Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи прогнозування нестаціонарних часових рядів на основі рекурентних нейронних мереж

(тема)

Виконав:

студент II курсу, групи СПм-22-3

Близнюк О.В.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Іващенко Г.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Близнюку Олександру Валерійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Методи прогнозування нестаціонарних часових рядів на основі рекурентних нейронних мереж

затверджена наказом по університету від “ 01 ” квітня 2024 р. № 257 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 червня 2024 р.

3. Вхідні дані до роботи

1) документація мови програмування Python;
2) документація бібліотеки Keras;
3) інтегроване середовище розробки JetBrains PyCharm 2023;
4) набори даних Yahoo!Finance з історією коливання курсів акцій Nasdaq.

4. Перелік питань, що потрібно опрацювати у роботі

1) аналіз предметної області;
2) огляд існуючих досліджень;
3) дослідження методів прогнозування часових рядів;
4) програмна реалізація обраних моделей прогнозування;
5) аналіз результатів дослідження;
6) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

Слайд-презентація – 13 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	02.04.24-08.04.24	
2	Вибір та обґрунтування методики дослідження	09.04.24-16.04.24	
3	Вибір інструментальних засобів	17.04.24-22.04.24	
4	Розробка програмного забезпечення	23.04.24-06.05.24	
5	Проведення експериментів	07.05.24-23.05.24	
6	Оформлення матеріалів кваліфікаційної роботи	24.05.24-03.06.24	
7	Подання кваліфікаційної роботи керівникові та	04.06.24-07.06.24	
	її попередній захист		
8	Подання кваліфікаційної роботи на	08.06.24-12.06.24	
	рецензування		

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Іващенко Г.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 79 с., 5 рис., 5 табл., 2 дод., 45 джерел.

ОБЧИСЛЮВАЛЬНИЙ ІНТЕЛЕКТ, ШТУЧНА НЕЙРОННА МЕРЕЖА, ЧАСОВІ РЯДИ, ПРОГНОЗУВАННЯ, LSTM, RNN, MAE, MAPE, KERAS, PYTHON.

Метою кваліфікаційної роботи є дослідження ефективності та точності прогнозування нестаціонарних часових рядів за допомогою рекурентних нейронних мереж.

У ході виконання кваліфікаційної роботи проведений аналіз існуючих методів обчислювального інтелекту, спрямованих на вирішення завдання прогнозування часових рядів, виявлені переваги та недоліки. Була реалізована модель для прогнозування нестаціонарних часових рядів на основі рекурентної нейронної мережі.

Дослідження проводилися із застосуванням даних щодо курсу акцій Nasdaq зібрані на сайті Yahoo!Finance.

Розроблений програмний продукт дозволяє прогнозувати часові ряди за допомогою моделей RNN та проводити порівняльний аналіз.

ABSTRACT

Master’s thesis: 79 pages, 5 figures, 5 tables, 2 appendices, 45 sources.

ARTIFICIAL INTELLIGENCE, ARTIFICIAL NEURAL NETWORK, TIME SERIES, FORECASTING, LSTM, RNN, MAE, MAPE, KERAS, PYTHON.

The major goal of this thesis is to investigate the efficiency and accuracy of forecasting non-stationary time series using recurrent neural networks.

In order to conduct the qualification work, an analysis of existing methods of computational intelligence aimed at time series forecasting, their advantages and disadvantages, was conducted. A model for forecasting non-stationary time series based on a recurrent neural network was implemented.

The research was conducted using data on the NASDAQ stock price collected from the Yahoo! Finance website.

The developed software allows forecasting time series using RNN models and conducting comparative analysis.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ARIMA – авторегресійна інтегрована змінна ковзна середня (англ., Autoregressive Integrated Moving Average)

CNN – згорткова нейронна мережа (англ., Convolutional Neural Network)

CPU – центральний процесор (англ., Central Processing Unit)

CSV – значення, розділені комами (англ., Comma-Separated Values)

DNN – глибока нейронна мережа (англ., Deep Neural Network)

GRU – згортковий рекурентний блок (англ., Gated Recurrent Unit)

GPU – графічний процесор (англ., Graphics Processing Unit)

JSON – об'єктна нотація JavaScript (англ., JavaScript Object Notation)

LSTM – довга короткострокова пам'ять (англ., Long Short-Term Memory)

MAE – середнє абсолютне відхилення (англ., Mean Absolute Error)

MAPE – середня абсолютна похибка у відсотках (англ., Mean absolute percentage error)

MSE – середнє квадратичне відхилення (англ., Mean Squared Error)

MLP – мультишаровий персептрон (англ., Multilayer Perceptron)

RNN – рекурентна нейронна мережа (англ., Recurrent Neural Network)

SQL – мова структурованих запитів (англ., Structured Query Language)

ВСТУП

У сучасних умовах, коли оперативна реакція на зміни та непередбачувані обставини є необхідною, прогнозування нестаціонарних часових рядів стає актуальним. Це вимагає розвитку та оптимізації методів прогнозування, які здатні ефективно адаптуватися до динамічних змін в середовищі. Одним з основних об'єктів цього дослідження є застосування рекурентних нейронних мереж для поліпшення процесів прогнозування та забезпечення достовірності в умовах невизначеності та ризику.

В даній роботі розглядається актуальне завдання прогнозування часових рядів, яке має важливе значення в різних сферах, включаючи економіку, фінанси, та науку про дані. Особлива увага приділяється нестаціонарності та динамічним змінам у часових рядах, а також необхідності врахування цих особливостей для досягнення високої точності прогнозів. Зокрема, розглядаються різні методи та підходи, включаючи застосування рекурентних нейронних мереж для аналізу та прогнозування часових рядів у складних умовах.

У зв’язку з необхідністю пошуку кращих методів прогнозування, це дослідження спрямоване на оцінку ефективності рекурентних нейронних мереж порівняно з іншими методами. Зокрема, досліджується їхня здатність адаптуватися до складних змін у часових рядах та забезпечувати достовірні прогнози для нестаціонарних даних.

Для досягнення мети проводиться аналіз широкого спектру наукових джерел з прогнозування часових рядів та глибокого навчання. У результаті дослідження можна визначити ключові переваги рекурентних нейронних мереж та їх значущість серед сучасних інструментів прогнозування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд теорії часових рядів

Важливими концепціями, що характеризують часові ряди, є стаціонарність, тренд, сезонність та інші [1].

Стаціонарність є однією з основних характеристик часових рядів. Стаціонарний часовий ряд має постійну середню та коливання, що не залежать від часу. Такі ряди легше прогнозувати та аналізувати, оскільки їх властивості залишаються сталими з часом [1].

Тренд представляє собою довгострокову зміну часового ряду в певному напрямку. Він може бути зростаючим або спадаючим і вказує на загальний напрямок розвитку даних з плином часу [2].

Сезонність відображає циклічні зміни у часовому ряді, які повторюються з однаковими або схожими інтервалами. Наприклад, сезонність може бути пов'язана з щорічними або квартальними циклами у даних [3].

Для аналізу часових рядів використовуються різноманітні методи, серед яких слід зазначити найпоширеніші – декомпозиція, фільтрація та аналіз автокореляції [4].

Декомпозиція дозволяє розділити часовий ряд на складові частини, такі як тренд, сезонність та випадкова складова. Це допомагає визначити загальну структуру та закономірності ряду [5]. Фільтрація використовується для видалення шумів та інших непотрібних складових з даних [7]. Аналіз автокореляції допомагає виявити взаємозв'язок між значеннями часового ряду у різні моменти часу.

Методи аналізу допомагають визначити основні характеристики часових рядів та використовуються для побудови моделей прогнозування та прийняття рішень на основі аналізу часових даних [8].

1.2 Сучасні методи прогнозування часових рядів

Сучасні методи прогнозування часових рядів охоплюють широкий спектр підходів, включаючи статистичні методи, методи машинного навчання та глибоке навчання [9].

Статистичні методи прогнозування, такі як експоненційне згладжування, ARIMA (Autoregressive Integrated Moving Average, авторегресійна інтегрована змінна ковзна середня) та економетричні моделі, доволі поширені у практиці [10]. Вони базуються на статистичних властивостях даних і дозволяють прогнозувати майбутні значення на основі попередніх спостережень. Однак ці методи часто мають обмеження у прогнозуванні нестаціонарних часових рядів через їхню складність та недосконаліть у врахуванні складних залежностей між даними.

Методи машинного навчання, такі як методи лінійної та поліноміальної регресії, дерева рішень, випадковий ліс та градієнтний бустінг, набули популярності завдяки їхній здатності працювати з великими обсягами даних та виявляти складні залежності [11]. Ці методи можуть бути ефективними в прогнозуванні нестаціонарних часових рядів, але вони також можуть бути схильними до перенавчання та вимагати великої кількості даних для ефективної роботи.

Глибоке навчання, зокрема нейронні мережі, в останні роки набуло значного впливу на область прогнозування часових рядів. Рекурентні нейронні мережі (RNN) та їхні модифікації, такі як Long Short-Term Memory (LSTM) та Gated Recurrent Unit (GRU), показали високу ефективність у роботі з послідовними даними, включаючи часові ряди [12]. Вони здатні виявляти складні закономірності в даних та забезпечують гнучкість у моделюванні нестаціонарних часових рядів. Кожен з цих методів має свої переваги та обмеження, і вибір конкретного методу прогнозування залежить від конкретних вимог задачі та характеристик даних.

1.3 Аналіз існуючих досліджень

Застосування рекурентних нейронних мереж (RNN) для прогнозування нестаціонарних часових рядів є важливою областю досліджень у сучасній науці. Огляд робіт на дану тему дозволяє визначити досягнення, тенденції та нерозв'язані проблеми у цій області.

1.3.1 Long Short-Term Memory Neural Networks for Water Demand Forecasting

Дослідження «Forecasting Water Demand With the Long Short-Term Memory Deep Learning Mode» [13] присвячене застосуванню рекурентних нейронних мереж (RNN), зокрема моделі LSTM (Long Short-Term Memory), для прогнозування споживання води. Автори досліджують ефективність цієї моделі порівняно з традиційними методами прогнозування, такими як статистичні моделі або класичні нейронні мережі. У рамках дослідження було зібрано історичні дані про споживання води за період у п'ять років, що включали щоденні, тижневі та місячні показники. Дані були очищені та попередньо оброблені для видалення аномалій та заповнення пропущених значень. Зокрема, було використано метод інтерполяції для заповнення відсутніх даних, що становили близько 2% від загального обсягу.

Методика розглянутого дослідження включає збір і аналіз накопичених історичних даних про споживання води, розробку та навчання моделей LSTM на основі цих даних, а також оцінку якості прогнозів на тестових даних. Модель LSTM була налаштована з використанням гіперпараметрів, таких як кількість шарів (два шари LSTM з 50 нейронами в кожному), кількість епох навчання (100 епох), та розмір batch (32). Для навчання моделі було використано 70% даних, решта 30% були залишені для тестування.

Основною метою роботи є виявлення переваг і обмежень застосування

LSTM в прогнозуванні нестаціонарних часових рядів, таких як зібрані показники щодо споживання води, з урахуванням сезонності, зовнішніх факторів та часових залежностей. Зокрема, дослідники враховували вплив погодних умов, святкових періодів та інших факторів, що можуть впливати на споживання води.

Результати дослідження показали, що модель LSTM значно перевершує традиційні методи прогнозування. Середньоквадратична помилка (MSE) для моделі LSTM склала 15,3, тоді як для класичної нейронної мережі це значення було 25,7, а для статистичної моделі ARIMA – 30,2. Ці результати дозволяють зробити висновок про високу точність і ефективність моделі LSTM в прогнозуванні споживання води на основі історичних даних. Це підтверджує можливість застосування рекурентних нейронних мереж у задачах прогнозування часових рядів та їх важливість для вирішення практичних завдань у сфері водопостачання та управління ресурсами.

Таким чином, дослідження підтверджує, що модель LSTM здатна адекватно обробляти складні часові залежності та сезонні коливання у даних про споживання води, що свідчить про її доцільність для прогнозування та планування в галузі управління водними ресурсами.

1.3.2 Forecasting stock prices using Long Short-Term Memory Recurrent Neural Networks

Дослідження «Прогнозування цін на акції за допомогою рекурентних нейронних мереж з довгостроковою короткочасною пам'яттю» [14] присвячене використанню рекурентних штучних нейронних мереж (Recurrent neural network, RNN), зокрема моделі LSTM (Long Short-Term Memory), для прогнозування цін на акції. Автори досліджують ефективність цієї моделі порівняно з традиційними методами прогнозування, такими як аналітичні та статистичні моделі.

У рамках дослідження було зібрано історичні дані про ціни на акції

декількох компаній за період у десять років. Дані були очищені від аномалій і приведені до єдиного формату для подальшого аналізу. В процесі попередньої обробки було використано нормалізацію, щоб дані знаходилися в одному діапазоні, що покращило б навчання моделі. Крім цього, була проведена декомпозиція часових рядів для виділення трендів і сезонних коливань.

Під час дослідження автори зібрали та проаналізували історичні дані про ціни на акції, після чого розробили та навчили моделі LSTM на основі цих даних. Модель була налаштована з використанням декількох гіперпараметрів: кількість шарів LSTM (три шари), кількість нейронів у кожному шарі (50 нейронів на шар), кількість епох навчання (200 епох), та розмір batch (64). Для навчання моделі було використано 80% даних, решта 20% були залишені для тестування.

Оцінка якості короткострокових прогнозів часових рядів зібраних даних була здійснена на тестових вибірках з використанням різних метрик оцінки якості прогнозу, таких як середня абсолютна помилка (MAE) і середньоквадратична помилка (MSE). За результатами тестування, модель LSTM продемонструвала високу точність прогнозування. Значення MAE для моделі LSTM становило 1,8, а MSE – 5,4, що значно краще порівняно з традиційними методами. Наприклад, MAE для класичної регресійної моделі становило 3,2, а для статистичної моделі ARIMA – 4,6.

Результати дослідження «Stock Market Prediction Using Long Short-Term Memory» дозволяють зробити висновок про високу точність і ефективність моделі ШНМ Long Short-Term Memory для вирішення задачі короткострокового прогнозування цін на акції.

Розглянуте дослідження свідчить про можливість застосування рекурентних штучних нейронних мереж у фінансовому прогнозуванні та їх важливість для вирішення практичних завдань у сфері фінансових ринків та інвестицій. Крім того, модель LSTM продемонструвала здатність враховувати складні часові залежності і тренди, що особливо важливо для

фінансових даних, які характеризуються високою волатильністю та нестаціонарністю.

1.3.3 Time series forecasting with deep learning

Дослідження «Financial time series forecasting with deep learning: A systematic literature review» [15] є систематичним оглядом літератури з питань прогнозування нестаціонарних часових рядів різного походження за допомогою підходів глибокого навчання. Автори провели ретельний аналіз наукових статей, присвячених застосуванню глибокого навчання в задачах прогнозування часових рядів.

Під час дослідження були розглянуті різні аспекти застосування глибокого навчання для часових рядів, такі як архітектури нейронних мереж, типи завдань прогнозування, використані методи оцінки якості прогнозів та інші. Автори систематизували та класифікували представлені в літературі підходи та методи, виявивши їх переваги, недоліки та тенденції розвитку. Особлива увага була приділена порівнянню різних архітектур нейронних мереж, включаючи LSTM, GRU (Gated Recurrent Unit) та CNN (Convolutional Neural Networks), а також їхніх гібридних варіантів.

Під час аналізу автори дослідили більше 100 наукових праць, охоплюючи період з 2010 по 2022 рік. Розглядалися такі аспекти, як ефективність моделей прогнозування часових рядів на різних наборах даних, вплив попередньої обробки даних на результати, а також адаптивні методи навчання моделей. Було відзначено, що більшість досліджень фокусуються на короткостроковому прогнозуванні, тоді як довгострокове прогнозування все ще є проблемою через зростаючу невизначеність у часі зі збільшенням горизонту прогнозування.

Результати дослідження дають можливість отримати огляд сучасних підходів і досягнень стосовно прогнозування часових рядів за допомогою глибокого навчання. Це дозволяє дослідникам та практикам орієнтуватися в

актуальних методах та тенденціях у цій галузі досліджень, зокрема у виборі відповідних моделей та алгоритмів для специфічних завдань прогнозування.

1.3.4 Electric load forecasting using neural networks

Дослідження «Term Electric Load Forecasting Using Neural Networks» [16] присвячене прогнозуванню електричного навантаження за допомогою нейронних мереж. Автори розглядають застосування різних архітектур нейронних мереж для прогнозування електричного навантаження, що є важливою задачею для енергетичних компаній та організацій, які займаються енергопостачанням.

Автори розглянутого дослідження проводять аналіз ефективності штучних нейронних мереж у порівнянні з традиційними методами прогнозування навантаження електричної мережі, такими як статистичні моделі прогнозування часових рядів. Автори також досліджують різні аспекти, такі як вплив вибору архітектури мережі, обсягу даних та методів навчання на точність прогнозів. Наприклад, було виявлено, що моделі LSTM демонструють вищу точність прогнозування у порівнянні з класичними методами ARIMA і ETS (Exponential Smoothing). У моделі LSTM середня абсолютна помилка (MAE) склала 1,25, у той час як для ARIMA та ETS ці показники становили 2,05 та 1,95 відповідно.

Автори також аналізують вплив обсягу даних на продуктивність моделей. Вони встановили, що моделі, навчені на більших наборах даних, зазвичай показують кращі результати. Наприклад, використання навчальних наборів даних розміром 10000 записів дозволило знизити середньоквадратичну помилку (MSE) до 1,8, тоді як при використанні 5000 записів цей показник становив 2,4. Використання методів крос-валідації та регуляризації допомогло уникнути перенавчання моделей і підвищити їхню здатність до узагальнення.

Результати дослідження «Term Electric Load Forecasting Using Neural

Networks» дозволяють оцінити застосовність штучних нейронних мереж для задач прогнозування електричного навантаження та виявити їх переваги та обмеження в даному контексті. Зокрема, автори відзначають, що нейронні мережі можуть краще враховувати складні нелінійні залежності в даних та адаптуватися до змін у споживанні електроенергії. Водночас, автори зазначають, що для досягнення максимальної ефективності необхідно ретельно налаштовувати параметри моделей та забезпечувати якісні вихідні дані для навчання.

Наприклад, для досягнення MAE нижче 1,5, важливо використовувати регулярні оновлення навчальних даних та проводити перенавчання моделей не рідше ніж раз на квартал. Це може бути корисним для енергетичних компаній та фахівців у сфері енергетики при розробці більш точних та ефективних методів прогнозування навантаження.

1.3.5 Traffic flow prediction with big data: A deep learning approach

Дослідження «Прогнозування потоку транспорту з використанням великих даних: підхід глибокого навчання» [17] присвячене прогнозуванню транспортного потоку за допомогою методів глибокого навчання на великих обсягах даних. Автори розглядають актуальну проблему прогнозування трафіку, яка має велике значення для управління транспортними потоками та оптимізації інфраструктури міського транспорту.

У дослідженні застосовуються різні архітектури глибоких нейронних мереж, такі як CNN (Convolutional Neural Networks), LSTM (Long Short-Term Memory) та GRU (Gated Recurrent Unit), для прогнозування обсягів потоку транспортних засобів на основі великих обсягів даних про рух транспорту. Автори вивчають ефективність таких методів на реальних даних про рух транспорту з різних джерел, таких як дорожні камери, датчики руху та інші. Наприклад, дані з 50 дорожніх камер та 100 датчиків руху протягом одного року були використані для навчання моделей.

Результати дослідження демонструють, що модель LSTM показала найкращу точність прогнозування з середньою абсолютною помилкою (MAE) 3,2 автомобіля на годину, тоді як моделі CNN та GRU мали MAE 4,1 та 3,7 відповідно. Крім того, середньоквадратична помилка (MSE) для LSTM складала 15,8, для CNN – 20,3, а для GRU – 18,6. Також автори дослідили вплив обсягу даних на точність прогнозування і виявили, що збільшення обсягу навчальних даних з 6 місяців до 12 місяців дозволило знизити MAE для LSTM моделі з 4,5 до 3,2 автомобіля на годину.

У рамках дослідження також було вивчено вплив різних гіперпараметрів, таких як кількість шарів та розмір вікна прогнозування, на точність моделей. Було виявлено, що використання трьох шарів LSTM з розміром вікна прогнозування 10 хвилин забезпечило найкращі результати. Зокрема, MAE зменшилася з 3,9 до 3,2 автомобіля на годину при переході з двох до трьох шарів LSTM.

Результати дослідження «Traffic flow prediction with big data: A deep learning approach» дозволяють оцінити застосовність методів глибокого навчання для прогнозування трафікового потоку та виявити їх переваги порівняно з традиційними методами прогнозування. Зокрема, LSTM моделі штучних нейронних мереж продемонстрували високу точність у порівнянні з класичними методами ARIMA та ETS, які мали MAE 5,5 та 5,2 автомобіля на годину відповідно. Це може бути корисним для міських адміністрацій та організацій, що займаються управлінням транспортними потоками, при прийнятті рішень з оптимізації дорожньої інфраструктури та поліпшення умов дорожнього руху.

1.3.6 Forecasting energy consumption using LSTM neural networks

Дослідження «Прогнозування споживання енергії за допомогою нейронних мереж LSTM» [18] присвячене прогнозуванню енергоспоживання з використанням нейронних мереж з довгостроковою короткочасною

пам'яттю (LSTM). Автори досліджують ефективність застосування LSTM для прогнозування енергоспоживання на основі історичних даних про споживання енергії.

У рамках дослідження проводиться аналіз часових рядів даних про споживання енергії, включаючи фактори, такі як погода, час доби, день тижня та святкові дні. Було зібрано дані за останні п'ять років, що містять інформацію про щогодинне споживання енергії, температуру повітря, вологість, а також календарні дані. Потім автори розробляють модель нейронної мережі LSTM та навчають її на історичних даних для прогнозування майбутнього споживання енергії. Навчання моделі здійснювалось на 80% зібраних даних, тоді як 20% даних використовувались для тестування.

Результати дослідження показали, що модель LSTM демонструє високу точність прогнозування споживання енергії. Середня абсолютна помилка (MAE) для прогнозів на один день вперед становила 2,5, а середньоквадратична помилка (MSE) – 9,6. Для порівняння, традиційні методи, такі як метод ковзного середнього (MA) та модель авторегресії (AR), мали MAE 4,2 та 3,8 відповідно.

Крім того, автори досліджували вплив різних гіперпараметрів, таких як кількість шарів LSTM та розмір вікна прогнозування, на точність моделі. Було виявлено, що оптимальна конфігурація включала три шари LSTM з розміром вікна 24 години, що забезпечувало найменшу помилку прогнозування. Зокрема, використання більшої кількості шарів (п'ять і більше) призводило до перенавчання моделі та збільшення похибок прогнозування.

Результати дослідження «Forecasting energy consumption using LSTM neural networks» дозволяють оцінити точність і надійність прогнозів споживання енергії, отриманих з використанням нейронних мереж LSTM. Наприклад, під час тестування модель LSTM передбачила пікове споживання енергії з відхиленням всього в 1,8 від фактичного значення. Це може бути

корисним для енергетичних компаній і організацій, що займаються плануванням виробництва та розподілом енергії, для оптимізації процесів управління енергоспоживанням та запобігання можливим проблемам з нестачею енергії.

1.3.7 Weather prediction using recurrent neural networks

Дослідження «Прогнозування погоди за допомогою рекурентних нейронних мереж» [19] присвячене прогнозуванню погоди з використанням рекурентних нейронних мереж (RNN). Автори досліджують ефективність застосування рекурентних нейронних мереж для прогнозування погодних умов на основі історичних метеоданих.

У рамках дослідження проводиться аналіз часових рядів даних про погоду, включаючи температуру, вологість, атмосферний тиск та інші параметри. Для цього було зібрано дані за останні десять років з щогодинною деталізацією, що містять інформацію про температуру, вологість, атмосферний тиск, швидкість вітру та опади. Потім автори розробляють модель рекурентної нейронної мережі та навчають її на історичних даних для прогнозування майбутніх погодних умов. Навчання моделі здійснювалося на 70% зібраних даних, тоді як 30% даних використовувалися для тестування.

Результати дослідження показали, що модель RNN демонструє високу точність прогнозування погодних умов. Середня абсолютна помилка (MAE) для прогнозів температури на один день вперед становила 1,2, а середньоквадратична помилка (MSE) – 2,5. Для порівняння, традиційні методи прогнозування, такі як лінійна регресія та методи ковзного середнього, мали MAE 2,4 та 2,1 відповідно.

Крім того, автори досліджували вплив різних гіперпараметрів, таких як кількість шарів RNN та розмір вікна прогнозування, на точність моделі. Було виявлено, що оптимальна конфігурація включала два шари RNN з розміром вікна 24 години, що забезпечувало найменшу помилку прогнозування.

Наприклад, використання трьох шарів RNN призводило до перенавчання моделі та збільшення похибок прогнозування до MAE 1,5.

Результати дослідження дозволяють оцінити точність та надійність прогнозів погоди, отриманих з використанням рекурентних нейронних мереж. Наприклад, під час тестування модель RNN передбачила швидкість вітру з відхиленням лише 1,3 від фактичного значення. Це може бути корисним для метеорологічних служб, сільськогосподарських підприємств, а також для широкого кола промислових та комерційних організацій, що залежать від точних прогнозів погоди для прийняття рішень.

1.4 Постановка задачі

Мета даного дослідження полягає у розробці ефективних методів прогнозування нестаціонарних часових рядів на основі рекурентних нейронних мереж (RNN). Основні цілі включають в себе аналіз існуючих підходів до прогнозування часових рядів, виявлення їх переваг та обмежень, а також дослідження попередніх робіт у даній області з метою визначення основних тенденцій та результатів.

Об'єктом дослідження є методи прогнозування нестаціонарних часових рядів, а предметом дослідження є використання рекурентних нейронних мереж (RNN), зокрема їх модифікацій, для підвищення точності прогнозування часових рядів. У межах цього дослідження планується провести огляд літератури з питань застосування рекурентних нейронних мереж для прогнозування часових рядів, виявити основні підходи та їх ефективність. Визначити переваги та обмеження існуючих методів прогнозування часових рядів на основі RNN, а також розробити модель RNN для прогнозування часових рядів, враховуючи специфіку даних.

Дослідження включатиме серію експериментів для оцінки впливу різних гіперпараметрів на ефективність прогнозування. Тестування розробленої моделі на реальних даних дозволить порівняти отримані

результати з традиційними методами прогнозування. Подальший аналіз результатів експериментів дозволить оцінити застосовність розробленої моделі в практичних задачах прогнозування часових рядів. Визначення можливих напрямків подальших досліджень у цій області стане заключним етапом роботи.

Збір та підготовка даних передбачає збирання історичних даних для різних типів часових рядів, таких як фінансові дані, дані про споживання енергії та погодні дані. Обробка даних включатиме нормалізацію та заповнення відсутніх значень. Розробка моделі RNN включатиме створення кількох варіантів RNN, включаючи прості RNN, LSTM та GRU, з різними архітектурами та кількістю шарів.

Експерименти спрямовані на визначення оптимальних гіперпараметрів, таких як розмір вікна прогнозування, кількість нейронів у шарах, швидкість навчання та кількість епох. Навчання та тестування моделей будуть проводитися на навчальних наборах даних, а оцінка ефективності здійснюватиметься за допомогою метрик MAE (середня абсолютна помилка) та MAPE (середня абсолютна відсоткова помилка).

Після завершення експериментів буде проведено аналіз результатів прогнозування для різних моделей та гіперпараметрів, що дозволить визначити найбільш ефективні конфігурації. Порівняння з традиційними методами прогнозування, такими як ARIMA та експоненціальне згладжування, дозволить оцінити переваги RNN. Результати дослідження будуть візуалізовані для полегшення інтерпретації та порівняння з реальними даними. На завершення буде підготовлено звіт з висновками та рекомендаціями щодо практичного застосування розроблених моделей.

2 АНАЛІЗ ВИКОРИСТОВУВАНИХ МЕТОДІВ

2.1 Огляд методів прогнозування

2.1.1 Статистичні методи прогнозування на основі експоненційного згладжування та ARIMA

Аналіз статистичних методів, таких як експоненційне згладжування та ARIMA, є важливою частиною дослідження в галузі прогнозування часових рядів [20].

Експоненційне згладжування – це один з базових методів прогнозування, який використовується для визначення тренду в часовому ряді та прогнозу майбутніх значень [16]. Він базується на понятті експоненційної ваги для попередніх значень ряду, де вага кожного з них зменшується експоненційно з кожним наступним кроком часу. Цей метод дозволяє враховувати найновіші дані більшою мірою, ніж старі, що робить його особливо корисним для прогнозування в умовах тренду, що змінюється, або сезонності.

ARIMA (Autoregressive Integrated Moving Average) – це статистичний метод, який включає в себе моделі авторегресії (AR), моделі рухомих середніх (MA) та інтегрованість (I) [14]. ARIMA моделі дозволяють моделювати складні залежності між значеннями часового ряду та здійснювати прогнози на основі цих залежностей. Цей метод особливо корисний для прогнозування часових рядів з вираженими сезонними або трендовими патернами.

Проведений аналіз статистичних методів прогнозування часових рядів дозволяє оцінити їхню ефективність та придатність для конкретних нестаціонарних часових рядів, а також визначити оптимальний підхід до прогнозування в кожному конкретному випадку.

2.1.2 Методи машинного навчання для прогнозування часових рядів

Вивчення методів машинного навчання для прогнозування, таких як регресійні моделі, дерева рішень та градієнтний бустінг, є необхідним для розробки моделей прогнозування часових рядів [18].

Регресійні моделі, такі як лінійна або поліноміальна регресія, є основою методів машинного навчання [18]. Вони моделюють залежність між вхідними змінними та вихідними значеннями, дозволяючи прогнозувати майбутні значення на основі відомих факторів.

Дерева рішень – це методи, які розділяють простір ознак на набір бінарних розділів [21]і розглядають кожний вузол як питання про функцію ознаки. Цей метод дозволяє автоматично визначати важливість ознак та знаходити найоптимальніші шляхи розділення даних для прогнозування.

Градієнтний бустінг – це ансамблевий метод машинного навчання, який базується на ітеративному покращенні моделей [22]. Він використовує метод послідовного навчання слабких моделей, які компенсують певні недоліки, з метою покращення точності кінцевої моделі.

Вивчення цих методів дозволяє вибрати підхід для конкретного набору даних та задачі прогнозування. Кожен з цих методів має свої переваги та обмеження, які необхідно врахувати при виборі оптимального методу для прогнозування часових рядів.

2.2 Порівняння методів

2.2.1 Критерії оцінки точності та ефективності методу прогнозування

Оцінка точності та ефективності кожного методу на використаних даних є критично важливою для визначення його придатності для конкретного набору даних і задачі прогнозування [23]. Для цього можна

використовувати різні метрики, такі як середньоквадратична помилка (MSE), середня абсолютна помилка (MAE), коефіцієнт детермінації (R^2), або інші, в залежності від специфіки даних та вимог щодо точності прогнозування.

Під час оцінки точності кожного методу важливо врахувати його здатність до моделювання складних залежностей у даних, а також його чутливість до різноманітних факторів, таких як тренди, сезонність, аномалії тощо [24].

Додатково, ефективність кожного методу може бути оцінена з урахуванням часу, необхідного для навчання та прогнозування. Методи, які вимагають менше часу, з практичної точки зору є більш доцільними.

Щоб отримати об'єктивну оцінку ефективності кожного методу, зазвичай використовують перехресну перевірку або розділення даних на тренувальний та тестовий набори [25]. Це допомагає уникнути перенавчання моделі та забезпечити більш об'єктивну оцінку її здатності до прогнозування на нових даних.

2.2.2 Порівняння часу навчання та прогнозування розглянутих методів

Порівняння часу навчання та прогнозування для кожного методу є важливим аспектом аналізу ефективності моделей прогнозування часових рядів.

Час навчання визначає час, необхідний для побудови моделі на тренувальних даних. Для статистичних методів, таких як експоненційне згладжування, час навчання може бути дуже коротким, оскільки ці методи не потребують складних операцій з обчисленням параметрів моделі [26]. З іншого боку, ARIMA може вимагати значно більшого часу на використання, оскільки вона включає в себе пошук оптимальних параметрів моделі.

Щодо часу прогнозування, він визначає час, необхідний для отримання прогнозованих значень на основі вже побудованої моделі. Експоненційне згладжування, як правило, є дуже швидким у прогнозуванні, оскільки воно

вимагає лише обчислення зважених середніх [27]. З іншого боку, ARIMA може вимагати більшого часу для прогнозування, оскільки це включає в себе складні операції з рекурсивного оновлення параметрів моделі.

Порівняння цих часових характеристик допомагає визначити найбільш ефективний метод прогнозування залежно від конкретних вимог щодо швидкості та точності прогнозу.

2.3 Переваги та обмеження використовуваних методів

Переваги та обмеження розглянутих методів прогнозування потрібно враховувати при виборі підходу для створення моделей прогнозування нестаціонарних часових рядів.

Регресійні моделі, такі як лінійна регресія, прості в інтерпретації та мають низьку обчислювальну складність [28]. Вони використовуються для роботи з даними, які мають лінійну або нелінійну залежність. Однак регресійні моделі можуть бути менш ефективними для складних, нелінійних або нестаціонарних часових рядів.

Дерева рішень відомі своєю здатністю автоматично враховувати важливість ознак та оброблювати як категоріальні, так і числові дані без необхідності попередньої обробки [29]. Глибокі дерева використовуються у випадках, коли є ризик перенавчання [29]. Крім того, дерева рішень можуть бути менш точними в порівнянні з іншими складнішими моделями [29].

Градієнтний бустінг дозволяє створювати складні ансамблі моделей, які можуть вирішувати різноманітні задачі [14]. Однак градієнтний бустінг може бути чутливим до перенавчання, а також може вимагати більшого обсягу даних та обчислювальних ресурсів для навчання.

Загалом, кожен метод має свої переваги та обмеження, і вибір оптимального підходу залежить від конкретних вимог до задачі прогнозування (необхідна точність, розмір горизонту прогнозування) та характеристик даних (кількість значень ЧР, наявність викривлень тощо).

2.4 Врахування специфіки даних

Врахування специфіки вихідних даних є важливим аспектом при виборі та налаштуванні методів прогнозування нестаціонарних часових рядів.

Перш за все, необхідно детально проаналізувати самі дані, включаючи їх структуру, розподіл, наявність трендів, сезонності, аномалій та інші важливі характеристики [30]. Це дозволить визначити природу даних та їх можливі взаємозв'язки.

Потрібно врахувати часові залежності (кореляцію та автокореляцію) та динаміку даних. Деякі методи, наприклад, ARIMA, можуть бути ефективнішими для моделювання часових залежностей та трендів, тоді як інші, такі як градієнтний бустінг, можуть працювати в умовах наявності складних нелинійних залежностей [31].

Також важливо врахувати наявність аномалій та викидів у даних, оскільки вони можуть вплинути на якість прогнозу. Деякі методи можуть бути чутливі до аномалій та вимагати додаткової уваги при їх обробці та моделюванні [32]. Крім того, необхідно розглянути рівень шуму в даних та можливість його фільтрації чи врахування в моделі. Шум може впливати на точність прогнозу та вимагати застосування методів фільтрації або регуляризації для покращення результатів [33].

Для успішного прогнозування нестаціонарних часових рядів потрібно враховувати специфіку даних, ретельно аналізувати їх та підбирати відповідний метод моделювання.

2.5 Рекурентні нейронні мережі

Рекурентні нейронні мережі (RNN) – це клас нейронних мереж, призначений для обробки послідовних даних, таких як текст, аудіо, відео або

часові ряди. Основною особливістю RNN є їхня здатність до збереження попередньої інформації у внутрішньому стані та використання її для обробки наступних вхідних даних. Це дозволяє їм ефективно працювати з даними, що мають послідовну структуру, такими як речення, часові ряди тощо [34].

Архітектура RNN складається з набору нейронів, які взаємодіють між собою через зворотні зв'язки. Кожен нейрон приймає вхідні дані та попередній стан мережі, обчислює зважену суму цих даних і застосовує функцію активації для генерації виходу. При цьому вихід нейрону також може бути використаний як вхід для наступного часового кроку [35].

Основні типи RNN включають Simple RNN (прості рекурентні мережі), LSTM (довгострокова краткочасна пам'ять) та GRU (гейтовані рекурентні одиниці). Кожен з цих типів має свої особливості та переваги. Simple RNN є найбільш базовим типом RNN і використовується для моделювання послідовних даних. LSTM та GRU є розширеннями Simple RNN, які були розроблені для вирішення проблем з затуханням або нестабільністю градієнтів та здатні до більш ефективного моделювання довгострокових залежностей в даних [36].

У прогнозуванні часових рядів RNN використовуються для моделювання та прогнозування майбутніх значень ряду на основі його попередніх значень. Вони можуть бути ефективними у виявленні складних залежностей у динаміці часових рядів і можуть бути адаптовані до різних типів прогнозування, включаючи короткострокові та довгострокові прогнози [37].

2.6 Технологічні аспекти застосування RNN у прогнозуванні

Технологічні аспекти застосування рекурентних нейронних мереж (RNN) у прогнозуванні часових рядів включають в себе використання різноманітних програмних інструментів та бібліотек для реалізації моделей RNN, а також технічні аспекти навчання та використання цих моделей [38].

У сфері програмних інструментів і бібліотек найпопулярнішими є TensorFlow, PyTorch та Keras. Ці бібліотеки надають широкий спектр можливостей для побудови, навчання та використання різноманітних архітектур нейронних мереж, включаючи RNN. Вони також забезпечують різноманітні інструменти для візуалізації даних, налаштування моделей та оцінки їхньої ефективності [39].

У технічних аспектах навчання та використання RNN для прогнозування нестаціонарних часових рядів важливою є попередня обробка даних. Підготовка даних включає в себе їхнє масштабування, нормалізацію, розбиття на тренувальний та тестовий набори та можливу агрегацію та видалення аномалій [15].

Гіперпараметризація визначає оптимальні значення параметрів моделі RNN, таких як кількість шарів, кількість нейронів у кожному шарі, швидкість навчання та інші. Цей процес може бути здійснений за допомогою методів налаштування гіперпараметрів, таких як зовнішній перехресний перевірочний набір або оптимізація гіперпараметрів з використанням алгоритмів оптимізації, наприклад, Random Search або Grid Search [40].

Оптимізація моделі включає в себе використання різноманітних алгоритмів, таких як стохастичний градієнтний спуск або метод оптимізації ADAM, для знаходження оптимальних ваг нейронів моделі, які мінімізують функцію втрати та покращують прогностичні здібності моделі [13].

Розгляд технологічних аспектів застосування RNN у прогнозуванні часових рядів є важливим для ефективної розробки та впровадження моделей прогнозування нестаціонарних часових рядів, таких як ряди фінансових даних, з використанням зазначеної технології.

2.7 Потенційні вдосконалення та дослідження

Після аналізу методів прогнозування та їхнього застосування для нестаціонарних часових рядів, важливо розглянути потенційні вдосконалення

та напрямки подальших досліджень.

Один з можливих напрямків удосконалення полягає у використанні глибоких нейронних мереж з більш складними архітектурами, такими як CNN (згорткові нейронні мережі), або Transformer, які можуть краще моделювати складні нестаціонарні залежності в часових рядах [41].

Важливою є розробка методів адаптивного прогнозування, які можуть адаптуватися до змін у структурі нових даних та автоматично коригувати параметри моделі відповідно до динаміки часового ряду [42].

Дослідження нових методів обробки та підготовки даних також може значно покращити якість прогнозів. Наприклад, застосування технік аугментації даних або автоматичного видалення аномалій може допомогти покращити стабільність та точність моделей [43].

Крім того, важливо дослідити ефективність різних підходів до ансамблювання моделей для прогнозування нестаціонарних часових рядів. Комбінація різних моделей може допомогти зменшити ризик перенавчання та підвищити загальну точність прогнозів [44].

Для підвищення точності та надійності прогнозування нестаціонарних часових рядів необхідно вдосконалювати існуючі методи та досліджувати нові підходи. Це включає використання глибоких нейронних мереж із складними архітектурами, розробку адаптивних методів прогнозування, вдосконалення методів обробки та підготовки даних, а також дослідження ефективності ансамблювання моделей.

3 РЕАЛІЗАЦІЯ МЕТОДІВ ПРОГНОЗУВАННЯ

3.1 Використані технології

3.1.1 Мова програмування Python

Мова програмування Python обрана для реалізації методів прогнозування через наявний широкий спектр бібліотек та фреймворків, спрямованих на обробку даних, машинне навчання та статистичний аналіз. Python відомий своєю простотою використання, зручним синтаксисом та великою спільнотою користувачів, що робить його популярним вибором для розробників у сфері аналізу даних та машинного навчання.

Використання Python дозволяє ефективно використовувати існуючі бібліотеки для обробки даних, візуалізації результатів та розробки моделей прогнозування. Бібліотеки, такі як Pandas, NumPy, Matplotlib, Seaborn, TensorFlow та scikit-learn, надають широкий набір інструментів для роботи з даними та створення моделей машинного навчання.

Python також підтримує різні парадигми програмування, включаючи процедурне, об'єктно-орієнтоване та функціональне програмування, що дозволяє розробникам створювати ефективні та модульні рішення для різних завдань прогнозування.

Вибір мови програмування Python є обґрунтованим для реалізації методів прогнозування через її зручність, ефективність та широкий спектр доступних інструментів та бібліотек.

3.1.2 Бібліотека TensorFlow для машинного навчання

Бібліотека TensorFlow є однією з найпопулярніших бібліотек для машинного навчання, зокрема, підходів глибокого навчання. Вона

розроблена компанією Google і використовується для різноманітних завдань, включаючи класифікацію, регресію, кластеризацію та прогнозування.

Основною перевагою TensorFlow є гнучкість та висока продуктивність. Вона має велику кількість оптимізованих алгоритмів для навчання нейронних мереж на графічних процесорах (GPU) та центральних процесорах (CPU), що дозволяє швидко та ефективно працювати з великими обсягами даних.

TensorFlow надає широкий набір інструментів для створення, навчання та оцінки різноманітних моделей машинного навчання та глибокого навчання. Вона підтримує різні типи нейронних мереж, включаючи звичайні шарові нейронні мережі (DNN), згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та багатошарові перцептрони (MLP).

Крім того, TensorFlow має велику та активну спільноту користувачів, що надає доступ до документації, навчальних матеріалів та готових рішень для різних завдань машинного навчання. Це дозволяє розробникам швидко засвоювати нові концепції та реалізовувати складні моделі з високою точністю отримуваного результату.

3.1.3 Бібліотека scikit-learn для статистичного аналізу та машинного навчання

Бібліотека scikit-learn є однією з широко використовуваних бібліотек для статистичного аналізу даних та машинного навчання в середовищі Python. Вона містить широкий спектр інструментів і алгоритмів, які дозволяють виконувати різноманітні завдання, такі як класифікація, регресія, кластеризація даних, відбір ознак, оцінка моделей та багато іншого.

Однією з основних переваг scikit-learn є простота використання та інтуїтивно зрозумілий інтерфейс. Бібліотека надає легкий доступ до різноманітних алгоритмів машинного навчання через однорідний API, що дозволяє проводити експерименти з різними методами машинного навчання

та варіантами параметрів.

Scikit-learn характеризується ефективністю та швидкістю роботи. Бібліотека оптимізована для роботи з великими обсягами даних та використанням розподілених обчислювальних ресурсів, що дозволяє швидко обробляти та аналізувати великі набори вихідних даних. Також перевагою scikit-learn є його розширюваність та гнучкість. Бібліотека містить багато різноманітних алгоритмів та моделей, а також набір інструментів для підготовки даних, валідації моделей та оцінки їх продуктивності.

Бібліотека scikit-learn є важливим інструментом для розробки та реалізації моделей машинного навчання в середовищі Python, і характеризується своєю простотою використання, ефективністю та розширюваністю.

3.1.4 Pandas для обробки та аналізу даних

Бібліотека Pandas є інструментом для обробки та аналізу даних в середовищі Python. Вона надає зручні засоби для роботи з табличними даними, що дозволяє здійснювати різноманітні операції з даними швидко та ефективно. Однією з основних переваг є здатність до роботи з великими обсягами даних. Вона дозволяє зчитувати дані з різних форматів файлів, таких як CSV, Excel, SQL, JSON тощо, та ефективно виконувати операції з ними навіть при великому обсязі інформації.

Pandas також має багато корисних функцій для обробки та підготовки даних, таких як очищення даних від пропущених значень, перетворення типів даних, об'єднання, розділення та групування даних, що спрощує виконання складних маніпуляцій з даними. Крім того, Pandas надає засоби для візуалізації даних, що дозволяє аналізувати вміст даних за допомогою графіків та діаграм.

Бібліотека Pandas є важливим інструментом для роботи з даними в середовищі Python, який забезпечує широкі можливості для обробки, аналізу

та візуалізації даних, що дозволяє ефективно виконувати різноманітні завдання в аналізі даних та машинному навчанні.

3.1.5 Matplotlib та Seaborn для візуалізації даних

Бібліотеки Matplotlib та Seaborn є основними засобами для візуалізації даних в середовищі Python. Вони надають можливості для створення різноманітних графіків, діаграм та інших візуальних представлень даних, що дозволяє аналізувати та візуалізувати інформацію з різних кутів огляду.

Matplotlib є однією з найпопулярніших бібліотек для візуалізації даних в Python. Вона надає широкий спектр інструментів для створення різноманітних графіків, включаючи лінійні графіки, гістограми, точкові діаграми, кругові діаграми та багато інших. Matplotlib також надає засоби для налаштування вигляду графіків, таких як колір, стиль та розмір шрифту, що дозволяє створювати естетичні візуалізації.

Seaborn – це високорівнева бібліотека для візуалізації даних, яка базується на Matplotlib і надає простий та зручний інтерфейс для створення складних графіків. Вона містить велику кількість вбудованих стилів та палітр кольорів, а також підтримує різноманітні типи графіків, такі як box plots, violin plots, pair plots тощо. Seaborn також дозволяє використовувати статистичні методи для автоматичного розрахунку та відображення додаткової інформації на графіках, що дозволяє зробити аналіз даних більш інформативним.

Matplotlib та Seaborn є важливими інструментами для візуалізації даних в Python, які надають широкі можливості для створення професійних та інформативних візуалізацій, що допомагають аналізувати та візуалізувати дані з різних кутів.

3.2 Структура проекту

3.2.1 Розробка у відповідних Jupyter Notebook або Python скриптах

Розробка у відповідних Jupyter Notebook або Python скриптах є основним етапом реалізації методів прогнозування. Jupyter Notebook є інтерактивним середовищем для виконання коду Python у вигляді окремих «клітинок», що дозволяє поєднувати виконання коду з текстовими та візуальними поясненнями. Використання Jupyter Notebook дозволяє реалізовувати методи прогнозування та проводити аналіз результатів.

Також розробка у відповідних Python скриптах є важливим елементом процесу реалізації методів прогнозування. Python підтримує велику кількість бібліотек та інструментів для роботи з даними та машинним навчанням. Реалізація методів прогнозування у вигляді Python скриптів дозволяє створювати автоматизовані та гнучкі рішення, які можна використовувати у різних середовищах та інтегрувати у різноманітні проекти.

Обидва підходи, як використання Jupyter Notebook, так і розробка у відповідних Python скриптах, мають свої переваги та використовуються залежно від конкретних потреб та умов проекту. Jupyter Notebook підходить для інтерактивного аналізу та документування процесу реалізації, тоді як Python скрипти є більш зручними для автоматизації процесів та використання у виробничому середовищі.

3.2.2 Організація коду у директоріях «data» та «models»

Організація коду у відповідних директоріях є важливим аспектом розробки методів прогнозування. Чітка структура проекту допомагає підтримувати порядок у файловій системі, полегшує розуміння та управління кодом, а також сприяє подальшому розвитку проекту.

Директорія «data» використовується для збереження вихідних даних, які використовуються для навчання моделей та проведення прогнозування. У цій директорії можуть зберігатися різні набори даних, які використовуються

для вдосконалення та тестування моделей.

Директорія «models» призначена для збереження навчених моделей. Після тренування моделі її можна зберегти у цій директорії для подальшого використання без необхідності повторного тренування. Це дозволяє ефективно управляти навченими моделями та швидко відновлювати їх для застосування в реальних умовах.

Крім того, можуть бути створені додаткові директорії для збереження інших елементів проекту, наприклад:

- «utils» для збереження допоміжних функцій та утиліт;
- «visualizations» для збереження візуалізацій результатів та діаграм;
- «configs» для збереження конфігураційних файлів і налаштувань;
- «logs» для збереження журналів та інших ведення даних під час навчання моделей.

Ця структура дозволяє ефективно організувати код проекту та забезпечити зручний доступ до різних елементів проекту, що допомагає підтримувати порядок у розробці та управлінні проектом.

3.3 Робота з даними

3.3.1 Завантаження та очистка вхідних даних з файлів CSV або відповідних джерел даних

Завантаження та очистка вхідних даних є важливим етапом у реалізації методів прогнозування часових рядів. Початкові дані можуть бути збережені у різних форматах, таких як файли CSV, Excel, бази даних або інші відповідні джерела.

У першу чергу, використовуючи вбудовані функції чи сторонні бібліотеки, такі як Pandas у Python, дані завантажуються з файлів CSV або інших джерел та зберігаються у відповідних структурах даних: pd.read_csv('nasdaq.csv', date_format = True). Це забезпечує зручність та

швидкість обробки.

Після завантаження даних проводиться їх очистка. Цей процес включає усунення пропущених значень, видалення дублікатів, а також може включати конвертацію типів даних, якщо це необхідно.

Після завантаження та очищення даних вони можуть бути додатково оброблені або підготовлені для використання у методах прогнозування. Наприклад, це може включати вибір певних колонок, розділення даних на тренувальний та тестовий набори, а також може бути застосовано масштабування або нормалізація даних, якщо це необхідно для певних методів прогнозування.

3.3.2 Розділення даних на тренувальний та тестовий набори

Після завантаження та очищення вхідних даних важливим кроком є їх розділення на тренувальний та тестовий набори. Це дозволяє оцінити ефективність моделі на даних, що не використовувалися під час навчання, і перевірити її здатність до узагальнення.

Зазвичай дані розділяються випадковим чином на два піднабори: тренувальний набір, який використовується для навчання моделі, і тестовий набір, який використовується для оцінки її ефективності. Зазвичай відводиться більша частина даних для тренування (наприклад, 70-80%), а решта – для тестування (20-30%).

Розділення даних на тренувальний (data[data['Date']<'2023-01-01'].copy()) та тестовий (data[data['Date']>='2023-01-01'].copy()) набори допомагає уникнути перенавчання моделі та дозволяє оцінити її загальну здатність до узагальнення на нових даних. Це важливий етап у процесі розробки та оцінки моделі прогнозування, оскільки дозволяє забезпечити об'єктивність та достовірність результатів.

Після розділення даних на тренувальний та тестовий набори можна переходити до навчання моделей на тренувальних даних та оцінки їх

ефективності на тестових даних. Відповідно до результатів тестування можуть бути внесені корективи у модель або у методи прогнозування в цілому для досягнення кращих результатів.

3.3.3 Використання методів для обробки та підготовки даних для подальшого аналізу та моделювання

Використання Pandas для обробки та підготовки даних є ключовою складовою розробки методів прогнозування. Ця бібліотека Python надає ефективні інструменти для роботи з табличними структурами даних, включаючи вибірку, фільтрацію, обчислення нових колонок, об'єднання, групування та агрегування даних. Pandas також підтримує роботу з часовими рядами, що дозволяє виконувати операції з індексами дат і часу, ресемплінг, згладжування та обчислення статистик за періоди. Використання Pandas забезпечує зручну та ефективну підготовку даних для аналізу та моделювання, підвищуючи продуктивність розробки методів прогнозування.

В моделі було виявлено надлишкові ознаки, такі як дата та ціна закриття (Adj Close), з набору даних. Після цього дані були нормалізовані за допомогою методу масштабування Min-Max. Цей процес дозволяє стандартизувати значення ознак так, щоб вони мали діапазон від 0 до 1, що полегшує навчання моделі і покращує її стійкість до великих значень.

Після обробки та нормалізації даних вони були розділені на вхідні (X_train) та вихідні (y_train) змінні для подальшого використання в моделі прогнозування (лістинг 3.1).

Лістинг 3.1 – Підготовка даних до використання у моделі

```
data_training = data_training.drop(['Date', 'Adj Close'], axis =
1)
scaler = MinMaxScaler()
data_training = scaler.fit_transform(data_training)
window_size = 60
```

```
data_training[0:5]
X_train = []
y_train = []
for i in range(window_size, data_training.shape[0]):
    X_train.append(data_training[i-window_size:i])
    y_train.append(data_training[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
```

3.4 Реалізація методів прогнозування

3.4.1 ARIMA та експоненційне згладжування

Після підготовки даних стає актуальним реалізація статистичних методів прогнозування, таких як ARIMA (авторегресійна інтегрована ковзна середня) та експоненційне згладжування. Метод ARIMA є одним з основних методів для прогнозування часових рядів, який включає аналіз трьох основних компонентів: авторегресії (AR), інтегрованої складової (I) та ковзної середньої (MA). Реалізація методу ARIMA передбачає вибір оптимальних параметрів моделі (p, d, q) шляхом аналізу автокореляційної та часткової автокореляційної функцій, а потім побудову та навчання моделі на тренувальних даних. Після навчання моделі можна використовувати її для прогнозування значень часового ряду на майбутні періоди.

Експоненційне згладжування є іншим популярним статистичним методом для прогнозування часових рядів, базується на згладжуванні змін, що стаються в часовому ряді, з метою виявлення загальних тенденцій. Реалізація експоненційного згладжування включає в себе вибір параметрів згладжування (наприклад, alpha для простого експоненційного згладжування або alpha та beta для подвійного експоненційного згладжування) та навчання моделі на тренувальних даних.

ARIMA та експоненційне згладжування вимагають ретельного аналізу та підготовки даних перед їх застосуванням, а також налаштування параметрів моделі для досягнення найкращих результатів прогнозування (лістинг 3.2).

Лістинг 3.2 – Використання методу ARIMA та експоненційного згладжування

```
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.holtwinters import ExponentialSmoothing
import matplotlib.pyplot as plt
data = pd.read_csv('time_series_data.csv', parse_dates=['Date'],
index_col='Date')
data = data.asfreq('D').fillna(method='ffill')
train_data = data[:'2022-12-31']
test_data = data['2023-01-01':]
arima_model = ARIMA(train_data, order=(p, d, q))
arima_fit = arima_model.fit()
arima_forecast = arima_fit.forecast(steps=len(test_data))
alpha, beta = 0.8, 0.2
exp_smooth_fit.forecast(steps=len(test_data))
```

3.4.2 Використання моделей машинного навчання, таких як лінійні регресії, дерева рішень та градієнтний бустінг

Для аналізу та прогнозування часових рядів можна використовувати моделі машинного навчання. Деякі з популярних моделей машинного навчання, які можна використовувати для прогнозування часових рядів, включають лінійні регресійні моделі, дерева рішень та градієнтний бустінг.

Лінійні регресійні моделі використовуються для прогнозування значень змінної на основі лінійних залежностей вхідних ознак. Вони широко застосовуються в прогнозуванні часових рядів, коли змінна, яку потрібно прогнозувати, має лінійну залежність від інших змінних (лістинг 3.3).

Лістинг 3.3 – Лінійна регресія

```
from sklearn.linear_model import LinearRegression
import numpy as np
X_train = np.array(train_data.index).reshape(-1, 1)
y_train = train_data.values
X_test = np.array(test_data.index).reshape(-1, 1)
lin_reg_model = LinearRegression()
lin_reg_model.fit(X_train, y_train)
lin_reg_forecast = lin_reg_model.predict(X_test)
```

Дерева рішень – це моделі, які представляють собою деревоподібну

структуру прийняття рішень, де кожен вузол представляє певну умову на вхідних ознаках, а кожний листок вузла представляє прогнозоване значення. Дерева рішень добре підходять для задач класифікації та регресії, включаючи прогнозування часових рядів (лістинг 3.4).

Лістинг 3.4 – Дерева рішень

```
from sklearn.tree import DecisionTreeRegressor
tree_model = DecisionTreeRegressor()
tree_model.fit(X_train, y_train)
tree_forecast = tree_model.predict(X_test)
```

Градієнтний бустінг – це ансамблевий метод машинного навчання, який поєднує кілька слабких моделей для створення сильної моделі прогнозування. Він працює шляхом послідовного навчання моделей, де кожна нова модель намагається виправити помилки попередніх моделей.

Градієнтний бустінг часто використовується для прогнозування часових рядів через його високу точність та здатність до роботи зі складними залежностями в даних (лістинг 3.5).

Лістинг 3.5 – Градієнтний бустінг

```
from sklearn.ensemble import GradientBoostingRegressor
gbr_model = GradientBoostingRegressor()
gbr_model.fit(X_train, y_train)
gbr_forecast = gbr_model.predict(X_test)
```

Використання цих моделей машинного навчання дозволяє покращити якість прогнозів часових рядів, особливо у випадку, коли дані мають складні нелинійні залежності або велику кількість ознак. Крім того, ці моделі можуть допомогти автоматизувати процес прогнозування та зробити його більш гнучким та ефективним.

3.4.3 Розробка рекурентних нейронних мереж (RNN) для прогнозування часових рядів

Рекурентні нейронні мережі (RNN) це моделі машинного навчання, які використовуються в прогнозування часових рядів. У порівнянні з традиційними методами, такими як ARIMA або експоненційне згладжування, RNN можуть автоматично виявляти складні залежності в даних та адаптуватися до різних структур часових рядів (лістинг 3.6).

Лістинг 3.6 – Модель рекурентної нейронної мережі для прогнозування часових рядів

```
model = Sequential()
model.add(LSTM(units = 60, activation = 'relu', return_sequences
= True, input_shape = (X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))
model.add(LSTM(units = 60, activation = 'relu', return_sequences
= True))
model.add(Dropout(0.2))
model.add(LSTM(units = 80, activation = 'relu', return_sequences
= True))
model.add(Dense(units = 1))
model.compile(optimizer='adam', loss = 'mean_squared_error')
model.fit(X_train, y_train, epochs=50, batch_size=32)
```

Основна перевага RNN полягає в їх здатності до роботи з послідовними даними та врахування попередніх станів. Це дозволяє їм враховувати не лише поточний момент часу, а й попередні зміни в часовому ряді. Крім того, RNN можуть працювати з даними різної довжини, що робить їх гнучкими для різноманітних завдань прогнозування.

Розробка моделі RNN для прогнозування часових рядів включає в себе кілька етапів. Спочатку необхідно визначити архітектуру мережі, включаючи кількість шарів, кількість нейронів у кожному шарі та тип шарів (наприклад, Simple RNN, LSTM або GRU). Потім необхідно підготувати дані для навчання, включаючи відбір вхідних та вихідних змінних та розбиття даних на тренувальний та тестовий набори (лістинг 3.7).

Лістинг 3.7 – Підготовка даних для навчання у моделі RNN

```
import numpy as np
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import SimpleRNN, LSTM, GRU, Dense
from keras.optimizers import Adam
import matplotlib.pyplot as plt
data = pd.read_csv('time_series_data.csv')
train, test = values[0:train_size],
values[train_size:len(values)]
```

Після цього проводиться навчання моделі прогнозування часового ряду на тренувальних даних, використовуючи відповідні алгоритми оптимізації та функції втрати (лістинг 3.8).

Лістинг 3.8 – Навчання моделі

```
def create_dataset(dataset, look_back=1):
    X, y = [], []
    for i in range(len(dataset) - look_back - 1):
        a = dataset[i:(i + look_back)]
        X.append(a)
        y.append(dataset[i + look_back])
    return np.array(X), np.array(y)
look_back = 10
X_train, y_train = create_dataset(train, look_back)
X_test, y_test = create_dataset(test, look_back)
X_train = np.reshape(X_train, (X_train.shape[0],
X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1],
1))
model = Sequential()
model.add(LSTM(50, input_shape=(look_back, 1)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
history = model.fit(X_train, y_train, epochs=100, batch_size=32,
validation_data=(X_test, y_test), verbose=2)
```

Після завершення навчання модель може бути використана для прогнозування значень часового ряду на майбутні періоди (лістинг 3.9).

Лістинг 3.9 – Прогнозування значень та відновлення даних

```
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)
train_predict = np.squeeze(train_predict)
test_predict = np.squeeze(test_predict)
```

Важливою перевагою RNN є їх здатність до автоматичного виявлення та використання складних залежностей в даних. Однак вони можуть вимагати значних обчислювальних ресурсів та часу для навчання, особливо при роботі з великими обсягами даних. Також важливо враховувати можливість перенавчання моделі на шумових або недостатньо об'ємних даних, що може призвести до погіршення її ефективності.

3.5 Попередня обробка вихідних даних

3.5.1 Нормалізація та стандартизація

Нормалізація та стандартизація даних – це важливий крок у підготовці даних для навчання моделей машинного навчання, включаючи рекурентні нейронні мережі (RNN). Цей процес допомагає забезпечити стабільність та ефективність навчання моделей, зокрема в умовах різниці масштабів між ознаками.

Нормалізація полягає у масштабуванні значень ознак у межі від 0 до 1. Це особливо важливо для моделей, які використовують функції активації з обмеженим діапазоном, такі як сигмоїдальна або гіперболічний тангенс, оскільки це допомагає уникнути проблеми «затухання градієнту» та прискорити процес навчання. Для нормалізації можна використовувати методи, такі як мінімаксне масштабування або середнє нульове масштабування (лістинг 3.10).

Лістинг 3.10 – Нормалізація та стандартизація даних

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data_training = scaler.fit_transform(data_training)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_standardized = scaler.fit_transform(data)
```

Стандартизація полягає у перетворенні значень ознак так, щоб вони

мали середнє значення 0 та стандартне відхилення 1. Це дозволяє уникнути впливу великих значень ознак на результат навчання (лістинг 3.10).

Обидва підходи до підготовки даних мають свої переваги та недоліки і можуть бути використані в залежності від конкретного завдання та властивостей даних. Важливо провести експерименти з обома методами та визначити оптимальний підхід для конкретного набору даних та моделі.

3.5.2 Обробка пропущених значень та видалення аномалій

Обробка пропущених значень та видалення аномалій є важливими кроками в підготовці даних для прогнозування часових рядів за допомогою рекурентних нейронних мереж (RNN) та інших моделей машинного навчання.

Пропущені значення можуть виникнути з різних причин, таких як помилки вимірювань, втрати даних або технічні проблеми. Вони можуть спричинити недостовірні результати прогнозування, тому їх потрібно врахувати в аналізі даних. Одним зі способів обробки пропущених значень є їх заповнення. Це може бути здійснено шляхом заповнення пропущених значень за допомогою статистичних міри центральної тенденції (наприклад, середнє або медіанне значення) або методами інтерполяції (data_training = data_training.interpolate(method='linear', limit_direction='forward', axis=0)). Іншим підходом є видалення рядків або стовпців з пропущеними значеннями, але це може призвести до втрати важливих даних.

Видалення аномалій також важливо для забезпечення точності прогнозів. Аномалії – це значення, які суттєво відрізняються від інших значень у часовому ряді і можуть бути результатом помилок або виняткових подій. Для виявлення аномалій можна використовувати різні методи, такі як виявлення за допомогою статистичних мір, виявлення за допомогою зовнішніх джерел даних або використання алгоритмів машинного навчання для виявлення відхилень від звичайного патерну (лістинг 3.11).

Лістинг 3.11 – Видалення аномалій

```
lower_bound = data['column_name'].mean() - 2 *
data['column_name'].std()
upper_bound = data['column_name'].mean() + 2 *
data['column_name'].std()
data = data[(data['column_name'] > lower_bound) &
(data['column_name'] < upper_bound)]
```

Після обробки пропущених значень та видалення аномалій дані придатні для подальшого аналізу. Це дозволяє покращити якість прогнозів та забезпечити стабільність роботи моделей машинного навчання, включаючи рекурентні нейронні мережі.

3.5.3 Розбиття даних на вхідні та вихідні змінні для побудови моделей

Розбиття даних на вхідні та вихідні змінні є головним етапом підготовки даних для побудови моделей прогнозування за допомогою RNN та інших підходів машинного навчання. Вхідні змінні представляють собою характеристики або ознаки, які будуть використані для прогнозування, тоді як вихідні змінні є значеннями, які має на меті модель прогнозувати.

При побудові моделей прогнозування часових рядів вхідні змінні можуть включати різні аспекти даних, такі як попередні значення часового ряду, вхідні змінні, що впливають на зміну, або додаткові фактори, які враховуються у моделі. Наприклад, у випадку прогнозування цін акцій попередні значення ціни можуть використовуватися як вхідні змінні для прогнозування майбутніх значень. Вихідні змінні визначаються завданням прогнозування та можуть бути одним або кількома значеннями, які модель намагається прогнозувати. Наприклад, у випадку прогнозування цін акцій вихідною змінною може бути наступне значення ціни на певний момент часу у майбутньому.

Після розбиття даних на вхідні та вихідні змінні проводиться подальший аналіз даних, включаючи вибір моделі, навчання моделі на

тренувальних даних та оцінку її ефективності на тестових даних. Цей процес допомагає створити модель прогнозування, яка може бути використана для прогнозування майбутніх значень часового ряду з високою точністю.

3.6 Виконання прогнозування за допомогою навченої моделі

Після підготовки навчальних даних та навчання моделі нейронної мережі, виконується процес прогнозування майбутніх значень на основі навченої моделі. Спочатку дані останніх декількох днів з навчального набору даних використовуються для створення початкового набору даних для прогнозування. Далі ці дані об'єднуються з тестовим набором даних, що містить майбутні спостереження. З цього початкового набору даних видаляються непотрібні стовпці, такі як дата та цінова статистика, щоб залишити лише вхідні змінні, які будуть використовуватися для прогнозування (лістинг 3.12).

Лістинг 3.12 – Підготовка до прогнозування

```
data_training = data[data['Date']<'2023-01-01'].copy()
past_days = data_training.tail(window_size)
df = pd.concat([past_days, data_test], ignore_index=True)
df = df.drop(['Date', 'Adj Close'], axis = 1)
```

Процес нормалізації вхідних даних – дані масштабуються за допомогою раніше навченої масштабуючої функції. Після цього визначається максимальний горизонт прогнозування, тобто максимальна кількість майбутніх часових кроків, які будуть прогнозуватися. Для кожного часового кроку в тестовому наборі даних формується вхідний вектор, що складається з попередніх значень за останні window_size днів. Ці вхідні вектори подаються на вхід моделі для отримання прогнозованих значень. Отримані прогнози після цього масштабуються назад до їхнього вихідного масштабу за допомогою раніше визначеного масштабу (лістинг 3.13).

Лістинг 3.13 – Прогнозування тестових даних

```
X_test = []
y_test = []
for i in range(window_size, min(inputs.shape[0], window_size +
max_forecast_horizon)):
    X_test.append(inputs[i-window_size:i])
    y_test.append(inputs[i, 0])
X_test, y_test = np.array(X_test), np.array(y_test)
y_pred = model.predict(X_test)
y_pred = y_pred*scale
y_test = y_test*scale
```

Цей процес виконується для кожного часового кроку у максимальному горизонті прогнозування, щоб отримати прогнозовані значення для кожного майбутнього часового кроку.

3.7 Відображення результатів прогнозування

Після виконання прогнозування за допомогою навченої моделі та отримання прогнозованих значень для майбутніх часових кроків, результати прогнозування відображаються та аналізуються для оцінки точності моделі. Спочатку формується DataFrame, що містить дати спостережень, фактичні та прогнозовані значення (лістинг 3.14). Цей DataFrame об'єднується з тестовим набором даних за допомогою об'єднання по даті за допомогою внутрішнього злиття.

Лістинг 3.14 – Об'єднання результатів прогнозування з тестовим набором даних

```
results_df = pd.DataFrame({'Date': data_test['Date'].values[-
y_test.shape[0]:],
                           'Actual': y_test,
                           'Predicted': y_pred.flatten()})
merged_df = pd.merge(data_test, results_df, on='Date',
how='inner')
```

Після об'єднання розраховуються середні абсолютні похибки (MAE) та середні абсолютні відсоткові похибки (MAPE) для оцінки точності прогнозування. Отримані значення MAE та MAPE додаються до першого рядка DataFrame для подальшого аналізу (лістинг 3.15).

Лістинг 3.15 – Виведення даних з результатами прогнозування та обчисленими метриками точності та збереження у файл CSV для подальшого аналізу

```
results_df = pd.DataFrame({'Date': data_test['Date'].values[-
y_test.shape[0]:],
                           'Actual': y_test,
                           'Predicted': y_pred.flatten()})
merged_df = pd.merge(data_test, results_df, on='Date',
how='inner')
mae = np.mean(np.abs(merged_df['Actual'] -
merged_df['Predicted']))
mape = np.mean(np.abs((merged_df['Actual'] -
merged_df['Predicted']) / merged_df['Actual'])) * 100
summary_df = summary_df.dropna(axis=1, how='all')
final_df = pd.concat([summary_df, merged_df], ignore_index=True)
final_df.to_csv('таблица.csv', index=False, columns=['Date',
'Actual', 'Predicted', 'MAE', 'MAPE'])
plt.figure(figsize=(14,5))
plt.plot(merged_df['Date'], merged_df['Actual'], color='red',
label='Реальна ціна акцій Nasdaq')
plt.plot(merged_df['Date'], merged_df['Predicted'],
color='blue', label='Прогнозована ціна акцій Nasdaq')
plt.title('Прогнозування ціни акцій Nasdaq')
plt.xlabel('Час')
plt.ylabel('Ціна акцій Nasdaq')
plt.legend()
plt.xticks(plt.xticks()[0][::5], rotation=25)
plt.savefig('графік.png', bbox_inches='tight')
```

Після цього всі результати записуються в CSV-файл для зручності подальшого використання. Крім того, результати візуалізуються на графіку, де червоним кольором позначені фактичні значення, а синім - прогнозовані. Це дозволяє візуально порівняти прогнозовані та реальні значення та зрозуміти, наскільки добре модель відтворює майбутні тренди на ринку.

4 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

4.1 Досліджувані параметри моделі

З метою проведення досліджень була створена модель рекурентної штучної нейронної мережі, з використанням мови Python, середовища Jupyter Notebook та засобів бібліотек Pandas, NumPy, Matplotlib, Seaborn, TensorFlow та scikit-learn. Експериментальні дослідження проводилися для виявлення залежностей між точністю прогнозування та такими параметрами, як horizont, epoch, units, window size, batch size.

Параметр horizont визначає розмір горизонту прогнозування, що відповідає кількості майбутніх значень, щодо яких виконується прогнозування.

Параметр epoch представляє собою кількість ітерацій, проведених під час процесу навчання моделі. Більше значення цього параметра зазвичай сприяє підвищенню точності моделі, проте велика кількість епох може призвести до перенавчання, коли модель втрачає здібність до узагальнення на нових даних.

Параметр units – кількість нейронів у шарі нейронної мережі. Чим більше значення цього параметру, тим більше пам'яті та обчислювальних ресурсів вимагається для моделі, що може призвести до підвищення її потужності, але також може зробити її більш витратною у відношенні до обчислювальних ресурсів.

Параметр window_size визначає кількість попередніх часових кроків, значення яких використовуються для прогнозування наступного значення. Вибір оптимального розміру вікна є важливим, оскільки занадто малий розмір може не враховувати достатньо інформації з минулого для точного прогнозування, тоді як занадто великий розмір може ускладнити модель і призвести до надмірної обчислювальної складності.

Параметр batch_size визначає кількість зразків, які обробляються моделлю за одну ітерацію перед оновленням ваг. Менший розмір пакета може призвести до більш частого оновлення ваг і швидшої конвергенції, але з більшою варіативністю градієнтів. Навпаки, більший розмір пакета забезпечує стабільніше оновлення ваг, але потребує більше пам'яті і може призвести до повільнішого навчання.

Для оцінки ефективності моделей прогнозування використовувалися такі метрики, як середня абсолютна помилка (MAE), середня абсолютна відсоткова помилка (MAPE) та середньоквадратична помилка (MSE). Вони дозволяють кількісно оцінити помилки прогнозування та порівняти продуктивність моделей.

4.2 Базові конфігурації моделі

Для моделі рекурентної нейронної мережі в якості базових конфігурацій визначені наступні значення параметрів: epochs – 50, batch_size – 32, функція активації – relu, optimizer – adam, loss – mse, horizont – 60, window_size – 60.

Функція активації ReLU (Rectified Linear Unit) характеризується простотою обчислень та має декілька важливих переваг у контексті навчання штучних нейронних мереж. Спрощеність обчислень забезпечує ефективність в роботі з великими обсягами даних, що є доцільним для застосування у складних моделях, таких як глибокі нейронні мережі. Крім того, функція ReLU допомагає уникнути проблеми втрати градієнту, яка може виникнути при навчанні моделей з багатьма шарами. Необхідно моделювати нелінійні залежності у вхідних даних для ефективного розв'язання багатьох завдань машинного навчання. Функція ReLU відповідає цим потребам, завдяки чому модель може розрізняти дані та виявляти складні закономірності. Також використання ReLU сприяє створенню розріджених активацій, що дозволяє знизити кількість параметрів моделі та запобігти її перенавчанню. Одним із

важливих аспектів функції ReLU є те, що вона залишає всі позитивні значення без змін, тоді як негативним значенням присвоює нуль. Це призводить до ефективного виділення сигналу від шуму та сприяє більш стабільному навчанню нейронних мереж.

Метод оптимізації Adam представляє собою ефективний алгоритм, який широко використовується для навчання нейронних мереж. Він поєднує в собі переваги інших методів оптимізації, таких як AdaGrad та RMSProp, і додає до них адаптивність.

При оцінюванні якості моделі одним з основних критеріїв є точність її прогнозів. Для оцінки використовуються різні критерії, такі як середньоквадратична помилка (MSE), середня абсолютна похибка (MAE) та середня абсолютна відносна похибка (MAPE). MSE обчислюється шляхом усереднення квадратів різниць між фактичними значеннями цільової змінної і прогнозованими значеннями моделі. MAE та MAPE надають додаткову інформацію про точність моделі, оцінюючи середню абсолютну похибку в одиницях вихідних даних та у відсотках відповідно.

Середньоквадратична помилка (MSE) – це метрика точності, що широко використовується в області машинного навчання для оцінки точності моделей прогнозування. Вона обчислюється шляхом усереднення квадратів різниць між фактичними значеннями цільової змінної і прогнозованими значеннями моделі. Підвищення значення MSE вказує на те, що прогнози моделі відхиляються від реальних значень у великій мірі.

Середня абсолютна похибка (MAE) також є метрикою оцінки точності і обчислюється шляхом усереднення абсолютних значень різниць між фактичними і прогнозованими значеннями. MAE використовують для інтерпретації, оскільки відображає середню похибку прогнозу в тих же одиницях, що і вихідні дані. Це дозволяє зрозуміти, наскільки в середньому модель відхиляється від реальних значень.

Середня абсолютна відносна похибка (MAPE) обчислюється як середнє значення відносних похибок між фактичними і прогнозованими значеннями,

виражених у відсотках. Цей показник дозволяє зрозуміти похибку прогнозу в відносному вираженні, що робить його корисним для порівняння точності моделей на різних наборах даних. MAPE надає додаткову інформацію про ефективність моделі, показуючи, наскільки відсотково відхиляються прогнози від реальних значень.

4.3 Вплив параметрів моделі на результати прогнозування

Результати прогнозування часових рядів залежать від різних параметрів моделі, таких як розмір вікна, кількість епох навчання, розмір пакетів, горизонт прогнозування, а також кількість шарів і нейронів. Розмір вікна визначає кількість попередніх спостережень для прогнозування майбутніх значень. Кількість епох навчання впливає на здатність моделі до узагальнення, а розмір пакетів визначає кількість зразків, що обробляються за одну ітерацію. Горизонт прогнозування визначається кількістю майбутніх часових кроків, які модель намагається передбачити.

Налаштування вказаних параметрів визначає здатність моделі адекватно моделювати часові ряди. Також важливою є обрана модель прогнозування та методи обробки даних, які також впливають на результати прогнозування.

Для дослідження впливу параметрів на результати прогнозування було проведено серію експериментів.

При налаштуванні моделей для проведення експериментів використовуються параметри моделі за замовчуванням: розмір batch складає 32, кількість епох навчання – 50, горизонт прогнозування – 60, а розмір вікна вхідних даних – 60. Архітектура моделі включає чотири шари LSTM: перший шар має 60 нейронів, другий – 60 нейронів, третій – 80 нейронів, а четвертий – 120 нейронів. Ці параметри залишаються незмінними, якщо не вказано інше в конкретному експерименті.

У першому експерименті проводилося дослідження впливу кількості

епох на результати прогнозування, було проведено серію обчислень, в яких варіювалась кількість епох навчання моделі нейронної мережі. У результаті експерименту проведено аналіз впливу кількості епох навчання на точність прогнозування (таблиця 4.1), зазначені значення кількості епох та відповідна помилка прогнозування. Потім на основі цих даних був побудований графік (рисунок 4.1), на якому по осі абсцис відображалася кількість епох, а по осі ординат – значення помилки.

Таблиця 4.1 – Залежність результатів прогнозування від кількості епох навчання

№	epochs	MAE
1	10	432,538105
2	20	538,8836417
3	30	720,51431
4	40	504,2608033
5	50	414,938895
6	100	798,7181417
7	200	387,615905
8	400	445,1998941

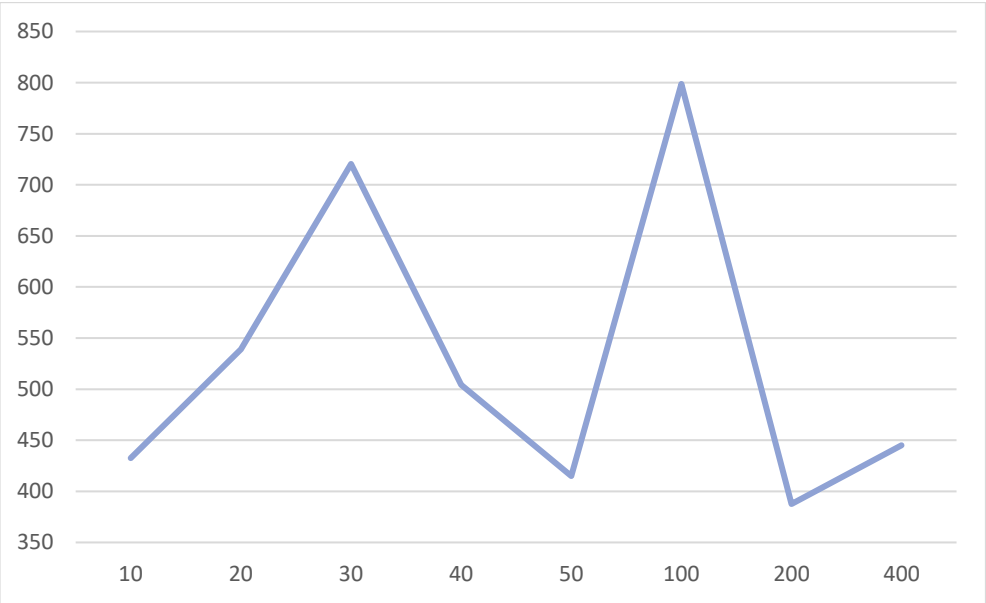

Рисунок 4.1 – Вплив параметру epochs на значення MAE

Аналізуючи залежність результатів прогнозування від кількості епох навчання, можна помітити, що MAE (середня абсолютна похибка) коливається при різної кількості епох. При малих значеннях (10 та 20) спостерігається відносно низький MAE (432,54 та 538,88 відповідно), однак при збільшенні кількості епох до 30 MAE зростає до 720,51. Збільшення кількості епох до 50 призводить до покращення результатів (414,94), що свідчить про недостатню кількість епох для повного навчання моделі. Найнижче значення MAE досягається при 200 епохах (387,62), після чого подальше збільшення кількості епох до 400 знову призводить до зростання MAE (445,20). Це може вказувати на перенавчання моделі при занадто великій кількості епох.

У відповідному експерименті досліджувався вплив параметра batch_size на результати прогнозування в штучній нейронній мережі. Різні значення цього параметра випробовувалися в серії ітерацій, залишаючи інші параметри незмінними, що дозволило окремо визначити його вплив на результати. У результаті експерименту були отримані дані про значення помилок прогнозування для кожного значення batch_size. Ці дані були відображені у таблиці 4.2, де кожному значенню batch_size відповідає певне значення помилки.

Таблиця 4.2 – Залежність результатів від batch_size

№	batch_size	MAE
1	4	376,8841933
2	8	226,0692767
3	16	298,6666667
4	32	538,8836417
5	64	838,9370217
6	128	735,8446367
7	256	692,5444767
8	512	464,2444767

На основі результатів був побудований графік (рисунок 4.2), де по осі абсцис відображалися значення batch_size, а по осі ординат – відповідні значення помилок.

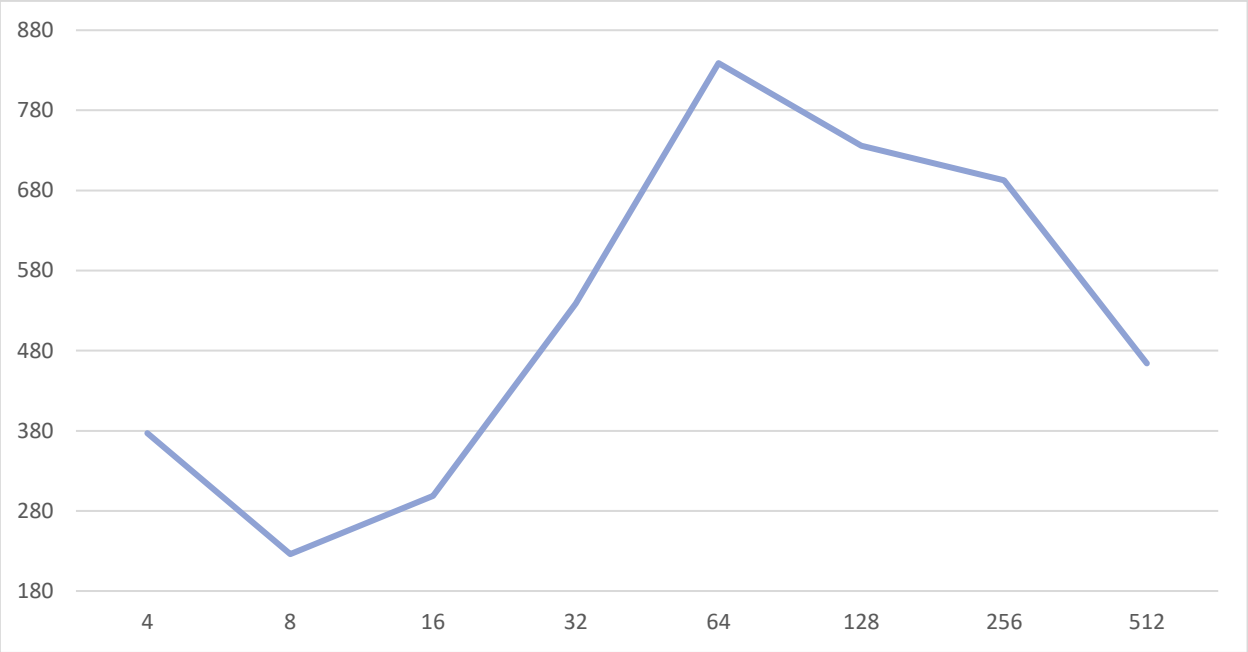

Рисунок 4.2 – Вплив batch_size на MAE

Залежність результатів від batch_size показує, що при малих batch_size (4 та 8) результати є найкращими, з найнижчими значеннями MAE (376,88 та 226,07 відповідно). При збільшенні batch_size до 16 і 32 MAE збільшується (298,67 та 538,88 відповідно), що може вказувати на погіршення навчання моделі через недостатнє оновлення ваг. Подальше збільшення batch_size до 64 і вище (128, 256, 512) призводить до значного зростання MAE, досягаючи максимального значення при batch_size 64 (838,94). Таким чином, можна зробити висновок, що для даного експерименту оптимальним є використання малого batch_size.

У наступному експерименті метою було дослідити вплив величини горизонту прогнозування на результати в моделі рекурентної штучної нейронної мережі. Для цього змінювалися значення параметра горизонту прогнозування, тобто кількість значень, які модель намагалася передбачити в

майбутньому. В інших аспектах експериментальні умови залишались сталі, щоб уникнути змін, які могли б вплинути на результати.

У результаті експерименту отримані дані про значення помилок прогнозування для кожного значення горизонту прогнозування. Ці дані були представлені у таблиці 4.3, де кожному варіанту значення горизонту прогнозування відповідає певне значення помилки.

Таблиця 4.3 – Залежність результатів прогнозування від горизонту прогнозування

№	horizont	MAE
1	10	380,64595
2	20	561,054165
3	30	591,5938533
4	40	567,704755
5	50	582,081448
6	60	538,8836417
7	70	735,6843657
8	80	578,2885225

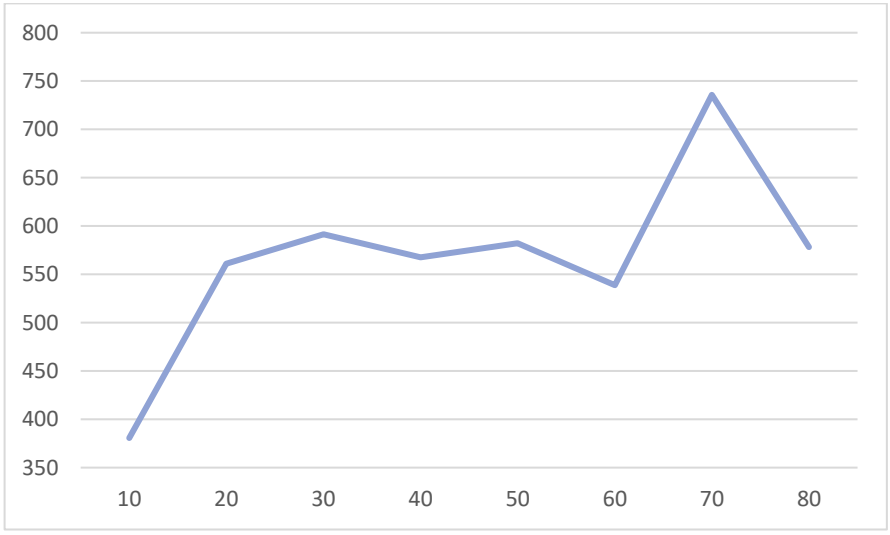

Рисунок 4.3 – Вплив параметру horizont на значення MAE

Результати використані для побудови графіку (рисунок 4.3), де на осі абсцис відображалися значення горизонту прогнозування, а на осі ординат відповідні значення помилок.

Під час аналізу результатів прогнозування виявлено закономірність: зі збільшенням горизонту прогнозування відбувається поступове збільшення середньої абсолютної похибки (MAE). Наприклад, при горизонті в 10 днів MAE становить 380,65, що позначається як оптимальний результат у порівнянні з іншими перевіреними значеннями горизонту. Однак зі збільшенням горизонту до 20, 30 та 40 днів спостерігається поступове зростання MAE (відповідно 561,05, 591,59 та 567,70). Найбільші значення MAE відзначаються при горизонтах 70 та 80 днів (відповідно 735,68 та 578,29). Таким чином, можна зробити висновок, що для досягнення кращих результатів з меншою похибкою рекомендується використовувати більш короткі горизонти прогнозування.

У наступному експерименті детально досліджується вплив параметрів моделі на результати прогнозування часових рядів. Основна увага приділяється аналізу впливу зміни розміру вікна на точність прогнозів. Важливо зазначити, що необхідно правильно обрати цей параметр для забезпечення здатності моделі ефективно моделювати часові ряди та здійснювати точні прогнози. Планується проведення аналізу результатів та формулювання висновків щодо оптимальних значень розміру вікна, необхідних для досягнення найкращих результатів прогнозування.

У результаті експерименту отримані дані про значення помилок прогнозування для кожного значення кількості попередніх часових кроків (window size). Ці дані були представлені у таблиці 4.4, де кожному варіанту значення window size відповідає певне значення помилки. Таблиця дозволяє детально побачити, як зміна розміру вікна впливає на точність прогнозування. Для більшої наочності результати також були використані для побудови графіку (рисунок 4.4). На графіку на осі абсцис відображені значення розміру часового вікна, а на осі ординат – відповідні значення

помилок (MAE). Така візуалізація дозволяє порівняти вплив різних розмірів вікна на точність моделі, наочно показуючи тенденцію до зміни помилки зі збільшенням або зменшенням window_size. Це допоможе визначити оптимальні значення розміру вікна, які забезпечують мінімальні похибки прогнозування, та зробити відповідні висновки щодо налаштувань моделі.

Таблиця 4.4 – Залежність результатів прогнозування від кількості попередніх часових кроків, використаних для прогнозування

№	Window_size	MAE
1	20	501,6021617
2	40	718,8064467
3	60	795,3677633
4	80	856,4461783
5	100	651,6773917
6	120	760,8953783
7	140	766,7327533
8	160	623,05766

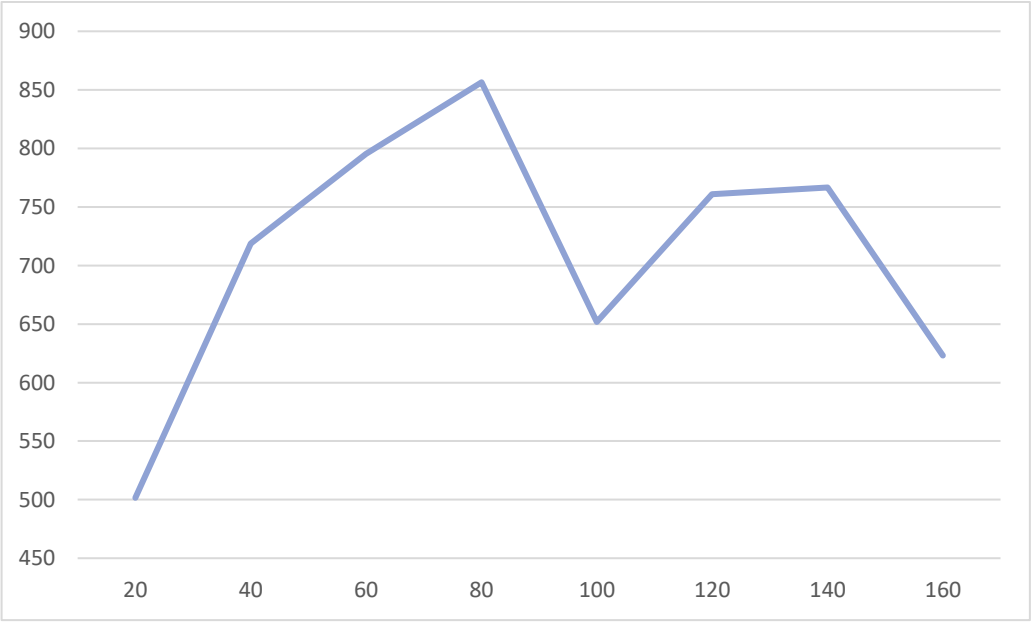

Рисунок 4.4 – Вплив розміру часового вікна на MAE

Згідно з отриманими значеннями залежності результатів прогнозування від кількості попередніх врахованих часових кроків (window_size), найменше значення MAE (501,60) досягається при найменшому window_size – 20. Зі збільшенням window_size до 40, 60 та 80 MAE зростає (718,81, 795,37 та 856,45 відповідно), що свідчить про негативний вплив занадто великої кількості часових кроків на точність прогнозування. Подальше збільшення window_size до 100 і більше (120, 140, 160) також не призводить до суттєвого покращення результатів MAE на рівні близько 623,06 – 766,73. Це вказує на те, що оптимальним для даного експерименту є використання меншого window size, оскільки велика кількість часових кроків може призвести до перенавчання моделі.

Для аналізу впливу кількості нейронів у шарах на результати прогнозування було проведено комплекс експериментів, в рамках яких змінювалась кількість нейронів у кожному шарі моделі нейронної мережі. Кожен з експериментів включав в себе навчання моделі з фіксованими іншими параметрами та детальне вимірювання різних показників якості прогнозування. Кількість шарів для кожного вимірювання дорівнює трьом. Такий підхід дозволив систематично вивчати вплив зміни кількості нейронів на результати моделі та зробити обґрунтовані висновки щодо оптимальних значень цього параметра. Наприклад, у лістингу 4.1 наведено приклад моделі, в якій для кожного експерименту змінювалась кількість нейронів.

Лістинг 4.1 – Модель для експерименту

```
model = Sequential()
model.add(LSTM(units=40, activation='relu',
return_sequences=True, input_shape=(X_train.shape[1],
X_train.shape[2])))
model.add(Dropout(0.2))
model.add(LSTM(units=40, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, y_train, epochs=50, batch_size=32)
```

В процесі експерименту отримані дані про значення помилок прогнозування для кожного значення кількості нейронів у шарі. Ці дані представлені у таблиці 4.5, де кожному варіанту кількості нейронів відповідає певне значення помилки. Таблиця дозволяє детально побачити, як зміна кількості нейронів впливає на точність прогнозування. Для більшої наочності результати також були використані для побудови графіку (рисунок 4.5). На графіку на осі абсцис відображені значення кількості нейронів, а на осі ординат – відповідні значення помилок (MAE). Така візуалізація дозволяє порівняти вплив різної кількості нейронів на точність моделі, наочно показуючи тенденцію до зміни помилки зі збільшенням або зменшенням кількості нейронів. Це допоможе визначити оптимальні значення кількості нейронів, які забезпечують мінімальні похибки прогнозування, та зробити відповідні висновки щодо налаштувань моделі. У подальшому буде проведено детальний аналіз отриманих даних та розроблено рекомендації щодо вибору оптимальних параметрів для прогнозування.

Таблиця 4.5 – Залежність результатів прогнозування від кількості нейронів у шарі, використаних для прогнозування

№	Нейронів у шарі (3 шари)	MAE	MAPE
1	40	524,9315837	8,272021379
2	60	628,4324137	9,873688085
3	80	379,9315999	5,959300303
4	100	201,9811442	3,232335981
5	120	219,8934163	3,534161258
6	140	518,7171387	8,154393758
7	160	268,1054362	4,239372226
8	180	269,90271	4,246480152

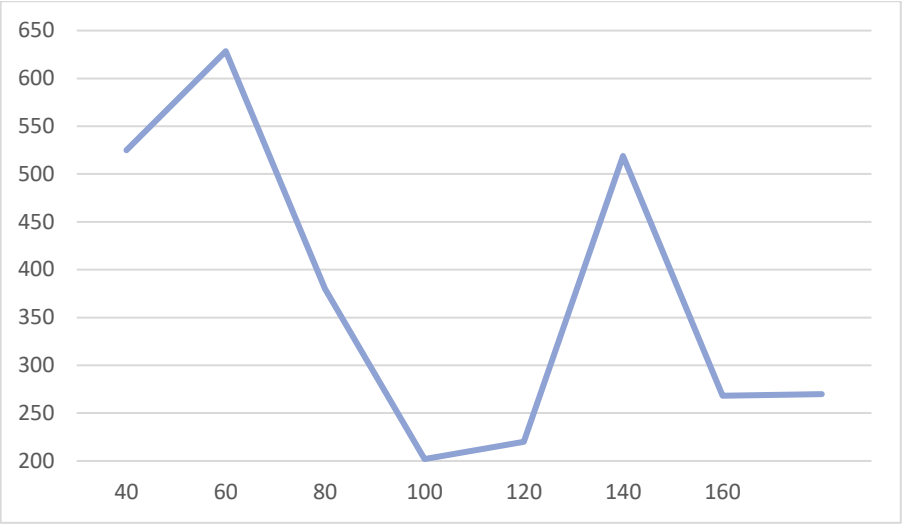

Рисунок 4.5 – Вплив кількості нейронів на MAE

Згідно з таблицею 4.5, значення MAE та MAPE демонструють різну поведінку залежно від розміру вікна.

Найменші значення MAE (201,98) та MAPE (3,23) спостерігаються при розмірі вікна 100, що вказує на оптимальний розмір вікна для точного прогнозування. Відхилення від цього оптимального значення, як у бік збільшення, так і зменшення розміру вікна, призводить до погіршення якості прогнозування, що виражається у вищих значеннях MAE та MAPE. Наприклад, збільшення розміру вікна до 140 і більше супроводжується значним зростанням MAE та MAPE, що свідчить про зниження ефективності моделі.

ВИСНОВКИ

Використання рекурентних нейронних мереж (RNN) виявляється ефективним та точним методом для прогнозування нестаціонарних часових рядів. Порівняно зі статистичними методами, такими як ARIMA та експоненційне згладжування, та методами машинного навчання, такими як лінійні регресії та дерева рішень, RNN показали кращі результати, зокрема щодо точності та робастності прогнозів.

Виявлені обмеження в застосуванні RNN, зокрема щодо складності навчання та потреби щодо обчислювальних ресурсів [45]. Однак з розвитком технологій та покращенням методів оптимізації можна очікувати подальшого зростання використання RNN в прогнозуванні нестаціонарних часових рядів.

На основі результатів проведених експериментів, де вивчався вплив різних гіперпараметрів на точність прогнозування, виявлено, що найменші значення помилки (MAE 201,98 та MAPE 3,23) спостерігаються при розмірі вікна 100, що вказує на оптимальний розмір вікна для точного прогнозування. Для досягнення кращих результатів з меншою похибкою рекомендується використовувати менші значення горизонту прогнозування.

Доцільним є подальше дослідження в галузі оптимізації та підвищення ефективності рекурентних нейронних мереж для прогнозування нестаціонарних часових рядів. Також важливо розглянути можливості використання комбінованих підходів, які об'єднують переваги різних методів для досягнення ще кращих результатів прогнозування. Це дозволить покращити точність прогнозів часових рядів, що має велике значення для прийняття ефективних управлінських рішень та прогнозування майбутніх подій у різних сферах людської діяльності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Brillinger, D. R. Time series. Data processing and theory [Текст] / D. R. Brillinger. – SIAM: Society for Industrial and Applied Mathematics, 1983. – 536 p.

2. Ebrahimpour, R. Mixture of MLP-experts for trend forecasting of time series: A case study of the Tehran stock exchange [Текст] / R. Ebrahimpour, H. Nikoo, S. Masoudnia // International Journal of Forecasting. – 2011. – Vol. 27. – P. 804–816.

3. Understanding outliers in time series analysis [Електронний ресурс] – Режим доступу : www/ URL: https://pro.arcgis.com/ru/pro-app/latest/toolreference /space-time-pattern-mining/understanding-outliers-in-time-seriesanalysis.htm. – 15.05.2024 р. – Загол. з екрану.

4. Mehtab, S. Analysis and Forecasting of Financial Time Series Using CNN and LSTM-Based Deep Learning Models [Текст] / S. Mehtab, J. Sen // Advances in Distributed Computing. – 2022. – Vol. 302. – P. 405–423.

5. Awad, M. Efficient learning machines: theories, concepts, and applications for engineers and system designers [Текст] / M. Awad, R. Khanna. – Apress Open, 2015. – 263 с.

6. Palit, A. Computational Intelligence in Time Series Forecasting [Текст] / A. Palit, D. Popovic. – Springer, Part of: Advances in Industrial Control, 2006. – 372 p. – ISBN: 978-1852339487.

7. Mostafa, M. A neuro-computational intelligence analysis of the ecological footprint of nations [Текст] / M. Mostafa, R. Nataraajan // Computational Statistics & Data Analysis. – 2009. – Vol. 53. – P. 3516–3531.

8. Chatfield, C. Time-Series Forecasting [Текст] / C. Chatfield. – Chapman and Hall/CRC, 2000. – 280 p.

9. Li, S. Deterministic fuzzy time series model for forecasting enrollments [Текст] / S. Li, Y. Cheng // Computers & Mathematics with Applications. – 2007.

– Vol. 53. – P. 1904–1920.

10. Cheng, C. Time series forecasting for nonlinear and non-stationary processes: a review and comparative study [Текст] / C. Cheng, A. Ngasoongsong, O. Beyca // IIE Transactions. – 2015. – Vol. 47. – P. 1053–1071.

11. Derbentsev, V. Machine learning approaches for financial time series forecasting [Текст] / V. Derbentsev, A. Matviychuk, N. Datsenko // Proceedings of the Selected Papers of the Special Edition of International Conference on Monitoring, Modeling & Management of Emergent Economy. – 2020. – № 2713. – P. 434–450.

12. Crash Course in Recurrent Neural Networks for Deep Learning [Електронний ресурс] – Режим доступу : www/ URL: https://machinelearningmastery.com/crash-course-recurrent-neural-networks-deeplearning/ – 15.05.2024 р. – Загол. з екрану.

13. Berastegi, G. From diagnosis to prognosis for forecasting air pollution using neural networks: Air pollution monitoring in Bilbao [Текст] / G. Berastegi, A. Elias, A. Barona // Environmental Modelling. – 2008. – Vol. 23. – P. 622–637.

14. Time-Series Forecasting: Predicting Stock Prices Using An LSTM Model [Електронний ресурс] – Режим доступу : www/ URL: https://towardsdatascience.com/lstm-time-series-forecasting-predicting-stockprices-using-an-lstm-model-6223e9644a2f/ – 15.06.2024 р. – Загол. з екрану.

15. Piccialli, F. Artificial intelligence and healthcare: Forecasting of medical bookings through multi-source time-series fusion [Текст] / F. Piccialli, F. Giampaolo, P. Edoardo // Information. – 2021. – Vol. 74. – P. 1–16.

16. Korablev, N. M. Parallel immune algorithm of short-term forecasting based on model of clonal selection [Текст] / N. M. Korablev, G. S. Ivaschenko // Radio Electronics, Computer Science, Control. – 2014. – № 2. – P. 73–78.

17. Lachtermacher, G. Back propagation in time-series forecasting [Текст] / G. Lachtermacher, D. Fuller // Journal of forecasting. – 1995. – P. 381–393.

18. Abbasimehr, H. Improving time series forecasting using LSTM and

attention models [Текст] / H. Abbasimehr, R. Paki // Journal of Ambient Intelligence and Humanized Computing. – 2021. – Vol. 13. – P. 673–691.

19. Rick, R. Energy forecasting model based on CNN-LSTM-AE for many time series with unequal lengths [Текст] / R. Rick, L. Berton // Engineering Applications of Artificial Intelligence. – 2022. – Vol. 113.

20. Livieris, I. A CNN–LSTM model for gold price time-series forecasting [Текст] / I. Livieris, E. Pintelas, P. Pintelas // Neural Computing and Applications. – 2020. – Vol. 32. – P. 17351–17360.

21. Cai, Q. A Novel Stock Forecasting Model based on Fuzzy Time Series and Genetic Algorithm [Текст] / Q. Cai, D. Zhang, B. Wu // Procedia Computer Science. – 2013. – Vol. 18. – P. 1155–1162.

22. Wong, F. Time series forecasting using backpropagation neural networks [Текст] // Neurocomputing. – 1991. – Vol. 2. – P. 147–159.

23. Ahmed, N. An Empirical Comparison of Machine Learning Models for Time Series Forecasting [Текст] / N. Ahmed, A. Atiya, N. Gayar // Econometric Reviews. – 2010. – Vol. 29. – P. 594–621.

24. Kirisci, M. A New CNN-Based Model for Financial Time Series: TAIEX and FTSE Stocks Forecasting [Текст] / M. Kirisci, O. Yolcu // Neural Processing Letters. – 2022. – Vol. 54. – P. 3357–3374.

25. Mohammed, A. S. Green Hybrid Models Based on Clonal Selection and Case-based Reasoning for Short-term Time Series Forecasting [Текст] / A. S. Mohammed, H. Ivashchenko, T. Filimonchuk // Journal of Green Engineering. – 2020. – Vol. 10. – P. 2139–2154.

26. Armstrong, J. S. Forecasting for Marketing [Текст] // Quantitative Methods in Marketing. – 1999. – P. 92–119.

27. Yang, J. Power System Short-term Load Forecasting: Thesis for Ph.d degree [Текст] // Elektrotechnik und Informationstechnik der Technischen Universität Darmstadt zur Erlangung vorgelegte Dissertation. – P. 139.

28. Durao, R. Forecasting O3 levels in industrial area surroundings up to 24 h in advance, combining classification trees and MLP models [Текст] / R. Durao,

M. Mendes, J. Pereira // Atmospheric Pollution Research. – 2016. – Vol. 7. – P. 961–970.

29. Voyant, C. Uncertainties in global radiation time series forecasting using machine learning: The multilayer perceptron case [Текст] / C. Voyant, G. Notton, C. Darras // Energy. – 2017. – Vol. 125. – P. 248–257.

30. How to Develop LSTM Models for Time Series Forecasting [Електронний ресурс] – Режим доступу : www/ URL: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-seriesforecasting/ – 15.05.2024 р. – Загол. з екрану.

31. How to Develop Convolutional Neural Network Models for Time Series Forecasting [Електронний ресурс] – Режим доступу : www/ URL: https://machinelearningmastery.com/how-to-develop-convolutional-neuralnetwork models-for-time-series-forecasting/ – 15.05.2024 р. – Загол. з екрану.

32. Gurney, K. An Introduction to Neural Networks // K. Gurney. – 1997. – 234 p.

33. Cao, J. Financial time series forecasting model based on CEEMDAN and LSTM [Текст] / J. Cao, Z. Li, J. Li // Physica A: Statistical Mechanics and its Applications. – 2019. – Vol. 519. – P. 127-139.

34. Recurrent Neural Network [Електронний ресурс] – Режим доступу : www/ URL: https://itwiki.dev/data-science/ml-reference/ml-glossary/recurrent-neural-network/ – 31.05.2024 р. – Загол. з екрану.

35. Obrubov, M. Using LSTM network for solving the multidimentional time series forecasting problem [Текст] / M. Obrubov, S. Kirillova // National Association of Scientists. – 2021. – Vol. 2, №68. – С. 43–48.

36. Zhao, R. Improving Neural Network Quantization without Retraining using Outlier Channel Splitting [Текст] / R. Zhao, Y. Hu, J. Dotzel // International Conference on Machine Learning. – 2019. – Vol. 97. – P. 7543–7552.

37. Wang, Z. RFPruning: A retraining-free pruning method for accelerating convolutional neural networks [Текст] / Z. Wang, X, Xie, G. Shi // Applied Soft Computing. – 2021. – Vol. 113.

38. Shi, X. Convolutional LSTM Network: a machine learning approach for precipitation nowcasting [Текст] / X. Shi, Z. Chen, H. Wang, D. Yeung // Proceedings of the 28th International Conference on Neural Information Processing Systems. – 2015. – Vol. 1. – P. 802–810.

39. Srivastava, N. Dropout: A Simple Way to Prevent Neural Networks from Overfitting [Текст] / N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever // Journal of Machine Learning Research. – 2014. – №15. – P. 1929–1958.

40. How to Check if Time Series Data is Stationary with Python [Електронний ресурс] – Режим доступу : www/ URL: https://machinelearningmastery.com/time-series-data-stationary-python/ – 15.05.2024 р. – Загол. з екрану.

41. Ivaschenko, G. S. Time series forecasting on the basis of the case-based reasoning using the models of artificial immune systems [Текст] / H. S. Ivaschenko, N. M. Korablev // System technologies. – 2014. – № 6. – P. 43–51.

42. Understanding LSTM Networks [Електронний ресурс] – Режим доступу : www/ URL: http://colah.github.io/posts/2015-08-UnderstandingLSTMs/ – 15.11.2021 р. – Загол. з екрану.

43. Jose, M. Adding external factors in Time Series Forecasting [Текст] / M. Jose. – School of Electrical Engineering and Computer Science (EECS), 2020. – 68 p.

44. Temporal Convolutional Networks, The Next Revolution for TimeSeries? [Електронний ресурс] – Режим доступу : www/ URL: https://towardsdatascience.com/temporal-convolutional-networks-the-nextrevolution-for-time-series-8990af826567. – 15.05.2024 р. – Загол. з екрану.

45. Іващенко, Г. С. Моделі глибокого навчання для прогнозування часових рядів [Текст] / Г. С. Іващенко, Д. О. Тимошенко, О. В. Близнюк, О. М. Кононенко // Системи управління, навігації та зв’язку. – 2024. – № 1(75). – С. 82–87.