КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

Кваліфікаційна робота

на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

АНАЛІЗ ТА РОЗРОБКА МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ ВИРІШЕННЯ ЗАДАЧ ФІНАНСОВОЇ МАТЕМАТИКИ

Виконав студент 4-го курсу
Кравченко Ярослав Сергійович

(підпис)

Науковий керівник:
асистент
Бобиль Богдан Володимирович

(підпис)

Консультант:
доцент кафедри математичної інформатики,
кандидат фіз.-мат. наук
Деревянченко Олександр Валерійович

(підпис)

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

Студент

(підпис)

Роботу розглянуто й допущено до захисту
на засіданні кафедри математичної інформатики
«___» _____ 2022 р.,
протокол № ___
Завідувач кафедри
М.В. Терещенко _____

(підпис)

Київ - 2022

РЕФЕРАТ

Обсяг роботи 48 сторінок, 16 ілюстрацій, 5 таблиць, 12 джерел посилання.

МАШИННЕ НАВЧАННЯ, ДАТАСЕТ, ЧАСОВІ РЯДИ, АНАЛІЗ ФІНАНСОВИХ ДАНИХ, МОДЕЛЬ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, ГЛИБОКЕ МАШИННЕ НАВЧАННЯ.

Об’єктом роботи є створення, аналіз та порівняння моделей машинного навчання для аналізу фінансових даних. Предметом роботи є рекурентні нейронні мережі декількох видів для аналізу часових рядів.

Метою роботи є оцінка точності предикативних моделей глибокого машинного навчання для передбачення ціни криптовалюти в доларовому еквіваленті у короткостроковій перспективі.

Методи розроблення: взаємодія з хмарними осередками зберігання даних, програмні бібліотеки для запису, вилучення та створення таких осередків, побудова гібридних моделей машинного навчання. Інструменти розроблення: python3, Azure Databricks IDE, Databricks File System, Colab, SKlearn, Tensorflow Keras, NBeats, AlphaRNN.

Результати роботи: виконано загальний огляд засобів створення аналітичних інструментів для обробки часових рядів, проаналізовані переваги та недоліки використання обраних технологій, визначені найбільш результативні моделі для вирішення поставленої задачі.

Результати роботи мають безпосереднє практичне значення, адже створені моделі машинного навчання можуть бути застосованими не тільки у випадку передбачення спекулятивних тенденцій для крипто валюти, а й для фондових ринків з двома рівнями фінансових даних.

Це дає підстави для проведення подальших досліджень іншими прогресивними методами аналізу фінансових активів з метою ведення автоматизованої торгівлі. Тобто безпосередня прикладна задача, яку вирішують мільйони людей кожного дня зводиться до такої, що може бути вирішена гібридними методами ML. Дійсно, результати даної роботі слід сприймати як основу для порівняння з RL та Inverse-RL методами. Вони часто використовуються в трейдингових ботах, які вчаться на своїх помилках в реальному часі. Щоб перевірити ефективність такого інструменту, не обов'язково відкривати рахунок в іноземному банку та ризикувати власними коштами. Замість цього, онлайн-брокери часто пропонують відкрити пробний рахунок, щоб дати можливість початківцям розібратися основами торгівлями фінансовими та криптовалютними активами. Також такі сервіси мають API для фінтех-розробників та аналітиків.

Сфера застосування технологій, розглянутих в цьому досліджені, не обмежується фінансовою математикою. Потенційними галузями є енергетика, біологія, логістика тощо. В молекулярній біології, наприклад, розповсюдженою задачею є побудова тривимірних моделей білків за заданими параметрами. Науковці зазначають, що лабораторні дослідження в даній області є кропіткими та коштовними і займають багато часу. Але комп'ютерне моделювання з елементами AI допомагає успішно вирішити цю задачу за порівняно невеликий проміжок часу.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

AI – Artificial Intelligence, Штучний інтелект;

API – інтерфейс прикладного програмування;

ML – Machine Learning, Машинне навчання;

NN – Neural Networks, Нейронні мережі;

CNN – Convolutional Neural Networks, Згорткові нейронні мережі;

RNN – Recurrent Neural Networks, Рекурентні нейронні мережі;

LSTM – Long Short Term Memory;

DBFS – Databricks File System;

MA – Moving Average, Рухоме середнє

ETS – Exponential Smoothing, Експоненційне згладжування

ARMA – Auto regression and Moving Average,

MAPE – Mean absolute percentage error, Середня абсолютна похибка;

DNN – Dense(Deep) Neural Network, Глибоке навчання;

SaaS – Software as Service, Програмне забезпечення як сервіс;

RL – Reinforcement Learning, Навчання з підкріпленням

QL – Q-Learning

SL – Supervised Learning

ЧР – Часовий ряд

ВСТУП

Оцінка сучасного стану об’єкта розробки. Рекурентні нейронні мережі(RNNs) за останні 10 років довели свою конкурентоспроможність у багатьох сферах машинного навчання. Широкий спектр застосування даної технології робить її очевидним вибором для таких задач як розпізнавання образів, комп’ютерна лінгвістика, передбачення послідовностей тощо. Основними конкурентами RNN для аналізу часових рядів є класичні методи статистичного аналізу, серед них: AR, MA, ETS, ARIMA та її варіації. Не можна стверджувати, що гібридні методи машинного навчання усунули потребу у використанні класичних моделей, оскільки вони мають свої переваги. Дійсно, часова складова часто грає важливу роль у розробці тих чи інших моделей, особливо у тих випадках, коли необхідно визначити саме тренд розвитку часового ряду, а точність не має великого значення. Тоді ETS ідеально підходить для вирішення задачі. Якщо ж часовий ряд має чітку сезонну складову, яку легко помітити на графіку декомпозиції, то має сенс застосувати модель із сімейства ARIMA. Яскравим прикладом датасету, для якого ARIMA робить доволі точні передбачення – це AirPassengers.

Тим не менш, часові ряди часто містять нечіткі тренди, які важко пояснити агрегуючими моделями. Оскільки вони використовують вхідні вікна обмеженої довжини(наївні вікна), то пам’ять таких моделей наповнюється недалекосяжними знаннями про дані. Саме RNN інкапсулює ідею обміну інформації всередині одного шару нейронів утворюючи допоміжні послідовності всередині однієї моделі. Така модифікація класичних NN значно покращує якість передбачення, хоча й вимагає більших обчислювальних можливостей для побудови якісних моделей.

Актуальність роботи та підстави для її виконання. Історія фондових ринків бере свій початок ще у 1300 році у Венеції, а в 17 столітті торгівля цінним паперами стала всесвітньо розповсюдженим явищем у зв'язку з діяльністю Ост-Індійських компаній. Але справжня революція в трейдингу відбулася в 1971 році, коли Нью-Йоркська фондова біржа(New-York Stock Exchange, NYSE) запустила електронну систему купівлі та продажу активів. Ця подія ініціювала створення окремого напрямку цифрових технологій – фінансова кібернетика. В той час як заможні компанії на Вол Стріт(Wall Street) займалися високочастотним трейдингом(High-frequency trading, HFT), науковці теоретичного спрямування, а саме математичного та статистичного, отримали можливість аналізувати історичні дані майже будь-якого активу. За понад 50 років існування NYSE, методи обробки інформації зазнали неймовірного розвитку, кількість доступних для аналізу даних експоненційно зросла, потужні обчислювальні можливості стали доступними кожному.

Зрозуміло, що нещодавні досягнення RL-технологій дозволяють моделювати поведінку віртуального агента, який вчиться приймати правильні рішення для досягнення певної мети і такий підхід приносить високі результати. Але він також має серйозний недоліки: довгий час навчання, потреба в додатковій ринковій інформації, архітектурна складність RL-моделей. RNNs, на противагу, мають простішу структуру та дозволяють робити точні передбачення у короткостроковій перспективі[1]. В такому випадку користувачу потрібно самостійно робити висновки щодо операцій з активом, тим не менш, такий підхід є актуальним.

Мета й завдання роботи. Метою дипломної роботи є аналіз, розробка, та тестування RNN моделей для вирішення розповсюдженої задачі фінансової математики – короткострокове передбачення розвитку ціни

обраної криптовалюти у доларовому еквіваленті. Для досягнення мети поставлено такі завдання:

- Дослідити й описати класичні методи аналізу часових рядів
- Визначити особливості архітектури RNN та її підвидів
- Спроектувати та натренувати необхідні моделі
- Проаналізувати результати роботи моделей

Об’єкт, методи й засоби розроблення. Об’єктом розроблення є сімейство рекурентних нейронних мереж для аналізу часових рядів. Під час проведення дослідження використана каскадна модель, заснована на таких принципах:

a) формуються вимоги до кожного етапу роботи з моделями, встановлюються терміни виконання та узгоджується формат вхідних даних;
b) результати тестування кожної моделі зберігаються для подальшого порівняння;
c) кінцевим етапом проведення дослідження є аналіз отриманих результатів та формування пропозицій щодо подальшої роботи

Можливі сфери застосування. Розглянуті RNN-моделі у даній роботі застосовуються для аналізу фінансових активів. За таким самим принципом відбувається моделювання енергетичних та екологічних процесів. Зрозуміло, що ідея передбачення майбутніх спостережень, відштовхуючись від історичних даних не завжди є доцільною, але якщо потрібно зробити точне короткострокове передбачення деякого об’єкта з чітким трендом, то результат, як правило, є дуже високим.

Розділ 1. Огляд конкуруючих систем

1.1 Рухоме середнє(MA)

Хоча метод MA не може бути ефективно застосованим до вирішення поставленої задачі, його варто розглянути як важливу складову загального процесу аналізу часових рядів. Ідея методу полягає у заміщенні простим або зваженим середнім $n = 2k + 1$ сусідніх членів, де n – ширина “вікна”.

$$Y_t = \frac{1}{2k+1} \sum_{j=-k}^{k} X_{t+j}$$

З формули видно, що неперервна множина точок ряду замінюється середнім значенням «вікна». Така модель скоріше описує тренд спостережуваних даних, аніж надає можливість робити точні передбачення. Тим не менш, вона широко використовується у складніших моделях[5].

1.2 Експоненційне згладжування(EST)

Варто зазначити, що метод EST, або модель Брауна, застосовують для часових рядів вигляду

$$X_t = b + \epsilon_t,$$

де b – деяка константа, ϵ_t – випадкова похибка.

Основною відмінністю EST від MA є кількість врахованих спостережень ряду, тобто, EST враховує всі спостереження, а MA тільки обмежену кількість. Тоді формула відповідного лінійного фільтра матиме вигляд

$$X_t^* = \alpha X_t + (1 - \alpha) X_{t-1}^*, t \geq 1$$

з початковим значенням $X_0^* = X_0$ і параметром α, який визначає гладкість відфільтрованого ряду.

1.3 Модель авторегресії інтегрованого рухомого середнього

Оскільки методи експоненційного згладжування не беруть до уваги кореляцію спостережень часового ряду, то їхні передбачення можна покращити, якщо припустити, що кореляція має місце. ARIMA інкапсулює два головних процеси: MA та AR. Потрібно зазначити, що моделі класу ARIMA визначені для стаціонарних часових рядів. Модель ARMA(p,q) має такий загальний вигляд:

$$X_t = Z_t + \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \theta_1 Z_{t-1} + \cdots + \theta_q Z_{t-q},$$

де $\phi_1 \ldots \phi_p$ –AR-компоненти, $\theta_1 \ldots \theta_q$ – MA-компоненти.

Якщо ряд X_t – нестаціонарний, то його потрібно диференціювати і звести до стаціонарного:

$$Y_t = \Delta^d X_t.$$

Якщо ряд Y_t можна описати моделлю ARMA(p,q), то початковий ряд характеризується моделлю ARIMA(p,d,q). Якщо ж часовий ряд містить сезонну компоненту, то до моделі ARIMA(p,d,q) додаються P авторегресійних компонент та Q компонент рухомого середнього, причому P і Q – сезонні параметри, тоді як p і q – несезонні параметри. Отже, маємо SARIMA(p,d,q)(P,D,Q)S, де S – період сезонної компоненти.

Точність передбачення можна покращити, якщо взяти до уваги зовнішні фактори, які впливають на спостережувану змінну[1]. Екзогенні змінні не мають прямого відношення до моделі, тому потрібно збирати дані з

декількох джерел як, наприклад, у випадку передбачення кількості спожитої енергії деякої електростанції. Так, температура має значний вплив на цільову змінну, тому метеоспостереження у відповідному регіоні, такі як температура, тиск, вологість тощо, дозволять зробити більш точний прогноз.

1.4 Prophet

Незважаючи на те, що моделі сімейства ARIMA можуть включати сезонні коваріати та екзогенні змінні, розробник-початківець може не мати необхідних навичок для побудови ефективного рішення. Тому розробники компанії Facebook запропонували “business-friendly” модель Prophet для передбачення часових рядів. Охарактеризуємо її рівнянням:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t.$$

Видно, що Prophet містить три складові:

1. $g(t)$ – функція, яка задає тренд, тобто неперіодичні зміни часового ряду;
2. $s(t)$ – періодичні зміни, наприклад, тижнева та щорічна сезонність;
3. $h(t)$ – позначає вплив свят, які відбуваються за нерегулярним графіком протягом одного або декількох днів;
4. ϵ_t – це випадкова похибка, яка не пояснюється моделлю; вважається, що її розподіл – нормальний.

Декомпозиція моделі на складові має декілька практичних переваг, серед них: гнучкість у підборі необхідної кількості періодів та змога робити різні припущення про тренди; можливість працювати з не рівновіддаленими

спостереженнями; дуже швидкий “підгін” моделі; її параметри легко інтерпретувати та змінити в залежності від припущення щодо передбачення.

Prophet – це регресійна модель, яка апроксимує лінійні та нелінійні функції замість того щоб шукати залежності в межах деякого вікна. В 2020 дослідники Стендфордського університету та Facebook розширили Prophet, додавши до моделі елементи DNN[7]. Точність передбачення NeuralProphet на 55 – 92% вища за Prophet на широкому наборі несимульованих датасетів.

1.5 Vector Autoregression(VAR)

VAR – це алгоритм для багатовимірного аналізу часових рядів, якщо два або більше ЧР мають взаємний вплив на один одного. Основна відмінність VAR від моделей сімейства ARIMA полягає в кількості цільових змінних – в ARIMA вона одна, а в VAR – декілька. Така відмінність відображається в рекурентному представленні VAR-моделі першого порядку - VAR(1) - для випадку двох ендогенних змінних(перше та останнє рівняння системи):

$$Y_{1,t} = \beta_1 + \beta_{11,1}Y_{1,t-1} + \beta_{12,1}Y_{2,t-1} + \cdots + \beta_{1n,1}Y_{m,t-1} + \epsilon_{1,t}$$

$$Y_{m,t} = \beta_m + \beta_{m1,1}Y_{1,t-1} + \beta_{m2,1}Y_{2,t-1} + \cdots + \beta_{mn,1}Y_{m,t-1} + \epsilon_{m,t},$$

де β_i – коефіцієнти перетину, β_{ij} – коефіцієнти перед лагами Y_i, ϵ_i – випадкова похибка.

З формул видно, що кожна змінна – це лінійна комбінація попередніх значень самої змінної та інших змінних системи. В даному випадку розглядається модель першого порядку, тобто поточне спостереження спирається тільки на попереднє. Але в реальних випадках кількість лагів потрібно підбирати за допомогою статистик Акаїке, Шварца-Байєса,

Ханнана-Квіна. Також важливим питанням є вибір ендогенних змінних, які потрібно включити в модель[6]. Тест Грангера допомагає визначити, чи має певна змінна вплив на іншу змінну системи. Варто зазначити, тест Гангера надає інформацію про міру впливу на результат передбачення моделі, а не істинну причинність. Іншими методами оцінки моделі є імпульсні перехідні функції та декомпозиція дисперсії помилки передбачення.

1.6 Transfer Learning(TL)

В статті [8] розглядається застосування TL для класифікації часових рядів. Цей підхід є популярним для вирішення задач комп'ютерного зору та комп'ютерної лінгвістики і досі не був ґрунтовно дослідженим у випадку аналізу ЧР. Так, потрібно дати відповідь на ряд запитань, а саме: чи може модель працювати з датасетами, які мають декілька ознак? Чи є обрана архітектура оптимальною як основа для подальшого файнтюнингу моделі? Чому тестування на певній множині даних дає кращі або гірші результати за базову модель? Які параметри повинні бути обрані для тренування базової моделі? Серед результатів даної наукової праці є певний успіх у застосуванні CNN для аналізу ЧР, що, власне, не є очевидним вибором для даної задачі, враховуючи послідовну структуру часових рядів. Зрозуміло, що TL потребує значних обчислювальних можливостей для тренування базової моделі, тому застосування цього підходу не є доцільним у випадку побудови короткострокових передбачень, які мають бути побудовані у короткий проміжок часу.

1.7 Temporal Fusion Transformer(TFT)

TFT – це новітня архітектура для аналізу ЧР, яка дозволяє працювати з різними типами вхідних даних, наприклад, статичні коваріати, екзогенні

змінні тощо одночасно та використовує рекурентні блоки для локальної обробки і self-attention шари для побудови довгострокових залежностей. TFT містить спеціальні компоненти для виділення релевантних особливостей, а також фільтраційні шари, щоб дискримінувати непотрібні елементи. Важливо, що результати перебігу тестів можна інтерпретувати, що дозволяє розробнику підібрати гіперпараметри моделі для покращення результату.

Розділ 2. Огляд використаних технологій

2.1 RNN

Згідно з Універсальною теоремою апроксимації, нейронна мережу прямого зв'язку з одним прихованим шаром може апроксимувати будь-яку неперервну функцію декількох змінних з довільною точністю. Але звичайні NN потребують значних обчислювальних ресурсів, пам'яті та часу для тренування на великому об'ємі даних. Тому модифікація архітектури NN, ідея якої полягає в утворенні зв'язків між нейронами одного шару[5], додає ефективності до класичної моделі та значно розширює її можливості.

RNN – це нелінійна модель ЧР, яка узагальнює класичні моделі ЧР, як, наприклад, AR:

$$\hat{y}_{t+h} = f(X_t),$$

де $X_t = (x_{t-T+1}, \dots, x_t)$, x_{t-j} – це j-те лагове спостереження x_t для $j = 0..T-1$.

Приклад RNN з одним прихованим шаром для $j = 0..5$ зображено на рисунку 2.1:

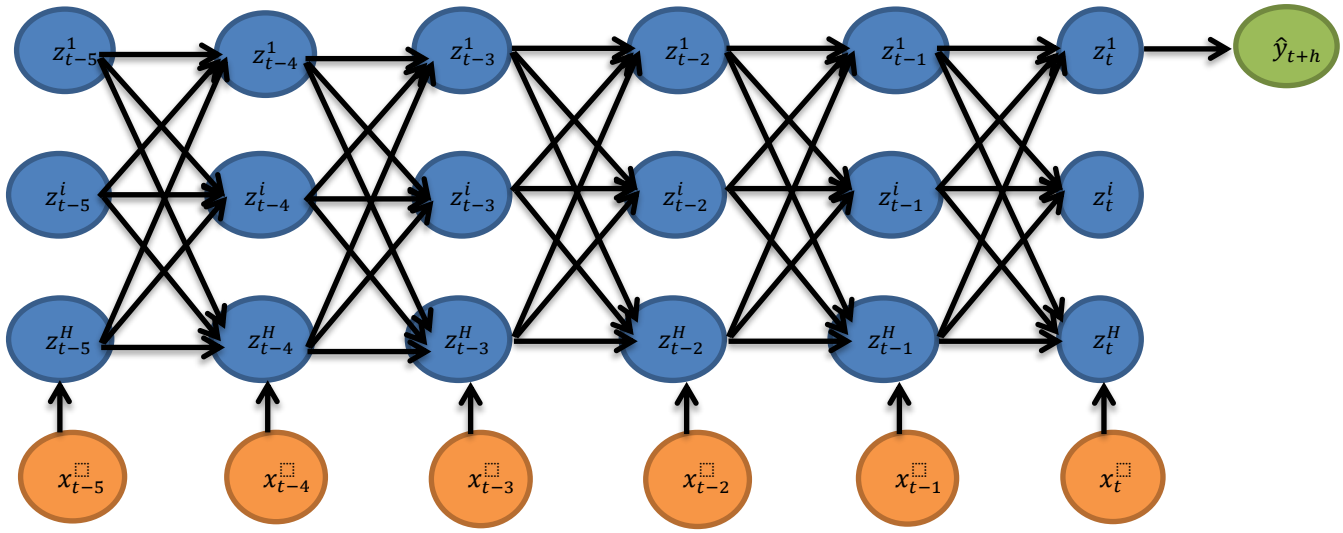

Рисунок 2.1 – Приклад RNN з одним прихованим шаром, розгорнутим над послідовністю з 6 часових кроків

Розглянемо даний приклад більш детально. Прихований шар містить H нейронів, де i-й вихід позначається як z_t^i. Зв'язки між нейронами – рекурентні та зважені матрицею $W_z^{(1)}$. На останньому часовому кроці t, результати обчислень «зливаються» в єдиний нейрон зі значенням $\hat{y}_{t+h}$. На кожному кроці $j = 0, \ldots, 5$, функція $f^{(1)}_{W^{(1)},b^{(1)}}(X_{t,j})$ генерує прихований стан z_{t-j} з поточного вводу та попереднього прихованого стану z_{t-1} і послідовності $x_{t-T+1}, \ldots, x_{t-j}$. Приховані стани z_{t-j} обчислюються за формулою:

$$z_{t-j} = f^{(1)}_{W^{(1)},b^{(1)}}(X_{t,j}) = \sigma^{(1)}\left(W_z^{(1)} z_{t-j-1} + W_x^{(1)} x_{t-j} + b^{(1)}\right),$$

де $W_x^{(1)} \in \mathbb{R}^{H \times P}$ – матриця зв'язків зовнішніх вводів x_t з прихованими шарами, а матриця $W_z^{(1)} \in \mathbb{R}^{H \times H}$ містить значення рекурентних ваг серед H прихованих нейронів.

Значення цільового нейрона визначено як

$$\hat{y}_{t+h} = f^{(2)}_{W^{(2)},b^{(2)}}(X_{t,j}) = \sigma^{(2)}\left(W^{(2)} z_t + b^{(2)}\right),$$

де $\sigma^{(2)}$ – це softmax-функція або мапа ідентичності в залежності від типу цільової змінної; $W^{(2)}$ позначає ваги для виходу нейронів на останньому кроці.

Отже, RNN зберігають внутрішній стан для кожного часового кроку та мають два джерела для оновлення вагових коефіцієнтів – нові спостереження ЧР та попередній стан NN. На відміну від класичних NN, RNN використовують однакові параметри $W = (W_x, W_z)$ на всіх кроках. Варто зазначити, градієнтна функція для оновлення ваг обчислюється не тільки на

поточному кроці, а й спирається на декілька попередніх лагів. Тим не менш, RNN мають проблеми з довгостроковим залежностями, які були вирішені в архітектурі LSTM та GRU.

2.2 Alpha-RNN

Alpha-RNN – тип нейронної мережі, дуже схожий зі звичайними RNN, але до множини параметрів даної NN додано скалярну величину α, яка надає RNN довшу пам'ять[10]. Таке розширення досягається за рахунок згладжування прихованих параметрів моделі між обчисленнями вагових коефіцієнтів. Тобто додаються проміжкові нейрони, які обчислюють згладжені коефіцієнти за формулою, схожою до тієї, що визначає ETS. При цьому дещо зміниться процес обчислення прихованих коефіцієнтів:

$$z_s = f^{(1)}_{W^{(1)},b^{(1)},\alpha}\left(X_{t,j}\right) = \sigma^{(1)}\left(W_z^{(1)}\tilde{z}_{s-1} + W_x^{(1)}x_s + b^{(1)}\right),$$

де $s = t - j, \dots, 0$ та $j = 0 \dots p$.

Згладжені коефіцієнти мають вигляд

$$\tilde{z}_s = \alpha z_s + (1-\alpha)\tilde{z}_{s-1},$$

а приховані коефіцієнти на першому кроці вікна знаходять за допомогою елементів вхідної послідовності:

$$z_{t-p+1} = \sigma^{(1)}\left(W_x x_{t-p+1}\right).$$

Схема архітектури Alpha-RNN зображена на рисунку 2.4:

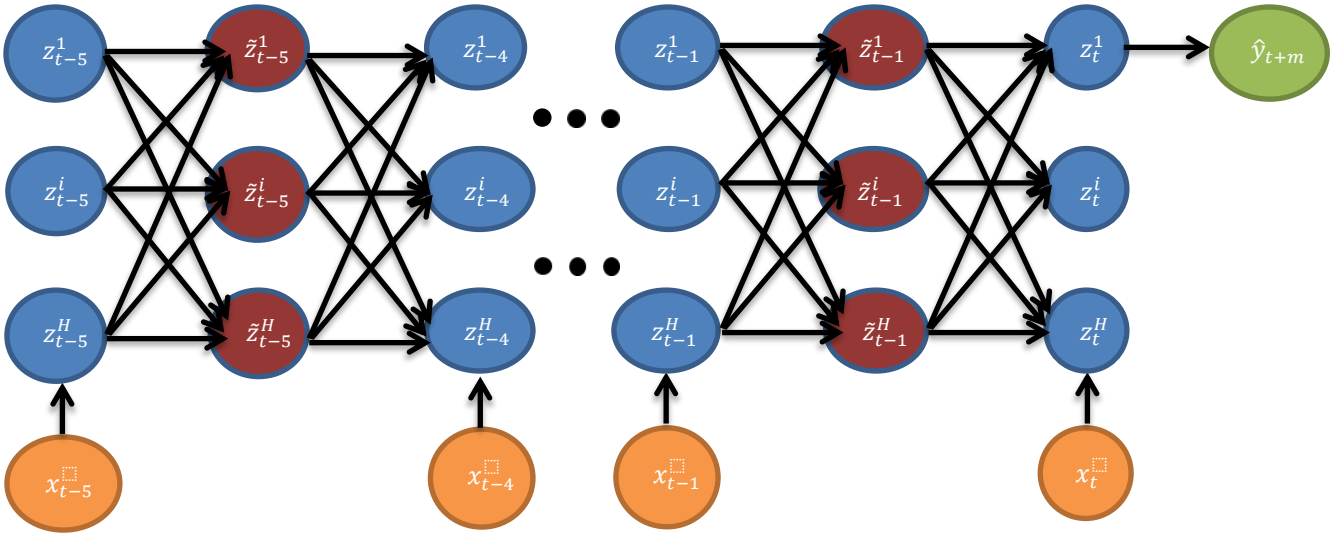

Рисунок 2.4 – Приклад Alpha-RNN з одним прихованим шаром, розгорнутим над послідовністю з 6 часових кроків

2.3 Alpha-t RNN

Вибір параметра α в моделі Alpha-RNN є апріорним та не змінюється під час тренування. Тому варто розглянути архітектуру, яка передбачає динамічну зміну даного параметра в залежності від часу на проміжку $[0, 1]$. Тоді оновлення згладжених параметрів матиме вигляд:

$$\tilde{z}_{t+1} = \alpha_t z_t + (1 - \alpha_t)\tilde{z}_t.$$

Тобто згладжування можна розглядати як динамічну корекцію помилки передбачення[10]. Якщо на деякому кроці параметр дорівнює нулю($\alpha_t = 0$), то значення попереднього згладженого параметра переноситься на поточний, що еквівалентно втраті інформації про попередні стани $\tilde{z}_{0..t-1}$. Якщо $\alpha = 1$, то значення попереднього прихованого стану z_{t-1} надається поточному згладженому. Простежити оновлення α_t можна з розширеної формули для $\tilde{z}_{t+1}$:

$$\tilde{z}_{t+1} = \alpha_t z_t + \sum_{s=1}^{t-1} \alpha_{t-s} \prod_{r=1}^{s} (1 - \alpha_{t-r+1}) z_{t-s} + \prod_{r=0}^{t-1} (1 - \alpha_{t-r}) \tilde{z}_1,$$

де $\tilde{z}_1 = z_1$. Зауважимо, що для будь-якого $\alpha_{t-r+1} = 1$, значення цільової змінної $\tilde{z}_{t+1}$ не залежатиме від лагів $z_{t-s} \dots z_t$, $s \geq r$.

2.4 Long Short Term Memory(LSTM)

LSTM – це особливий тип RNN, які вміють ефективно запам'ятовувати довгострокові залежності. Вони були запропоновані в [9] в 1997. Основна відмінність полягає в тому, що до структури рекурентного блоку було додано три прихованих шари, які гнучко взаємодіють між собою.

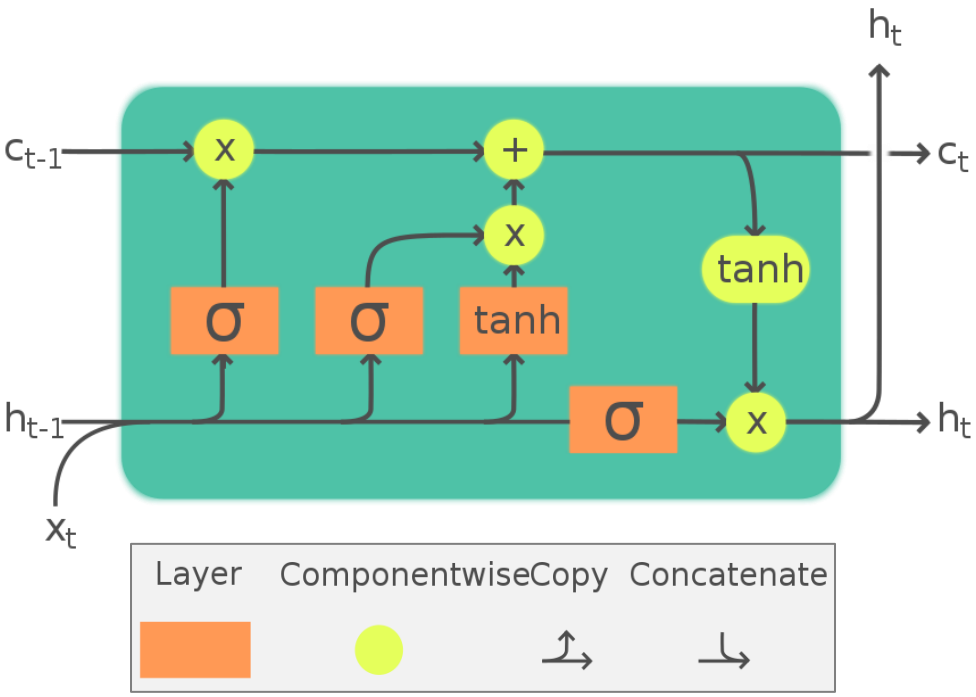

Рисунок 2.2 – Архітектура блоку LSTM

Розглянемо складові рекурентного блоку, зображеного на рисунку 2.2, які утворюють багаторівневий пайплайн для обробки вхідних даних. У

верхній частині блоку знаходиться ключовий компонент в LSTM-моделі – стан клітини(або блоку) $c_{t-1} \to \times + \to c_t$. Він з'єднаний з попереднім та наступним блоком, модифікується всередині поточного блоку через т.з. ворота(gates) і впливає на його прихований параметр h_t. На першому кроці визначаємо, яку інформацію потрібно забути. Операцію «забування»(або "f") виконує перший сігмоїд-шар. Його вихід обчислюється таким чином:

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

Далі в два етапи вирішуємо, яку нову інформацію будемо зберігати в стані клітини. По-перше, застосовуємо сігмоїд-функцію в т.з. «вхідному» шарі(input gate layer, "i") для того, щоб визначити, які значення потрібно оновити. По-друге, шар гіперболічних тангенсів(tanh-layer) утворює вектор значень(операція "g"), які можуть бути доданими то поточного стану. Тоді

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

є ідентифікатором корисних вхідних даних, а

$$\tilde{c}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

генерує масив кандидатів на оновлення стану блоку. Передостаннім кроком є оновлення поточного стану клітини. В ньому відфільтровуються неактуальні попередні знання та обираються нова релевантна інформація. Маємо вираз:

$$c_t = f_t * C_{t-1} + i_t * \tilde{c}_t.$$

Кінцевим кроком є вивід відфільтрованого стану блоку. Спершу шар сігмоїдів фільтрує комбінації попереднього виводу h_{t-1}(він же вхід поточного блоку) та нових даних x_t:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_0)$$

Значення поточного стану фільтруються гіперболічним тангенсом та множаться на гейт сігмоїдів. Результат зберігається в поточному блоці та передається в наступний:

$$h_t = o_t * \tanh(c_t).$$

Архітектура LSTM допомагає уникнути проблем, які виникають під час тренування об'ємних RNN/DNN, а саме, проблеми нульового та експоненційного градієнта. Якщо значення часткових похідних є дуже великими, то подальші ітерації алгоритму градієнтного спуску призведуть до його експоненційного росту. В іншому випадку похідні можуть мати незначні коефіцієнти та зовсім не впливати на хід тренування.

Згідно з об'ємним дослідженням [11], стандартна LSTM-модель загалом показує не гірші результати в 5400 експериментах за її модифікації на різних датасетах. Тим не менш, дві модифікації – CIFG та NP – зменшують кількість параметрів моделі без наслідків для ефективності. Також зазначається, що операція "f" та активація виходу є найважливішими компонентами блоку, тому що вони гарантують стабільний перебіг навчання.

2.5 Gated Recurrent Unit(GRU)

В попередньому розділі був розглянутий особливий тип RNN – LSTM. Моделі даного класу можуть запам'ятовувати довгострокові залежності та якісно маніпулювати вхідними даними. Звичайно ж, за такий широкий функціонал доводиться платити більшим обчислювальним ресурсом, необхідним для тренування моделі. Покращення LSTM в плані

продуктивності було втілене в архітектурі GRU за рахунок зменшення кількості шарів до двох. Розглянемо особливості цієї моделі на рисунку 2.3.

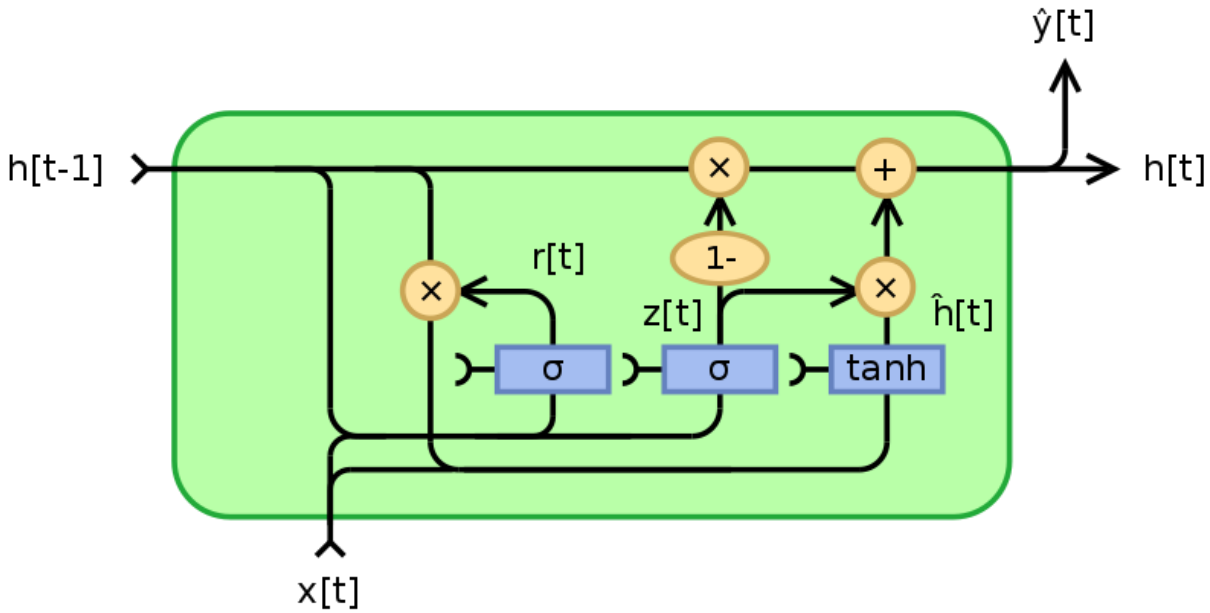

Рисунок 2.3 – Архітектура блоку GRU[Wiki]

Щоб вивчити щось нове, спочатку потрібно забути непотрібне старе. Тому на першому кроці, аналогічному до першого кроку LSTM, застосовуємо шар сігмоїдів для нових зовнішніх даних x_t та стану попереднього шару h_t. Тоді маємо ”r”-функцію(reset-function):

$$r_t = \sigma(W_{xr}x_t + b_{xr} + W_{hr}h_{t-1} + b_{hr}).$$

Функція оновлення “z”(update-function) є аналогічною до “r”-функції за винятком відповідних вагових матриць, що множаться на x_t та h_{t-1}.

$$z_t = \sigma(W_{xz}x_t + b_{xz} + W_{hz}h_{t-1} + b_{hz}).$$

Передостанній компонент інкапсулює дві частини: перша схожа вхідний шар, а друга застосовує вектор r_t до, по суті, стану попереднього блоку.

$$n_t = \tanh\big(W_{xn}x_t + b_{xn} + r_t \odot (W_{hn}h_{t-1} + b_{hn})\big).$$

Тут оператор $\odot$ позначає добуток Адамара. Останнім кроком є оновлення стану поточного блоку:

$$h_t = (1 - z_t) \odot n_t + z_t \odot h_{t-1}.$$

З формули видно, що якщо $z_t \to 1$, то стан попереднього блоку переноситься на поточний, $h_t \coloneqq h_{t-1}$. У протилежному випадку, якщо $z_t \to 0$, то попередній стан майже ігнорується. Отже, z_t виражає міру впливу на поточний стан, причому r_t відфільтровує попередні стани та поточні вводи, а n_t визначає скільки пам'яті слід залишити в поточному блоці.

Ідея GRU на 17 років молодша за LSTM, тим не менш, вони обидві показують високі результати на багатьох датасетах і часто саме підбір гіперпараметрів має визначальне значення[2], а не тип архітектури. Тим не менш, різні задачі вимагають різних інструментів, і у випадку невеликих датасетів з короткими послідовностями тренування об'ємних LSTM-моделей не дає відчутних переваг. Якщо ж ЧР містить довгострокові залежності, то саме LSTM зможе ефективно передавати їх на «довгі дистанції».

В нещодавньому дослідженні[12] було запропоновано замінити "r"-функцію на контентно-адаптивний шар. CARU-модель має менше параметрів за GRU та показує дещо кращі результати в деяких задачах комп'ютерної лінгвістики.

2.6 N-BEATS

N-BEATS – це модель глибокого машинного навчання, архітектура якої ґрунтується на залишкових блоках та ієрархічно організованих шарах NN. Цікаво, що вона не містить класичних елементів аналізу ЧР або інших послідовностей та має легко розширювальну структуру, яка може бути доповнена методами аналізу ЧР[3]. Оглянемо схему архітектури N-BEATS:

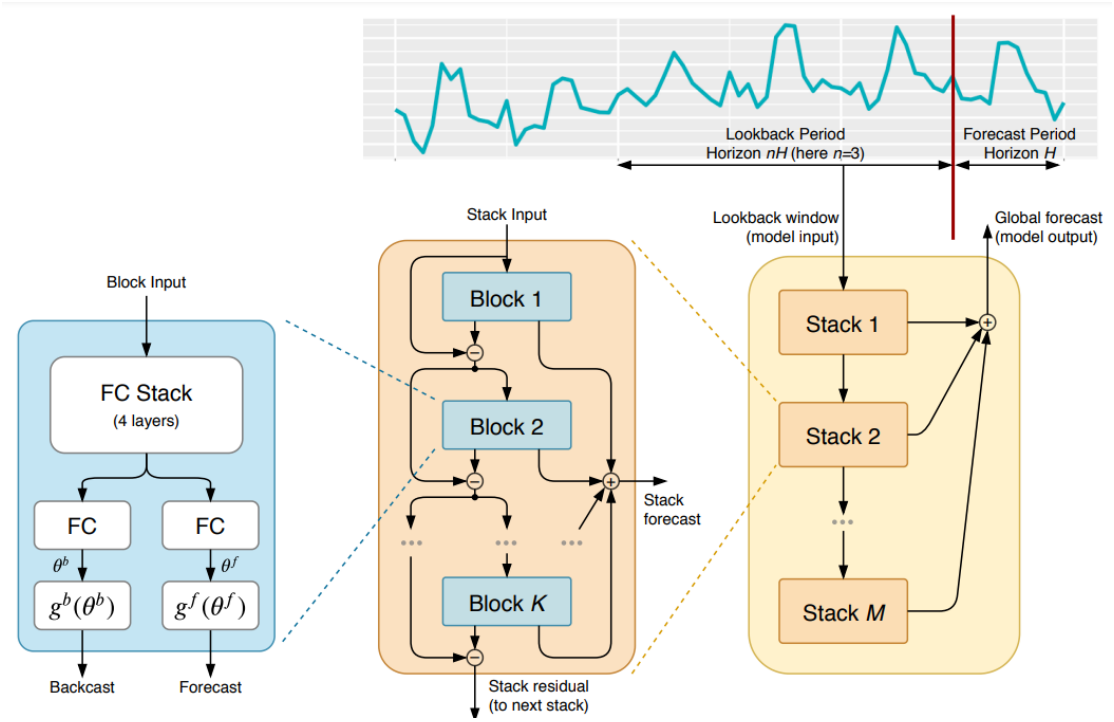

Рисунок 2.4 – Схема архітектури N-BEATS

Базовим елементом даної NN є блок, який складається з декількох повністю з'єднаних шарів з ReLu-активаторами. Він прогнозує предиктори θ^g та θ^f як базиси для передбачення для минулого та майбутнього відповідно. Блоки організовані в стеки з за принципом подвійного залишку. Стеки агрегують часткові передбачення та передають один одному. На останньому кроці, дані підпослідовності збираються в єдиний прогноз.

Розділ 3. Призначення системи

Призначенням системи для аналізу фінансових даних є побудова короткострокового передбачення ціни криптовалюти Ethereum(ETH) в доларовому еквіваленті для надання рекомендацій щодо операцій з даним активом. Основні етапи аналізу вхідних даних зображені на рисунку 3.1.

Дипломна робота передбачає:

a) аналіз методів, методик і моделей, що застосовуються для розв’язання задачі передбачення розвитку ЧР;

b) дослідження існуючих програмних засобів у галузі машинного навчання для вирішення даної задачі;

c) проектування та тренування як класичних, так і гібридних моделей обробки ЧР;

d) тестування моделей для визначення якості кожної з них;

e) порівняння отриманих результатів для вибору найбільш оптимального варіанту, враховуючи контекст задачі.

3.1 Цілі створення системи

Аналіз та розробка методів машинного навчання для вирішення задач фінансової математики – це дослідження, яке має за мету:

a) визначити переваги конфігурації кожної розглянутої моделі;

b) пояснити результати тестування виходячи з теоретичних відомостей про предмет дослідження;

с) визначити найбільш перспективну технологію для практичного застосування та визначити напрямок подальших досліджень.

Рисунок 3.1 – Етапи аналізу ЧР

Розділ 4. Вимоги до системи

4.1 Вимоги до системи вцілому

Система повинна аналізувати та обробляти як стаціонарні, так і не стаціонарні ЧР. Моделі навчання повинні бути побудовані з урахуванням особливостей вхідних даних та мати MAPE(Mean Absolute Percentage Error)-поріг 20%. Також залишки повинні мати гомоскедастичну природу і їхній розподіл має бути близьким до розподілу білого шуму. Етап передобробки даних передбачає вилучення або заповнення пустих значень, виведення повідомлень про недопустимий формат вхідних даних.

За необхідності, система повинна автоматично завантажувати нові дані з відкритих джерел. Тоді інфраструктура обробки даних має бути розроблена відповідно до парадигми Extract-Transform-Load та вести облік експлуатації системи з метою виявлення аномалій, неконсистентності або некоректності.

Окрім функціональних вимог, виділимо технічні:

a) тренування моделей не повинно займати більше ніж 48 годин на стандартному single-node кластері з 0,75 DTU (8 ядер, 14 GB ОП);

b) усі натреновані моделі мають бути завантажені до DBFS для подальшого використання;

c) якщо тренування займає більше ніж 48 годин, то потрібно змінити формат вхідних даних або корегувати параметри моделі;

d) універсальний доступ до оброблених даних та ML моделей.

4.2 Вимоги до функцій, які виконуються системою

4.2.1 Складові середовища розробки

В нижченаведеній Таблиці 4.1 представлені функції та задачі, які виконує Azure Databricks IDE.

Таблиця 4.1 – Перелік задач Azure Databricks IDE

Функція	**Задача**
Підтримка Jupyter Notebooks	Надавання доступу до сховища навіть за умови вимкнених обчислювальних потужностей
Підготовка та аналіз датасетів	Імпорт, обробка та збереження даних в автоматизованому режимі
Навчання ML моделей	Моніторинг та менеджмент процесу тренування, валідації та тестування ML-моделей

4.2.2 Вимоги до розробника

Таблиця 4.2 – Перелік вимог до ML-аналітика ЧР.

Етап обробки	**Вимоги**
Дискриптивні статистики, зображення	Узагальнення та візуалізація даних; Висунення гіпотез щодо ЧР

Прогноз та контроль	Побудова ML-моделей та валідація результатів
Аналіз	Візуальний аналіз залишків та проведення статистичних тестів; обчислення метрик залишків
Порівняння	Виокремлення ключових параметрів кожної моделі та вибір найбільш релевантного рішення; порівняння метрик, де це доцільно

4.3 Технічні вимоги

Azure Databricks IDE є середовищем розробки, яке підтримується такими інтернет-браузерами: MS Internet Explorer 5.5 і вище, Mozilla 1.7 і вище, Opera 7.54 і вище, Chrome 81.0.4044 і вище.

Дане середовище повинно бути доступним в регіоні, зазначеному розробником під час ініціалізації відповідного сервісу. На етапах попередньої обробки даних, аналізу та запуску, необхідно спостерігати за навантаженням на хостингову систему з метою дотримання принципів оптимальної утилізації доступних ресурсів.

Для використання потужностей API хмарних сервісів Azure, зокрема Azure Databricks, в локальних додатках, написаних на популярних мовах програмування, необхідно завантажити бібліотеки, доступні на сторінці документації Azure.

Розділ 5. Реалізація системи

5.1 Перевірка статистичних гіпотез

Оскільки датасет містить хронологічно впорядковані спостереження, то кожне спостереження не є незалежним. Всього маємо одну цільову змінну та одну змінну для тренування, тому задача передбачення є одновимірною(з однією ознакою).

5.1.1 Стаціонарність

Характеристики стаціонарних процесів не залежать від часу. Дуже важливо визначити її наявність, оскільки вона може сильно вплинути на коректність інтерпретації моделі та її результати. З практичних міркувань, відсутність стаціонарності часто негативно впливає на результати передбачення класичних методів. Навіть ML-моделі з недостатньо гнучкою структурою, як, наприклад, прості RNN можуть не дати бажаних результатів.

Проведемо тест Дікі-Фулера(Augmented Dickey-Fuller test) на строгу стаціонарність ЧР.

$$X(t_i) \cong X(t_i + \tau)$$

$$H_0\colon \gamma \neq 0.$$

Нульова гіпотеза стверджує, що ЧР не описує деякий часовий тренд. Тобто всі випадкові величини, що визначають випадковий процес є однаково розподіленими[6]. Згідно з альтернативною гіпотезою, ЧР має тренд, тому, щоб звести його до стаціонарного, потрібно провести відповідну кількість диференціювань і про вести тест знову. Цікаво, що модель ARFIMA дозволяє провести нецілу кількість диференціювань, наприклад, 0.5.

В нашому випадку, p-value складає 0.6, що дає підставу для прийняття нульової гіпотези, тобто ЧР – не стаціонарний. Цей висновок відповідає візуальному аналізу ЧР. Дійсно, на рисунку 5.1 спостерігаємо зростаючий тренд по локальних мінімумах.

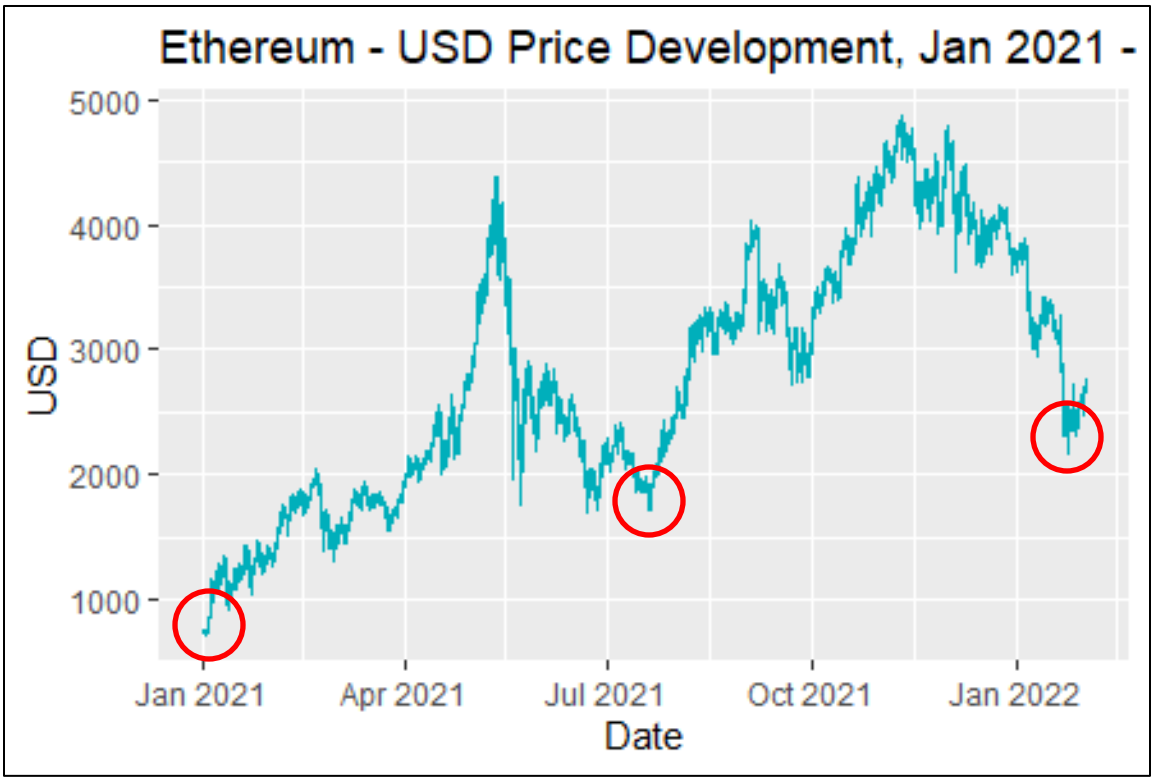

Рисунок 5.1 – Ціна Ethereum в доларовому еквіваленті, Січень 2021 – Січень 2022

5.1.2 Ідентифікація параметрів авторегресії ЧР

Щоб визначити ширину вікна для RNN-подібних моделей, потрібно визначити кількість лагів зі значущою автокореляцією. Для цього розглянемо частинну автокореляційну функцію(ЧАКФ), яка на відміну від звичайної корелограми дає більш «чисту» картину періодичних залежностей. Критичні

межі ЧАКФ визначаються залежно від рівня значущості та кількості ознак датасету за формулою:

$$\frac{\left(N{\sim}(0,1)\right)^{-1}(\alpha)}{\sqrt{n}},$$

де $N{\sim}(0,1)$ – стандартний нормальний розподіл, $(1-\alpha)$ – рівень значущості, n – кількість ознак в датасеті. Тоді задача знаходження оптимальної ширини вікна зводиться до пошуку першого лагу, абсолютне значення якого менше за α. Якщо індекс цього лагу - i, то ширина вікна дорівнює $i-1$(кореляція попередніх лагів є значущою). Розглянемо автокорелограму та ЧАКФ даного ЧР на рисунках 5.2 та 5.3 відповідно.

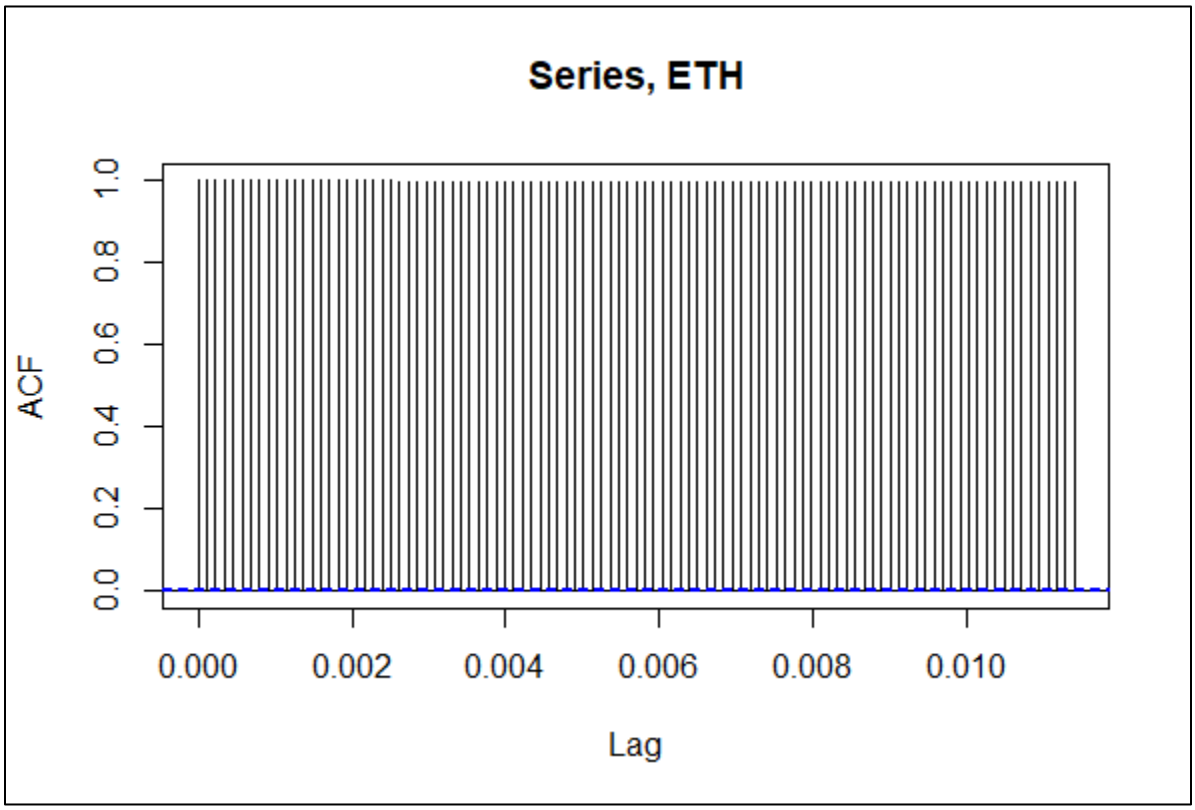

Рисунок 5.2 – Автокореляційна функція

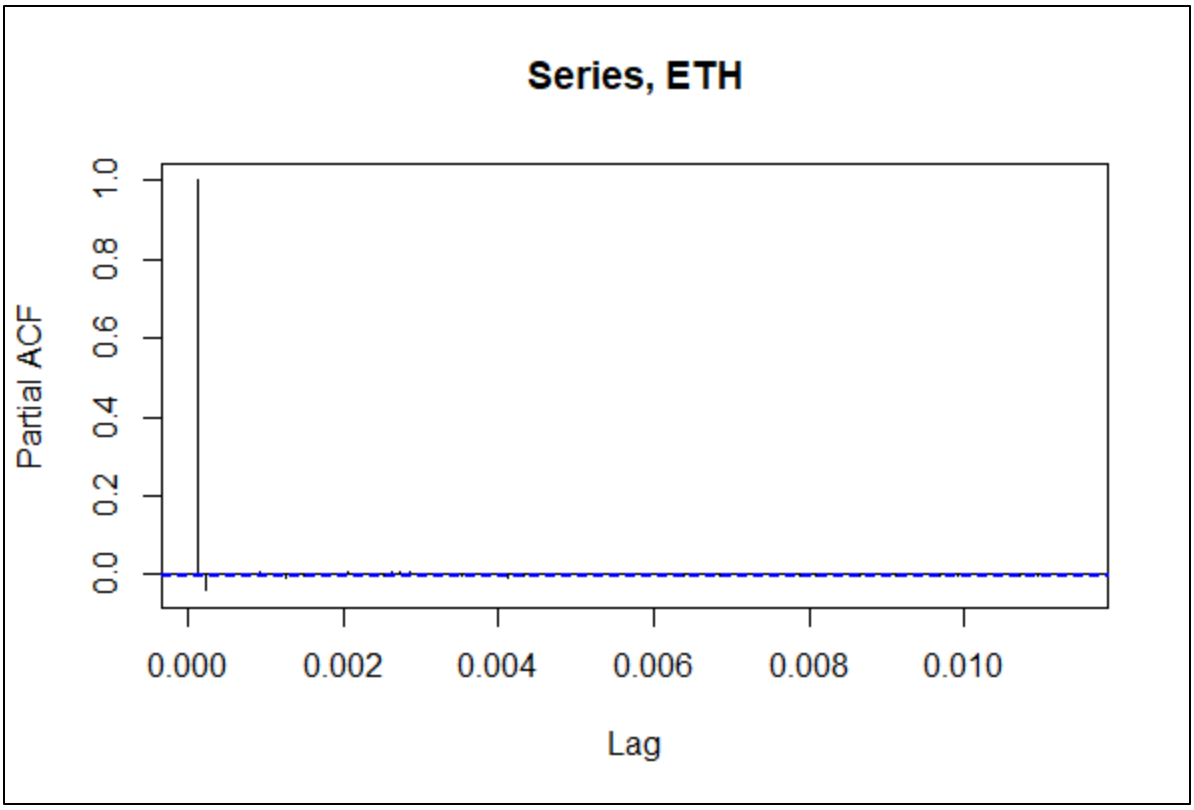

Рисунок 5.3 – Часткова автокореляційна функція

З рисунку 5.2 видно, що кореляція між спостереженнями сягає 1, що свідчить про тісну взаємодію між елементами ЧР. Коефіцієнти кореляції в даному випадку не є спадними в межах заданого вікна. Можливо, збільшення максимального лагу допомогло б помітити зменшення кореляції, але зважаючи на кількість спостережень(456000), збільшення повинно сягати половини довжини ЧР. Якщо побудувати графік випадкового блукання

$$\{S_t, t = 0,1,2, \dots\},$$

де S_t – сума незалежних однаково розподілених випадкових величин, причому $S_0 = 0$ та $\{X_t, t = 1,2, \dots\}$, тобто

$$S_t = X_1 + X_2 + \cdots + X_t,$$

то графік корелограма такого процесу буде схожим з графіком на рисунку 5.2. Якщо ж $\{X_t, t = 1,2, \dots\}$ – це симетричний бінарний процес, то S_t є простим симетричним блуканням на прямій. Таке блукання може бути описано як позиція пішохода, який мандрує вулицями Манхетена та на кожному перехресті випадково обирає подальший напрямок руху: наліво або направо. В термінах фондових ринків, такий процес можна спроектувати на зміну ціни деякої акції, облігації тощо. Можемо зробити висновок, що спостереження нашого часового ряду відповідають цілком природнім процесам, але разом з тим, описати їх класичними методами навряд вдасться.

На рисунку 5.3 спостерігаємо ЧАКФ, АКФ без залежності між проміжними(всередині лагу) спостереженнями. Тобто з АКФ буї видалений вплив автокореляцій з меншими лагами. Вже четверте спостереження знаходиться за критичною межею, що говорить про низьку кореляцію між лагами в заданому діапазоні.

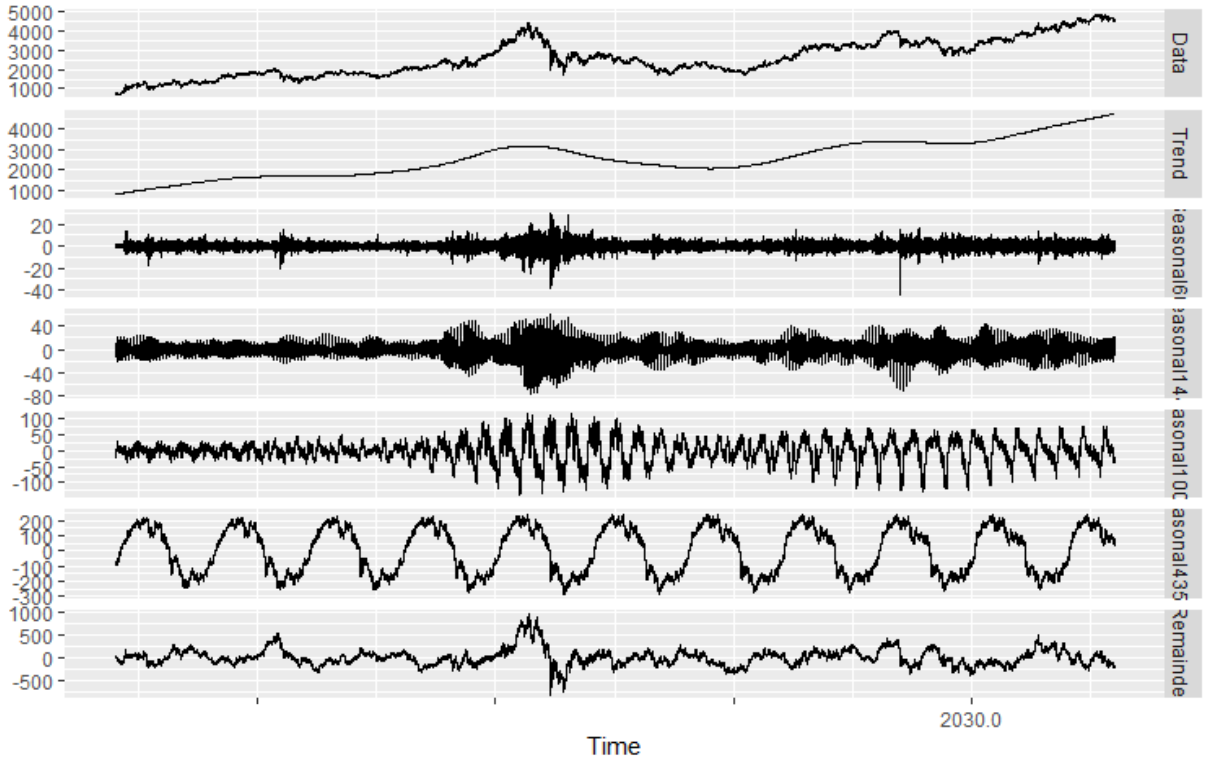

Рисунок 5.4 – Декомпозиція ЧР

На рисунку 5.4 зображена декомпозиція даного ЧР на три глобальні складові: тренд, сезонність, залишок, причому сезонність містить три розклади. Варто зазначити, що сезонні компоненти 60(щогодини) та 60*24(щодня) схожі на щільний білий шум, оскільки виділити циклічність візуально дуже складно. Тим не менш, сезонні компонент 60*24*7 та 60*24*30(щотижня та щомісяця) носять більш систематичний характер. Але вони не достатньо добре пояснюють дисперсію спостережень ЧР. Дійсно, значення залишкової компоненти належать проміжку $(-500; 500)$, що в 2.5 рази більше за розмах щомісячної компоненти. З рисунку видно, що саме залишок має визначальний вплив на значення ЧР, але пояснити його такими класичними моделями, як, наприклад, SARIMA не вийде. Це пов'язано з вимогами до залишкового члена ϵ_t адитивної моделі ARIMA.

5.2 Тестування SARIMA

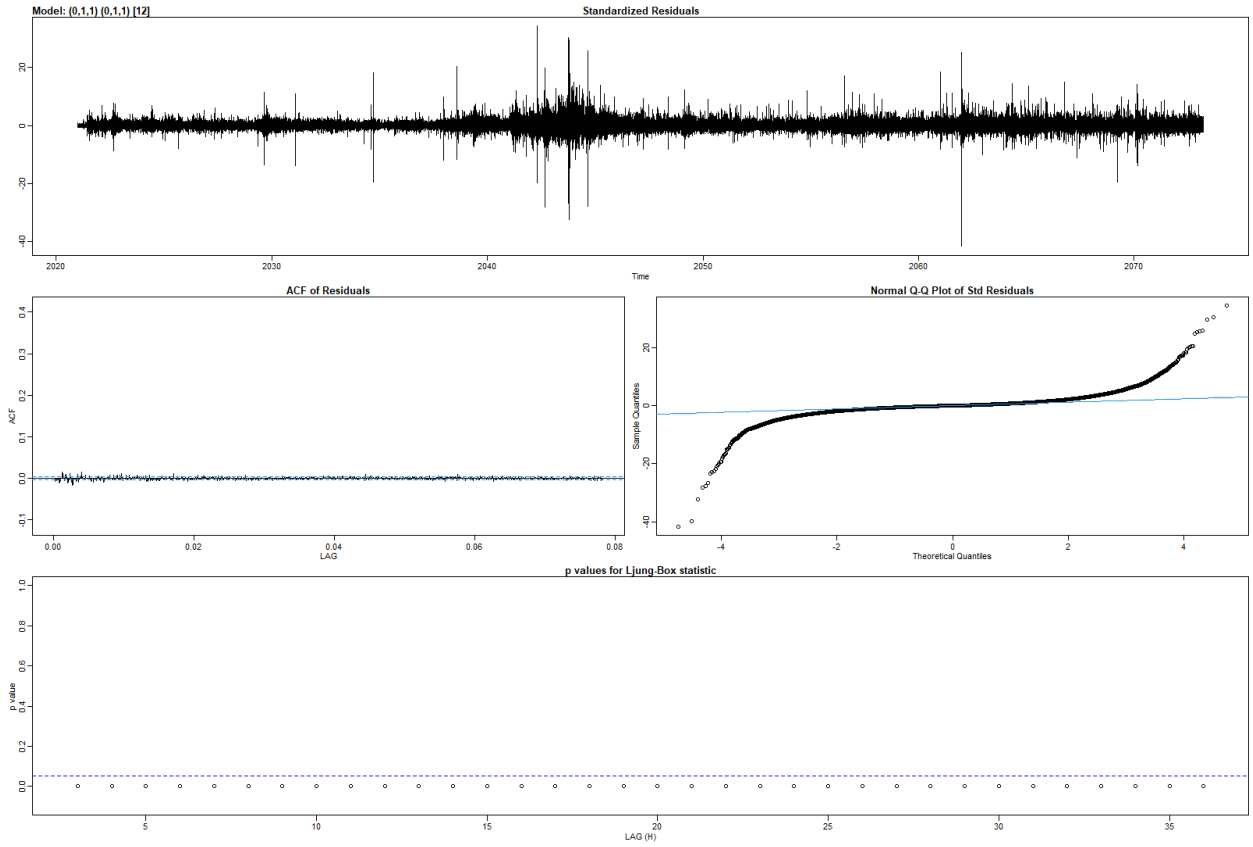

Рисунок 5.5 – Оцінка якості SARIMA-моделі

На рисунку 5.5 зображені графічні метрики для оцінки якості підгонки SARIMA-моделі. На графіку зверху видно стандартизовані значення залишків. Спостерігаємо значну кількість аномалій та очевидну відмінність від розподілу білого шуму. Дійсно, корелограма відображає неконсистентну кореляцію поза критичними межами. Q-статистика Льюнга-Бокса не перевищує критичного рівня на жодному лазі, що дає підставу прийняти нульову гіпотезу і зробити висновок про те, що залишки не корелюють між собою.

5.3 Реалізація RNN-подібних моделей

5.3.1 Підготовка до тренування

Елементи даного ЧР потрібно стандартизувати з метою уникнення проблем з підгонкою моделі. Цей крок є особливо важливим, якщо датасет містить декілька ознак, які мають різні одиниці вимірювання. Стандартизуємо спостереження за формулою:

$$\frac{X - \mu}{\sigma},$$

де μ – це середнє по X, а σ – стандартне відхилення.

Наступний крок – формування множин для тренування та тестування. Вони складаються з послідовностей спостережень, які перетинають один одного та мають довжину, визначену на кроці ідентифікації авторегресійної моделі. В термінах ML, такі послідовності або вектори називають тензорами[4]. В нашому випадку, множини тренування та тестування мають такі розмірності: [(456557, 3, 1), (456557, 1)], [(114135, 3, 1), (114135, 1)].

5.3.2 Специфікація моделей

Раніше ми описали структуру кожної моделі, використаної для вирішення поставленої задачі. Тепер необхідно задовольнити певні технічні вимоги перед початком процесу тренування, а саме: вибір параметрів моделей.

За ініціалізацію ваг та вільних векторів відповідають параметри kernel_initializer та bias_initializer відповідно. Розподіл значень відбуватиметься за принципом Хавьєра(Xavier initialization), мета якого забезпечити константу дисперсію після активації всередині кожного шару. Такий підхід допомагає уникнути проблеми зникаючого або експоненційно зростаючого градієнту. Ініціалізація Хавьєра має такий загальний вигляд:

$$W_x^{[l]} = \mathrm{N}\left(0, \frac{1}{n^{[l-1]}}\right).$$

Ініціалізатор рекурентної матриці $W_z^{[l]}$, recurrent_initializer, визначаємо як ортогональну матрицю з довільним початковим станом. Регуляризатором функції втрат обираємо L1, оскільки він допомагає ефективно нейтралізувати велику кількість непотрібних ваг. Тоді функція втрат MSE визначається таким чином:

$$\frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 + \lambda\sum_{j=1}^{p}\left|\beta_j\right| \rightarrow min.$$

Оптимізатор алгоритму градієнтного спуску – Adam. Його призначення полягає в тому, щоб динамічно обчислювати крок навчання для кожного параметра моделі. Він комбінує попередні розробки AdaGrad та RMSProp та успішно працює з нестабільними градієнтами на шумних даних.

Також важливо ініціалізувати об’єкт для відслідковування виконання критерію ранньої зупинки тренування. Тобто якщо після 100 кроків виконання алгоритму градієнтного спуску функція втрат не зміниться, то модель відновить найкращі ваги та процес тренування завершиться.

Наступний крок – це проведення крос-валідації для визначення оптимального коефіцієнта регуляризації λ та розмірності входу на останнього шару моделі. Елементи ЧР не можна перемішувати з метою формування підмножин для крос-валідації через кореляцію спостережень. Тому відбувається розбиття на декілька(в нашому випадку 5) підмножини з впорядкованими елементами для тренування і однією підмножиною для тестування. Причому кожна наступна множина для тренування містить попередню. Процес підгонки параметрів займає багато часу, оскільки потрібно оцінити ефективність кожної з 5-ти підмножин для 12 можливих комбінацій гіперпараметрів. Так, пошук для всіх моделей зайняв майже 7 годин. Таблиця 5.1 містить значення гіперпараметрів для відповідних моделей. Бачимо, що λ в усіх випадках дорівнює 0, що свідчить про відсутність перетренування моделей.

Модель	Розмірність входу останнього шару	Коефіцієнт регуляризації λ
RNN	20	0
AlphaRNN	10	0
Alpha-tRNN	5	0
GRU	10	0
LSTM	10	0

Таблиця 5.1 – Оптимальні гіперпараметри RNN-подібних моделей

5.3.3 Тренування

Процес тренування п’яти RNN-подібних моделей тривав 45 хвилин. Варто зазначити, що час тренування залежить від кількості епох, тому потрібно обрати оптимальну кількість, щоб не перенавантажити обчислювальний кластер. Так, 1000 епох можуть значно уповільнити процес тренування, який загалом триватиме до 24 годин. При цьому точність не зазнає істотного покращення, а обчислювальні потужності будуть витрачені даремно. На рисунку 5.6 зображені підігнані RNN-подібні моделі у випадку тренування на 500 епохах.

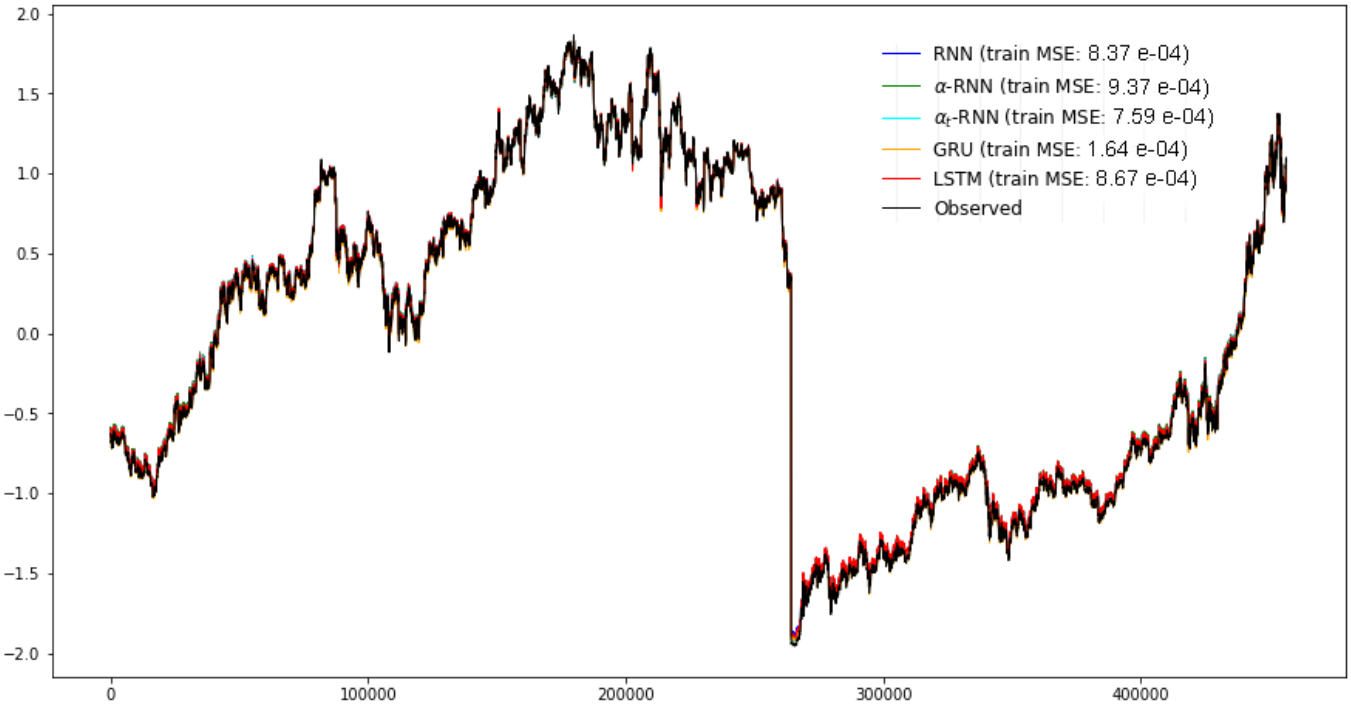

Рисунок 5.6 – Спостереження та тренування

З рисунку 5.6 видно, що моделі добре повторюють коливання спостережень на ділянках сталого росту до стрімкого спаду 20.07.2021. Після цього спостерігаємо легке відхилення від фактичних даних на наступній ділянці підйому. Графік похибки тренування зображений на рисунку 5.7.

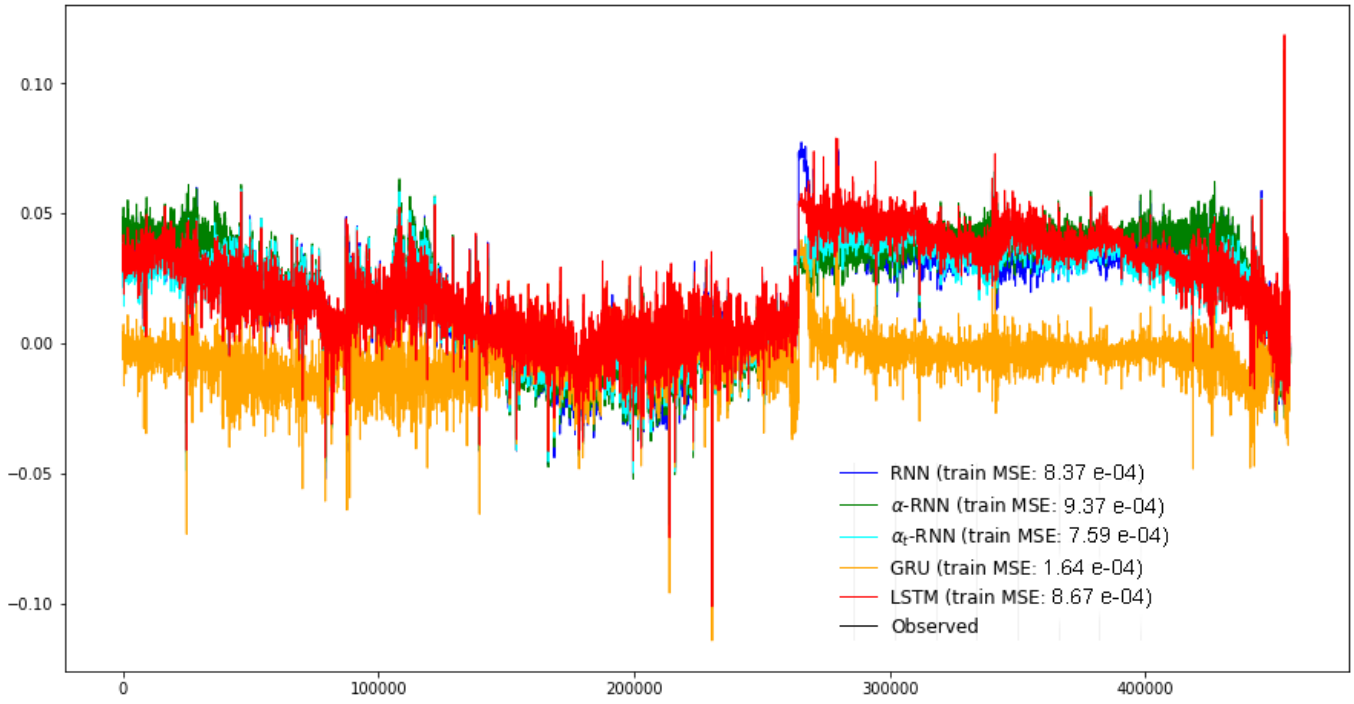

Рисунок 5.7 – Похибки тренування

Варто зазначити, що GRU доволі швидко адаптувалася до спаду 20 липня. Разом з тим, дана модель з самого початку тренування мала порівняно невелику похибку. В таблиці 5.2 наведені значення похибок тренування, які обчислюються за метриками: MSE, RMSE, MAPE.

Модель	MSE	RMSE	MAPE
RNN	8.37E-04	2.89E-02	9.5%
AlphaRNN	9.37 E-04	3.06E-02	9.4%
Alpha-tRNN	7.59 E-04	2.75E-02	9.3%
GRU	**1.64 E-04**	**1.28E-02**	**6.4%**
LSTM	8.67 E-04	2.94E-02	10.3%

Таблиця 5.2 – Похибки тренування за трьома метриками

5.3.4 Тестування

Множина спостережень для тестування містить 114135 елементів, 20% від загальної довжини ЧР, що складає майже 80 днів. На рисунку 5.8 можна візуально оцінити результати тестування моделей.

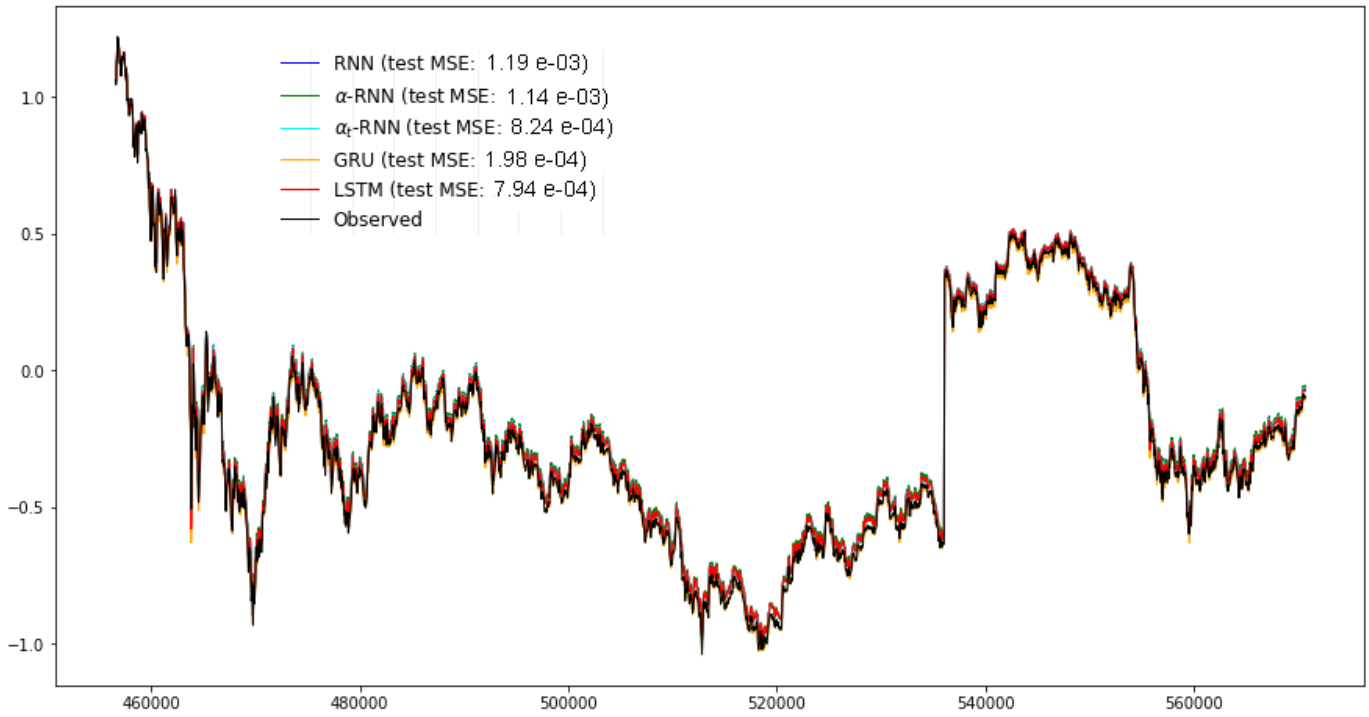

Рисунок 5.8 – Передбачені та фактичні значення

Загалом, всі моделі, окрім GRU, показують схожий результат. Найскладнішим завдання для нейронних мереж було передбачення падіння в околі 460000 спостереження(початок жовтня 2021). Також варто зазначити локальний мінімум біля 525000 спостереження(31 грудня 2021) та стрімкий ріст майже одразу після цього, який завершився корекцією до попереднього рівня. Спостерігаються систематичні відхилення, які повторюють розвиток фактичних спостережень та негативно впливають на похибку передбачення. Розглянемо графік похибок тестування на рисунку 5.9.

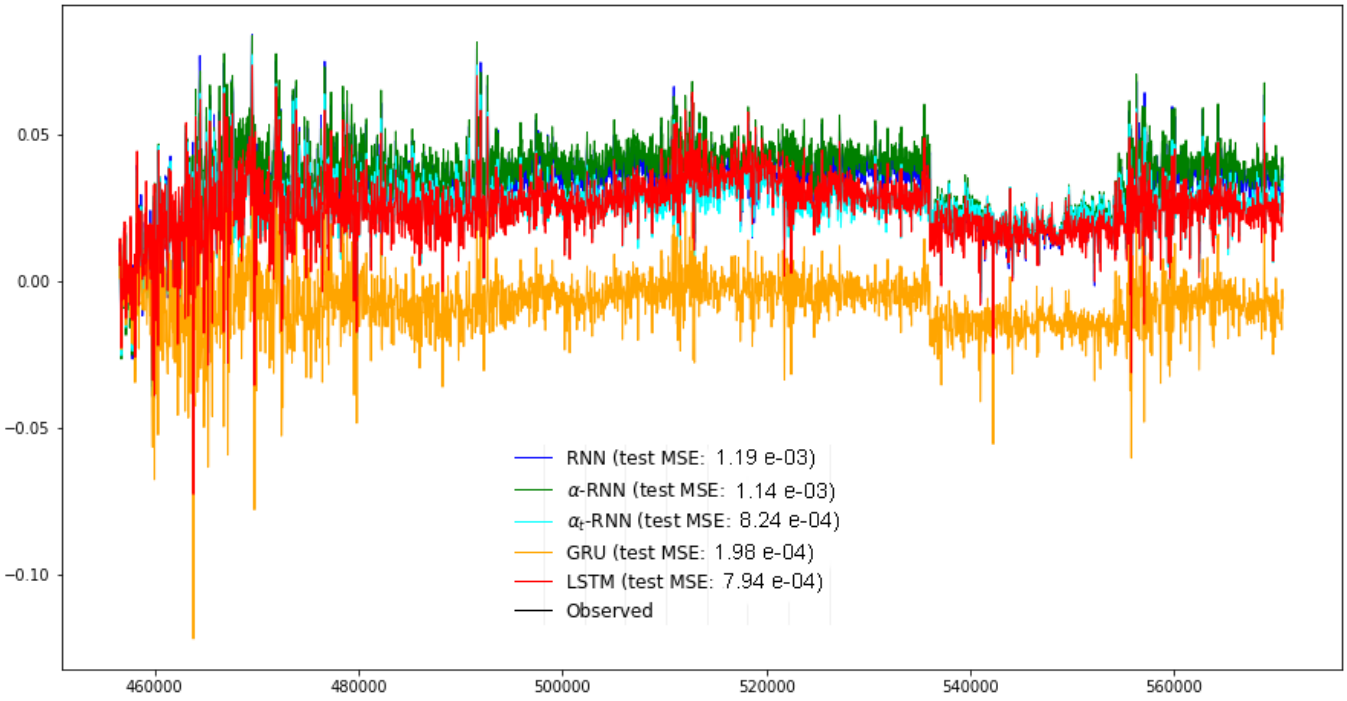

Рисунок 5.9 – Похибки тестування

З рисунку 5.9 видно, що тренд похибки GRU лежить найближче до 0, хоча модель робить аналогічні помилки як і інші. Графік немов підтверджує той факт, що ми працюємо з моделями з дуже схожою архітектурою, але в силу структурних відмінностей, результати тренування, а отже і тестування – різні. Розглянемо таблицю 5.3 з похибками тестування.

Модель	MSE	RMSE	MAPE
RNN	1.19E-03	4.37E-02	36.4%
AlphaRNN	1.14 E-03	3.37E-02	40%
Alpha-tRNN	8.24 E-04	2.87E-02	32.2%
GRU	**1.98 E-04**	**1.4E-02**	**16.1%**
LSTM	7.94 E-04	2.81E-02	26.8%

Таблиця 5.3 – Похибки тестування за трьома метриками

5.4 Реалізація моделі N-BEATS

Процес тренування даної моделі займає до 10 десяти годин, якщо кількість епох дорівнює 100. Встановлення меншої кількості епох не призведе до якісного підгону моделі. На рисунках 5.9 та 5.10 зображені результати передбачення з моделлю N-BEATS.

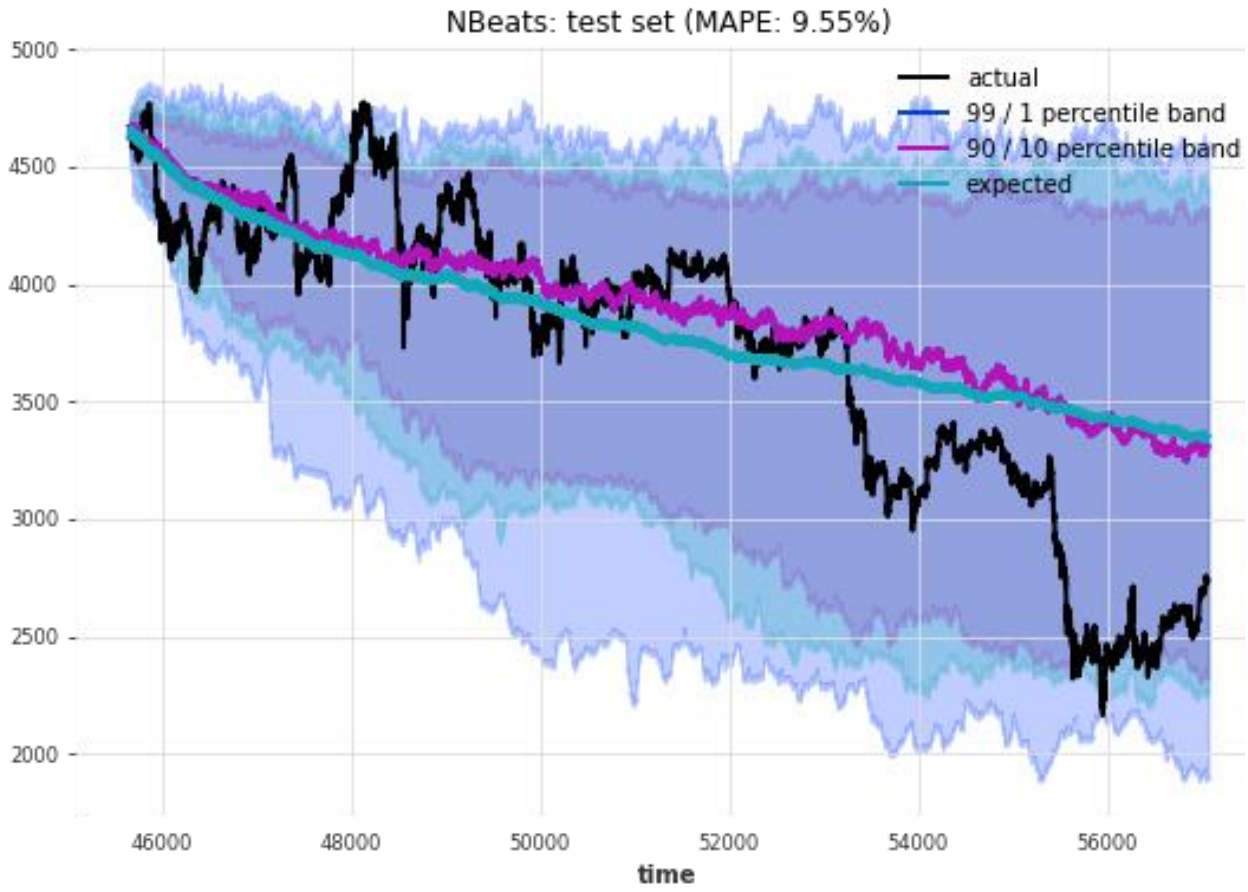

Рисунок 5.9 – Передбачення з N-BEATS

На графіку зображені фактичні та передбачені значення, а також значення, які матимуть місце з вірогідністю 90%. Зазначимо, що передбачення описують скоріше тренд розвитку часового ряду, але разом з тим, MAPE складає 9.55%, що на 6.55% менше за найкращий результат рекурентних моделей.

Розділ 6. Аналіз та порівняння результатів

Архітектура RNN зважає на структуру вхідних даних та визначає їхні особливості під час навчання. Це впливає на результат передбачення кожної моделі. В наших експериментах, жодна модель не показала кращу точність на тестових даних аніж на тренувальних, що говорить про стабільність навчання кожної з них. З графіків похибок тренування та навчання видно, що GRU найкраще адаптувалася до спостережень з самого початку навчання і це призвело до найкращих результатів в її класі. Якщо порівнювати GRU з іншими NN її класу, то ця модель має менше пам’яті за LSTM, але більше за RNN та їй подібних. Тобто облегшена та дещо модернізована архітектура LSTM дає найкращий результат за трьома метриками, тоді як базові моделі фактично не справляються з завданням.

Зазначимо, що тренування моделі N-BEATS проходило на зменшеній версії датасету – були обрані тільки 10-хвилинні спостереження для зменшення навантаження на кластер. Результати тестування вказують на задовільний показник метрики MAPE – 9.55%. При цьому візуальний аналіз говорить про недостатню кількість епох для більш якісного передбачення. Оскільки N-BEATS та RNN-подібні моделі використовували, по суті, різні дата сети, то порівняння їхніх метрик не є доцільним. Дійсно, не можна стверджувати апріорі, що RNN-подібні моделі матимуть гірший або кращий результат, якщо натренувати їх на зменшеному датасеті. Але всі ці моделі об’єднує відсутність вимог щодо розподілу або авторегресійних властивостей ЧР, тому вони можуть моделювати навіть нестаціонарні часові послідовності. Так, виявлення неконсистентних закономірностей – це перевага обраних ML-методів, яку вони успішно продемонстрували у вирішенні поставленої задачі.

ВИСНОВКИ

За результатами виконаної роботи, були досліджені та описані класичні та гібридні методи аналізу ЧР. Проведений порівняльний аналіз архітектури RNN-подібних моделей, а також розглянуті особливості моделі N-BEATS, яка використовує елементи DNN. Були спроектовані та натреновані відповідні моделі; проведено тестування та обчислення ключових метрик для визначення ефективності кожної з них.

За результатами емпіричних досліджень, визначена найбільш ефективна NN для успішного вирішення поставленого завдання – GRU. Окреслені перспективи для подальшого дослідження в сфері фінансової математики з урахуванням отриманих результатів.

Розглянуті способи обробки часових рядів є найбільш актуальними та широко розповсюдженими, а також легко розширюваними. Тому дослідження даних методів є значущим з науково-практичної точки зору.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Python for Algorithmic Trading / Yves Hilpisch – “O’Reilly”, 2021. – 458с.
2. Artificial Intelligence in Finance / Yves Hilpisch – “O’Reilly”, 2021. – 477с.
3. Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, Yoshua Bengio: ICLR 2021 / N-BEATS: Neural Basis Expansion Analysis For Interpretable Time Series.
4. AI as a Service / Peter Elger, Eoin Shanaghy – “Manning”, 2020. – 328с.
5. Deep Learning from Scratch / Seth Weidman – “O’Reilly”, 2019. – 269с.
6. An Introduction to Statistical Learning / Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani – “Springer”,2013. – 426с
7. Oskar Triebe, Hansika Hewamalagec, Polina Pilyuginad, Nikolay Laptevb, Christoph Bergmeirc, Ram Rajagopal / NeuralProphet: Explainable Forecasting at Scale, 2021.
8. Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, Pierre-Alain Muller / Transfer learning for time series classification, 2018.
9. Sepp Hochreiter, Jürgen Schmidhuber: Neural Computation 9(8):1735-1780 / Long Short-term Memory, 1997.

10. Matthew Dixon, Justin London: Frontiers of Applied Mathematics and Statistics / Financial Forecasting With α-RNNs: A Time Series Modeling Approach, 2021.

11. Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, Jürgen Schmidhuber: Transactions On Neural Networks And Learning Systems / LSTM: A Search Space Odyssey.

12. Ke, W.; Chan, K.-H.: Applied Sciences / A Multilayer CARU Framework to Obtain Probability Distribution for Paragraph-Based Sentiment Analysis, 2021.