МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки

Центр післядипломної освіти
Кафедра Програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження методів прогнозування і аналітики кривих ціноутворення цифрових активів

Виконав:
студент 2 курсу групи ІПЗздм-21-1

Гриньова М. О.
(прізвище, ініціали)

Спеціальність	121 – Інженерія програмного забезпечення
Тип програми	Освітньо-наукова
Керівник	проф. Шубін І. Ю.

(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ З.В. Дудар

2023 р.

Харківський національний університет радіоелектроніки

Факультет Центр післядипломної освіти

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма

Освітньо-наукова програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____2023 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Гриньовій Марії Олегівні
(прізвище, ім'я, по-батькові)

1. Тема роботи (проекту) Дослідження методів прогнозування і аналітики кривих ціноутворення цифрових активів

затверджена наказом по університету від «03» квітня 2023 р. № 83 Стз

2. Термін подання студентом роботи до екзаменаційної комісії «12» травня 2023 р.

3. Вихідні дані до роботи *електронні ресурси за обраною тематикою, алгоритми прогнозування кривих ціноутворення цифрових активі, принципи розробки програмного забезпечення*

4. Перелік питань, що потрібно опрацювати в роботі *реферат, вступ, аналіз предметної галузі, дослідження існуючих алгоритмів прогнозування, підготовка даних та встановлення метрик, проведення експериментального дослідження, проєктування архітектури ансамблю моделей, тренування моделей, розробка програмної системи, висновки, перелік джерел посилань.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Аналіз предметної галузі	03 квітня 2023 р.	*виконано*
2.	Огляд існуючих методів	10 квітня 2023	*виконано*
3.	Розробка алгоритмів, проектування та розробка ПЗ	15 квітня 2023	*виконано*
4.	Підготовка пояснювальної записки	20 квітня 2023 р.	*виконано*
5.	Спецчастина	28 квітня 2023 р.	*виконано*
6.	Підготовка презентації та доповіді	03 травня 2023 р.	*виконано*
7.	Попередній захист	05 травня 2023 р.	*виконано*
8.	Нормоконтроль, рецензування	07 травня 2023	*виконано*
9.	Занесення роботи в електронний архів	08 травня 2023	*виконано*
10.	Допуск до захисту в зав. кафедри	11 травня 2023 р.	*виконано*

Дата видачі завдання «03» квітня 2023 р.

Студент _____ Марія ГРИНЬОВА

Керівник роботи _____ Ігор ШУБІН

РЕФЕРАТ/ABSTRACT

Кваліфікаційна робота магістра містить: 93 с., 35 рис., 3 табл., 32 джерела.

АНАЛІТИКА, АНСАМБЛЬ МОДЕЛЕЙ, БЛОКЧЕЙН, КРИВІ НЕЙРОННІ МЕРЕЖІ, ПРОГНОЗУВАННЯ, ПРОГРАМНА СИСТЕМА, ЦИФРОВІ АКТИВИ, ARIMA, GRU, LSTM, SVR.

Об’єктом дослідження є алгоритми прогнозування і аналітика кривих ціноутворення цифрових активів.

Метою роботи є дослідження предметної області для аналізу можливостей розвитку ринку цифрових активів

В результаті запропоновано найоптимальніше рішення для розв’язання поставленої задачі, що дозволить отримувати актуальну інформацію про динаміку змін на ринку криптовалют та ефективно передбачати можливі зміни для кожної монети окремо.

ANALYTICS, ARIMA, BLOCKCHAIN, CRYPTOCURRENCIES, DIGITAL ASSETS, FORECASTING, GRU, LSTM, NEURAL NETWORKS, PRICE CURVES, SOFTWARE SYSTEM, SVR.

The object of the research is forecasting algorithms and analytics of pricing curves for digital assets.

The aim of the work is to investigate the subject area for analyzing the development possibilities of the digital asset market, analyzing existing approaches

As a result of its implementation, the most optimal solution for solving the stated problem has been proposed, which will allow obtaining up-to-date information on the dynamics of changes in the cryptocurrency market and effectively predicting possible changes for each coin individually.

Умови публікації пояснювальної записки

Я, Гриньова Марія Олегівна
(прізвище, ім'я, по батькові)

студентка групи ІПЗздм-21-1 здобувачка вищої освіти на другому (магістерському) рівні

кафедра програмної інженерії,
(повна назва кафедри)

заявляю: моя кваліфікаційна робота на тему Дослідження методів прогнозування і аналітики кривих ціноутворення цифрових активів,
(назва роботи)

що буде представлена до ЕК для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ARIMA –. Autoregressive integrated moving average.

kNN –. k-Nearest neighbor.

SVR – Support vector regression.

RF –. Random Forest.

LSTM –. Long short-term memory.

GRU –. Gated recurrent unit.

RMSE – Root mean square error.

MAE – Mean absolute error.

MAPE – Mean absolute percentage error.

ML – Machine learning.

DL – Deep learning.

UWP – Universal Windows platform.

ВСТУП

На початку XXI століття світова економічна система відчула необхідність в якісно нових змінах, і відтоді розвиток технології блокчейн стрімко набирає обертів в економіках багатьох країн. Діджиталізація світової економіки та швидкий розвиток сфери ІКТ стимулюють виникнення цифрових валют та операцій, що пов'язані з ними. Цифрові валюти – це електронні гроші, які можуть використовуватися як додаткова або альтернативна валюта; часто їхня вартість прив'язана до національних валют та може слугувати базою обміну [1].

Необхідно зазначити, що на сьогодні налічується понад 20 тисяч криптовалют, що є доказом швидкого розвитку цього сектору. У порівнянні з 2018 роком, коли було створено лише близько 1,5 тисячі криптовалют, це вражає своїм темпом росту. Криптовалюти є новим різновидом електронних та віртуальних грошей. Створення Bitcoin у 2009 році, автором якого є Сатоші Накамото, стало початком криптовалютного руху [2]. Метою Сатоші було створення нової електронної платіжної системи, яка дозволить здійснювати цифрові фінансові транзакції між користувачами без посередників, таких як банківські та інші фінансові установи.

Технологія електронних грошей забезпечила новий напрямок розвитку бізнесу, що зумовило формування нової групи покупців та динамічний розвиток міжнародної торгівлі. Криптовалюти стали досконалим інструментом уникнення оподаткування та зборів, що дозволило здійснювати миттєві міжнародні фінансові операції без жодних обмежень. У зв'язку з цим, актуальним стає дослідження проблематики розвитку міжнародного ринку криптовалют в інформаційному середовищі сьогодення

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Загальна характеристика технології блокчейн

Blockchain – термін утворений поєднанням двох англійських слів: «block» – блок та «chain» – ланцюг. Блок – файл, який в залежності від системи має певні характеристики: частота створення, розмір, дані. Дані у блоках зашифровані з використанням криптографічних алгоритмів. Ланцюг – поєднання блоків у логічну послідовність, кожен новостворений блок містить посилання на попередній блок. Такий спосіб зберігання інформації майже унеможливлює підробку даних. Зміна даних в будь-якому блоці призведе до зміни його посилання, таким чином буде запущена «ланцюгова реакція» і зміна буде відхилена [3].

Бази даних використовують таблиці для зберігання інформації, а блокчейн – блоки, що пов’язані один з одним. Блокчейн – це децентралізована база даних (P2P), тобто уся інформація міститься на великій кількості машин, які можуть знаходитись у будь-якій точці світу, головний сервер відсутній. Ці комп’ютери називають вузлами. Блокчейн – прозора система. Кожен може побачити інформацію про будь-який блок. Блокчейн використовує криптографічні алгоритми для забезпечення роботи системи. Вони гарантують незмінність блоку транзакцій, керують аутентифікацією. Хешування – криптографічна техніка, яка перетворює вхідний масив даних довільної довжини у стрічку фіксованої довжини. Хеш-функції, що використовуються у блокчейні мають задовольняти умови відсутності колізій: не можна отримати однаковий хеш з різного набору вхідних даних. Хеші використовуються для вказівок суміжних блоків у зв’язному списку. Уся інформація у блоці хешується для створення вказівника за допомогою бінарного дерева хешів (дерево Меркла), а отже зміна будь-якого шматку інформації призведе до зміни хешу самого блоку і, відповідно, усього ланцюжку. Також у блокчейні використовуються цифрові підписи, що базуються на криптографії з відкритим ключем [4]. Відкритий ключ – використовується для перевірки електронного підпису, доступний усім. Закритий ключ зберігається в

таємниці, потрібен для створення електронного підпису. Відкритий ключ можливо обрахувати за допомогою закритого, проте зворотний процес має бути неможливим (тільки за допомогою методу brute-force). Ініціатор транзакції підписує дані своїм закритим ключем, шифрує повідомлення за допомогою публічного ключа одержувача, отримувач же розшифровує дані за допомогою свого закритого ключа. Таким чином єдиний хто може розшифрувати повідомлення – власник приватного ключа, якому адресована транзакція. Підпис використовується для того, щоб довести, що саме цей користувач ініціював транзакцію. Маючи повідомлення, підпис, та публічний ключ підписанта можна підтвердити його авторство.

1.2 Криптовалюти та їх різновиди

Криптовалюта – різновид віртуальної валюти, створена за допомогою криптографічних методів та математичних обчислень переважно на базі блокчейну. Це і цифрова, і віртуальна валюта, тому що вона існує в інтернеті та створена за допомогою криптографічних алгоритмів.

Всі види криптовалют умовно можна розділити на три основні типи: біткоїн, альткоїни (серед них стейблкоїни), токени (включаючи DeFi-токени). Кожна з цих груп має свої особливості.

За визначенням В. Пола Біткоїн – це тип електронних грошей, який є найбільшою за вартістю криптовалютою у світі та першою децентралізованою криптовалютою. Він був створений у 2008 році Сатоші Накамото, неіменованим індивідом або групою індивідів. Він був запущений після того, як архітектура валюти стала відкритою для загального доступу як програмне забезпечення з відкритим кодом в 2009 році. Біткоїн – це децентралізована цифрова валюта, яку можна передавати від одного користувача до іншого через пірингову мережу біткоїн без використання посередника

Біткоїн — це глобальна однорангова електронна платіжна система, яка дозволяє сторонам здійснювати угоди без посередників від імені банку чи іншої фінансової організації [5]. Ця криптовалюта часто розглядається як цифрова альтернатива фіатним валютам та золоту. У біткоїна є низка важливих відмінностей від традиційних паперових грошей:

– обмежена емісія. Традиційні (фіатні) гроші друкують центробанки, ФРС та інші державні установи. Іноді вони друкують їх так багато, що це призводить до інфляції. Біткоїн же, навпаки, — ресурс вичерпний. Його виробництво (емісія) обмежене програмно, і максимально можлива сума в обігу становить 21 млн BTC;

– відсутність посередників. Біткоїн – це монета, яку користувачі можуть передавати один одному безпосередньо, з електронного гаманця на гаманець. Їм не потрібна допомога банків, платіжних систем (на кшталт SWIFT, Visa) та інших фінансових агентів. Система є децентралізованою, а копії блокчейна (історії всіх транзакцій, що здійснювалися) зберігаються у різних користувачів на пристроях.

Альткоїни – це решта криптовалют, створених після біткоїну, частка яких у 2022 році на крипторинку досягла приблизно 40%.

Запуск біткоїну та його відкритого вихідного коду у 2008 році проклав шлях до створення тисяч інших криптовалют, які стали альтернативними до біткоїна монетами (альткоїнами). Як і біткоїн, всі альткоїни можуть працювати незалежно у власних мережах, використовуючи технологію розподіленого реєстру (DLT). Найвідомішим і найчастіше використовуваним типом DLT є технологія блокчейн. При цьому варіації в базовому коді кожного протоколу – це те, що робить кожен альткоїн унікальним.

Стейблкоїни – це альткоїни, курс яких чимось забезпечений. Наприклад, сильними фіатними валютами (доларом США, євро тощо), чи товарними цінностями (наприклад, золотом), чи іншими криптовалютами. Ціна на стейблкоїн забезпечена іншим фінансовим активом, який не є настільки волатильним, а отже, курс стейблкоїну відповідатиме курсу активу, який його забезпечує. Один із найвідоміших стейблкоїнів – криптовалюта USDT від

компанії Tether. Вона забезпечена американським доларом і її курс відповідає ринковому курсу долара.

1.3 Актуальність проблеми

Нейронні мережі та глибоке навчання для прогнозування та торгівлі криптовалютою – це новий і актуальний напрямок досліджень. Вони були широко вивчені та використовуються для прогнозування та торгівлі цінами на акції. Аналіз попиту на криптовалюти, зокрема на Біткоїн, також виконується різними алгоритмами.

Оскільки особливості Біткоїну настільки складні, а його ціна змінюється з часом, існує лише кілька алгоритмів, які можуть прогнозувати вартість криптовалют Bitcoin. На рисунку 1.1 зображено, як змінювалась ціна Біткоїну за всі часи . Тому, для підтримки сучасних інвестиційних рішень, стало необхідним точно прогнозувати вартість Біткоїну, який є найпоширенішою криптовалютою. Оскільки ця проблема належить до категорії прогнозування часових рядів, можуть бути розглянуті механізми глибокого навчання та моделі нейронних мереж, особливо при прогнозуванні цін даних, які є часовими за своєю природою.

Рисунок 1.1 – Інфографіка змін ціни Біткоїну за останні роки

Трейдери та інвестори зацікавлені в точному прогнозуванні цін на криптовалюту для збільшення прибутку та мінімізації ризику. Однак через їх невизначеність, волатильність та динаміку, прогнозування цін на криптовалюту є складною задачею аналізу часових рядів. Дослідники запропонували методи передбачення на основі статистичних підходів, машинного навчання (ML) та глибокого навчання (DL), але література є обмеженою. Насправді, вона є обмеженою тому, що фокусується на передбаченні цін лише кількох найвідоміших криптовалют. Крім того, вона розкидана, оскільки порівнює різні моделі на різних криптовалютах непослідовно, і вона не має загальності, тому що рішення є занадто складними та важкими в реалізації.

1.4 Постановка задачі

Метою атестаційної роботи є дослідження існуючих методів прогнозування і аналітики кривих ціноутворення цифрових активів, розробка власного ансамблю моделей на основі класичних алгоритмів, а також розробка програмної системи для демонстрації результатів виконання найефективніших алгоритмів.

Під час аналізу предметної галузі, а саме літератури та існуючих рішень та постановці проблеми були сформовані вимоги для подальшої роботи, також були виявлені основні напрями та задачі для проведення порівняння. Основним завдання було поставлено пошук підходу, який дає найкращий показник у якості, для роботи з різними тестовими даними. Також у даній роботі потрібно реалізувати та проаналізувати різні варіації нейромережевих підходів та алгоритмів для аналітики та прогнозування кривих ціноутворення. Щоб об’єктивно оцінити та порівняти результат реалізованих варіантів, потрібно розглянути їх роботу на різних вхідних даних. Для цього аналізуватимемо зміни ціни різних криптовалют за різний проміжок часу. Окрім того, необхідно реалізувати новий алгоритм, а саме створити ансамбль моделей та порівняти його ефективність з усіма розглянутими методами,

щоб зробити висновок про доречність даного підходу, довести його ефективність і продемонструвати результати в десктопному додатку.

При реалізації нейромережевих підходів слід виконати наступні завдання:

– для навчання нейронної мережі треба провести збір даних;

– в якості нейронної мережі – ансамблю моделей слід розробити та реалізувати мережу з архітектурою, що підходить для роботи зі статистичними даними та прогнозування;

– потрібно провести комплексну оцінку ефективності отриманих моделей за допомогою обраних метрик.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ПРОГНОЗУВАННЯ

2.1 Стаціонарні підходи

Авторегресійна інтегрована рухома середня (ARIMA) – це узагальнення простішої моделі ARMA (авторегресійної рухомої середньої). Традиційний трьохкроковий процес побудови моделей ARIMA включає ідентифікацію моделі, оцінювання параметрів та, нарешті, діагностику симуляції та її верифікацію. В основному, передбачення значення є лінійною комбінацією значень y_{t_i} до моменту часу t та помилок передбачення, зроблених для тих самих y_{t_i}значень. Приклади використання ARIMA включають прогнозування попиту на авіаперевезення, прогнозування довгострокового заробітку та прогнозування цін на електроенергію на наступний день [6].

ARIMA моделі – це загальний клас моделей, що використовуються для прогнозування часових рядів даних. Зазвичай ARIMA моделі позначаються як ARIMA (p,d,q), де p – це порядок авторегресійної моделі, d – ступінь диференціювання, а q – порядок моделі плаваючого середнього. ARIMA моделі використовують диференціювання для перетворення нестаціонарного часового ряду в стаціонарний, а потім прогнозують майбутні значення на основі історичних даних. Ці моделі використовують "авто" кореляції та плаваючі середні по залишкових помилках в даних для прогнозування майбутніх значень.

Можливі переваги використання моделей ARIMA:

– вимагається лише вхідний часовий ряд для прогнозування;

– показує добрі результати на короткострокових прогнозах;

– моделює нестаціонарні часові ряди.

По-перше, ARIMA – це авторегресійна модель, у якій до попередніх значень y, до часу p, використовуються як передбачувані змінні. Тут p – це кількість відстаючих спостережень у моделі, ε – білий шум в часі t, c – константа, а φs – параметри.

$$\hat{y}_t = c + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \cdots + \varphi_p y_{t-p} + \epsilon_t$$

По-друге, ARIMA – це стаціонарна модель. Стаціонарний часовий ряд – це той, у якого статистичні характеристики залишаються сталими протягом часу. Ці характеристики можуть включати середнє значення, дисперсію та автокореляцію. Коли часовий ряд є стаціонарним, моделювання та аналіз його поведінки з часом стає значно простішим.

Розглянемо два графіки з книги "Forecasting: Principles and Practice" (2-е видання) від Роба Дж. Гіндмана та Джорджа Атанасопулоса. Графік на рисунку 2.1 показує ціну на акції Google протягом 200 послідовних днів. Це нестаціонарний часовий ряд. Графік на рисунку 2.2 показує щоденну зміну ціни акцій Google протягом 200 послідовних днів. Другий графік є стаціонарним, оскільки його значення не залежить від часу спостереження. У цьому прикладі, порядок диференціювання дорівнює одиниці, оскільки перший диференційований ряд є стаціонарним.

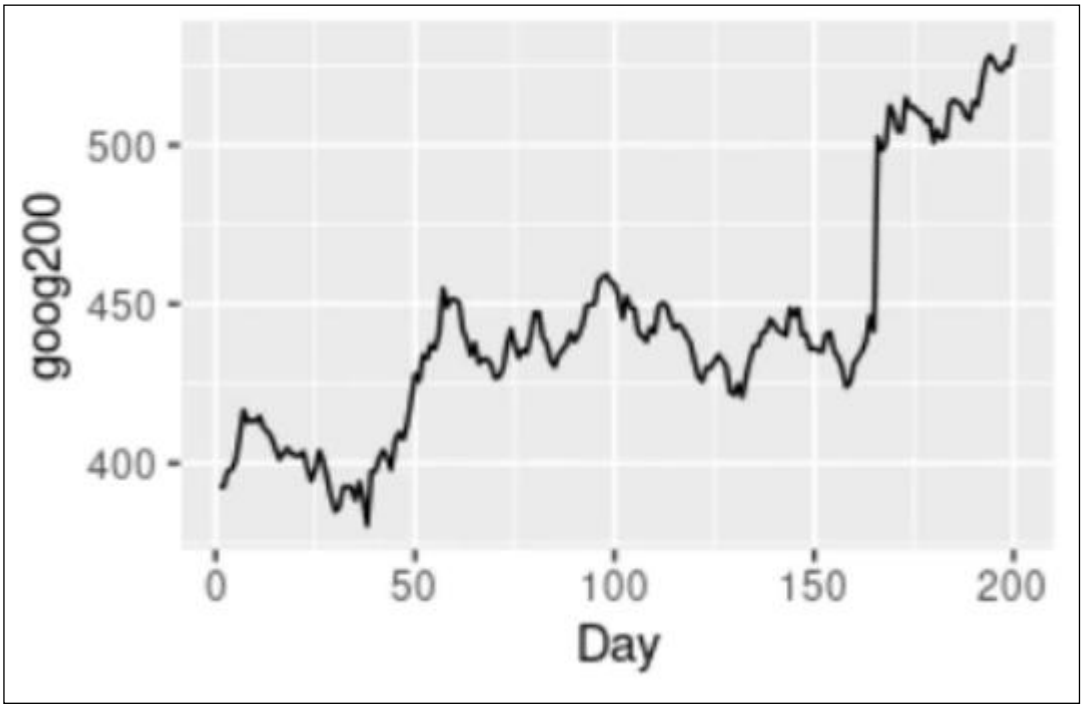

Рисунок 2.1 – Ціна на акції Google протягом 200 послідовних днів

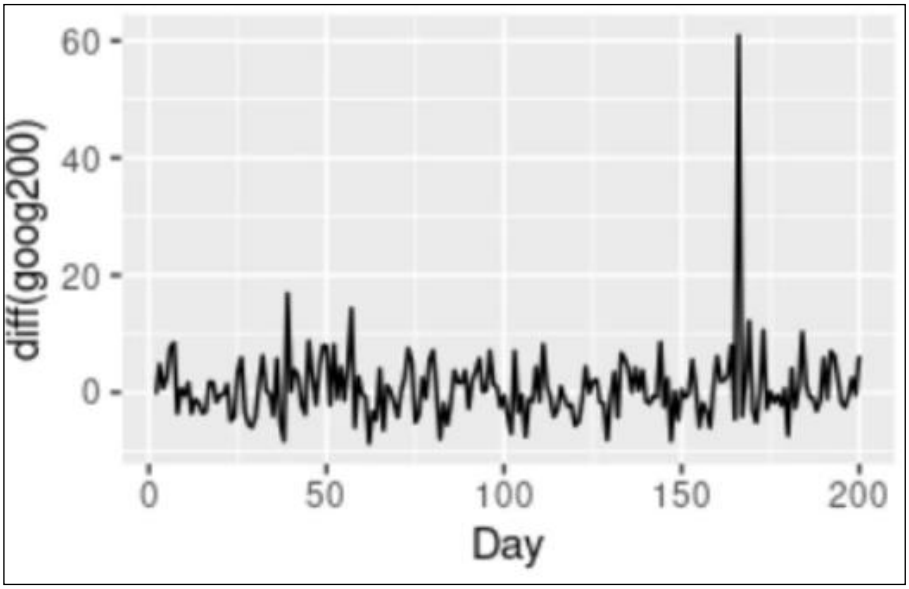

Рисунок 2.2 – Щоденна зміна ціни акцій Google протягом 200 днів

По-третє, ARIMA використовує модель, подібну до регресії, на минулих прогнозованих помилках. Тут ε – це білий шум в час t, c – стала, а θs – параметри.

$$\hat{y}_t = c + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_p \epsilon_{t-p} + \theta_q \epsilon_{t-q}$$

Комбінування трьох типів моделей, згаданих вище, призводить до отримання моделі ARIMA(p,d,q):

$$\hat{y}_t{}' = c + \varphi_1 y'_{t-1} + \varphi_2 y'_{t-2} + \cdots + \varphi_p y'_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \theta_q \epsilon_{t-q} + \epsilon_t$$

Часто використовуються декілька варіацій моделі ARIMA. Якщо використовується декілька часових рядів, то y_t можна розглядати як вектори, і може бути відповідним VARIMA-модель. У випадку, коли під час аналізу даних є підозра на наявність сезонного ефекту в моделі, у такому випадку, загалом вважається краще використовувати SARIMA (сезонну ARIMA) модель, ніж збільшувати порядок частин AR або MA моделі. Якщо підозрюється довготривала залежність часового ряду, тоді параметру d можна дозволити мати невід'ємне

значення в моделі авторегресії фрактально-інтегрованого ковзного середнього, яка також називається моделлю фрактальної ARIMA (FARIMA або ARFIMA).

Алгоритм k-Nearest Neighbor (kNN) – це алгоритм машинного навчання, який може використовуватись для класифікації та регресії. У кожному з наборів даних, kNN шукає k-найближчих сусідів для точки, яку необхідно передбачити або класифікувати. Найчастіше kNN використовують для розв'язання задач класифікації, де точка призначена до класу, який найбільш часто зустрічається серед її k-найближчих сусідів.

Формула kNN для прогнозування може бути записана як:

$$y = \frac{1}{k} * \sum y_i,$$

де *y* – прогнозоване значення для точки, яку необхідно передбачити;

y_i – і-те значення з k-найближчих сусідів;

k – кількість найближчих сусідів, яку необхідно вибрати для передбачення.

Для класифікації, kNN використовується для вибору класу, який найчастіше зустрічається серед k-найближчих сусідів. Формула kNN для класифікації може бути записана як:

$$y = argmax\left(\frac{1}{k}\right) * \sum I(y_i = c),$$

де *y* – клас, який має бути призначений до точки, яку необхідно класифікувати;

c – можливий клас;

y_i– і-те значення з k-найближчих сусідів;

k – кількість найближчих сусідів, яку необхідно вибрати для класифікації;

функція I вказує, чи є y_i рівним класу *c*.

2.2 Механізми машинного навчання

Support Vector Regression (SVR) – алгоритм машинного навчання, що використовується для розв'язання задач регресії. Він був запропонований у 1996 році та є однією з форм SVM (Support Vector Machine) [7].

Суть алгоритму полягає в знаходженні оптимальної гіперплощини, яка найкраще розділяє дані на дві класи. У випадку SVR, на відміну від класичної SVM, ми не розділяємо дані на класи, а намагаємося побудувати гіперплощину, яка якомога точніше описує залежність між вхідними даними та їх відповідними значеннями цільової змінної.

Загалом, процес навчання SVR складається з кількох кроків:

– побудова гіперплощини: SVR намагається знайти оптимальну гіперплощину, яка якомога точніше описує залежність між вхідними даними та їх відповідними значеннями цільової змінної;

– визначення функції ядра: у випадку SVR ми використовуємо функцію ядра, яка дозволяє перетворити вхідні дані у вищу просторову вимірність, де легше знайти оптимальну гіперплощину;

– встановлення гіперпараметрів: як і у більшості алгоритмів машинного навчання, SVR має декілька гіперпараметрів, які потрібно встановити перед навчанням моделі. Ці параметри впливають на точність та швидкість навчання моделі;

– оптимізація: після встановлення гіперпараметрів, SVR проводить оптимізацію функції витрат за допомогою методу опорних векторів.

У свою чергу, SVR є потужним алгоритмом машинного навчання, який використовується для аналізу регресії. На відміну від традиційних алгоритмів регресії, які намагаються мінімізувати помилку між прогнозованими та фактичними значеннями, SVR намагається вписати якомога кращу лінію в межах визначеної межі помилки.

Основна ідея за SVR полягає в тому, щоб перетворити вхідні дані в простір ознак більш високої розмірності та знайти гіперплощину, яка розділяє точки даних на різні класи. Гіперплощина вибирається таким чином, щоб максимізувати зазор між найближчими точками даних з кожного класу.

У випадку регресії SVR намагається знайти гіперплощину, у межах зазору якої знаходиться максимальна кількість точок даних, і в той же час, зберігає помилку між прогнозованими та фактичними значеннями. Допустима помилка, яка може бути здійснена, контролюється параметром C. Більша значення C призводять до меншої допустимої помилки, але можуть перенаситити модель, тобто занадто добре підігнати її під навчальні дані, що може призвести до поганої загальної продуктивності на нових даних. Оптимальні значення C і параметр ядра можна знайти за допомогою крос-валідації.

Для досягнення мети вмістити якомога більше екземплярів в межах зазору SVR використовує функцію втрат, яка штрафує екземпляри, що виходять за межі зазору. Функція втрат також містить термін регуляризації, який запобігає перенавчанню.

Рішення для SVR визначається таким чином:

$$f(x) = w^T{}_x + b,$$

де w – вектор ваги;

x – вектор вхідних даних;

b – зсув.

Для вирішення задачі оптимізації та знаходження гіперплощини, яка максимізує зазор, SVR використовує метод множників Лагранжа. Розв'язання полягає у пошуку опорних векторів, які є навчальними екземплярами, що лежать на зазорі або є помилково класифікованими, та використовуючи їх для обчислення вектора ваги та зсуву.

SVR також може працювати з нелінійними задачами регресії, використовуючи функцію для перетворення вхідних даних у простір вищої

розмірності, де дані можуть бути легше відокремлені за допомогою гіперплощини. Популярні ядерні функції включають лінійні, поліноміальні та функції радіальної базисної функції (RBF).

За допомогою графічного зображення можна продемонструвати, як працює алгоритм SVR. На графіку показано дві гіперплощини, взірці яких різні, але обидві мають зазор, що містить максимальну кількість точок даних (рисунок 2.3). Точки, що знаходяться на межі зазору, називаються опорними векторами.

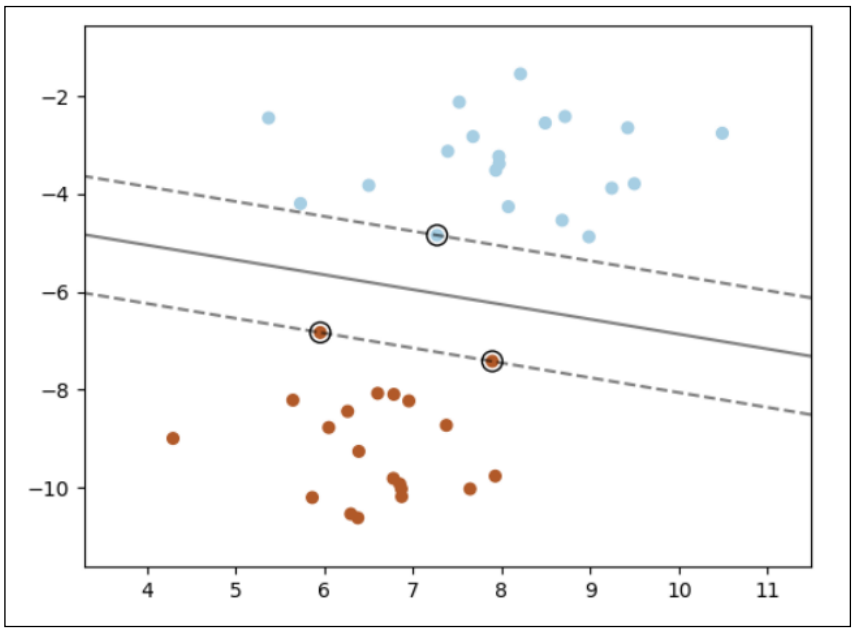

Рисунок 2.3 – Демонстрація роботи методу SVR

Узагальнюючи, SVR є типом SVM, який використовується для задач регресії. Він знаходить гіперплощину, що максимізує зазор навколо тренувальних даних, і використовує функцію втрат з регуляризаційним членом, щоб запобігти перенавчанню. Він також може обробляти нелінійні задачі регресії, використовуючи ядрену функцію для перетворення вхідних даних в простір вищої розмірності, де дані можуть бути легше розділені гіперплощиною. Популярні ядерні функції включають лінійне, поліноміальне та функцію радіальної базисної функції (RBF).

Random Forest (RF) – це один з найбільш популярних алгоритмів машинного навчання. Він широко використовується як для задач класифікації, так і для

регресії. У цьому дослідженні розглянемо, як використовувати Random Forest для задач регресії.

Random Forest – це ансамбль рішень дерев, який дозволяє вирішувати задачі регресії та класифікації. RF розділяє дані на випадкові підвибірки та будує дерева на кожній підвибірці. Потім RF об'єднує прогнози кожного дерева, щоб отримати більш точний результат.

Для задач регресії Random Forest будує дерева, які прогнозують числові значення змінної відгуку. Як і в будь-якому дереві рішень, на кожному рівні RF розділяє дані на дві групи, використовуючи певну характеристику. Однак у RF для кожного розділення використовується випадковий піднабір змінних, а не усі змінні. Це дозволяє зменшити кореляцію між деревами та покращити їхню прогностичну здатність.

RF також дозволяє оцінювати важливість змінних, що використовуються для прогнозування значень відгуку. Важливість змінної визначається як середнє зменшення дисперсії, що відбувається під час розділення дерева за цією змінною. Чим більше зменшення дисперсії, тим більш важливою є змінна.

Random Forest Regressor є варіацією RF, яка використовується для розв'язання задач регресії.

RF – це ансамбль рішучих дерев, що діють разом. Кожне дерево в RF будується на підмножині тренувальних даних і залежно від випадкових властивостей вибору функції із підмножини функцій, а також вибору підмножини ознак.

RF Regressor працює наступним чином. Спочатку, вибирається випадкова підмножина тренувальних даних, на яких буде будуватися дерево. Потім з цих даних випадковим чином вибирається підмножина ознак, на яких буде проводитися розгалуження у дереві. У кожному вузлі дерева, RF Regressor обчислює середнє значення відповідних вихідних даних і використовує його як передбачуване значення для тих даних, які потрапляють в цей вузол. Оцінка результуючого передбачення визначається як середнє значення передбачень, отриманих від кожного дерева.

RF Regressor може допомогти уникнути проблем перенавчання, з якими можуть стикнутися інші алгоритми машинного навчання, завдяки використанню випадковості у процесі побудови дерев. Крім того, RF Regressor може добре працювати з великими обсягами даних та датасетами з багатьма ознаками, що робить його популярним в багатьох галузях.

RF Regressor використовує ансамбль рішень дерев для побудови моделі регресії. Кожне дерево дає прогноз значення цільової змінної на основі вхідних ознак. Прогнози кожного дерева об'єднуються, щоб отримати кінцевий прогноз.

RF Regressor має дві основні параметри: кількість дерев та глибину дерева. Кількість дерев визначається параметром n_estimators, а глибина дерева – параметром max_depth. У RF Regressor кожне дерево побудоване на випадковій підмножині вхідних ознак та випадковій підмножині тренувальних прикладів. Це допомагає зменшити джерела зміщення та дисперсії та забезпечує стійкість до перенавчання.

Формула прогнозу для RF Regressor виглядає так:

$$f(x) = \frac{1}{N} * \sum m \ (predict\_m(x)),$$

де N – кількість дерев;

m – номер дерева;

predict_m(x) – прогноз значення дерева m на вхідних ознаках x.

Таким чином даний підхід краще всього використовувати, коли дані мають нелінійну тенденцію, а екстраполяція за межі тренувальних даних не є важливою. В той самий чином використання даного алгоритму не є доцільним у випадку роботи з даними, що мають форму часових рядів. Проблеми з часовими рядами вимагають ідентифікації зростаючої або зменшуючої тенденції, яку алгоритм Random Forest не зможе сформулювати

Звичайна нейронна мережа може мати проблеми зі зберіганням та використанням інформації, що була зібрана раніше. Це особливо важливо для

аналізу часових рядів, де інформація з попередніх моментів часу може бути корисною для прогнозування майбутніх значень.

Для вирішення цієї проблеми були розроблені рекурентні нейронні мережі (RNN). Ці мережі можуть зберігати інформацію в пам'яті та використовувати її для прийняття рішень на кожному кроці часу.

Одним з найбільш відомих та ефективних типів RNN є мережа довгострокової пам'яті (LSTM). LSTM має додаткові компоненти порівняно зі звичайною RNN, що дозволяє їй зберігати більше інформації та запобігати проблемі зникнення градієнту. Основна ідея LSTM полягає в тому, що вона має «ворота» (gates), що контролюють потік інформації в та з пам'яті. Ці ворота дозволяють мережі регулювати, яка інформація має бути збережена в пам'яті, а яка – проігнорована.

LSTM складається з чотирьох основних компонентів: клітини пам'яті, воріт забування (forget gate), воріт входу (input gate) та вихідний шар (output gate). Кожен з цих компонентів складається зі своїх власних нейронів та зв'язків.

Ключовим елементом LSTM мережі є її внутрішня структура, що називається "клітина" (cell). Клітина зберігає та передає інформацію через часові кроки, а ворота контролюють потік інформації, яка потрапляє до клітини та виходить з неї.

Ворота забування визначають, яку інформацію треба видалити з клітини. Ворота входу визначають, яку інформацію потрібно додати до клітини. І, нарешті, вихідний шар визначає, яку інформацію потрібно отримати з клітини.

Формула для воріт забування:

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

де f_t – вектор ворота забування на часі t;

W_f – матриця ваг;

h_{t-1} – вектор стану на попередньому часовому кроці;

x_t – вектор вхідних даних на часі t;

b_f – зміщення.

Формула для воріт входу:

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

$$\widetilde{C_t} = \tanh(W_C * [h_{t-1}, x_t] + b_C),$$

де $i(t)$ – вектор воріт входу на момент часу t;

W_i – вагова матриця.

Формула для клітини:

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t},$$

де C_t- вектор клітини на часі t;

C_{t-1} – вектор клітини на попередньому часовому кроці.

Формула для вихідного шару:

$$O_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = O_t * \tanh(C_t)$$

.

LSTM блоки використовуються в нейронних мережах, що працюють з послідовностями даних, оскільки вони здатні зберігати та використовувати довгострокові залежності між елементами послідовності. Це дає їм перевагу у моделюванні та прогнозуванні часових рядів, мовних послідовностей, звукових сигналів та інших послідовних даних, де контекст та залежності на довгій відстані мають важливе значення.

Загалом, навчання RNN з використанням LSTM блоків у режимі спостереження за допомогою оптимізаційних алгоритмів, таких як градієнтний спуск, є ефективним способом моделювання та прогнозування послідовних даних, зокрема у випадку, коли важлива довгострокова залежність між елементами послідовності.

Gated Recurrent Unit (GRU) – це тип рекурентної нейронної мережі, яка була розроблена для зменшення проблем зі зникненням/вибуванням градієнтів, що зазвичай виникають у довгих послідовностях вхідних даних. GRU була запропонована в 2014 році Кйона Чуном та Хієном Гоу.

GRU має дві важливі складові: гейти (gates) забування (forget) та оновлення (update), які дозволяють моделі забувати або зберігати певні інформаційні риси з попередніх кроків.

Гейт забування відповідає за видалення непотрібної інформації з попередніх кроків. Він використовує сигмоїдальну функцію для визначення того, які значення потрібно зберегти, а які можна викинути.

Чим більше значення сигмоїди близьке до 0, тим більше інформації буде забуте, і навпаки, чим більше воно близьке до 1, тим більше інформації буде збережено.

Гейт оновлення відповідає за додавання нової інформації до попередніх кроків. Він також використовує сигмоїдальну функцію для визначення того, які значення потрібно оновити, а які можна залишити без змін.

Чим більше значення сигмоїди близьке до 0, тим менше інформації буде додано, і навпаки, чим більше воно близьке до 1, тим більше інформації буде додано.

Загальний вигляд одного юніту GRU зображено на рисунку 2.4.

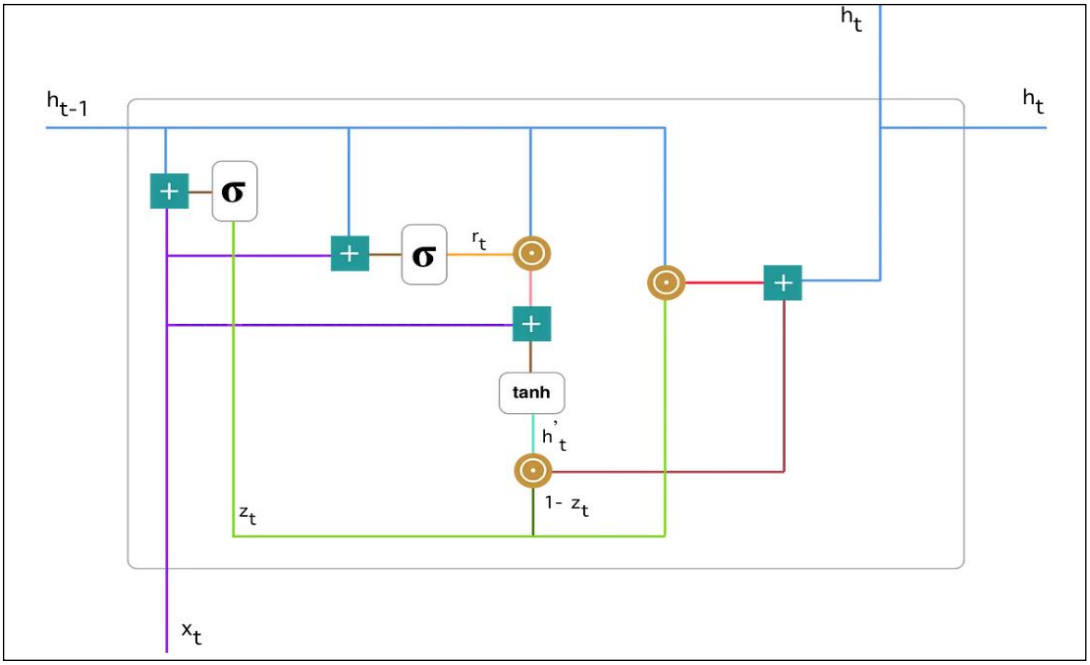

Рисунок 2.4 – GRU

Формула для обчислення нового значення стану h_t в GRU має наступний вигляд:

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

$$h'_t = tanh(Wx_t + r_t \odot Uh_{t-1})$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t$$

де x_t – вхідний вектор у момент часу t;

h_t – стан у момент часу t;

z_t – вектор затвору оновлення;

r_t – вектор затвору забування;

h'_t – тимчасовий вектор стану;

W, U – ваги та зміщення;

σ – функція сигмоїди;

◦ – покомпонентне множення.

Крім того, GRU має менше параметрів ніж LSTM, що робить його більш ефективним для тренування та використання в обчислювально обмежених умовах. Формули для GRU дещо складніші, ніж у LSTM. Але якщо врахувати, що GRU заснований на LSTM, то вони дещо схожі між собою. Відомо, що GRU працює краще за LSTM в деяких випадках, особливо коли є обмеження на ресурси пам'яті та обчислювальної потужності. Він може бути особливо корисним для завдань, що вимагають багато даних та ресурсів, таких як обробка мовленнєвих даних або аналіз часових рядів.

У підсумку, Gated Recurrent Unit (GRU) є потужним інструментом в галузі машинного навчання, який може бути використаний для вирішення різноманітних завдань, таких як машинний переклад, розпізнавання мови, синтез мови та багато інших. В порівнянні з LSTM, GRU має меншу кількість параметрів та може бути більш ефективним у виконанні завдань з великою кількістю даних. Крім того, GRU може бути більш стійким до проблем з градієнтами, що дозволяє йому досягати кращих результатів при роботі з довільними послідовностями даних.

Загалом, GRU є важливим кроком в еволюції рекурентних нейронних мереж і відкриває нові можливості для використання нейронних мереж в різних галузях. За допомогою GRU можна досягнути значних покращень у багатьох завданнях, що потребують обробки послідовностей даних, та допомогти вирішувати складні проблеми, що вимагають великої кількості обчислень та аналізу великих обсягів даних.

3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Збір даних

Для якісного порівняння ефективності в розв’язанні здачі прогнозування кривих ціноутворення цифрових активів описаних в попередньому розділі алгоритмів необхідно обрати надійний історник данних для дата сету, провести їх попередню обробку, а також визначити метрики оцінювання.

Збір даних є важливою та необхідною складовою процесу тренування нейромережі. У контексті розробки та навчання нейромережевих моделей, правильно підібрані та якісно підготовлені дані є ключовим фактором успіху.

Збір даних включає в себе процес отримання, структурування та підготовки даних для використання в тренуванні нейромережі. Цей процес може вимагати залучення різних джерел даних, таких як бази даних, сенсори, веб-скрапінг або зовнішні API.

Один з важливих кроків в зборі даних – це визначення цільової змінної або вихідного параметра, який нейромережа буде навчатися передбачати. Це може бути прогнозування числового значення (регресія) або класифікація у певні категорії (класифікація).

Крім цього, під час збору даних необхідно враховувати якість та репрезентативність даних. Недостатньо об'єму даних або невірність даних можуть призвести до незадовільних результатів під час тренування нейромережі. Додатково, необхідно враховувати можливі проблеми зі збалансованістю класів у випадку класифікаційних завдань.

Загалом, якісний збір даних є ключовим етапом у тренуванні нейромережі, який визначає початкову основу для розвитку та оптимізації моделі. Правильний вибір та підготовка даних можуть позитивно вплинути на точність та ефективність навчання нейромережі, а отже, на її здатність до виконання поставлених завдань, як прогнозування, класифікація або генерація нових даних. Оптимально підготовлені дані забезпечують нейромережі необхідний контекст та

репрезентативність для вивчення залежностей у даних та утворення придатних модельних уявлень.

Джерелом даних було вирішено використовувати веб-сайти Binance.com (https://www.binance.com/en, звернення від 13 грудня 2022 року) та Investing.com (https://www.investing.com, звернення від 13 грудня 2022 року). Binance.com є найбільшим та найпопулярнішим порталом обміну криптовалютами для щоденної торгівлі в світі. Він надає ряд функцій, специфічних для продуктів криптовалют, які включають інформацію про ринок тисяч криптовалют. Investing.com діє як глобальний портал для інформації про фондовий ринок та аналізу на багатьох світових фінансових ринках. Для дослідження було обрано п'ять популярних криптовалют, а саме XRP, Bitcoin (BTC), Litecoin (LTC), Ethereum (ETH) та Monero (XMR).

Процес збору даних використовував Binance API як основний ресурс та доповнювався інформацією, отриманою з Investing.com, коли виникали пропущені значення (наприклад, коли ціна закриття XMR не була доступна для певного дня). Часовий період зібраних даних охоплює період з 1 січня 2017 року по 31 грудня 2022 року, тобто п'ять років.

3.2 Попередня обробка даних

Попередня обробка даних є важливою стадією в процесі тренування нейромережі. Цей етап включає в себе ряд дій, спрямованих на підготовку даних для подальшого використання в моделі. Якісна попередня обробка даних може покращити якість моделі та забезпечити більш точні прогнози або класифікацію.

Процес попередньої обробки даних включає в себе наступні кроки:

– видалення відсутніх значень: Це важливий крок, де видалення або заповнення відсутніх значень допомагає уникнути проблем з невизначеністю та некоректними результатами. Це можна зробити шляхом використання стратегій,

таких як видалення рядків з відсутніми значеннями або заповнення їх середніми значеннями чи значеннями з близьких прикладів;

– нормалізація та стандартизація: Цей крок полягає в приведенні значень даних до одного масштабу, щоб уникнути перекосів та забезпечити рівномірну обробку. Нормалізація використовується для приведення значень до певного діапазону, наприклад, від 0 до 1. Стандартизація змінює розподіл значень на нормальний з медіаною 0 та стандартним відхиленням 1 [9].

– кодування категоріальних змінних: Якщо дані містять категоріальні змінні, вони повинні бути перетворені на числові значення, щоб їх можна було використовувати в нейромережі;

– видалення викидів: Викиди або аномальні значення можуть спотворити модель та призвести до неправильних висновків. Ці значення можна видалити або замінити на значення на основі статистичних методів, таких як правило 3-х середніх квадратичних відхилень або міжквартильний розмах.

– вибірка даних: У великих наборах даних можуть бути зайві або непотрібні атрибути, які не мають суттєвого впливу на результат. Вибірка даних дозволяє обрати лише необхідні атрибути для моделі, що сприяє зменшенню обчислювальної складності та покращенню швидкості тренування.

– балансування даних: У випадку незбалансованих класів, коли один клас має значно меншу кількість прикладів ніж інші, можна застосувати методи балансування даних, такі як випадкова підвибірка з більшого класу або синтез нових прикладів з меншого класу.

При прогнозуванні часових рядів їх стаціонарність є вирішальною для ефективного моделювання . Часовий ряд з середнім значенням та дисперсією, що не змінюються з часом, називається стаціонарним. Навпаки, часовий ряд, середнє значення, частота та дисперсія якого коливаються з часом та часто відображають високу волатильність, тенденції та гетероскедастичність, називається нестаціонарним [10]. Зазвичай, традиційні статистичні методи прогнозування, такі як ARIMA, потребують стаціонарності часових рядів, щоб успішно відображати їх властивості. На відміну від цього, стаціонарність сприяє навчанню в

нестатистичних моделях, таких як ML та DL, що використовуються у цій роботі. З цих причин ми проводимо статистичний тест на стаціонарність розширеного Дікі–Фуллера (ADF), щоб визначити, чи є наші набори даних стаціонарними. Результати показують, що всі набори даних, крім набору даних XRP, є нестаціонарними.

Як було зазначено раніше, важливим кроком попередньої обробки даних є перетворення наборів даних на стаціонарні шляхом застосування детрендінгу, тобто процесу видалення тенденції з часового ряду. Зокрема, застосовується техніка диференціювання – найпростіший метод детрендінгу, що генерує новий часовий ряд, де нове значення y'_{t_i} на момент часу t_i обчислюється як різниця між початковим спостереженням та спостереженням $y_{t_{i-1}}$ на попередньому кроці часу, тобто:

$$y'_{t_i} = y_{t_i} - y_{t_{i-1}}$$

На рисунку 3.1 зображено оригінальний часовий ряд Bitcoin жовтим кольором та його диференційовану версію червоним кольором ADF, який був проведений на детрендованих наборах даних, що підтверджує їх стаціонарність.

Тепер розглянемо етап нормалізації даних в контексті підготовки даних для тренування нейромережі прогнозування змін кривих ціноутворення цифрових активів. Щоб її виконати необхідно застосувати нормалізацію Мін-Макс до всіх значень y_{t_i} кожного набору даних, так щоб значення були відображені в діапазоні (0, 1) за формулою:

$$y_{t_i} = \frac{y_{t_i} - y_{min}}{y_{max} - y_{min}}$$

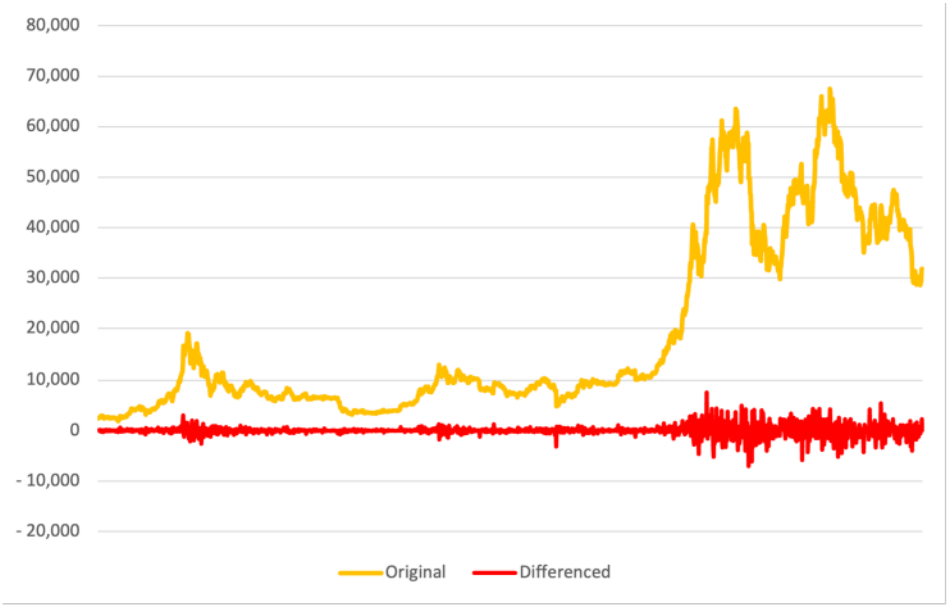

Рисунок 3.1 – Ціна Bitcoin щоденно з січня 2017 року по грудень 2022 року

Загальною метою попередньої обробки даних є покращення якості та готовності даних для тренування нейромережі. Це дозволяє забезпечити надійну та репрезентативну вибірку, що підвищує ефективність та точність моделі при виконанні завдань прогнозування, класифікації або генерації нових даних.

3.3 Метрики оцінювання алгоритмів

Метрики оцінювання грають важливу роль у визначенні ефективності та якості алгоритмів машинного навчання. Вони допомагають визначити, наскільки добре модель працює на тестових даних та як точно вона виконує поставлені завдання, такі як прогнозування або класифікація.

У цьому підрозділі ми розглянемо основні метрики оцінювання алгоритмів та їх значення. Кожна метрика має свої особливості та використовується для вимірювання певних аспектів моделі.

Для оцінки якості передбачень моделі було обчислено середньоквадратичну помилку (RMSE), середню абсолютну помилку (MAE), середню абсолютну

відсоткову помилку (MAPE) та коефіцієнт детермінації (R^2) на кожному етапі оцінки, описаному в попередньому розділі, наступним чином:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_{t_i} - \acute{y}_{t_i})^2}{n}},$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_{t_i} - \acute{y}_{t_i}|,$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_{t_i} - \acute{y}_{t_i}|}{|y_{t_i}|} * 100,$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_{t_i} - \acute{y}_{t_i})^2}{\sum_{i=1}^{n}(y_{t_i} - \bar{y}_{t_i})^2},$$

де y_{t_i} – це реальна ціна криптовалюти після нормалізації;

$\acute{y}_{t_i}$ – передбачуване значення;

$\bar{y}$ – середнє значення прогнозованих значень;

n – кількість прогнозованих значень.

Варто зазначити, що метрика коефіцієнта детермінації відображає дисперсію моделі відносно загальної дисперсії. Тому, на відміну від інших метрик помилок, чим вище значення R^2, тим краще виконання моделі.

3.4 Отримані результати

Таблиця 3.1 показує середні результати кожної моделі, обчислені для всіх криптовалют. Моделі відсортовані за зростанням RMSE.

Можна побачити, що ранжування моделей однакове для всіх показників точності (з дуже небагатьма винятками). LSTM показує найкращі результати зі стійкою перевагою порівняно з іншими моделями. Для кожного показника значення досить близькі, оскільки ми обчислюємо їх на нормалізованому передбачуваному ціновому рівні, а не на детрендованих даних. Моделі рекурентних нейронних мереж займають перші три позиції в ранжуванні, за ними йдуть KNN та згорткова мережа. Цікаво, що ARIMA показує кращі результати порівняно з RF та SVR.

Стосовно часу, необхідного для навчання та застосування моделей, DL підходи є дорожчими, порівняно з методами машинного навчання та статистичними методами, як і очікувалося. В цілому, всі моделі надають передбачення в достатньо короткий час, тому вони можуть бути придатні для роботи в деяких онлайн-налаштуваннях. ML моделі виконуються значно швидше. KNN забезпечує хороший компроміс між точністю та обчислювальною вартістю.

Таблиця 3.1 – Середній показник ефективності окремих моделей, впорядкований за зростанням RMSE.

Модель	RMSE	MAE	MAPE	R^2	Тренування (с)	Інференція (мс)
LSTM	0.02224	0.0173	3.862%	0.735	173.765	1.862
GRU	0.02285	0.0176	3.939%	0.720	254.520	1.550
KNN	0.02332	0.0179	4.003%	0.711	<0.01	0.074
ARIMA	0.02343	0.0180	4.010%	0.708	4.035	0.109
RF	0.02402	0.0184	4.095%	0.697	2.121	0.586
SVR	0.02452	0.0189	4.240%	0.681	<0.01	0.008

Таблиця 3.2 показує результати RMSE для різних криптовалют. Рейтинг топ-3 моделей є стабільним для всіх криптовалют. Однак, у нижніх позиціях можна спостерігати деяку змінність, наприклад, SVR показує особливо хороші результати для BTC.

Таблиця 3.2 – RMSE результати виконання окремих моделей для кожної криптовалюти

	BTC	ETH	LTC	XMR	XPR	Середнє
LSTM	0.0239	0.030	0.0189	0.0236	0.0148	0.0222
GRU	0.0245	0.0309	0.0193	0.0243	0.0153	0.0229
KNN	0.0249	0.0319	0.0197	0.0245	0.0155	0.0233
ARIMA	0.0251	0.0320	0.0198	0.0244	0.0158	0.0234
RF	0.0266	0.0322	0.0199	0.0251	0.0157	0.0240
SVR	0.0248	0.0342	0.0207	0.0268	0.0160	0.0245

Отримані результати демонструють, що ранжування ефективності моделей залишається стійким для різних криптовалют, і їхня середня ефективність підтверджує це ранжування. Рекурентні підходи глибокого навчання домінують у завданні прогнозування цін на криптовалюти з точки зору всіх метрик точності. Зокрема, LSTM є найкращою моделлю з середнім RMSE 0,0222 і значно перевершує інші архітектури мереж, які мають вищу похибку. Природа цих архітектур може пояснити їхню погану ефективність.

Другим кращим підходом до прогнозування цін на криптовалюти є GRU – рекурсивна мережа, менш складна за LSTM. Її середня помилка RMSE на 2,7% вище, ніж у LSTM, але для її виконання потрібна схожа обчислювальна потужність. Отже, результати для моделей глибинного навчання свідчать про те, що більш дорогі та складні архітектури можуть бути надмірними для цього типу завдання з часовими рядами.

Ці результати показують, що KNN надає відмінний баланс між точністю передбачення та обчислювальним зусиллям. Помилка його передбачень на 4,8% вища, ніж у LSTM, але не вимагає часу на навчання і має 25 разів швидший час інференції. Інші моделі машинного навчання (SVR та RF) знаходяться внизу рейтингу і, досить дивно, перевершує їх базова модель ARIMA. Це, мабуть, тому, що вони не можуть захопити значущі патерни в часовому ряді, який є шумним та містить викиди (SVR працює краще, оскільки менше схильний до викидів). На

відміну від цього, через свої лінійні припущення, передбачення ARIMA мають напрямок і є більш точними для короткострокового аналізу. У висновку, ARIMA забезпечує хороший баланс між точністю та зменшеним обчислювальним попитом.

У цьому дослідженні пропонується створити ансамбль моделей на основі LSTM та GRU для покращення прогнозування змін цін на фінансових ринках. Попередні випробування показали, що обидва типи моделей демонструють високу точність та здатність виявляти складні залежності у часових рядах криптовалют.

Для створення ансамблю моделей, ми пропонуємо об'єднати LSTM та GRU шляхом комбінування їх прогнозів. При такому підході, кожна модель LSTM та GRU прогнозує зміни цін окремо, а їх прогнози потім комбінуються за допомогою певної агрегаційної стратегії.

Впровадження ансамблю моделей буде включати створення множини LSTM та GRU моделей, які будуть тренуватися з використанням попередньо визначених параметрів і гіперпараметрів. Після тренування кожна модель буде застосована до перевірочного набору для прогнозування змін цін.

Завершуючи експериментальний аналіз, ми порівняємо результати ансамблю моделей LSTM та GRU з результатами окремих моделей.

4 ПОБУДОВА АНСАМБЛЮ МОДЕЛЕЙ НА ОСНОВІ МУЛЬТИСКЕЙЛОВОГО АНАЛІЗУ ТА ГЛИБОКОГО НАВЧАННЯ

4.1 Загальні відомості

Ансамбльне навчання було запропоновано як елегантне рішення для подолання високої дисперсії окремих моделей прогнозування та зменшення загальної помилки узагальнення. Основний принцип будь-якої стратегії ансамблю полягає в тому, щоб зважити кілька моделей та комбінувати їх індивідуальні прогнози для покращення результативності прогнозування; при цьому ключовим моментом для ефективності ансамблю є те, що його компоненти повинні відрізнятися точністю та різноманітністю своїх прогнозів. Загалом, комбінація прогнозів кількох моделей додає зміщення, яке в свою чергу компенсує дисперсію окремої навченої моделі. Тому, зменшуючи дисперсію в прогнозах, ансамбль може працювати краще, ніж будь-яка окрема краща модель.

Сучасний розвиток машинного навчання та штучного інтелекту вимагає від інженерів та дослідників розробки нових підходів до аналізу та обробки даних. Одним з таких підходів є побудова ансамблю моделей на основі мультискейлового аналізу та глибокого навчання. Ця технологія дозволяє підвищити якість передбачення та роботи систем штучного інтелекту, забезпечуючи аналіз даних на різних рівнях деталізації та залучаючи багатошарові нейронні мережі для вирішення складних задач. У даному розділі ми розглянемо базові відомості про архітектуру та роботу ансамблю моделей на основі мультискейлового аналізу та глибокого навчання.

Мультискейловий аналіз – це метод аналізу даних, який використовується для вивчення інформації на різних рівнях деталізації або масштабу. Замість того, щоб досліджувати дані на одному єдиному рівні, мультискейловий аналіз дозволяє розглядати дані на більшій кількості рівнів деталізації.

Одним з основних принципів мультискейлового аналізу є використання ієрархії масштабів. Кожен рівень масштабу відповідає певній ступені деталізації даних. Наприклад, у випадку зображень, нижні рівні можуть представляти загальні

контури та структуру, тоді як вищі рівні можуть відображати більш дрібні деталі та текстуру.

Мультискейловий аналіз може бути використаний для різних завдань, таких як обробка зображень, аналіз текстів, виявлення об'єктів та багато інших. Використання мультискейлового підходу дозволяє отримати більш повну та комплексну інформацію з даних, враховуючи їх структуру та характеристики на різних рівнях деталізації.

У контексті побудови ансамблю моделей на основі мультискейлового аналізу, цей метод може бути використаний для створення різних моделей, які аналізують дані на різних рівнях деталізації, і об'єднання їх результатів для отримання кращих прогнозів або результатів.

Глибоке навчання – це галузь машинного навчання, яка зосереджується на навчанні та використанні глибоких нейронних мереж. Глибокі нейронні мережі складаються з багатошарових структур, які намагаються моделювати роботу людського мозку шляхом шарування багатошарових штучних нейронів.

Одна з ключових властивостей глибокого навчання – це здатність до автоматичного витягування корисних ознак або представлення даних на різних рівнях абстракції. За допомогою глибоких нейронних мереж можна розв'язувати складні завдання, які вимагають великої кількості даних та складних взаємозв'язків між ними. Глибоке навчання виявилося особливо ефективним у таких областях, як комп'ютерний зір, обробка природної мови, розпізнавання мови, рекомендаційні системи та інші.

Основним інструментом глибокого навчання є штучні нейронні мережі з багатошаровою архітектурою. Ці мережі можуть бути навчені на великих наборах даних за допомогою алгоритмів, таких як зворотне поширення помилки (backpropagation), що дозволяє автоматично налаштовувати ваги мережі для досягнення оптимального прогнозу або класифікації.

Глибоке навчання продовжило революцію в галузі штучного інтелекту та машинного навчання. Воно дозволяє вирішувати завдання, які раніше вважалися

складними або недосяжними, і досягати вражаючих результатів у багатьох областях.

Однією з основних переваг глибокого навчання є його здатність до самостійного витягування ознак з даних. Замість того, щоб самостійно визначати та вибирати ознаки, необхідні для розв'язання задачі, глибокі нейронні мережі можуть автоматично навчитися репрезентувати дані на власний спосіб. Це дозволяє моделі розпізнавати складні закономірності та взаємозв'язки між даними, які можуть бути недоступні для людського спостереження.

Глибоке навчання також вигідно використовувати в ситуаціях, коли доступно велике число даних. Великі набори даних дозволяють глибоким нейронним мережам навчитися вищого рівня абстракцій та репрезентувати більш складні концепти. Це дозволяє досягти високої точності та знизити помилки в передбаченнях.

Наряду з цим, глибоке навчання здатне ефективно працювати з різноманітними типами даних, включаючи зображення, текст, аудіо та інші. Це робить його універсальним інструментом для багатьох завдань, від обробки зображень та розпізнавання мови до рекомендаційних систем та обробки природної мови.

Загалом, глибоке навчання відкриває широкі можливості для розвитку штучного інтелекту та досліджень у сфері машинного навчання. Воно продовжує розширювати свої горизонти і застосовується в різних галузях, включаючи медицину, фінанси, автомобільну промисловість, робототехніку та багато інших. Воно допомагає вирішувати складні проблеми, забезпечує високу точність та надійність результатів і відкриває нові можливості для інновацій.

Проте, важливо зазначити, що глибоке навчання також має свої виклики і обмеження. Навчання глибоких нейронних мереж вимагає значних обчислювальних ресурсів і тривалого часу. Великі набори даних можуть бути необхідні для досягнення хороших результатів, що може бути складно в деяких випадках. Крім того, інтерпретованість результатів глибоких моделей може бути

викликом, оскільки вони працюють на високоабстрактному рівні, що може ускладнювати розуміння причинно-наслідкових зв'язків у даних.

4.2 Огляд архітектури

При побудові архітектури ансамблю моделей на основі мультискейлового аналізу та глибокого навчання можуть використовуватися різні підходи. Основна мета полягає у поєднанні прогнозів та результатів різних моделей для отримання кращої якості передбачень. Деякі з популярних підходів включають:

– багатошарові ансамблі (Stacked Ensembles): Цей підхід передбачає побудову ансамблю з кількох шарів моделей, де кожен шар складається з набору базових моделей. На першому шарі кожна модель навчається незалежно на різних підмножинах даних. Потім результати цих моделей стають входом для моделей другого шару, які навчаються враховуючи ці прогнози. Процес може продовжуватися на декількох шарах. Загальні прогнози отримуються шляхом агрегування прогнозів з останнього шару;

– бустінг (Boosting): Цей підхід базується на послідовному навчанні слабких моделей, при цьому кожна наступна модель фокусується на виправленні помилок попередніх моделей. Кожна модель навчається на вибірці, зваженій за результатами попередньої моделі. Потім прогнози моделей об'єднуються шляхом комбінування їх результатів з додатковими вагами;

– беггінг (Bagging): Цей підхід полягає у незалежному навчанні кількох моделей на випадкових підмножинах даних. Кожна модель навчається на своєму підмножині, і прогнози цих моделей комбінуються шляхом усереднення або голосування;

– випадковий ліс (Random Forest): Це один з популярних методів ансамблю моделей, який поєднує ідеї беггінгу та розбиття дерев. Випадковий ліс складається з набору рішень на основі дерев рішень, де кожне дерево навчається на випадковій

підмножині даних та випадковій підмножині ознак. Комбінація результатів цих дерев дозволяє отримати кінцевий прогноз;

– стекінг (Stacking): Цей підхід полягає у комбінуванні прогнозів різних моделей, використовуючи додаткову модель вищого рівня, відому як мета-модель або агрегаційна модель. Перші моделі навчаються незалежно, а їх прогнози стають вхідними даними для мета-моделі, яка навчається передбачати кінцевий результат. Мета-модель може використовувати різні алгоритми, наприклад, лінійну регресію або нейронну мережу.

Ці підходи до побудови архітектури ансамблю моделей дозволяють комбінувати прогнози різних моделей для отримання кращих результатів, покращення стійкості до шуму та зниження ризику перенавчання. Важливими аспектами є різноманітність моделей в ансамблі, правильний підбір параметрів та використання регуляризації для зменшення перенавчання.

При побудові архітектури ансамблю моделей важливо враховувати особливості конкретної задачі, типу даних та ресурсів, доступних для навчання. Оптимальний вибір архітектури може допомогти досягнути кращих результатів та покращити загальну ефективність ансамблю моделей.

В даній роботі використовується ансамбль-модель під назвою "stacked generalization " або стекінг. Цей вибір легко обґрунтувати наступними причинами:

– узгодженість з особливостями задачі: Стекінг є потужним методом ансамблю моделей, який дозволяє комбінувати прогнози різних моделей, залежно від їхньої ефективності. У задачі прогнозування кривих цінотворення, де точність і надійність передбачень є ключовими, використання стекінгу дозволяє отримати кращі результати шляхом комбінування інформації з різних джерел;

– здатність до моделювання складних залежностей: Стекінг дозволяє створювати моделі вищого рівня, які можуть уявляти більш складні залежності у даних. У задачі прогнозування цінотворення цифрових активів можуть бути складні взаємозв'язки та нелінійність. Використання стекінгу дозволяє створювати більш гнучкі та потужні моделі, які можуть краще апроксимувати ці залежності та забезпечувати точніші передбачення;

– здатність до адаптації до змін: Криптовалюти та інші цифрові активи можуть підлягати швидким та складним змінам на ринку. Стекінг може бути корисним в таких випадках, оскільки його можна легко адаптувати до нових умов. Нові моделі можуть бути додані або існуючі моделі можуть бути оновлені, щоб врахувати нові тренди та зміни на ринку. Це дозволяє ансамблю моделей бути більш гнучким та актуальним у прогнозуванні цінотворення цифрових активів;

– підтримка неоднорідних джерел даних: У задачі прогнозування цінотворення цифрових активів можуть бути доступні різні джерела даних, такі як історичні дані, новини, соціальні мережі та інші. Використання стекінгу дозволяє інтегрувати інформацію з цих різних джерел, що сприяє отриманню більш повного та різноманітного набору ознак для моделей. Це допомагає забезпечити більш точні та стійкі передбачення;

– можливість управління недоліками моделей: Кожна окрема модель може мати свої переваги та недоліки. Використання стекінгу дозволяє компенсувати слабкі сторони окремих моделей шляхом комбінування їх прогнозів з іншими моделями. Якщо одна модель має тенденцію до певного виду помилок, інша модель може її скоригувати. Таким чином, стекінг дозволяє збалансувати та поліпшити якість прогнозів, зменшуючи вплив недоліків окремих моделей;

– легкість інтерпретації та зрозумілість: У порівнянні з іншими складнішими методами ансамблю, стекінг є досить простим у розумінні та інтерпретації. Модель вищого рівня (мета-модель) може бути побудована з використанням стандартних методів машинного навчання, що дозволяє легко враховувати та аналізувати її результати. Це особливо корисно, якщо інтерпретація прогнозів є важливою для прийняття рішень в контексті цифрових активів.

Враховуючи ці фактори, вибір стекінгу для побудови архітектури ансамблю моделей для прогнозування кривих цінотворення цифрових активів є цілком обґрунтованим. Мотивацією цього підходу є обмеження простого ансамблю-середнього, тобто кожна модель розглядається рівнозначно при передбаченні ансамблю, незалежно від того, наскільки добре вона виконує своє завдання. Таким чином, стекінг використовує модель вищого рівня для експлуатації та

комбінування передбачень компонентних моделей ансамблю. Зокрема, моделі, що складають ансамбль (моделі рівня 0), навчаються окремо з використанням одного і того ж навчального набору (навчального набору рівня 0). Після цього зібрані виходи компонентних класифікаторів утворюють навчальний набір рівня 1.

Огляд архітектури показано на рисунку 4.1.

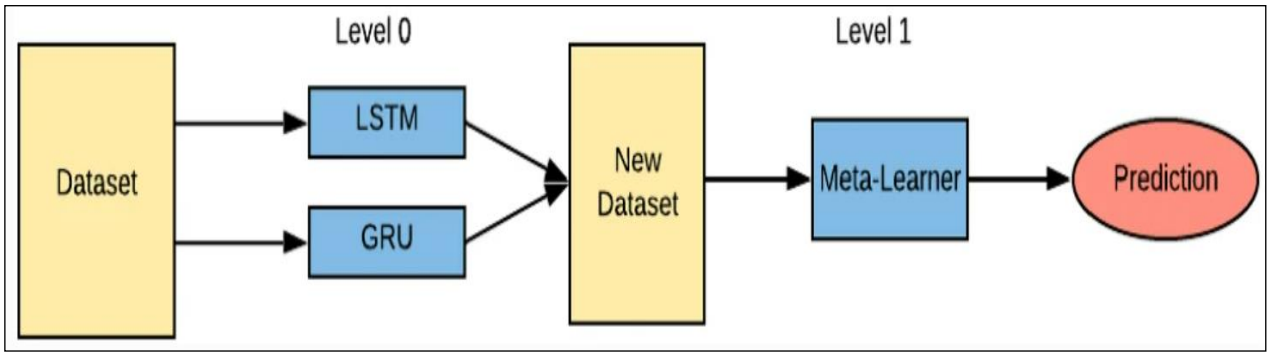

Рисунок 4.1 – Огляд архітекури

Було протестовано багато різних комбінацій конфігурацій під час наших експериментів. На основі емпіричних досвідів було обрано параметри, такі як розмір епохи, кількість нейронів та кількість шарів.

Модель зіставленого ансамблю має два рівні. Перший рівень містить дві RNN-моделі: підмодель 1 – це LSTM, а підмодель 2 – це модель GRU. Набір даних розділено на три частини: навчальні дані, дані для перевірки та тестові. Кожен набір даних необхідний для навчання моделі зіставленого ансамблю. Навчальні дані використовуються для навчання підмоделей рівня 1: моделі LSTM та моделі GRU. Після тренування на першому рівні використовуються навчені підмоделі рівня 1, щоб зробити передбачення на даних для перевірки, які фактично є даними для навчання моделі на рівні 2. А тестові дані використовуються для утворення кінцевого прогнозу та розрахунку точності.

Спочатку використовуємо тренувальні дані, щоб навчити підмодель 1: модель LSTM. Ця модель LSTM має всього чотири шари, кожен з яких містить 50 нейронів. Ми додаємо dropout рівний 0,2 для кожного прихованого шару та тренуємо модель зі 100 епохами. Після того, як ми навчили модель LSTM, ми вводимо в неї набір даних для перевірки (validation dataset), щоб зробити першу

передбачення з набору даних для перевірки. Перше передбачення, яке зробила модель LSTM, використовуючи дані для перевірки, називається перевірочними передбаченнями (LSTM validation predictions).

Другий етап полягає у тренуванні другої підмоделі – моделі GRU. Модель GRU також містить чотири шари, кожен з яких містить 50 нейронів. Для кожного прихованого шару ми також застосовуємо 0,2 dropout і тренуємо модель протягом 100 епох. Тренувальний процес для моделі GRU такий самий, як і для LSTM. Ми вводимо тренувальний набір даних у GRU, а після отримання навченої моделі GRU вводимо набір даних для перевірки, щоб отримати GRU-прогнози для перевірки.

Після того, як отримано прогнози валідації LSTM та GRU, ми комбінуємо їх в новий набір даних для тренування другого рівня, що складається з p×m елементів (p – кількість прогнозів, m – кількість моделей). Ці дані будуть передані на другий рівень для тренування мета-навчального алгоритму, який також називається другорівневою моделлю. Мета-навчальний алгоритм – це повністю зв'язана нейронна мережа з трьома шарами, а функція активації для цієї моделі – це функція ReLU. Після тренування мета-навчальної моделі, тестовий набір даних буде введений у підмоделі знову, щоб отримати проміжні дані для мета-навчальної моделі. Потім мета-навчальна модель використовує проміжні прогнози тесту від підмоделей, щоб зробити остаточний прогноз.

4.3 Навчання моделі

Щоб порівняти ефективність отриманого в рамках експерименту ансамблю з раніше дослідженими алгоритмами, навчання та тестування проводитиметься на тому самому наборі даних, що було використано під час дослідження попередніх моделей.

Розвідувальний аналіз даних (EDA) – це процес огляду та візуалізації даних для кращого розуміння їх властивостей, закономірностей та зв'язків. Він включає в себе використання статистичних та графічних методів для підсумовування та

дослідження основних функцій набору даних, таких як їх розподіл, змінність та кореляція між змінними. EDA є важливим кроком у аналізі даних, оскільки він допомагає виявити будь-які проблеми з якістю даних, аномалії або тенденції, які можуть впливати на аналіз та інтерпретацію даних [11].

Ми хочемо оцінити деякі параметри наших даних, оскільки це може бути корисним у подальшому проектуванні моделі. Перша важлива річ при прогнозуванні часових рядів – перевірити, чи є дані стаціонарними. Це означає, що наші дані піддаються впливу таких факторів, як тенденція або сезонність.

У наступному прикладі на рисунку 4.2 ми об'єднуємо дані тренувального та тестового наборів, щоб проводити аналіз та перетворення одночасно.

In [61]:
```
working_data = [df_train, df_test]
working_data = pd.concat(working_data)

working_data = working_data.reset_index()
working_data['date'] = pd.to_datetime(working_data['date'])
working_data = working_data.set_index('date')
```

Рисунок 4.2 – Розділення тренувальних та тестових даних

У наступних кількох комірках ми проводимо сезонний розбір даних для оцінки їх тенденції та сезонності. Далі необхідно провести аналіз автокореляції. Це вимірювання подібності між спостереженнями в залежності від часового затримки між ними. Даний етап важливий для виявлення зразків у даних, що повторюються. Результат автокореляції та часткової кореляції можна побачити на рисунках 4.3 та 4.4 відповідно.

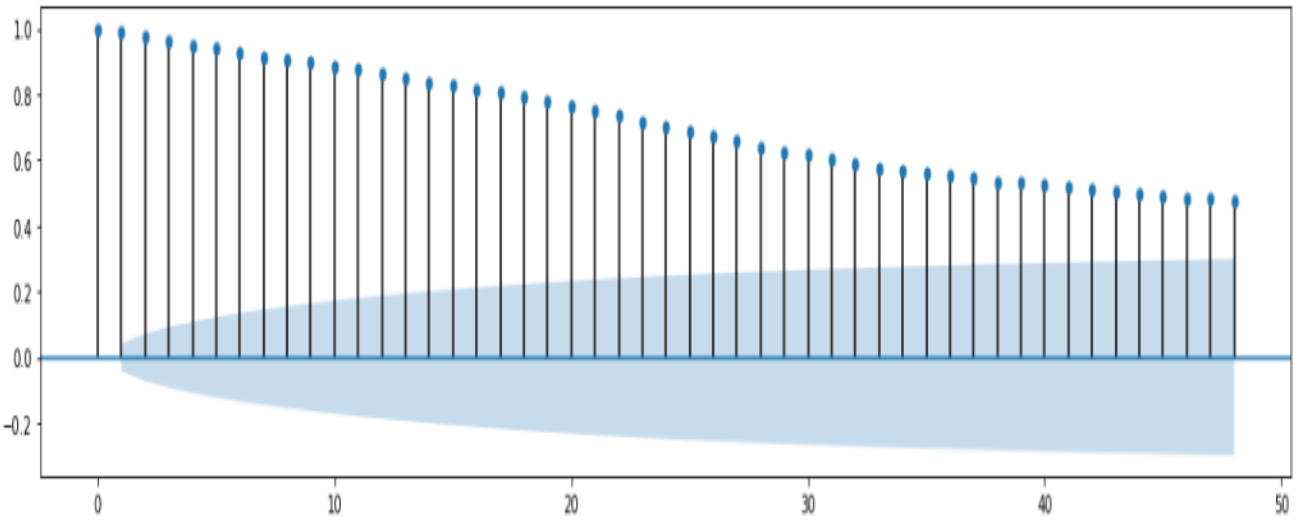

Рисунок 4.3 – Результат виконання автокореляції

Часткова автокореляція просто є частковою кореляцією часового ряду в двох різних моментах часу. Підвищуючи на рівень вище, це кореляція між часовим рядом в двох різних затримках без урахування ефекту будь-яких проміжних затримок. Наприклад, часткова автокореляція для затримки 2 – це лише кореляція, яку не пояснює затримка 1.

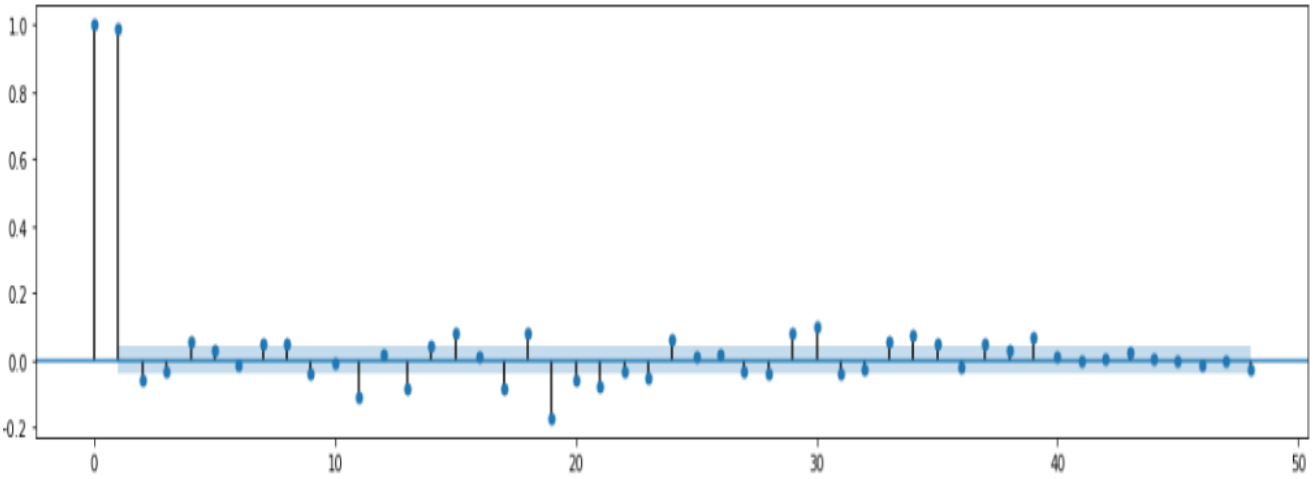

Рисунок 4.4 – Результат виконання часткової автокореляції

Після проведення підготовки даних переходимо до етапу тренування отриманої дворівневої нейронної мережі.

Для тренування моделі використовуватимемо фреймворк Keras для глибокого навчання. Дана модель складається з двох шарів (перший шар – LSTM, а другий шар – GRU), кожен з яких має 256 одиниць, та плотно зв'язаного вихідного

шару з однією нейронною одиницею. Також будемо використовувати оптимізатор Adam та MSE як функцію втрат. Крім того, якщо результат не покращується протягом 20 ітерацій навчання (епох), то необхідно задіяти підхід раннього зупинення навчання [12].

Після проведення кількох експериментів було встановлено, що оптимальні значення кількості епох та розміру пакету (batch_size) дорівнюють відповідно 100 та 16. Варто зазначити, що важливим є встановлення параметру shuffle=False, для того щоб не перемішувати дані часового ряду.

Код для ініціалізації моделі з визначеними вище значеннями параметрів можна побачити на рисунку 4.5.

```
# initialize sequential model, add 2 stacked LSTM layers and densely connected output neuron
model = Sequential()
model.add(LSTM(256, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(LSTM(256))
model.add(Dense(1))

# compile and fit the model
model.compile(loss='mean_squared_error', optimizer='adam')
history = model.fit(X_train, Y_train, epochs=100, batch_size=16, shuffle=False,
                    validation_data=(X_test, Y_test),
                    callbacks = [EarlyStopping(monitor='val_loss', min_delta=5e-5, patience=20, verbose=1)])
```

Рисунок 4.5 – Побудова моделі

Після навчання моделі можемо побачити, що вона має хорошу продуктивність навіть після кількох ітерацій. На рисунку 4.6 зображено фрагмент результату виконання функції втрат, яка виконується для тренувального та тестового наборів даних з метою порівняння даних на кожній ітерації процесу навчання. Очевидно, що після деяких ітерацій значення втрат на тренувальному та тестовому наборах даних стають дуже схожими, що є добрим знаком (це означає, що ми не перенавчаємося на тренувальному наборі даних). Далі ми використовуємо нашу модель для передбачення міток для тестового набору даних. Потім ми зворотньо масштабуємо наші дані до початкового масштабу.

```
Train on 2137 samples, validate on 59 samples
Epoch 1/100
2137/2137 [==============================] - 5s 2ms/step - loss: 9.7324e-04 - val_loss: 0.2857
Epoch 2/100
2137/2137 [==============================] - 4s 2ms/step - loss: 0.0202 - val_loss: 0.3650
Epoch 3/100
2137/2137 [==============================] - 3s 2ms/step - loss: 0.0061 - val_loss: 0.0493
Epoch 4/100
2137/2137 [==============================] - 3s 2ms/step - loss: 1.3302e-04 - val_loss: 0.0259
Epoch 5/100
2137/2137 [==============================] - 3s 2ms/step - loss: 2.9849e-04 - val_loss: 0.0269
Epoch 6/100
2137/2137 [==============================] - 3s 2ms/step - loss: 8.5363e-05 - val_loss: 0.0174
Epoch 7/100
2137/2137 [==============================] - 3s 2ms/step - loss: 1.8244e-04 - val_loss: 0.0141
Epoch 8/100
2137/2137 [==============================] - 3s 2ms/step - loss: 1.0668e-04 - val_loss: 0.0114
Epoch 9/100
2137/2137 [==============================] - 3s 2ms/step - loss: 1.1613e-04 - val_loss: 0.0105
Epoch 10/100
```

Рисунок 4.6 – Фрагмент результату виконання обчислення значень функцією втрат для тренувального та тестового набору даних

Нижче на рисунку 4.7 наведено код для обчислення середньоквадратичної помилки (RMSE) та отриманий результат обчислення. Значення цього показника показує те, якою є середня відстань між передбаченими точками на тестовому наборі та фактичними (істинними) мітками. Іншими словами, це показує ступінь помилки. Чим менше це число, тим краще. За результатами обчислень бачимо, що середньоквадратична помилка нашої моделі не дуже велика (зважаючи на те, що ціна в нашому наборі даних виражена в тисячах доларів США, ми помиляємося лише на десятки доларів США).

```
In [72]: RMSE = sqrt(mean_squared_error(Y_test2_inverse, prediction2_inverse))
         print('Test RMSE: %.3f' % RMSE)

Test RMSE: 688.899
```

Рисунок 4.7 – Обчислення RMSE

4.4 Порівняння результатів з класичними моделями

В цьому розділі проведемо порівняння розробленого ансамблю моделей з класичними підходами, описаними в попередньому розділі. Теоретично ансамбль моделей може мати переваги, оскільки він поєднує декілька моделей, що працюють разом для досягнення кращої точності передбачень. Кожна модель може мати свої власні слабкі сторони, і комбінування результатів може зменшити вплив недоліків окремих моделей та забезпечити більш точні передбачення.

В таблиці 4.1 наведено порівняння розробленого ансамблю (в таблиці під назвою LSTM-GRU) з класичними моделями, які показали найкращі результати під час експериментальних досліджень. Результати не впорядковані.

Таблиця 4.1 – Порівняння результатів

Модель	RMSE	MAE	MAPE	R^2	Тренування (с)	Інференція (мс)
LSTM	0.02224	0.0173	3.862%	0.735	173.765	1.862
GRU	0.02285	0.0176	3.939%	0.720	254.520	1.550
LSTM-GRU	0.02223	0.0171	3.890%	0.740	289.177	1.493

Можемо побачити, що порівняння результатів ефективності класичних підходів та ансамблю моделей на основі LSTM та GRU підтверджує переваги використання ансамблю моделей в задачах прогнозування часових рядів.

Класичні підходи до прогнозування часових рядів, такі як ARIMA, базуються на статистичних методах і мають свої обмеження. Ці підходи недостатньо ураховують складні динамічні залежності та нелінійність в даних, що зазвичай притаманні цифровим активам. Тому їх точність прогнозування може бути обмеженою.

У порівнянні з класичними підходами, ансамбль моделей на основі LSTM та GRU виявляється більш ефективним. Здебільшого це зумовлено тим, що LSTM та

GRU є типами рекурентних нейронних мереж, які добре працюють з послідовними даними, такими як часові ряди.

Основні переваги використання ансамблю моделей на основі LSTM та GRU включають:

– моделювання складних залежностей: LSTM та GRU мають здатність до захоплення довготривалих залежностей та нелінійних взаємозв'язків у часових рядах. Це особливо корисно в задачах, де цифрові активи можуть мати складні та непередбачувані динамічні зміни;

– здатність до виявлення контексту: LSTM та GRU здатні зберігати та використовувати корисну інформацію про попередні стани в часовому ряді. Це дозволяє їм краще розуміти контекст та динаміку даних, що сприяє точнішому прогнозуванню цінотворення цифрових активів;

– стійкість до шуму та аномалій: Ансамбль моделей на основі LSTM та GRU може бути більш стійким до шуму та аномалій у часових рядах. Через свою здатність до моделювання складних залежностей та використання попереднього контексту, вони можуть здатися кращими у виявленні та урахуванні непередбачуваних змін у даних, що допомагає покращити точність передбачень;

– узгодженість з ансамбльним підходом: Використання ансамблю моделей, зокрема на основі LSTM та GRU, дозволяє комбінувати прогнози різних моделей для отримання кращої універсальності та точності. Кожна модель може мати свої переваги та обмеження, але їх комбінація в ансамблі може забезпечити кращу здатність передбачення.

Незважаючи на те, що класичні підходи можуть бути простішими та швидшими у виконанні, ансамбль моделей на основі LSTM та GRU показує переваги у точності та стійких прогнозів. З їхньою здатністю до моделювання складних залежностей та використання контексту, ансамбль моделей може бути кращим в прогнозуванні кривих цінотворення цифрових активів.

4.5 Розробка програмної системи

Під час проведення досліджень, особливо у сфері машинного навчання та обробки даних, важливим етапом є демонстрація та візуалізація отриманих результатів. Для графічного відображення прогнозування та аналітики кривих ціноутворення цифрових додатків було вирішено використовувати десктопний додаток під операційну систему Windows, а саме UWP застосунок. Це рішення було зумовлено зручністю використання та можливістю інтеграції з фреймворком Keras, який було використано під час глибокого навчання моделей.

Універсальна платформа Windows (Universal Windows Platform, UWP) є однією з найпопулярніших платформ для розробки додатків під Windows. UWP дозволяє створювати додатки, які працюють на різних пристроях, таких як ПК, планшети, смартфони, Xbox, HoloLens та інші. Один з важливих аспектів розробки UWP додатків – це їхній інтерфейс, який може бути адаптований до різних розмірів екранів та пристроїв.

Інтеграція Keras, популярної бібліотеки для глибокого навчання, в UWP додаток дозволяє поєднати потужність нейронних мереж зі зручним інтерфейсом UWP. Keras надає високорівневий інтерфейс для побудови та навчання моделей глибокого навчання, що спрощує розробку складних алгоритмів ML.

Розроблений додаток надає користувачам зручний інтерфейс для перегляду інформації про обрану криптовалюту. Користувачі можуть швидко отримати доступ до основних даних, таких як поточна вартість криптовалюти, обсяг торгів, зміна вартості за останні 24 години та інші важливі показники.

Однак, функціональність додатка не обмежується лише наданням інформації. Він також пропонує графічну візуалізацію динаміки змін вартості обраної криптовалюти за обраний проміжок часу. Користувачі можуть вибрати потрібний період (наприклад, останню годину, день, тиждень або місяць) і переглянути графік, який ілюструє коливання ціни активу протягом цього періоду.

Основною функцією розробленого додатку є також перегляд прогнозованих змін в цінах криптовалют. Ці прогнози розраховуються за допомогою розроблених нейромереж, які були описані раніше.

На рисунку 4.8 наведено загальний вигляд інтерфейсу.

Рисунок 4.8 – Вигляд головної сторінки додатку

Для того, щоб побачити прогнозовану динаміку змін стосовно конкретної обраної валюти, необхідно з головної сторінки додатку перейти на сторінку цієї криптовалюти. На рисунку 4.9 зображено графік прогнозування ціни Bitcoin на останній тиждень грудня 2022 року.

Також користувачам надається можливість імпортувати своє портфоліо, щоб спостерігати за станом свого портфелю. Також застосунок пропонує перегляд головних новин зі світу цифрових активів, таким чином користувач матиме можливість ознайомитись з поточною ситуацію на ринку та зробити висновки стосовно купівлі або продажу певних криптовалют.

Для зручності користування інтерфейсом пропонується пошукова система (щоб швидше знаходити потрібні новини або монети), а також в налаштуваннях користувач має можливість встановити такі параметри як валюта (фіатні гроші, в яких відображатиметься вартість криптовалют), час автоматичного оновлення

сторінок, а також параметри адаптування інтерфейсу згідно вподобань користувача (основний колір додатку, тема тощо).

Рисунок 4.9 – Прогнозована ціна Біткоїну

Таким чином, користувачі додатку мають можливість переглядати отримані прогнози і використовувати їх як додатковий інструмент для прийняття рішень щодо купівлі, продажу або тримання криптовалют. Прогнози можуть бути представлені у вигляді графіків або числових значень, що дозволяє користувачам оцінити ймовірність певних змін і адаптувати свої інвестиційні стратегії відповідно. Враховуючи прогнози, надані розробленими нейромережами, користувачі можуть бути більш освіченими і усвідомленими при прийнятті рішень на ринку криптовалют.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було проведено аналіз предметної області, визначено найпоширеніші моделі нейронних мереж, які використовуються для прогнозування, та проведено експериментальне дослідження з метою аналітики кривих ціноутворення цифрових активів на однаковому наборі даних.

У даній кваліфікаційній роботі порівнюються моделі глибокого навчання (DL), машинного навчання (ML) та статистичні моделі для прогнозування щоденних цін на криптовалюти. Використана методика оцінки передбачень на крок наперед є інкрементальною та працює за графіком щомісячної перетренування. Ми протестували понад 12 місяців даних. Результати показують, що в цілому рекурентні DL підходи є найкращими моделями для цієї задачі. Зокрема, LSTM є найкращою моделлю з найменшими витратами на навчання серед інших DL моделей з найближчою до нього продуктивністю. KNN та ARIMA забезпечують гарний баланс між точністю та витратами на обчислення.

Остаточна частина експерименту підкреслює, що комбінування різних регресорів в ансамбль може підвищити продуктивність, але також сильно підвищує часові витрати. Цей підхід спрямований на компенсацію недоліків моделі, шляхом усереднення її з іншими, які є більш точними у певних випадках. Однак, якщо регресор надає більш точні прогнози у великій більшості випадків, то усереднювання його з значно менш точними моделями негативно впливає на його продуктивність. Тому стратегія навчання ансамблю моделей, що складається з найефективніших класичних моделей, дає незначну перевагу .

В результаті виконання кваліфікаційної роботи також було розроблено програмну систему – десктопний застосунок для демонстрації результатів прогнозування. Таким чином, користувачі розробленої системи матимуть змогу приймати рішення щодо змін в своєму портфелі згідно з прогнозами, отриманими в результаті навчання моделей.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Chuen, D. L. K., Wang, Y., & Huang, W. "Forecasting cryptocurrency time-series prices using a novel ensemble model combining deep learning with traditional methods." Applied Soft Computing, 107389, 2021.

2. Antonopoulos, A. M. (2014). Mastering Bitcoin: Unlocking Digital Cryptocurrencies. O'Reilly Media, Inc.

3. Багнюк В. Якою буде наша енергетична стратегія? // Вісн. НАН України. - 2019. - № 9. - С. 29 – 37.

4. Галак О.В., Козирєв А.Д., Орлов Я.В., Шубін І.Ю. Інформаційна технологія визначення зон ураження під час надзвичайних ситуацій. // Інформаційні технології: наука, техніка, технологія, освіта, здоров’я: тези доповідей XXVII міжнародної науково-практичної конференції MicroCAD-2019, 15–17 травня 2019р.: у 5 ч. Ч.V. / за ред. проф. Сокола Є.І. –Харків: НТУ “ХПІ”. – 158с.

5. І.Ю. Шубін, А.Д. Козирєв, О.В. Галак. Методи створення інтелектуальної системи екологічного моніторингу та аналізу побудови складних границь територій. // Науково-технічний журнал «Біоніка інтелекту» ХНУРЕ, 2020

6. Vigna, P., & Casey, M. J. (2015). The Age of Cryptocurrency: How Bitcoin and Digital Money are Challenging the Global Economic Order. St. Martin's Press.

7. Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction. Princeton University Press.

8. Kim, T. H., & Kim, H. K. (2018). Predicting bitcoin price fluctuation with twitter sentiment analysis. Journal of Information Processing Systems, 14(1), 199-207.

9. Kuo, T. C., & Lin, Y. H. (2019). An empirical study of bitcoin price volatility. Journal of Risk Finance, 20(1), 39-49.

10. Mokhtarimehr, M., & Ghorbani, A. "A comparison of time series forecasting methods for cryptocurrencies prices." Applied Economics Letters, 27(19), 1585-1592, 2020.

11. Skovgaard, J., Da Silva, R. B., & Li, X. "Forecasting cryptocurrency prices with deep learning." arXiv preprint arXiv:1906.06524, 2019.

12. Zhang, X., Qi, Y., & Li, B. "A comparative study of LSTM and ARIMA for time series prediction on stock market data." Journal of Physics: Conference Series, 1168(1), 012065, 2019.

13. Zhao, Y., Zhang, Y., & Qi, Y. "Hybrid model for forecasting cryptocurrency price with attention-based LSTM and EEMD." IEEE Access, 8, 235335-235344, 2020.

14. Teorey T., Lightstone S., Nadeau T. Database Modeling and Design. – Elsevier, 2006. – 296 P. – ISBN 978-0-12-685352-0.

15. Witten D., James G., Hastie T. An introduction to Statistical Learning. Springer, 2013. – 357 P. – ISBN 978-1-4614-7138-7. 10. Teorey T.

16. Гиренко П.І., Писклакова В.П., Халявин В.А. Створення розподіленої бази даних територіальної інформаційно-аналітичної системи надзвичайних ситуацій // АСУ й прилади автоматики. - 2002. - Вип. 120. - С. 10 – 14.

17. Клименко Е.Г. Програмно-алгоритмічні засоби інтелектуального аналізу даних // Радіоелектроніка й інформатика. - 2001. - № 3. - С. 64-67.

18. Уваров Р.А. Моделювання екологічної обстановки з урахуванням турбулентного руху в атмосфері // Радіоелектроніка й інформатика. - 2001. - № 3. - С. 129 – 134.

19 Ejchert J., Haumann D. Animation Aerodynamics // ACM SIGGRAPH Computer Graphics. 1991. 25. N 4. P. 19–22.

20 Bert D., Musgrave K., Peachy D., Perlin K., Worley S. Texturing and Modeling: A Procedural Approach. Orlando, Florida: AP Professional, 1994. 350 p.

21 Aeger L., Upson C. Combining Physical and Visual Simulation. Creation of the Planet Jupiter for the Film 2010 // ACM SIGGRAPH Computer Graphics. 1986. 20. N 4. P. 85–93.

22 Grass M., Miller G. Rapid, Stable Fluid Dynamics for Computer Graphics // ACM SIGGRAPH Computer Graphics. 2018. 24. N 4. P. 49–57.

23 Curtis C., Anderson S., Seims J., Fleischer K., Salesin D. Computer-Generated Watercolor // Proc. of the 24th annual conference on Computer graphics and interactive techniques. 2017. P. 421–430.

24 Tam J. Stable Fluids // Proc. of the 26th annual conference on Computer graphics and interactive techniques. 2019. P. 121–128.

25 Tam J, A Simple Fluid Solver based on the FFT // Journal of Graphics Tools. 2011. 6. N 2. P. 43–52.

26 Selle A., Fedkiw R., Kim B., Liu Y., Rossignac J. An Unconditionally Stable MacCormack Method // Journal of Scientific Computing. 2018. 35. N 2-3. P. 350–371.

27 Tam J. Real-Time Fluid Dynamics for Games // Proc. of the Game Developer Conference. 2003 URL: http://www.dgp.toronto.edu/people/stam/reality/Research/pdf/GDC03.pdf.

28 Rane K., Llamas I., Tariq S. Real-Time Simulation and Rendering of 3D Fluids // GPU Gems N 3. Massachusetts: Addison–Wesley, 2017. P. 633–675.

29 Erlin K. Improving Noise // ACM Transactions on Graphics. 2020. 21. N 3. P. 681–682.

30 Cerlin K. An Image Synthesizer // ACM SIGGRAPH Computer Graphics. 2015. 19. N 3. P. 287–296.

31 Cerlin K. Making Noise // Proc. of the Game Developer Conference. 2019 URL: http://www.noisemachine.com/talk1.

32 Cerlin K. Course in Advanced Image Synthesis // ACM SIGGRAPH Conference. 2020. 18. N 3.

33 Green S. Implementing Improved Perlin Noise // GPU Gems N 2. Massachusetts: Addison–Wesley, 2015. P. 409–416.

34 Chetverikov G., Puzik O., Vechirska I. Multiple-valued structures of intellectual systems //Proceedings of the with Internations Computer Sciences and Information Technologies (CSIT). 2016, 7589907. -pp. 204-207