припиняємо.

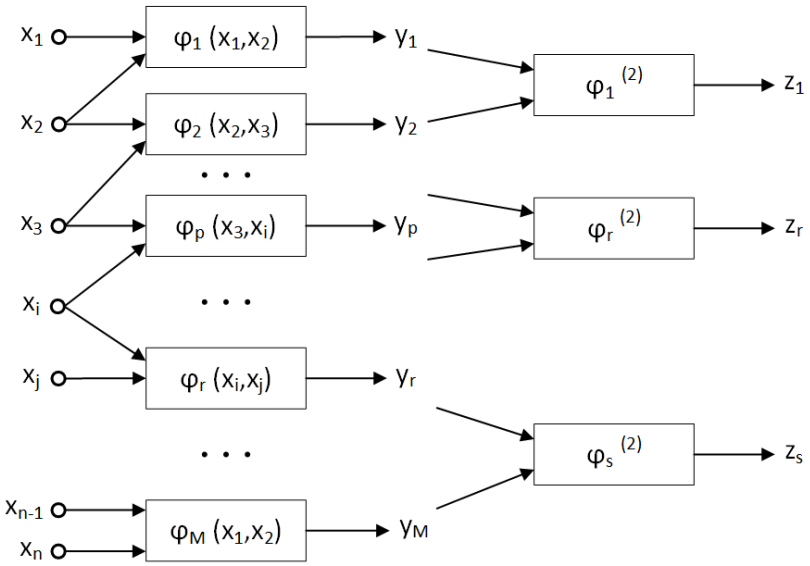

Рисунок 2.2 – Багаторядний алгоритм МГУА

Перевагами даного методу є можливість відновлення невідомої довільно складної залежності по обмеженій вибірці та можливість адаптації параметрів моделі при одержанні нових даних експериментів.

2.2 Синтез структури гібридної мережі глибокого навчання на основі самоорганізації

МГУА-нео-фазі мережа глибокого навчання надає такі переваги:

1. Отримання структури нейронної мережі, готової до прогнозування, у процесі навчання.
2. Можливість використання для навчання (та синтезу структури) відносно невеликої вибірки даних.
3. Зменшення обчислювальних витрат при адаптації параметрів у вузлах мережі.
4. Висока швидкість навчання за рахунок невеликої кількості вхідних сигналів.

Враховуючи, що кількість вхідних сигналів має бути (бажано) мінімальною, розглянемо процес еволюції структури мережі на прикладі з чотирма входами $n = 4$ (рис. 2.3).

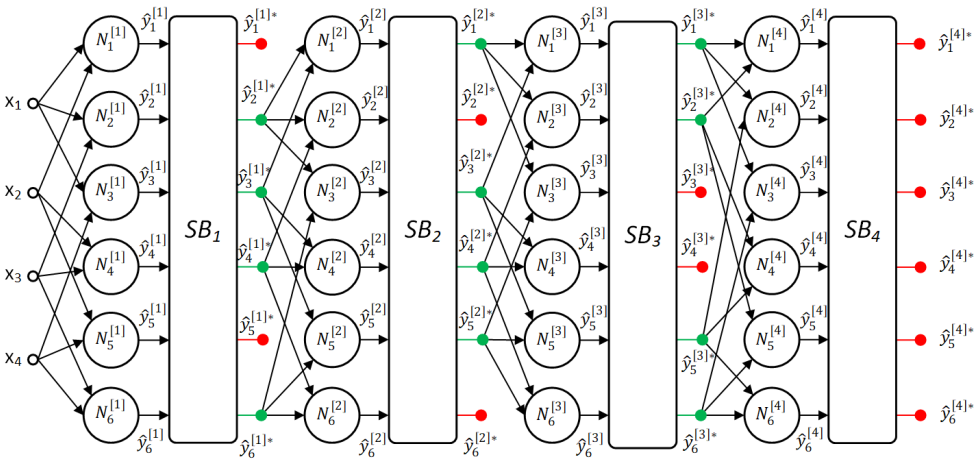

Рисунок 2.3 – Нечітка нейронна мережа МГУА-нео-фазі

Перший шар відіграє роль рецепторного, тобто він приймає вектор вхідних сигналів розмірності (4×1): $x = (x_1, x_2, x_3, x_4)^T$.

Після проходження рецепторного шару вектор сигналів надходить на перший прихований шар, що містить вузли $N^{[1]}$ з двома входами. Їх кількість визначається за формулою $n_1 = c_n^2$ (для даного випадку n_1=6). Перший шар формує вихідні сигнали $\hat{y}_l^{[1]}$, де $l = 1,2,\dots,0.5n(n-1)$, які надходять до першого селекційного блоку $SB^{[1]}$. Він призначений для відбору кращих варіантів $n_1^* < n_1$ з множини $\hat{y}_l^{[1]}$ за величиною дисперсії $\sigma^2_{\hat{y}_l^{[1]}}$. Далі відбувається формування комбінацій $\hat{y}_l^{[1]*}$, $\hat{y}_p^{[1]*}$ у кількості $n_2 = c_{n_1^*}^2$, причому $(n \le n_2 \le 2n)$. Сформовані сигнали надходять до другого прихованого шару з вузлами $N^{[2]}$. Далі сигнали $\hat{y}_l^{[2]}$, отримані після навчання вузлів даного прихованого шару, переходять до селекційного блоку $SB^{[2]}$, де обираються кращі варіанти за точністю, яка повинна перевищувати точність кращого сигналу попереднього шару $\hat{y}_l^{[1]*}$. Процес формування сигналів на наступних прихованих шарах відбувається аналогічно і продовжується до моменту, коли сформований сигнал на деякому блоці селекції $SB^{[s+1]}$ виявиться гіршим за кращий сигнал на попередньому шарі s за умовою $\sigma^2_{\hat{y}_l^{[s+1]}} > \sigma^2_{\hat{y}_l^{[s]}}$. У даному випадку це сталося після проходження четвертого блоку селекції $SB^{[4]}$.

Після цього процес переходить у зворотний напрямок (рис. 2.4).

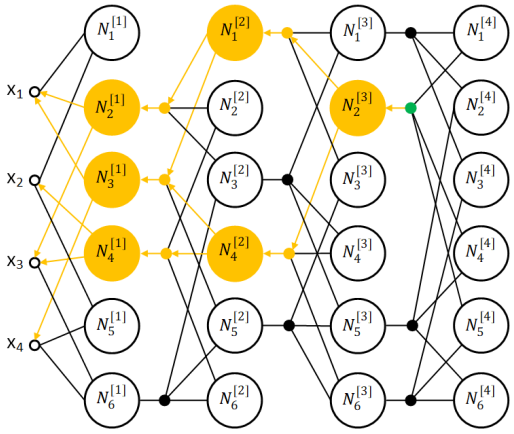

Рисунок 2.4 – Зворотний напрямок синтезу структури

Спочатку визначається кращий вузол $N^{[s]}$, який знаходиться на попередньому шарі і має вихідний сигнал $\hat{y}^s$. Потім, завдяки відомим зв'язкам з нейронами попереднього прихованого шару, відбувається послідовний прохід всіх шарів до початкового. У результаті зворотного руху з'являється структура мережі глибокого навчання МГУА-нео-фазі (рис. 2.5).

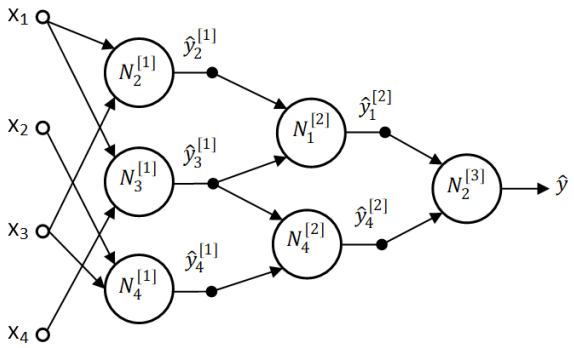

Рисунок 2.5 – Структура мережі глибокого навчання МГУА-нео-фазі

Розглянутий підхід дає можливість отримати не лише оптимальну структуру, а й добре навчену мережу за рахунок алгоритму МГУА. Її можна одразу застосовувати для прогнозування. Іншою перевагою є вирішення проблеми зникнення градієнта за рахунок послідовного пошарового навчання. Оскільки відбувається навчання невеликої кількості нейронів з двома входами, то це дозволяє використовувати невеликі навчальні вибірки та суттєво скоротити час навчання [18].

2.3 Нео-фазі нейрон у якості вузла гібридної мережі на основі самоорганізації

Ефективність навчання вузлів та синтезу структури гібридної мережі глибокого навчання на основі МГУА можна підвищити, якщо її вузлами будуть нео-фазі нейрони. Такий підхід дозволяє адаптувати тільки вагові коефіцієнти, не змінюючи параметри функцій належності.

Перейдемо до розгляду архітектури NFN (рис. 2.6), який має два входи ($n = 2$) та один вихід.

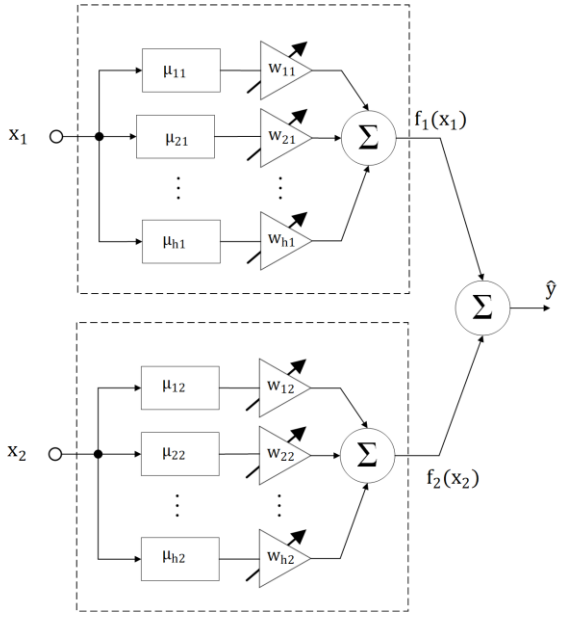

Рисунок 2.6 – Нео-фазі нейрон (NFN)

Вихідне значення NFN у загальному випадку визначається за формулою:

$$\hat{y} = \sum_{i=1}^{n} f_i(x_i).$$

Нелінійні синапси NS_n являються структурними блоками NFN. Вони призначені для перетворення сигналу від n-го входу. Вихідне значення NS_n розраховується за формулою:

$$f_i(x_i) = \sum_{j=1}^{h} w_{ji}\mu_{ji}(x_i).$$

Результатом синаптичного перетворення є нечіткий висновок, який можна інтерпретувати у такій формі: якщо X_i є x_i, то на виході отримаємо w_{ji}, причому значення x_{ji} характеризується функцією належності μ_{ji}, а синаптичною вагою виступає w_{ji}.

Векторний сигнал $x = (x_1, \dots, x_n)^T$, який подається на входи NFN, визначає його вихід під впливом функцій належності $\mu_{ji}(x_i)$ та синаптичних ваг w_{ji}, які налаштовуються і визначаються при навчанні на попередній епосі:

$$\hat{y} = \sum_{i=1}^{n}\sum_{j=1}^{h} w_{ji}\mu_{ji}(x_i).$$

В результаті NFN характеризується синаптичними вагами у кількості hn, які підлягають визначенню.

До важливих переваг NFN можна віднести високу швидкість навчання, простоту обчислень та можливість знаходити глобальний мінімум критерію навчання.

2.4 Функції належності

Фазифікація попередньо нормалізованих вхідних сигналів x_i (зазвичай $0 \le x_i \le 1$) відбувається за рахунок функції належності в антецеденті. Традиційно використовуються трикутні функції належності, але вони

забезпечують лише кусково-лінійну апроксимацію, а цей факт у більшості випадків може призвести до зниження точності отриманих результатів [19]. Щоб мінімізувати його негативний ефект, можна збільшити кількість функцій. Але це призводить до збільшення кількості синаптичних вагових коефіцієнтів, тому складність архітектури зростає разом з часом, необхідним для її навчання. Пропонується розглянути інші типи функцій належності та визначити їх оптимальну кількість.

2.4.1 Дзвоноподібна функція

Дзвоноподібна функція визначається формулою:

$$\mu(x) = \frac{1}{1 + (\frac{|x - c|}{a})^{2b}}.$$

Розглянемо графік даної функції (рис. 2.7).

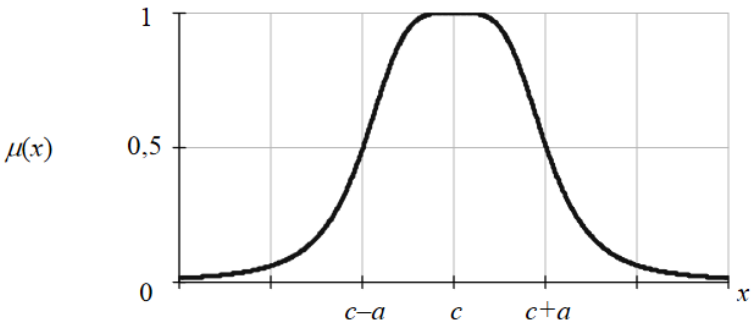

Рисунок 2.7 – Дзвоноподібна функція

На графіку помітно, що вона має три параметри:

1) c – модальна змінна ($\mu(c) = 1$);

2) $a > 0$ – позначає відстань до точок переходу від пікової точки ($\mu(c \pm a) = 0.5$);

3) $b \geq 0$ – визначає крутизну (або нахил).

Побудова дзвоноподібної функції належності потребує ідентифікації трьох точок:

1) модальної змінної c;

2) перехідної точки $x_{0.5}$, що визначає параметр a;

3) довільної точки, відмінної від c та $x_{0.5}$, на основі якої розраховується параметр b.

2.4.2 Функція Гауса

Функція належності Гауса (рис. 2.8) подібна до графіка щільності нормального ймовірнісного розподілу (він також називається гаусівським).

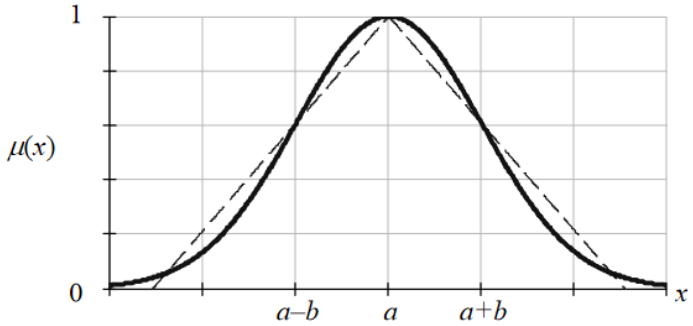

Рисунок 2.8 – Функція Гауса

Дану функцію визначають параметри a і b ($a > b$). Також варто зазначити, що існує деяка точка $c = 1$ і точки перегину функції визначаються виразом $x = a \pm b$.

Щоб задати функцію належності Гауса, необхідно ініціалізувати точки c та d (відстані до точок переходу від точки c).

Значення b можна розрахувати за формулою (враховуючи значення точки d):

$$b = \frac{d}{\sqrt{2 \ln 2}}.$$

Функцію належності Гауса описує формула:

$$\mu(x) = exp\left[-\frac{(x-a)^2}{2b^2}\right].$$

2.4.3 Трикутна функція

Графік трикутної функції належності наведено на рис. 2.9. Вона визначається на відрізку $X = [a_1; a_2]$ і набуває максимуму в точці $a = 1$.

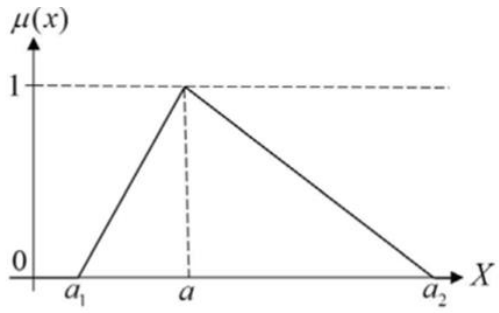

Рисунок 2.9 – Трикутна функція

Вона виражається формулою:

$$\mu(x) = \begin{cases} 1 - \dfrac{a - x}{a - a_1}, \text{якщо } a_1 \le x \le a; \\ 1 - \dfrac{x - a}{a_2 - a}, \text{якщо } a \le x \le a_2; \\ 0, \text{в інших випадках,} \end{cases}$$

де a_1, a, a_2 – деякі числа.

Трикутну функцію зручно застосовувати для опису термів типу «середнє значення», «схоже на об'єкт», «скоріше за все» та інших подібних.

2.5 Алгоритм навчання

Навчання вузлів мережі глибокого навчання на основі самоорганізації відбувається за рахунок мінімізації цільової функції, тобто критерію навчання. Ним являється стандартна локальна квадратична функція:

$$E(k) = \frac{1}{2}(y(k) - \hat{y}(k))^2 = \frac{1}{2}e(k)^2 = \frac{1}{2}(y(k) - \sum_{i=1}^{2}\sum_{j=1}^{h} w_{ji}\mu_{ji}(x_i(k)))^2$$

Мінімізацію можна виконати за допомогою застосування алгоритму стохастичного градієнтного спуску. Прискорення навчання можна досягти, використовуючи однокроковий алгоритм Відроу-Гоффа. Цей алгоритм характеризується згладжувальними та фільтрувальними властивостями.

Пакетний режим. У разі наявності навчальної вибірки, яка визначена попередньо, навчання можна провести протягом однієї епохи за допомогою методу найменших квадратів (МНК):

$$w^{[1]}(N) = \left(\sum_{k=1}^{N} \mu^{[1]}(k)\mu^{[1]T}(k)\right)^{+} \sum_{k=1}^{N} \mu^{[1]}(k)y(k) = P^{[1]}(N) \sum_{k=1}^{N} \mu^{[1]}(k)y(k),$$

де $P^1(N)$ – псевдообернена матриця Мура-Пенроуза, $y(k)$ – дійсне значення зовнішнього опорного сигналу.

Режим on-line. Використання рекурентного методу найменших квадратів (РНМК) дозволяє проводити навчання, коли дані спостереження подаються послідовно:

$$\begin{cases} w_l^{ij}(k) = w_l^{ij}(k-1) + \dfrac{P^{ij}(k-1)(y(k) - \left(w_l^{ij}(k-1)\right)^T \varphi^{ij}(x(k)))\varphi^{ij}(x(k))}{1 + \left(\varphi^{ij}\left(x(k)\right)\right)^T P^{ij}(k-1)\varphi^{ij}(x(k))}, \\ P^{ij}(k) = P^{ij}(k-1) - \dfrac{P^{ij}(k-1)\varphi^{ij}\left(x(k)\right)\left(\varphi^{ij}\left(x(k)\right)\right)^T P^{ij}(k-1)}{1 + \left(\varphi^{ij}\left(x(k)\right)\right)^T P^{ij}(k-1)\varphi^{ij}\left(x(k)\right)}. \end{cases}$$

2.6 Висновки до розділу 2

У розділі розглянуто математичні основи гібридних мереж на основі самоорганізації та алгоритм МГУА, який лежить в основі процесу синтезу їх структури. Визначено, що кількість вхідних сигналів впливає на розмірність результуючої мережі, тому перевагою буде використання якомога меншого числа входів. Також розглянуто нео-фазі нейрон у якості вузла. Визначено його переваги, особливості фазифікації вхідного сигналу та досліджено алгоритм навчання. Попередньо встановлено, що збільшення кількості функцій належності позитивно впливає на точність прогнозування, але спонукає до зростання обчислювальних витрат. Тому запропоновано

дослідити застосування різних типів функцій належності, які також були розглянуті.

РОЗДІЛ 3. ВИКОНАННЯ ТА АНАЛІЗ ОБЧИСЛЮВАЛЬНИХ ЕКСПЕРИМЕНТІВ

Проводилися експериментальні дослідження гібридних мереж глибокого навчання на основі самоорганізації з метою оптимізації їх параметрів та структури. Вирішувалася задача короткострокового та середньострокового прогнозування фондових індексів на біржі.

В процесі проведення експериментів змінювалися функції належності в антецедентах правил, число входів, кількість нечітких наборів та відношення навчальна/тестова вибірка.

Для оцінки ефективності застосування досліджуваних моделей гібридних мереж глибокого навчання на основі самоорганізації у задачі прогнозування аналогічні прогнози було отримано з використанням рекурентної нейронної мережі LSTM, оскільки вона визнана ефективним інструментом моделювання для короткострокового та середньострокового прогнозування нестаціонарних часових рядів і набула широкого використання в задачах прогнозування.

3.1 Дані для дослідження

В якості вхідних даних було взято корпоративний індекс прибутковості Emerging Markets Bond total Return Index (EMBRI) фондової біржі NASDAQ [20] за період з січня по серпень 2022 року (рис. 3.1). В результаті було отримано часовий ряд даних розміром 168 точок.

Рисунок 3.1 – Emerging Markets Bond total Return Index (EMBRI)

Для визначення ступенів взаємозв'язку залишків кожного наступного значення з попереднім було розраховано коефіцієнти автокореляції для часового ряду. Оскільки вважається, що доцільним для забезпечення статистичної достовірності коефіцієнтів використовувати правило, згідно якого максимальний лаг не повинен перевищувати $n/4$ (n – число примірників у вибірці), то розраховувалися коефіцієнти для 42 лагів. Графік коефіцієнтів автокореляції зображено на рис. 3.2.

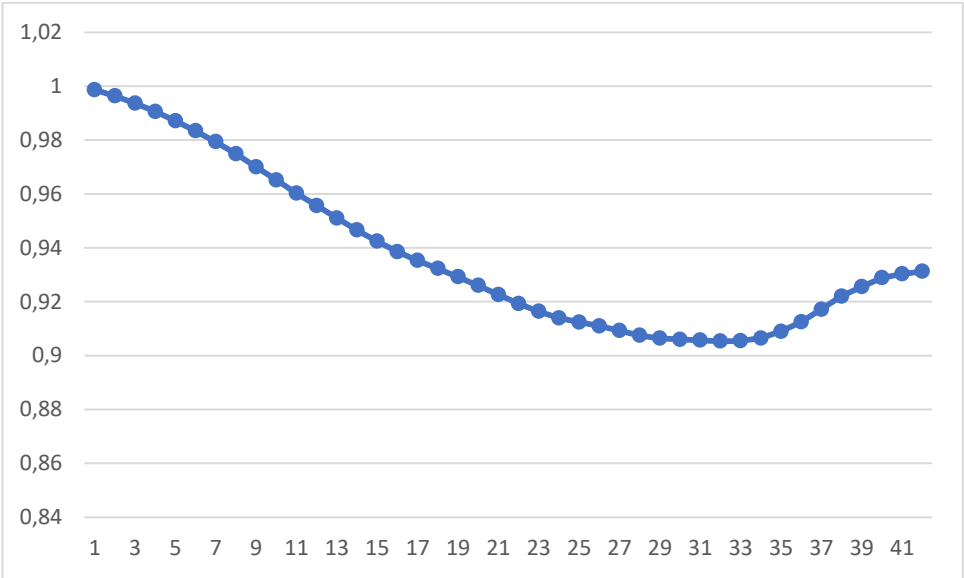

Рисунок 3.2 – Корелограма рівнів часового ряду

Значення коефіцієнтів для перших 20 лагів показано у таблиці 3.1, а повну таблицю наведено у Додатку Б.

Таблиця 3.1 – Коефіцієнти автокореляції для 20 лагів

Lag	Coefficient
1	0,9987
2	0,9964
3	0,9937
4	0,9906
5	0,9872
6	0,9834
7	0,9794
8	0,9749
9	0,97
10	0,9651
11	0,9602
12	0,9557
13	0,951
14	0,9466
15	0,9424
16	0,9385
17	0,9353
18	0,9323
19	0,9292
20	0,926

3.2 Розробка програмного продукту

Код програмного продукту написаний мовою програмування Python [21]. При виборі мови програмування враховувалися такі критерії:

- визнання світовою спільнотою програмістів;
- чистота синтаксису;
- можливість застосовувати об’єктно-орієнтований підхід;
- зручність виконання математичних обчислень;
- кросплатформність.

Мова Python набрала високу популярність за останні роки, в період активного розвитку та застосування штучного інтелекту. Ще однією важливою особливістю даної мови програмування є зручна розробка програмних модулів (модульність). На даний час існує велика кількість Python-бібліотек, за допомогою яких можна виконувати різної складності математичні обчислення, застосовувати матричні перетворення при роботі з масивами даних, використовувати поширені алгоритми при розробці власних програм [22].

У якості середовища розробки обрано PyCharm[23]. При виборі враховувалися такі критерії:

- зручність написання програмного коду;
- вартість ліцензії;
- наявність розширень, що дозволяються оптимізувати розробку.

PyCharm можна використовувати як безкоштовно, так і за ліцензією, завдяки чому він користується шаленою популярністю серед розробників. Важливим моментом також є той факт, що при розробці нових версій враховуються побажання спільноти програмістів. PyCharm можна інсталювати на будь-якій операційній системі, додавати різні корисні розширення та контролювати версії програмного коду під час розробки.

Для виконання операцій, пов’язаних з машинним навчання, було обрано бібліотеку Scikit-learn[24]. Вона є стабільною та добре зарекомендувала себе. Scikit-learn надає інструменти для візуалізації даних, виконання складних розрахунків та дозволяє працювати з тензорами.

Реалізація нечіткої логіки виконувалася за допомогою бібліотеки SciKit-Fuzzy[25]. Вона містить набір поширених алгоритмів, функцій належності, завдяки чому зручно виконувати фазифікацію і дефазифікацію сигналів.

В результаті було розроблено програмний продукт, який має модульну структуру, а його код написаний в об’єктно-орієнтованому стилі. Загальний алгоритм роботи програмного продукту наведено у Додатку В.

На початку роботи програми проходить ініціалізація даних, а саме: завантажується набір даних, генеруються параметри експериментів, налаштовуються системні шляхи для збереження результатів. Варто зазначити, що обробка набору даних проходить у кілька послідовних етапів:

- отримання даних з ресурсу;
- підготовка та збереження часового ряду у визначеному форматі для подальшого опрацюванні;
- генерування графіків динаміки зміни індексів;
- нормалізація даних;
- створення даних для навчання.

На цьому етапі вибірка даних спостережень розділяється на навчальну, тестову та контрольну. В залежності від параметрів експерименту, підвибірки мають різні розмірності.

Після опрацювання даних для дослідження відбувається розрахунок коефіцієнтів автокореляції та побудова корелограми. Якщо корелограма є незадовільною, то програма завершується формуванням звіту. В іншому випадку відбувається перехід до проведення експериментів.

На першому кроці проходить визначення параметрів моделі, на основі яких має відбуватися навчання.

На другому кроці запускається процедура навчання та синтезу структури мережі глибокого навчання на основі самоорганізації.

Отримана мережа (навчена) на попередньому кроці проходить оцінку точності за метриками. Якщо бажана точність не досягнута, то відбувається перехід до наступного експерименту. Цикл продовжується до тих пір, поки не пройдуть всі експерименти або буде досягнута бажана точність прогнозування на контрольній вибірці. Після цього результати проведених експериментів візуалізуються, формуються звіти і робота програми завершується.

Дослідження рекурентної нейронної мережі LSTM відбувається за таким самим алгоритмом.

3.3 Експериментальні дослідження гібридних мереж глибокого навчання на основі самоорганізації

Експериментальні параметри гібридної мережі глибокого навчання на основі самоорганізації та рекурентної мережі LSTM наведено у таблиці 3.2.

Таблиця 3.2 – Експериментальні параметри

Мережа	Параметр	Значення
МГУА-нео-фазі	Функція належності	Дзвоноподібна
		Гауса
		Трикутна
МГУА-нео-фазі, LSTM	Інтервал прогнозування	1
		3
		5
		7
		20
МГУА-нео-фазі, LSTM	Період прогнозування	3
		5
		7

		20
МГУА-нео-фазі, LSTM	Число входів	3
		4
		5
МГУА-нео-фазі	Кількість нечітких наборів	3
		4
		5
МГУА-нео-фазі, LSTM	Співвідношення навчальна/тестова вибірка	0,6 (60%)
		0,7 (70%)
		0,8 (80%)

Першу серію експериментів було проведено з гібридною мережею глибокого навчання з нео-фазі-нейронами у якості вузлів. Під час експериментів змінювали такі параметри: співвідношення навчальна/тестова вибірка, кількість входів , кількість нечітких наборів на змінну та функції належності: дзвоноподібна, Гауса та трикутна. Метою експериментів було знайти оптимальні значення параметрів.

Прогнозний період приймався 5 днів, а показники точності були взяті MSE та MAPE.

У першому експерименті було досліджено дзвоноподібну функцію. Після експерименту знайдено оптимальні параметри гібридної мережі глибокого навчання: кількість входів – 3, кількість нечітких наборів – 3, коефіцієнт – 0,8. При цих значеннях параметрів досягалася найкраща точність на дослідному зразку: MSE = 0,424, MAPE = 0,155.

Наступні експерименти були проведені для гібридної мережі з функцією належності Гауса. Після проведення експериментів оптимальні параметри отримано такі: кількість входів – 3, кількість правил – 4, співвідношення навчання/тест – 0,6. З цими параметрами результати прогнозування представлені в таблиці 3.3.

Таблиця 3.3 – Найкращий прогноз із функцією належності Гауса (кількість входів: 3; правила: 4; співвідношення: 0,6)

Date	Real	Forecast	MSE	MAPE
17.08.2022	399,15	398,44	0,504	0,178
18.08.2022	399,12	398,17	0,902	0,238
19.08.2022	397,97	398,50	0,281	0,133
22.08.2022	396,82	397,40	0,336	0,146
23.08.2022	396,62	396,93	0,096	0,078
		Min:	0,096	0,078
		Avg:	0,424	0,155
		Max:	0,902	0,238

У таблиці 3.4 представлено процес генерації структури найкращої мережі з оптимальними параметрами.

Таблиця 3.4 – Процес генерації структури мережі для найкращого результату з функцією належності Гауса (кількість входів: 3; правила: 4; співвідношення: 0,6)

NFN	SB1	SB2	SB3
(0, 1)	6,458466		
(0, 2)	2,746775		
(1, 2)	4,979467		
((0, 1), (0, 2))		0,020353	
((0, 1), (1, 2))		0,003018	
((0, 2), (1, 2))		0,000982	
(((0, 1), (0, 2)), ((0, 1), (1, 2)))			0,028025
(((0, 1), (0, 2)), ((0, 2), (1, 2)))			0,045476
(((0, 1), (1, 2)), ((0, 2), (1, 2)))			0,049383

Як випливає з таблиці 3.4, оптимальна структура складається з 2 шарів: 2 вузли на першому шарі і один вузол на другому шарі. На рис. 3.3 представлено схему синтезу структури. Показані нео-фазі нейрони, які

пройшли селекцію у напрямку прямого руху. Зеленим кольором позначені вузли, які були відібрані при зворотному ході синтезу. Оцінка результатів прогнозування на основі такої структури представлена на рисунку 3.4.

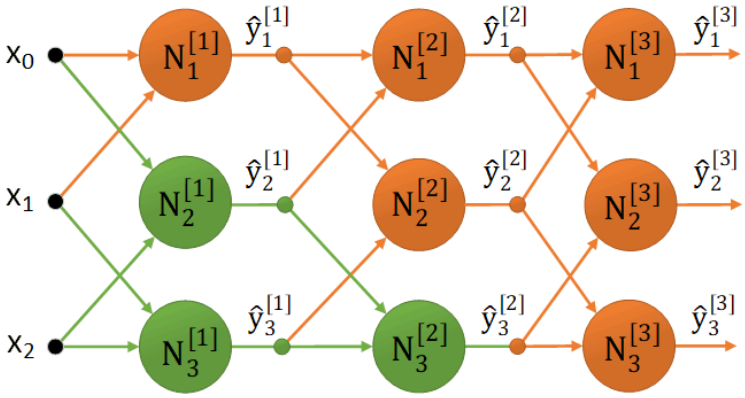

Рисунок 3.3 – Синтез оптимальної структури мережі з функцією належності Гауса (кількість входів: 3; правила: 4; співвідношення: 0,6)

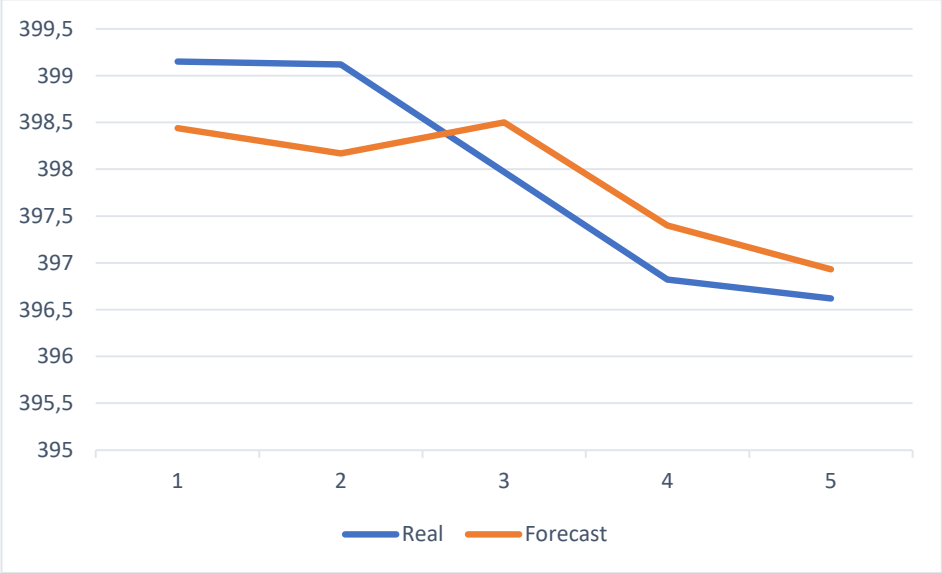

Рисунок 3.4 – Оцінка результатів прогнозування з використанням гібридної мережі глибокого навчання з функцією належності Гауса (кількість входів: 3; правила: 4; співвідношення: 0,6)

Наступний експеримент проводився з гібридною мережею глибокого навчання з трикутною функцією належності. Після проведення експерименту

були знайдені оптимальні параметри та структура гібридної мережі: 5 входів, 3 правила та коефіцієнт 0,8. Після експериментів порівнювалася точність гібридних мереж з різними функціями належності. Результати порівняння представлено на рисунку 3.5.

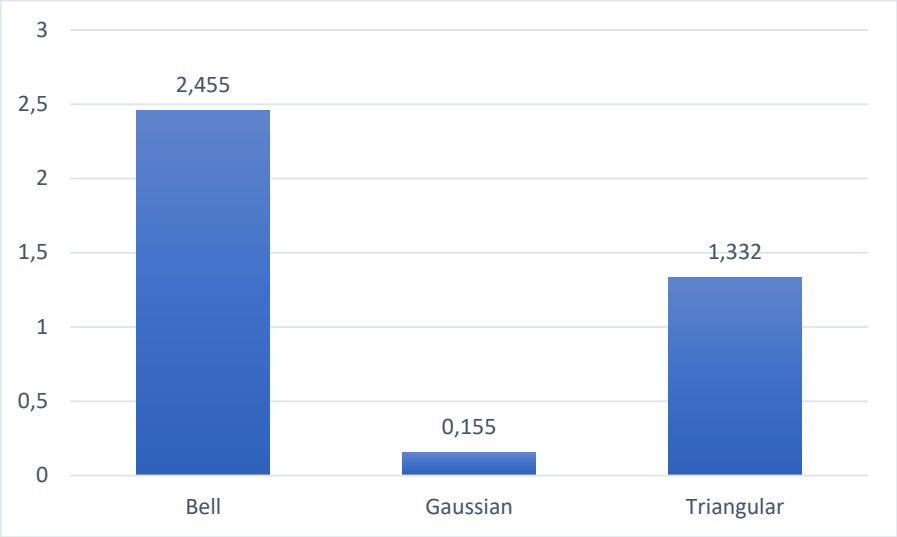

Рисунок 3.5 – Порівняння найкращих значень MAPE (середнє) при застосуванні різних функцій належності

У наступній серії експериментів були досліджені мережі LSTM. Метою експериментів було знайти оптимальні параметри. Змінювалися такі параметри: кількість входів (3-5), співвідношення навчання/тест (0,6; 07; 0,8). Знайдено оптимальні значення параметрів: кількість входів 3, коефіцієнт 0,7.

Після цього для навчання та прогнозування було застосовано LSTM з оптимальними параметрами. Результати представлені в таблиці 3.5.

Таблиця 3.5 – Найкращий прогноз LSTM (вхідні дані=3; співвідношення: 0,7)

Date	Real	Forecast	MSE	MAPE
17.08.2022	399,15	397,08	4,285	0,519
18.08.2022	399,12	397,58	2,372	0,386
19.08.2022	397,97	397,52	0,203	0,113

22.08.2022	396,82	396,71	0,012	0,028
23.08.2022	396,62	396,48	0,019	0,035
		Min:	0,012	0,028
		Avg:	1,378	0,216
		Max:	4,285	0,519

Процес навчання та валідації зображено на рисунку 3.6.

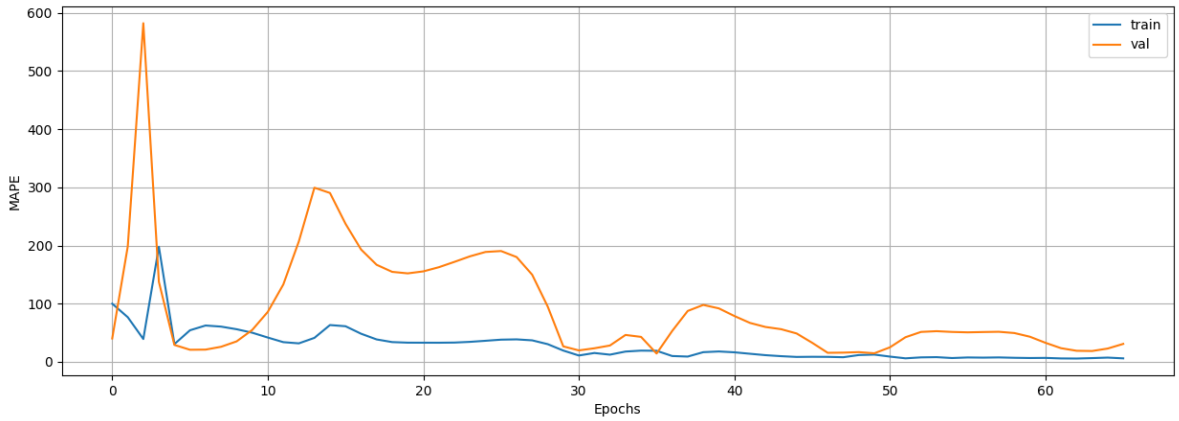

Рисунок 3.6 – Навчання мережі LSTM (MAPE)

У наступних експериментах було досліджено ефективність найкращих моделей гібридної мережі глибокого навчання та LSTM, а також порівняно з різними співвідношеннями навчальна/тестова вибірка. Результати для різних входів представлені в таблицях 3.6 та 3.7 відповідно.

Таблиця 3.6 – MSE для різного співвідношення навчання/тест та 3 входів

Ratio training/test	GMDH-neo-fuzzy	LSTM
60/40	0,424	4,074
70/30	1,242	1,378
80/20	1,248	2,023

Таблиця 3.7 – MSE для різного співвідношення навчання/тест та 4 входів

Ratio training/test	GMDH-neo-fuzzy	LSTM
60/40	2,894	4,715
70/30	1,783	2,107
80/20	3,491	5,412

Аналізуючи ці результати, можна зробити висновок, що мережа МГУА-нео-фазі має кращу точність прогнозування на інтервалі 5, ніж LSTM для різних співвідношень.

У наступних експериментах було досліджено ефективність прогнозування гібридних мереж глибокого навчання та LSTM на різних інтервалах прогнозування.

У таблиці 3.8 представлені результати прогнозування МГУА-нео-фазі мережі та LSTM на інтервалі 7 днів, а на рисунку 3.7 – точність прогнозування.

Таблиця 3.8 – Найкращий прогноз на період 7 – MAPE

Date	Real	GMDH-neo-fuzzy	LSTM
15.08.2022	400,32	0,412	1,806
16.08.2022	400,70	0,317	1,899
17.08.2022	399,15	0,704	1,518
18.08.2022	399,12	0,807	1,511
19.08.2022	397,97	0,859	1,224
22.08.2022	396,82	0,935	0,932
23.08.2022	396,62	0,630	0,877

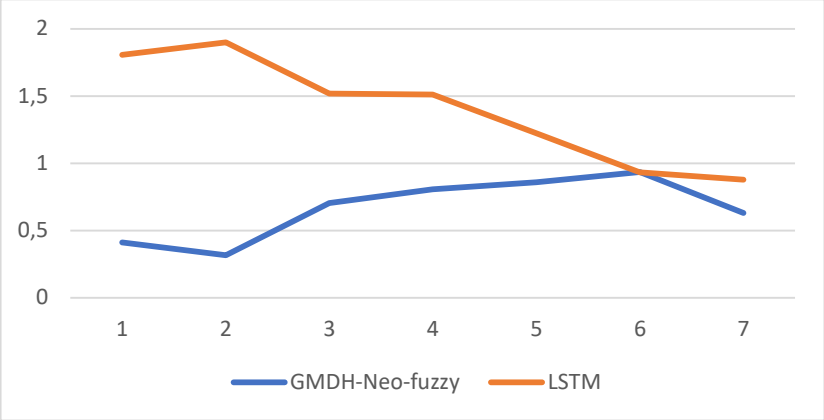

Рисунок 3.7 – Найкращий прогноз на період 7 – MAPE

Як випливає з представлених результатів, гібридна нео-фазі мережа має кращу точність, ніж LSTM на інтервалі 7 днів.

У наступних експериментах досліджувалася точність прогнозування обох мереж при середньостроковому прогнозуванні з інтервалом 20 днів. Точність за MAPE представлена на рисунку 3.8.

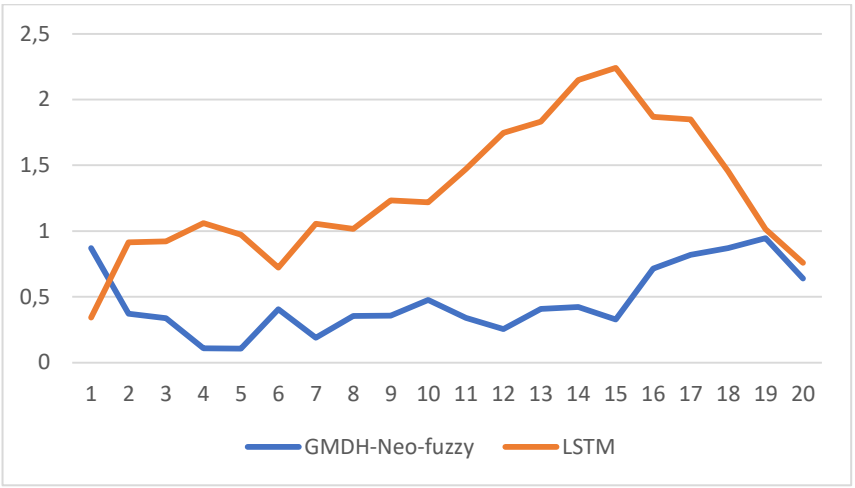

Рисунок 3.8 – Найкращий прогноз на період 20 – MAPE

3.4 Порівняння результатів експериментів гібридних мереж глибокого навчання на основі самоорганізації та LSTM

У підсумкових експериментах порівнювалася точність прогнозування гібридної мережі МГУА-нео-фазі та LSTM на різних інтервалах прогнозування (короткострокових і середньострокових). Відповідні результати за критеріями MSE та MAPE представлені в таблицях 3.9 та 3.10.

Таблиця 3.9 – Середнє значення MSE для різних інтервалів (періодів)

		Point 5	Point 7	Point 20
Period 3	GMDH-nf	0,319	0,678	1,251
	LSTM	0,702	0,408	2,374
Period 5	GMDH-nf	1,178	2,819	0,953
	LSTM	2,674	3,054	2,374
Period 7	GMDH-nf	0,238	2,714	0,863
	LSTM	5,888	6,325	2,983
Period 20	GMDH-nf	2,241	1,238	0,595
	LSTM	6,182	4,178	3,134

Таблиця 3.10 – Середнє значення MAPE для різних інтервалів (періодів)

		Point 5	Point 7	Point 20
Period 3	GMDH-nf	0,094	0,192	0,262
	LSTM	0,201	0,127	0,349
Period 5	GMDH-nf	0,111	0,337	0,224
	LSTM	0,316	0,358	0,352
Period 7	GMDH-nf	0,231	0,475	0,203
	LSTM	0,489	0,510	0,364
Period 20	GMDH-nf	0,392	0,368	0,172
	LSTM	0,453	0,408	0,416

3.5 Висновки до розділу 3

Аналіз результатів показує, що в цілому гібридна мережа глибокого навчання на основі самоорганізації має кращу точність, ніж LSTM на різних коротких і середніх інтервалах прогнозування. Це свідчить про доцільність її застосування для вирішення класу задач щодо короткострокового і середньострокового прогнозування у фінансовій сфері.

Практичні результати підтвердили теоретичні відомості, а саме: можливість отримати вищу точність прогнозу за рахунок використання не тільки трикутної функції належності, а й інших (наприклад, функції Гауса); зменшення обчислювальних витрат при навчанні мережі за рахунок адаптації лише вагових коефіцієнтів нео-фазі нейронів.

Проведені експерименти показали, що за оптимізація параметрів відіграє значну роль при синтезі структури глибокої мережі, наслідком чого є вища точність та менша кількість обчислень.

РОЗДІЛ 4. РОЗРОБКА СТАРТАП-ПРОЄКТУ

Ідея і розроблений мінімальний робочий продукт є основою для запуску стартап-проєкту. На початковому етапі кінцевому користувачу достатньо отримати сучасний та технологічний продукт. Але більшість стартап-проєктів зазнають невдач і виходять з ринку. Зазвичай їх місце займають конкуренти, які підійшли більш організовано до представлення свого продукту цільовій аудиторії. Для успішної презентації стартап-проєкту необхідно описати його ідею, провести технологічний аудит, проаналізувати ринкові можливості, розробити ринкову стратегію та маркетингову програму.

4.1 Опис ідеї стартап-проєкту

Основною ідеєю стартап-проєкту є вирішення задачі прогнозування фондових індексів на біржах. Опишемо детально ідею у вигляді таблиці 4.1.

Таблиця 4.1 – Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Система, яка комплексно використовується для прогнозування фондових індексів на біржах, внутрішніх та зовнішніх фінансових часових рядів.	Державні органи управління	Прогноз допомагає у плануванні бюджету та витрат
	Фінансові установи	Прогноз може слугувати визначним фактором для розробки задач
	Приватні підприємства	Прогноз допомагає при плануванні стратегій розвитку

Проаналізуємо конкурентів стартап-проєкту. Результати аналізу наведемо у вигляді таблиці 4.2.

Таблиця 4.2 – Слабкі (W), нейтральні (N) та сильні (S) характеристики ідеї стартап-проєкту

Техніко-економічні характеристики	(потенційні) товари/концепції конкурентів			W	N	S
	Стартап-проєкт	Конкурент 1	Конкурент 2			
Швидкість	Висока	Низька	Середня			+
Точність	Висока	Середня	Середня			+
Ціна	Низька	Середня	Висока			+
Інтеграція	Середня	Висока	Середня		+	
Гнучкість	Середня	Середня	Середня		+	

Визначені слабкі, нейтральні та сильні характеристики ідеї можна розглядати, як основу для формування конкурентоспроможності.

4.2 Технологічний аудит ідеї стартап-проєкту

Проведемо аудит технологій, за допомогою яких можна реалізувати ідею стартап-проєкту (таблиця 4.3).

Таблиця 4.3 – Технологічна здійсненність ідеї стартап-проєкту

№ п/п	Ідея стартап-проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Система для прогнозування індексів на фондових ринках	Побудова нейронної мережі та її навчання	Наявна	Загальнодоступна та безкоштовна

		(Python, sklearn)		
2		Побудова прогнозу (Python, pandas)	Наявна	Загальнодоступна та безкоштовна
3		Візуалізація даних (Python, pyplot)	Наявна	Загальнодоступна та безкоштовна
Обрана технологія реалізації ідеї стартап-проєкту: для успішної реалізації стартап-проєкту обрано мову програмування Python та бібліотеки sklearn, pandas, pyplot.				

4.3 Аналіз ринкових можливостей запуску стартап-проєкту

Дослідимо потенційний ринок. Для цього проведемо аналіз у вигляді таблиці 4.4.

Таблиця 4.4 – Попередня характеристика потенційного ринку

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	2
2	Загальний обсяг продаж, грн/ум.од	5000 ум.од
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по	31

	ринку), %	

У таблиці 4.5 наведемо характеристику потенційних клієнтів.

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проєкту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Короткострокове інвестування	Громадяни країн	Цікавлять короткострокові прогнози	Простота у використанні
2	Середньострокове інвестування	Гравці на фондовому ринку	Цікавлять середньострокові прогнози	Детальний опис та візуалізація даних

Необхідно дослідити позитивні та негативні фактори ринку. Проаналізуємо загрози у вигляді таблиці 4.6.

Таблиця 4.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренція	Вихід на ринок конкурентів, які будуть продавати аналогічний продукт за меншу ціну	Продати свою компанію або додати новий функціонал
2	Ресурси	Неправильна архітектура може призвести до значних витрат ресурсів	Передбачити масштабування та незалежність від платформи

За допомогою таблиці 4.7 розглянемо фактори, які можуть відкрити можливості для продукту.

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Низький поріг входу	Користувач повинен мати змогу користуватися інтуїтивно зрозумілим продуктом, не вивчаючи довідкових матеріалів	Створення зручного інтерфейсу для користувача, який містить підказки
2	Хмарні технології	Обчислення та зберігання інформації можуть виконуватися на окремому віддаленому сервері	Прийняти архітектурні рішення щодо хмарних обчислень

Визначимо загальні риси конкуренції на ринку. Для цього проведемо аналіз у вигляді таблиці 4.8.

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції: досконала конкуренція	На ринку представлено багато продуктів та експертів	Розробити широкий набір функцій, які можна обирати і за рахунок цього регулювати ціну
2. За рівнем конкурентної боротьби: міжнародний	Використання продукту не залежить від локалізації користувача	Додати переклади на декілька поширених мов, за рахунок чого розширити цільову аудиторію

3. За галузевою ознакою: внутрішньогалузева	Продукт переважно застосовується інвесторами	Персоналізація продукту
4. Конкуренція за видами товарів: товарно-видова	Конкуренцію складають сервіси з прогнозування	Додати різні види прогнозів, візуалізацію та можливість збереження у різних форматах
5. За характером конкурентних переваг: неціновa	Конкуренти мають різну точність прогнозування, швидкість та можливості роботи з даними	Зменшити похибку прогнозу, прискорити процес прогнозування, додати інтеграцію з найбільш відомими хмарними сервісами
6. За інтенсивністю: не марочна	На ринку не представлені бренди	Розробити стратегію для створення та розвитку власного бренду

Після аналізу конкуренції більше детально проаналізуємо умови конкуренції в галузі. М. Портер виділив п’ять основних факторів, які впливають на привабливість вибору ринку з огляду на конкуренцію. На основі п’яти сил за М. Портером розробимо перелік факторів конкурентоспроможності (таблиця 4.9).

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти у галузі	Потенційні конкуренти	Постачальники	Клієнти	Товарозамінники
Висновки	Мають контроль над основною частиною ринку	Широкі можливості для появи нових конкурентів	Активно не впливають на ринок	Потребують високої точності прогнозу, якості продукту та інтеграції зі сторонніми сервісами	Аналогічний функціонал за меншу ціну

На основі таблиці 4.8, а також враховуючи характеристики ідеї проєкту, вимоги споживачів та фактори маркетингового середовища, обґрунтуємо перелік факторів конкурентоспроможності у вигляді таблиці 4.10.

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проєктів значущими)
1	Точність прогнозу	Система прогнозує показники з вищою точністю, враховує історію прогнозів та порівняння з аналогічними продуктами
2	Низький поріг входу	Сервіс не вимагає вивчення інструкції користувача, містить покрокові підказки, які можна вимкнути
3	Масштабованість	Систему можна легко налаштувати, враховуючи побажання користувача щодо використання хмарних технологій та інтеграції зі сторонніми сервісами

Порівняємо сильні та слабкі сторони стартап-проєкту (таблиця 4.11).

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін RFS

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з нашим підприємством						
			-3	-2	-1	0	+1	+2	+3
1	Точність прогнозу	20		+					
2	Низький поріг входу	15			+				
3	Масштабованість	15		+					

На основі виділених ринкових загроз та можливостей, сильних і слабких сторін складемо таблицю 4.12 для SWOT-аналізу.

Таблиця 4.12 – SWOT-аналіз стартап-проєкту

Сильні сторони: - точність прогнозів; - низький поріг входу; - масштабованість.	Слабкі сторони: - відсутність бренду; - відсутність клієнтської бази; - відсутність партнерів.
Можливості: - багатомовність продукту; - широкий набір функцій; - персоналізація.	Загрози: - конкуренти з нижчими цінами; - ріст ресурсів.

Розглянемо альтернативи ринкового впровадження (таблиця 4.13).

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проєкту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Вихід на ринок з базовим функціоналом	75%	5 місяців
2	Багатомовність та розширений набір функцій	35%	7 місяців
3	Масштабованість та персоналізація	25%	5 місяців

4.4 Розробка ринкової стратегії стартап-проєкту

Проведемо опис та вибір цільових груп потенційних клієнтів (таблиця 4.14).

Таблиця 4.14 – Вибір цільових груп споживачів

№ п/п	Опис цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу в сегмент
1	Гравці фондового ринку	Висока	30%	Висока	Середня
2	Громадяни країн	Низька	15%	Низька	Висока
3	Підприємства, які вийшли на IPO	Середня	20%	Середня	Середня
Обрано цільові групи: 1, 3.					

Для роботи в обраних сегментах ринку сформуємо базову стратегію розвитку (таблиця 4.15).

Таблиця 4.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Створення зрозумілого інтерфейсу продукту, широкого набору функцій та можливості персоналізації продукту	Ринкове позиціонування	Гнучкість щодо ціноутворення за рахунок певного набору функцій, можливість інтеграції з поширеними хмарними сервісами	Удосконалення існуючого та розробка нового функціоналу, пошук партнерів, які займають передові позиції у рейтингах на ринку хмарних

			обчислень

В таблиці 4.16 визначимо базову стратегію конкурентної поведінки.

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

Чи є проєкт «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента (якщо так, то які)?	Стратегія конкурентної поведінки
Ні	Так	Так, базовий функціонал буде схожий	Гнучка цінова політика, персоналізація для користувача

Визначимо стратегію позиціонування (таблиця 4.17).

Таблиця 4.17 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту (три ключових)
Висока точність прогнозу, швидкість обробки даних, масштабованість	Додавання функцій, враховуючи потреби цільової аудиторії	Персоналізація, гнучка політика ціноутворення, хмарні сервіси	Прогнозування, масштабованість, персоналізація, гнучкість, простота

4.5 Розробка маркетингової програми стартап-проєкту

Представимо ключові переваги концепції потенційного товару (таблиця 4.18).

Таблиця 4.18 – Ключові переваги концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Точність прогнозу	Вирішення основної задачі користувача	Достовірність отриманої інформації
2	Простота у використанні	Зручність отримання інформації	Мінімізація часу для отримання інформації
3	Гнучка політика ціноутворення	Користувач платить тільки за той функціонал, який він використовує	Економія коштів

Розробимо трирівневу маркетингову модель товару. Для цього уточнимо ідею продукту, фізичні складові та особливості процесу його надання. Опис трьох рівнів моделі товару представимо у вигляді таблиці 4.19.

Таблиця 4.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
І. Товар за задумом	Система збирає статистичні дані, аналізує та будує прогнози. Від користувача очікується лише налаштування системи.

II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх/Тл/Е/Ор
	Ефективність Точність роботи Безпека	-	-
	Якість: відповідно до стандарту ISO 4444 буде проведено тестування		
	Маркування відсутнє		
III. Товар із підкріпленням	Пробна безкоштовна підписка та безкоштовна ліцензія для навчальних закладів		
	Технічна підтримка користувачів		
Захист продукту: закритий код, комерційна ліцензія.			

При встановленні ціни на потенційний товар необхідно керуватись визначеними ціновими межами. Наведемо їх у таблиці 4.20.

Таблиця 4.20 – Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
18000	21000	470000	17000

На наступному кроці визначимо оптимальну систему збуту, в межах якої приймається рішення. Для цього необхідно розуміти специфіку закупівельної поведінки клієнтів, функції збуту постачальника, глибину каналу збуту, оптимальні характеристики системи. Сформуємо систему збуту у вигляді таблиці 4.21.

Таблиця 4.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Платна підписка (для	Продаж	1 (напряму)	Власна, щоб

приватних осіб, для команди, персональна)			мінімізувати витрати

Розробимо концепцію маркетингових комунікацій і наведемо її у вигляді таблиці 4.22.

Таблиця 4.22 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Пошук доступних спеціалізованих систем	Професійні канали комунікації, реклама, форуми	Точність, простота, гнучкість, персоналізація	Виділитися на фоні конкурентів за рахунок своїх переваг	Реклама у соціальних мережах на цільову аудиторію, інформація від лідерів думок

4.6 Висновки до розділу 4

На основі проведеного аналізу можна сказати, що вихід стартап-проєкту на комерційний ринок є доцільним. Це доводить інформаційна карта проєкту, його ідея та технологічна здійсненність. Але варто враховувати продукти конкурентів, адже ідея не нова, тому для цього було проведено необхідні розрахунки. В цілому ідея просування розглянутого продукту на ринок є перспективною. Для більшої впевненості в досягненні успіху варто звернути увагу на новий функціонал, щоб утримати клієнтів, а також стимулювати маркетингову активність.

ВИСНОВКИ

У роботі проводилася оптимізація параметрів та структури гібридних мереж глибокого навчання на основі самоорганізації на прикладі вирішення задачі короткострокового та середньострокового прогнозування у фінансовій сфері.

В процесі її вирішення було визначено актуальність проблеми, проведено аналіз предметної області, зроблено огляд публікацій результатів аналогічних досліджень. Також розглянуто особливості аналізу та прогнозування часових рядів, визначено вимоги до експериментальних даних та розглянуто критерії якості, на основі яких оцінювалася ефективність моделей.

Проведено математичне моделювання гібридних мереж глибокого навчання на основі самоорганізації. Розглянуто ідеї та основні принципи МГУА, який лежить алгоритму синтезу структури досліджуваних моделей. Визначено параметри, оптимізація яких впливає на точність прогнозування та обчислювальну витрати при навчанні та прогнозуванні. Детально розглянуто сам процес синтезу структури МГУА-нео-фазі. Визначено важливу роль нео-фазі нейрона у якості вузла системи та розглянуто алгоритм його навчання.

Експериментальні дослідження проводились над задачею прогнозування індексу EMBRI на фондовій біржі NASDAQ на період з січня по серпень 2022 року. Для проведення експериментів було розроблено програмне забезпечення.

Під час експериментів проведено оптимізацію параметрів LSTM та гібридних мереж. Методом МГУА побудовано оптимальну структуру гібридної мережі глибокого навчання.

Експериментальні дослідження оптимізованих LSTM та гібридних мереж проводились на різних інтервалах прогнозування та порівнювалася їх точність.

У результаті було встановлено, що застосування гібридних мереж глибокого навчання на основі самоорганізації дозволяє отримати кращу точність, ніж LSTM у задачах короткострокового та середньострокового прогнозування на фондових біржах. Отже, оптимізація параметрів та структури відіграє важливу роль для таких моделей.

Виконано аналіз для запуску стартап-проєкту на комерційному ринку та на його основі побудовано систему рекомендацій для успішного просування розробленого продукту. Розроблена стратегія дозволяє оцінити строки та результат виходу продукту на рівень конкурентів.

Результати проведених досліджень представлено на міжнародних конференціях [26, 27]. Рекомендується використовувати їх при вирішенні задач короткострокового та середньострокового прогнозування у фінансовій сфері та подальшого дослідження гібридних мереж глибокого навчання на основі самоорганізації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Michael Z. Zgurovsky, Victor Sineglazov, Elena Chumachenko: Artificial Intelligence Systems Based on Hybrid Neural Networks -Theory and Applications. Studies in Computational Intelligence 904, Springer 2021, ISBN 978-3-030-48452-1, pp. 1-510
2. Кузьменко О.В., Зайченко Ю.П. Дослідження гібридних мереж глибокого навчання в задачах прогнозування у фінансовій сфері // Системні науки та інформатика: збірник доповідей I науково-технічної конференції «Системні науки та інформатика», 22-29 листопада 22 року, Київ. - К., НН ІПСА КПІ ім. Ігоря Сікорського, 2022. - 490 с.
3. Yuriy Zaychenko, Helen Zaychenko. Investigation of Fuzzy Inductive Modeling Method in Forecasting Problems. doi 10.5772/intechopen.86348.
4. Yuriy Zaychenko, Helen Zaychenko. Fuzzy GMDH and its application to forecasting financial processes. ISSN 1681–6048 System Research & Information Technologies, 2019. P. 91-109.
5. Бодянський Є.В., Тищенко О.К., Бойко О.О. Еволюційна каскадна система на основі нейро-фаззі вузлів // p-ISSN 1607-3274. Радіоелектроніка, інформатика, управління. 2016. №2. С. 40-45.
6. Бодянський Є.В., Антоненко Т.Є. Глибока нео-фаззі нейронна мережа та її навчання // Бионика интеллекта. 2019. №1(92). С. 3-8.
7. Yevgeniy Bodyanskiy, Nonna Kulishova and Olha Chala. Neo-Fuzzy System with Special Type Membership Functions Adaptation and Fast Tuning of Synaptic Weights in Emotion Recognition Task // II International Scientific Symposium «Intelligent Solutions» IntSol-2021, September 28–30, 2021, Kyiv-Uzhhorod, Ukraine. P. 158-166.
8. Anita Yadav, C K Jha, Aditi Sharan. Optimizing LSTM for time series prediction in Indian stock market // International Conference on Computational Intelligence and Data Science (ICCIDS 2019) - P. 2091-2100.

9. Qi Tang, Tongmei Fan, Ruchen Shi, Jingyan Huang, Yidan Ma: Prediction of financial time series using LSTM and data denoising methods. CoRR abs/2103.03505 (2021)
10. Robert H. Shumway, David S. Stoffer. Time Series Analysis and Its Applications. Fourth Edition. Springer International Publishing AG, Switzerland, 2017 - 567 p.
11. Mikhail Z. Zgurovsky, Yuriy P. Zaychenko. Big Data: Conceptual Analysis and Applications. Studies in Big Data 58. Springer Nature Switzerland AG 2020. - 298 p.
12. Bianchi, Filippo Maria & Maiorino, Enrico & Kampffmeyer, Michael & Rizzi, Antonello & Jenssen, Robert. (2017). Recurrent Neural Network Architectures. 10.1007/978-3-319-70338-1_3.
13. Бідюк, П.І. Аналіз часових рядів [Електронний ресурс] : навчальний посібник / П.І. Бідюк, В.Д. Романенко, О.Л. Тимощук ; НТУУ «КПІ». – Електронні текстові дані (1 файл: 8,42 Мбайт). – Київ : НТУУ «КПІ», 2010. – Назва з екрана.
14. О.Г. Руденко, О.О. Безсонов, О.Г. Лебедєв, О.С. Романюк Критерії вибору персептронної моделі для прогнозування: аналіз і практичні рекомендації щодо їх використання // Біоніка інтелекту 2 (91) - с. 31-40
15. Ивахненко А.Г. Метод группового урахування аргументів - конкурент методу стохастичної апроксимації // Автоматика. - 1968. - № 3. - С. 58-72.
16. Yevgeniy Bodyanskiy, Olena Boiko, Yuriy Zaychenko, Galib Hamidov, Anna Zelikman The Hybrid GMDH-Neo-fuzzy Neural Network in Forecasting Problems in Financial Sphere // 2020 IEEE 2nd International Conference on System Analysis & Intelligent Computing (SAIC). - 2020 - P. 1-6.
17. Зайченко Ю.П. Основи проектування інтелектуальних систем. Навчальний посібник. - К.: Видавничий Дім «Слово», 2004. - 352 с.

18. Mikhail Z. Zgurovsky, Yuriy P. Zaychenko. The Fundamentals of Computational Intelligence: System Approach. Studies in Computational Intelligence 652. Springer International Publ. AG, Switzerland. - 389 p.
19. I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT PRESS, 2016. URL: http://www.deeplearningbook.org.
20. Emerging Markets Corporate Bond Total Return Index [Електронний ресурс] / Nasdaq Data Link. URL: https://data.nasdaq.com/data/ML/EMCTRIemerging-markets-corporate-bond-total-return-index.
21. Welcome to Python.org. – URL: https://www.python.org. – Назва з екрану.
22. Himanshu Singh, Yunis Ahmad Lone. Deep Neuro-Fuzzy Systems with Python: With Case Studies and Applications from the Industry. – Apress, 2019. – 260 p.
23. PyCharm: IDE для профессиональной разработки на Python от JetBrains. – URL: https://www.jetbrains.com/ru-ru/pycharm. – Назва з екрану.
24. Scikit-learn: machine learning in Python. – URL: https://scikit-learn.org/stable. – Назва з екрану.
25. Skfuzzy 0.2 docs. – URL: https://pythonhosted.org/scikit-fuzzy. – Назва з екрану.
26. Yuriy Zaychenko and Oleksii Kuzmenko: IEEE 17th International Conference on Computer Science and Information Technologies (CSIT): Investigation of Hybrid Deep Learning Networks and LSTM in the Short-Term Forecasting in Financial Sphere - Lviv, Ukraine, 10-12 November 2022.
27. Yuriy Zaychenko, Helen Zaichenko and Oleksii Kuzmenko: The IX International Conference "Information Technology and Implementation" (IT&I-2022): Investigation of Artificial Intelligence Methods in the Short-Term and Middle-Term Forecasting in Financial Sphere - Kyiv, Ukraine, November 30 - December 2, 2022.

ДОДАТОК А. ЛІСТИНГ КОДУ ПРОГРАМИ

```
# requirements.txt


contourpy==1.0.6
cycler==0.11.0
fonttools==4.38.0
joblib==1.2.0
kiwisolver==1.4.4
matplotlib==3.6.2
networkx==2.8.8
numpy==1.23.5
packaging==21.3
pandas==1.5.2
Pillow==9.3.0
pyparsing==3.0.9
python-dateutil==2.8.2
pytz==2022.6
scikit-fuzzy==0.4.2
scikit-learn==1.1.3
scipy==1.9.3
six==1.16.0
threadpoolctl==3.1.0


# gmdh_neo_fuzzy.py


import abc
import random
import itertools
import numpy as np
import skfuzzy as skf
import pandas as pd
import math
import matplotlib.pyplot as plt

from copy import deepcopy
from typing import Any, Tuple
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import (
    mean_absolute_percentage_error,
    mean_squared_error
)


class DataProcessor:
    def __init__(self, src: str, validation_split: float = 0.4,
                 interval: int = 1, window_size: int = 10,
                 test_size: int = 20) -> None:
        self.src = src
        self.validation_split = validation_split
        self.interval = interval
        self.window_size = window_size
        self.test_size = test_size
        self.scaler = MinMaxScaler(feature_range=(0.1, 1))
        self.df = self._set_df()
```

```
        self.ts = self._set_ts()
        self._split_ts()
        self.n_lags = math.ceil(len(self.df) / 4)

    def _set_df(self) -> np.ndarray:
        return pd.read_csv(self.src).sort_index(ascending=False)

    def _set_ts(self) -> np.ndarray:
        ts = self.df.copy()
        ts.drop('Date', inplace=True, axis=1)
        return self.scaler.fit_transform(ts)

    def _split_ts(self) -> None:
        t = len(self.ts) - (self.window_size +
                            self.interval + self.test_size - 1)
        v = int(len(self.ts[:t]) * (1 - self.validation_split))
        train = self.ts[:v]
        val = self.ts[v:t]
        test = self.ts[t:]
        self.x_train, self.y_train = self._prepare_sample(train)
        self.x_val, self.y_val = self._prepare_sample(val)
        self.x_test, self.y_test = self._prepare_sample(test)

    def _prepare_sample(self, ts: np.ndarray) \
            -> Tuple[np.ndarray, np.ndarray]:
        x = []
        y = []
        i = 0
        j = self.window_size
        k = j + self.interval - 1
        while k < len(ts):
            x.append(ts[i:j])
            y.append(ts[k])
            i += 1
            j += 1
            k += 1
        return np.array(x), np.array(y)

    def get_test_df(self):
        return self.df[:self.test_size]

    def make_correlation(self, src: str,
                         n_lags: int = None,
                         n_lag_plot: int = None):
        if n_lags is None:
            n_lags = self.n_lags
        source = self.df.copy()

        source_df = pd.Series(source['Close'].to_numpy())

        lags = []
        coefficients = []

        for i in range(n_lags):
            lag = i + 1
            lags.append(lag)
            coefficient = source_df.autocorr(lag=lag)
            coefficients.append(coefficient)

        res_df = pd.DataFrame({
            'Lag': lags,
            'Coefficient': coefficients
```

```
        })

        res_df.to_csv(src, index=False)

        if n_lag_plot is not None:
            pd.plotting.lag_plot(source_df, lag=n_lag_plot)
            plt.show()


class Metrics:
    def __init__(self, y_true: Any, y_pred: Any):
        self.y_true = y_true
        self.y_pred = y_pred

    def mse(self) -> float:
        return mean_squared_error(y_true=self.y_true,
                                  y_pred=self.y_pred)

    def mape(self) -> float:
        return mean_absolute_percentage_error(y_true=self.y_true,
                                              y_pred=self.y_pred) * 100


class MembershipFunction:
    @abc.abstractmethod
    def calculate(self, x: Any) -> Any:
        pass


class Bell(MembershipFunction):
    def __init__(self, a: float, b: float, c: float) -> None:
        self.a = a
        self.b = b
        self.c = c

    def calculate(self, x: Any) -> Any:
        return skf.gbellmf(x, self.a, self.b, self.c)


class Gaussian(MembershipFunction):
    def __init__(self, mean: float, sigma: float) -> None:
        self.mean = mean
        self.sigma = sigma

    def calculate(self, x: Any) -> Any:
        return skf.gaussmf(x, self.mean, self.sigma)


class Sigmoid(MembershipFunction):
    def __init__(self, b: float, c: float) -> None:
        self.b = b
        self.c = c

    def calculate(self, x: Any) -> Any:
        return skf.sigmf(x, self.b, self.c)


class Triangular(MembershipFunction):
    def __init__(self, params: Any) -> None:
        self.params = params

    def calculate(self, x: Any) -> Any:
```

```
        return skf.trimf(x, self.params)


class Rule:
    def __init__(self):
        self.input = None
        self.output = None
        self.weight = None
        self.membership_function = None
        self.membership_function_name = 'gaussian'

    def init_weight(self) -> None:
        self.weight = np.random.random()

    def init_membership_function(self) -> None:
        if self.membership_function_name == 'bell':
            self.membership_function = Bell(a=0.2, b=0.3, c=0.5)
        elif self.membership_function_name == 'gaussian':
            self.membership_function = Gaussian(mean=0.1, sigma=0.9)
        elif self.membership_function_name == 'sigmoid':
            self.membership_function = Sigmoid(b=0.1, c=0.9)
        elif self.membership_function_name == 'triangular':
            self.membership_function = Triangular(params=[0.2, 0.3, 0.5])

    def forward(self):
        self.output = self.weight * \
                      self.membership_function.calculate(self.input)


class Synapse:
    def __init__(self):
        self.input = None
        self.output = None
        self.rules = []
        self.number_of_rules = 3

    def init_rules(self) -> None:
        for _ in range(self.number_of_rules):
            rule = Rule()
            rule.init_weight()
            rule.init_membership_function()
            self.rules.append(rule)

    def forward(self):
        output = []
        for i in range(len(self.rules)):
            self.rules[i].input = self.input
            self.rules[i].forward()
            output.append(self.rules[i].output)
        output = np.array(output)
        self.output = sum(output)


class History:
    def __init__(self):
        self.epoch_number = None
        self.start_time = None
        self.end_time = None
        self.train_metrics = None
        self.val_metrics = None
        self.synapses = None
```

```
class NeoFuzzyNeuron:
    def __init__(self):
        self.input = None
        self.output = None
        self.signal_numbers = None
        self.x_train = []
        self.y_train = []
        self.x_val = []
        self.y_val = []
        self.synapses = []
        self.metrics = None
        self.is_active = True
        self.target = None
        self.stories = []
        self.best_history = None
        self.epochs = 500
        self.learning_rate = 0.1
        self.patience = None

    def init_synapses(self) -> None:
        for _ in self.signal_numbers:
            synapse = Synapse()
            synapse.init_rules()
            self.synapses.append(synapse)

    def forward(self) -> None:
        output = []
        for i in range(len(self.synapses)):
            self.synapses[i].input = self.input[i]
            self.synapses[i].forward()
            output.append(self.synapses[i].output)
        output = np.array(output)
        output = sum(output)
        self.output = output

    def backward(self) -> None:
        error = self.target - self.output
        for i in range(len(self.synapses)):
            for j in range(len(self.synapses[i].rules)):
                weight = self.synapses[i].rules[j].weight
                output = self.synapses[i].rules[j].output
                rate = self.learning_rate
                new_weight = weight + rate * error * output
                self.synapses[i].rules[j].weight = new_weight

    def train(self):
        patience = 0
        for epoch in range(self.epochs):
            history = History()
            history.epoch_number = epoch + 1
            predicted = []
            target = []
            x_train_range = range(len(self.x_train))
            train_data_indices = [i for i in x_train_range]
            random.shuffle(train_data_indices)
            for index in train_data_indices:
                self.input = self.x_train[index]
                self.target = self.y_train[index]
                self.forward()
                predicted.append(self.output)
                target.append(self.target)
```

```
                self.backward()
            history.train_metrics = Metrics(y_true=np.array(target),
                                            y_pred=np.array(predicted))
            self.evaluate()
            history.val_metrics = self.metrics
            history.synapses = deepcopy(self.synapses)
            self.stories.append(history)
            self.generate_epoch_report(history)

            if self.patience is not None:
                if self.best_history is None:
                    self.best_history = history
                    patience += 1
                else:
                    curr_metrics = history.val_metrics
                    curr_criteria = curr_metrics.mse()
                    best_history = self.best_history
                    best_metrics = best_history.val_metrics
                    best_criteria = best_metrics.mse()
                    if curr_criteria < best_criteria:
                        self.best_history = history
                        patience = 0
                    else:
                        patience += 1
            else:
                self.best_history = history

            if patience == self.patience:
                self.synapses = deepcopy(self.best_history.synapses)
                break

        self.generate_training_report()

    def evaluate(self) -> None:
        predicted = []
        target = []
        x_val_range = range(len(self.x_val))
        val_data_indices = [i for i in x_val_range]
        random.shuffle(val_data_indices)
        for index in val_data_indices:
            self.input = self.x_val[index]
            self.forward()
            predicted.append(self.output)
            target.append(self.y_val[index])
        self.metrics = Metrics(y_true=np.array(target),
                               y_pred=np.array(predicted))

    def generate_epoch_report(self, history: History) -> None:
        epoch = history.epoch_number
        message = f"Epoch {epoch}/{self.epochs}:\n"
        train_metrics = history.train_metrics
        train_mse = train_metrics.mse()
        train_mape = train_metrics.mape()
        message += f"Train: MSE = {train_mse}; "
        message += f"MAPE = {train_mape}\n"
        val_metrics = history.val_metrics
        val_mse = val_metrics.mse()
        val_mape = val_metrics.mape()
        message += f"Validation: MSE = {val_mse}; "
        message += f"MAPE = {val_mape}\n"
        print(message)
```

```
    def generate_training_report(self) -> None:
        message = f"Node training was stopped"
        message += f" and best weights were restored:\n"
        history = self.best_history
        metrics = history.val_metrics
        message += f"Best epoch: {history.epoch_number};\n"
        message += f"Best MSE: {metrics.mse()};\n"
        message += f"Best MAPE: {metrics.mape()};\n"
        print(message)


class Layer:
    def __init__(self):
        self.input = None
        self.output = None
        self.number_of_inputs = 3
        self.x_train = []
        self.y_train = []
        self.x_val = []
        self.y_val = []
        self.nodes = []
        self.node_number_of_inputs = 2

    def init_nodes(self) -> None:
        sn_for_nodes = self.generate_signal_numbers_for_nodes()
        for signal_numbers in sn_for_nodes:
            node = NeoFuzzyNeuron()
            node.signal_numbers = signal_numbers
            node.init_synapses()
            self.nodes.append(node)

    def generate_signal_numbers_for_nodes(self) -> np.ndarray:
        return np.array(list(itertools.combinations(
            [i for i in range(self.number_of_inputs)],
            self.node_number_of_inputs))
        )

    def set_input_data_for_nodes(self):
        for i in range(len(self.nodes)):
            node_input_data = []
            for s in self.nodes[i].signal_numbers:
                node_input_data.append(self.input[s])
            self.nodes[i].input = np.array(node_input_data)

    def set_train_data_for_nodes(self):
        for i in range(len(self.nodes)):
            node_x_train = []
            for x_train_item in self.x_train:
                node_x_train_item = []
                for s in self.nodes[i].signal_numbers:
                    node_x_train_item.append(x_train_item[s])
                node_x_train.append(np.array(node_x_train_item))
            self.nodes[i].x_train = np.array(node_x_train)
            self.nodes[i].y_train = self.y_train

    def set_validation_data_for_nodes(self):
        for i in range(len(self.nodes)):
            node_x_val = []
            for x_val_item in self.x_val:
                node_x_val_item = []
                for s in self.nodes[i].signal_numbers:
                    node_x_val_item.append(x_val_item[s])
```

```
                node_x_val.append(np.array(node_x_val_item))
            self.nodes[i].x_val = np.array(node_x_val)
            self.nodes[i].y_val = self.y_val

    def forward(self) -> None:
        output = []
        self.set_input_data_for_nodes()
        for i in range(len(self.nodes)):
            if not self.nodes[i].is_active:
                continue
            self.nodes[i].forward()
            output.append(self.nodes[i].output)
        self.output = np.array(output)

    def forward_multiple(self, list_of_input_data: np.ndarray) -> np.ndarray:
        output = []
        for input_data in list_of_input_data:
            self.input = input_data
            self.forward()
            output.append(self.output)
        return np.array(output)

    def train(self):
        self.set_train_data_for_nodes()
        self.set_validation_data_for_nodes()
        for i in range(len(self.nodes)):
            print(f"Node #{i + 1} training:")
            self.nodes[i].train()

    def evaluate(self) -> None:
        self.set_validation_data_for_nodes()
        for i in range(len(self.nodes)):
            self.nodes[i].evaluate()


class SelectionBlock:
    def __init__(self):
        self.freedom_choice = 4
        self.input_layer = None
        self.best_metrics = None
        self.output_layer = None

    def filter(self):
        output_layer = deepcopy(self.input_layer)
        mse = []
        for node in output_layer.nodes:
            mse.append(node.metrics.mse())
        mse = np.array(mse)
        mse = np.sort(mse)[0:self.freedom_choice]

        for node in output_layer.nodes:
            if node.metrics.mse() not in mse:
                node.is_active = False
            else:
                node.is_active = True
                if node.metrics.mse() == mse[0]:
                    self.best_metrics = node.metrics
        self.output_layer = output_layer


class GMDHNeoFuzzy:
    def __init__(self):
```

```
        self.input = None
        self.output = None
        self.x_train = []
        self.y_train = []
        self.x_val = []
        self.y_val = []
        self.x_test = []
        self.y_test = []
        self.layers = []
        self.selection_blocks = []
        self.structure = []
        self.building_report = None
        self.random_seed = 11

    def build(self) -> None:
        np.random.seed(self.random_seed)
        random.seed(self.random_seed)
        self.evolve()
        self.build_structure()
        self.generate_building_summary()

    def evolve(self) -> None:
        layer = Layer()
        layer.number_of_inputs = self.get_number_of_inputs()
        layer.x_val = np.concatenate((self.x_train, self.x_val))
        layer.y_val = np.concatenate((self.y_train, self.y_val))
        layer.init_nodes()
        layer.evaluate()
        selection_block = SelectionBlock()
        selection_block.input_layer = layer
        selection_block.filter()
        self.layers.append(selection_block.output_layer)
        self.selection_blocks.append(selection_block)

        while 1:
            layer = Layer()
            layer.x_train = self.layers[-1].forward_multiple(self.x_train)
            layer.y_train = self.y_train
            layer.x_val = self.layers[-1].forward_multiple(self.x_val)
            layer.y_val = self.y_val
            layer.init_nodes()
            layer.train()
            layer.evaluate()
            selection_block = SelectionBlock()
            selection_block.input_layer = layer
            selection_block.filter()
            self.layers.append(selection_block.output_layer)
            self.selection_blocks.append(selection_block)
            curr_sb = self.selection_blocks[-1]
            prev_sb = self.selection_blocks[-2]
            curr_mse = curr_sb.best_metrics.mse()
            prev_mse = prev_sb.best_metrics.mse()
            if curr_mse >= prev_mse:
                for node in self.layers[-1].nodes:
                    node.is_active = False
                break

    def build_structure(self) -> None:
        layers = deepcopy(self.layers)
        layers.pop()
        for node in layers[-1].nodes:
            last_sb = self.selection_blocks[-2]
```

```
            last_node_mse = last_sb.best_metrics.mse()
            if last_node_mse == node.metrics.mse():
                continue
            node.is_active = False
        for layer in layers:
            self.structure.append(layer)

    def get_number_of_inputs(self) -> int:
        return len(self.x_train[0])

    def generate_building_summary(self) -> None:
        total_layers = 0
        total_nodes = 0
        total_synapses = 0
        total_rules = 0
        for layer in self.structure:
            total_layers += 1
            for node in layer.nodes:
                total_nodes += 1
                for synapse in node.synapses:
                    total_synapses += 1
                    total_rules += len(synapse.rules)
        number_of_inputs = self.get_number_of_inputs()
        message = "The neural network structure was synthesized:\n"
        message += f"Number of inputs: {number_of_inputs};\n"
        message += f"Total layers: {total_layers};\n"
        message += f"Total nodes: {total_nodes};\n"
        message += f"Total synapses: {total_synapses};\n"
        message += f"Total rules: {total_rules};\n"
        print(message)

    def make_building_report(self) -> None:
        layers, metrics = [], []
        for layer in self.layers:
            layer_nodes, layer_mse = [], []
            for node in layer.nodes:
                if 0 == len(layers):
                    node_inputs = node.signal_numbers
                else:
                    node_inputs = []
                    for n in node.signal_numbers:
                        node_inputs.append(tuple(layers[-1][n]))
                layer_nodes.append(tuple(node_inputs))
                layer_mse.append(node.metrics.mse())
            layers.append(layer_nodes)
            metrics.append(layer_mse)
        metrics = np.array(metrics).astype(str)
        nodes, sb = [], []
        for i in range(len(layers)):
            for node in layers[i]:
                nodes.append(node)
            selection_block = []
            for j in range(len(metrics)):
                for mse in metrics[j]:
                    if i == j:
                        selection_block.append(mse)
                    else:
                        selection_block.append('')
            sb.append(selection_block)
        report_dict = {"Nodes": nodes}
        for i in range(len(sb)):
            report_dict[f"Selection block {i + 1}"] = sb[i]
```

```
        self.building_report = pd.DataFrame(report_dict)

    def predict(self, x: np.ndarray) -> np.ndarray:
        y = []
        for i in x:
            self.input = i
            output = self.input
            for layer in self.structure:
                layer.input = output
                layer.forward()
                output = layer.output
            self.output = output
            y.append(self.output[0])
        return np.array(y)


# main.py


import os
import shutil
import pandas as pd

from gmdh_neo_fuzzy import DataProcessor, GMDHNeoFuzzy, Metrics
from matplotlib import pyplot as plt


if __name__ == '__main__':
    dp = DataProcessor(src='source/dji-close.csv', validation_split=0.3,
interval=1,
                       window_size=3, test_size=50)
    dp.make_correlation("results/correlation.csv")

    model = GMDHNeoFuzzy()
    model.x_train = dp.x_train
    model.y_train = dp.y_train
    model.x_val = dp.x_val
    model.y_val = dp.y_val
    model.x_test = dp.x_test
    model.y_test = dp.y_test
    model.build()
    model.make_building_report()

    dir_name = f"results"
    forecast_dir_name = f"{dir_name}/forecast"

    for dr in [dir_name, forecast_dir_name]:
        if os.path.isdir(dr):
            shutil.rmtree(dr)
        os.makedirs(f"{dr}", exist_ok=True)

    model.building_report.to_csv(f"{dir_name}/build.csv", index=False)

    y_pred = model.predict(dp.x_test)
    y_pred = dp.scaler.inverse_transform(y_pred)
    y_pred = y_pred.ravel().round(2)
    mse, mape = [], []

    real = dp.df[-dp.test_size:].to_numpy()

    for i in range(len(real)):
        metrics = Metrics(y_true=[real[i][1]], y_pred=[y_pred[i]])
```

```
    mse.append(metrics.mse())
    mape.append(metrics.mape())

forecast = pd.DataFrame({
    "Date": real.T[0],
    "Real": real.T[1],
    "Forecast": y_pred,
    "MSE": mse,
    "MAPE": mape
})

forecast.to_csv(f"{forecast_dir_name}/forecast.csv", index=False)

plt.figure(figsize=(12, 6))
plt.grid(True)
ax = plt.gca()
forecast.plot(x="Date", y="Real", ax=ax)
forecast.plot(x="Date", y="Forecast", ax=ax, linestyle="dashed")
ax.set_xlabel("Date")
ax.set_ylabel("Index")
plt.grid(True)
plt.savefig(f"{forecast_dir_name}/forecast.png")
plt.show()

metrics = Metrics(y_true=real.T[1], y_pred=y_pred)
file = open(f"{forecast_dir_name}/metrics.txt", "a+")
file.write(f"MSE: {metrics.mse()}\n")
file.write(f"MAPE: {metrics.mape()}\n")
file.close()

plt.figure(figsize=(12, 6))
plt.plot(forecast.get("MSE"))
plt.title("MSE")
plt.grid(True)
plt.savefig(f"{forecast_dir_name}/mse.png")
plt.show()

plt.figure(figsize=(12, 6))
plt.plot(forecast.get("MAPE"))
plt.title("MAPE")
plt.grid(True)
plt.savefig(f"{forecast_dir_name}/mape.png")
plt.show()
```

ДОДАТОК Б. КОЕФІЦІЄНТИ АВТОКОРЕЛЯЦІЇ

Lag	Coefficient
1	0.9986789764095034
2	0.9963667443170001
3	0.9936526877775418
4	0.9906116659006236
5	0.98715784615292
6	0.9834145807335093
7	0.9793610688266553
8	0.974920659557106
9	0.9700483284218511
10	0.9651413850725918
11	0.9602388970157845
12	0.95566272786669
13	0.951009457934368
14	0.9465642930267032
15	0.9424057164206792
16	0.9385427607336984
17	0.9352763391412338
18	0.9323004795080055
19	0.9292358822370629
20	0.9259843554177976
21	0.9226134914987297
22	0.9192904860239395
23	0.9163683647484371
24	0.9139328263074594
25	0.9124159980750811
26	0.9109628775820217
27	0.9092472238780557
28	0.9074583883042371
29	0.906398459947176
30	0.9059885224167096
31	0.9056906251982272
32	0.9053640497762524
33	0.9054648919532775
34	0.9064937648088877
35	0.9089675358173925
36	0.9124575885887678
37	0.917175708659256

38	0.9220335320506552
39	0.9255451110547177
40	0.9288647218280132
41	0.9302847123554993
42	0.931223628630232

ДОДАТОК В. АЛГОРИТМ РОБОТИ ПРОГРАМИ