МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ

завідувач кафедри
мережевих та інтернет технологій

_____ Ю.В. Кравченко

«_____»_____2021 року

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

галузі знань 17 «Електроніка та телекомунікації»
за спеціальністю 172 «Телекомунікації та радіотехніка»

на тему:

УПРАВЛІННЯ ІНВЕСТИЦІЙНИМ ПОРТФЕЛЕМ З ВИКОРИСТАННЯМ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

Виконав: студент групи МІТ -41

Ігнатюк Сергій Володимирович

(прізвище ім'я по-батькові) (підпис)

Керівник: доцент кафедри мережевих та інтернет технологій

к.т.н. Герасименко О.Ю.

(посада, прізвище ім'я по-батькові) (підпис)

Київ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій
Кафедра мережевих та інтернет технологій

ЗАТВЕРДЖУЮ
завідувач кафедри
мережевих та інтернет технологій

_____ Ю.В. Кравченко

«_____»_____2021 року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ

Здобувачу вищої освіти Ігнатюку Сергію Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи:
«Управління інвестиційним портфелем із використанням алгоритмів машинного навчання»
затверджена на засіданні кафедри МІТ «4» грудня 2020 р. протокол № 8
2. Термін здачі закінченої роботи «31» травня 2021 р.
3. Вихідні дані до проекту (роботи)
Дані котування акцій

4. Зміст пояснювальної записки (перелік питань, що їх потрібно розробити, обсяг – 35-40 стор.)
1. Аналіз задачі використання алгоритмів машинного навчання для управління інвестиційним портфелем
2. Аналіз літературних джерел. Визначення шляхів та методів вирішення задачі
3. Розробка моделі для прогнозування ціни акції з використанням алгоритмів машинного навчання
4. Шляхи практичного використання запропонованої моделі. Розробка телеграм-бота

5. Перелік графічного матеріалу 8-10 слайдів
Постановка задачі, порівняльний аналіз алгоритмів машинного навчання для прогнозування часових рядів, LSTM-нейромережа, розробка прогностичної моделі для ціни акції, розробка діаграми станів та діаграми кооперації телеграм-бота, створення телеграм-бота, висновки

Дата видачі завдання _____
Керівник роботи Доцент кафедри мережевих та інтернет технологій к.т.н. О.Ю. Герасименко
(підпис) (посада, прізвище, ім'я, по батькові)

Завдання прийняв до виконання _____
(підпис) (прізвище, ім'я, по батькові)

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ РОБОТИ

Номер	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Підготовчий	29.01.2020	
2	Розділ 1	01.03.2020	
3	Розділ 2	01.04.2020	
4	Розділ 3	01.05.2020	
5	Розділ 3	20.05.2020	
6	Доповідь та слайди	27.05.2020	
7	Пояснювальна записка	31.05.2020	

Здобувач вищої освіти _____Ігнатюк Сергій Володимирович
(підпис) (прізвище, ім'я, по батькові)

Керівник _____Герасименко Оксана Юріївна
(підпис) (прізвище, ім'я, по батькові)

РЕФЕРАТ

Пояснювальна записка: 63 с., 15 рис., 3 додатки, 18 джерел.

Об’єкт дослідження: динаміка цін акцій на фондовому ринку.

Предмет дослідження: алгоритми машинного навчання для прогнозування вартості акції компаній.

Мета роботи (проекту): спроектувати та реалізувати прогностичну модель для передбачення у короткостроковій перспективі ціни акції компанії на фондовому ринку; створення боту для її практичного використання.

Методи дослідження: системний підхід, порівняльний аналіз, алгоритми машинного навчання.

У спеціальній частині розглянуто сучасний стан і особливості алгоритмів для передбачення часових рядів.

У роботі проведено аналіз підходів та інструментів для створення та налаштування моделі машинного навчання.

Запропоновано використання рекурентної нейронної мережі для короткострокового прогнозування вартості акції.

Спроектовано модель нейронної мережі для передбачення ціни акції, створено телеграм-бота для практичного застосування моделі.

Реалізовано код програми мовою програмування Python з використанням бібліотек машинного навчання Keras і TensorFlow.

Практичне значення роботи полягає у розробці телеграм-бота для допомоги інвесторам різних рівнів у аналізі та вирішенні питань інвестиційного портфеля.

Результати здійснених у дипломному проекті досліджень можуть бути використані як продукт для інвесторів.

Галузь використання – інвестування, аналітика даних.

Напрямки подальшого розвитку роботи полягають у вдосконаленні створеної нейромережі з метою підвищення ефективності; додавання додаткових

метрик для гнучкого аналізу; додавання функціоналу передбачення цін криптовалют.

Ключові слова: АЛГОРИТМ МАШИННОГО НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, ПРОГНОЗ ЧАСОВИХ РЯДІВ, ТЕЛЕГРАМ-БОТ, ДОВГА КОРОТКОЧАСНА ПАМ'ЯТЬ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

LSTM	Long short-term memory
ДКП	Довга Короткочасна Пам'ять

ВСТУП

Отримання пасивного заробітку є однією з найпопулярніших тем сьогодення. Використання накопиченого людьми великого об’єму даних допомагає розв'язувати проблеми ефективніше та оминати помилки через людський фактор, у тому числі у сфері інвестування.

Розвиток сучасних технологій, таких як машинне навчання та нейромережі, сприяв тому, що люди мають можливість вкладати свої кошти набагато ефективніше та безпечніше. Завдяки розробленим моделям, нейромережа може майже точно передбачити подальшу ціну акції, використовуючи для навчання історичні дані по ціні акцій.

Використовуючи моделі машинного навчання та ботів як сервіс отримання результату нейромережі, можна розв'язувати проблему людей, які не мають великого досвіду у сфері економіки та інвестицій. Замість того, щоб витрачати роки за аналізом та стеженням за ринком великих компаній та навчатися на помилках, комп’ютерна програма може навчитися цьому за день, користуючись зібраними даними та відповідним алгоритмом.

Алгоритм вивчає та визначає закономірності у цінах акцій компаній за певний проміжок часу, а потім його можна використати для передбачення ціни на акцію на короткий термін. Таким чином, людина, виходячи з результату, який видає модель, основана на машинному навчанні, дає можливість зробити вибір у купівлі або продажі акції.

Нині допомога у прогнозуванні ціни акції є актуальним завданням. Це обумовлюється необхідністю аналізу історичних даних великих об’ємів. Одним з підходів для розв'язання даної задачі є використання довгої короткочасної пам'яті. Економія часу в результаті автоматизації аналізу великої кількості даних дозволить виявити закономірність даних, більш якісно спланувати операцію, що є визначальним для отримання заробітку. Бот, який надасть можливість будь-кому скористатися прогностичною моделлю, робить розробку широко доступною.

1. АНАЛІЗ ЗАДАЧІ ВИКОРИСТАННЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ ДЛЯ УПРАВЛІННЯ ІНВЕСТИЦІЙНИМ ПОРТФЕЛЕМ

1.1 Особливості використання алгоритму машинного навчання для передбачення ціни акції

Машинні алгоритми на сьогодні є одним з ефективних методів розв'язання проблеми, які допомагають людям у будь-якій задачі, починаючи з прибирання та закінчуючи складним проєктуванням будинку або автомобіля, отримуючи при цьому високу ефективність. Для отримання аналогічного результату, людина витрачає багато часу на створення та тестування свого продукту, коли ж за допомогою алгоритму можна здобути набагато ефективніший продукт за короткий проміжок часу.

Прогнозування фондового ринку – одна з найскладніших задач у сучасному світі. Аналітики роками досліджують та створюють різноманітні методи прогнозування поведінки фондового ринку, але через різке технологічне зростання, деякі з методів перестають діяти. Багато різних факторів впливає на результат, що робить ціни на акції нестабільними та важкими для прогнозування з високим ступенем точності.

Інвестиції поділяють на короткострокові (short term) та довгострокові (long term). Короткострокові полягають у тому, щоб купити по низькій ціні для отримання максимальної вигоди. Тоді як суть довгострокових інвестицій є знаходження перспективних компаній та вкладання у них грошей строком від 3 років.

Для розуміння та прогнозування ціни акції аналітики використовують так званий технічний аналіз, який містить в собі аналіз графіків ціни акцій та використання статистичних показників для виявлення тенденцій на фондовому ринку, або фундаментальний аналіз, що передбачає аналіз майбутньої прибутковості компанії на основі її поточного ділового середовища та фінансових результатів.

Короткострокові інвестиції є ефективним способом інвестування за допомогою машинного навчання, бо є багато факторів, які впливають на довгострокове інвестування. По динаміці ціни акції можна розпізнати ріст або падіння за певними ознаками, тобто особливими характеристиками, за якими слідує вирішальне прогнозування росту або падіння ціни.

Метрика – спосіб визначення відстані між елементами універсальної множини. Чим менше ця відстань, тим більше схожими є сутності. Ефективність розпізнавання напряму залежить від вибору точок на графіку і від реалізації метрики.

Найбільш ефективним у прогнозуванні на основі історичних даних на короткостроковий час є нейронні мережі. Навчання нейромережі є одним з ключових факторів у роботі машинного алгоритму. Основна ідея полягає в тому, що мережі можуть вчитися на основі даних, визначати закономірності та приймати рішення з мінімальним втручанням людини.

Навчання – це процес, у якому мережа налаштовує свої параметри шляхом моделювання середовища, завдяки чому модель навчається на своїх помилках та через кожних прохід змінює свої показники на більш точні.

Адаптація – це процес зміни параметрів та структури системи на основі поточної інформації з метою досягнення певного стану системи при початковій невизначеності та в мінливих умовах роботи.

Людський мозок є прикладом роботи нейромережі, де нейрони з’єднані між собою й утворюють зв’язки, по яким передається інформація.

Спектр завдань, які можуть бути вирішені за допомогою визначення закономірностей часових рядів, надзвичайно широкий. Особливо актуальними стають задачі, які мають нетривіальну постановку, для яких відсутні оптимальні алгоритми їх розв’язку. Власне кажучи, при розв’язуванні саме таких задач найчастіше використовують нейронні мережі.

Штучні нейронні мережі можуть моделювати роботу біологічних нейронів та дозволяють наблизитися до можливостей обробки інформації людським мозком. Такі штучні нейрони можуть розпізнавати або класифікувати образи;

навчатись на основі досвіду; прогнозувати події; обробляти рукописний текст та багато іншого. Нині існує багато прикладів використання штучних нейронних мереж. Прикладом успішного застосування нейронних мереж у галузі економіки є прогноз ситуації на фондовому ринку, оцінка вартості нерухомості, прогнозування динаміки біржових курсів, оптимізація товарних і грошових потоків, автоматичне зчитування чеків і форм тощо.

Штучні нейрони можуть бути використані майже у всіх сферах діяльності людини, але більш доцільно їх використовувати там, де не відомі принципи розв'язання поставленої задачі, але накопичено достатньо прикладів їх розв’язання, або існують неповні дані.

Найчастіше нейронні мережі застосовують для вирішення наступних задач:

1. Розпізнавання образів та класифікація. У якості образів можуть виступати різні за своєю природою об’єкти: символи тексту, зображення, зразки звуків і т. д. При навчанні мережі пропонуються різні зразки образів із зазначенням того, до якого класу вони відносяться. Коли мережі пред’являється якийсь образ, на одному з її виходів повинна з’явитися ознака того, що образ належить цьому класу. Одночасно на інших виходах повинна бути ознака того, що образ до даного класу не належить.
2. Кластеризація. Близькі образи відносять в одну групу. Після навчання така мережа здатна визначати, до якого класу належить вхідний сигнал.
3. Прогнозування. Мережа здатна передбачити деяке значення у деякий майбутній момент часу, на основі попередніх значень або діючих чинників.
4. Оптимізація. Завданням оптимізації є знаходження такого рішення, яке задовольняє заданим обмеженням і максимізує чи мінімізує цільову функцію.
5. Стиснення даних і асоціативна пам’ять. Здатність нейромереж до виявлення зав'язків між різними параметрами дає можливість узагальнити дані, якщо ці данні тісно пов'язані між собою. Асоціативна пам'ять – це зворотній до стиснення процес.

Таблиця 1.1 – Переваги і недоліки нейронних мереж

Нейронні мережі		
№	**Переваги**	**Недоліки**
1	Можуть керувати процесом, який ще не має математичного опису або іншого обґрунтування.	Не мають достатнього рівня надійності, можуть “помилятись”.
2	Адаптується до зовнішніх умов керування процесом і постійно переналагоджується під умови зовнішнього середовища, які змінюються.	Не можна повністю гарантувати, що розроблена нейронна мережа є найбільш оптимальною.
3	Досягається економія часу внаслідок гнучкості переналагодження та шляхом виконання паралельних обчислень.	У загальному випадку невідома кількість потрібних внутрішніх шарів та кількість нейронів у цих шарах.
4	Мають здатність до узагальнення, навчання та самонавчання у реальному часі. Штучна нейронна мережа працює автоматично внаслідок своєї структури, а не внаслідок використання інтелекту людини у формі комп’ютерних програм.	Немає гарантії, що мережа є оптимальною і може бути навчена за визначений час; немає впевненості, що мережа не потрапить у локальний мінімум і тому може отримати гірше навчання, у порівнянні з оптимумом.
5	Толерантні до помилок шляхом розподілу інформації у зв’язках та вагах нейронних мереж. Стан окремого нейрону визначається станом багатьох інших нейронів, пов’язаних з ним. Тому втрата одного або кількох зв’язків, або перекручення інформації і відмова окремих елементів не впливає кардинально на результат роботи у цілому.	Процес навчання не завжди може завершитись успіхом.

Зазвичай нейронні мережі є спеціалізованими, тобто призначені для виконання конкретних задач. Мережі також мають певні переваги на недоліки у використанні, і їх нераціонально використовувати для вирішення певних типів задач. У таблиці 1.1 коротко розглянуто переваги та недоліки нейромереж.

1.2 Аналіз існуючих підходів до навчання нейромережі

Як і їх біологічні прототипи, штучні нейронні мережі мають здатність навчатися, тобто вдосконалювати свою роботу під впливом середовища, змінюючи свої параметри. Існує багато визначень терміна "навчання", але наступне найбільше стосується штучної нейромережі.

Процес, за допомогою якого вільні параметри нейронної мережі адаптуються в результаті її безперервного стимулювання навчальною вибіркою, називається навчанням. Тип навчання визначається тим способом, яким виробляються зміни параметрів.

Можливість навчання є однією з головних переваг нейронних мереж перед традиційними алгоритмами. Існує безліч різних алгоритмів навчання, які діляться на два великі класи: детерміновані і стохастичні. У першому з них підстроювання ваг являє собою жорстку послідовність дій, у другому – підстроювання відбувається на основі дій, які підкоряються деякому випадковому процесу. Коротко розглянемо підходи до навчання нейромережі.

1.3.1 Навчання з учителем

Під час навчання з учителем нейронна мережа вчиться з міченого набору даних і прогнозує відповіді, які використовуються для оцінки точності алгоритму на навчальних даних.

Навчання з учителем – це тип мережевого навчання, в якому його ваги змінюються так, що відповіді в мережі мінімально відрізняються від готових правильних відповідей.

Розглянемо алгоритм навчання з учителем (рис. 1.1).

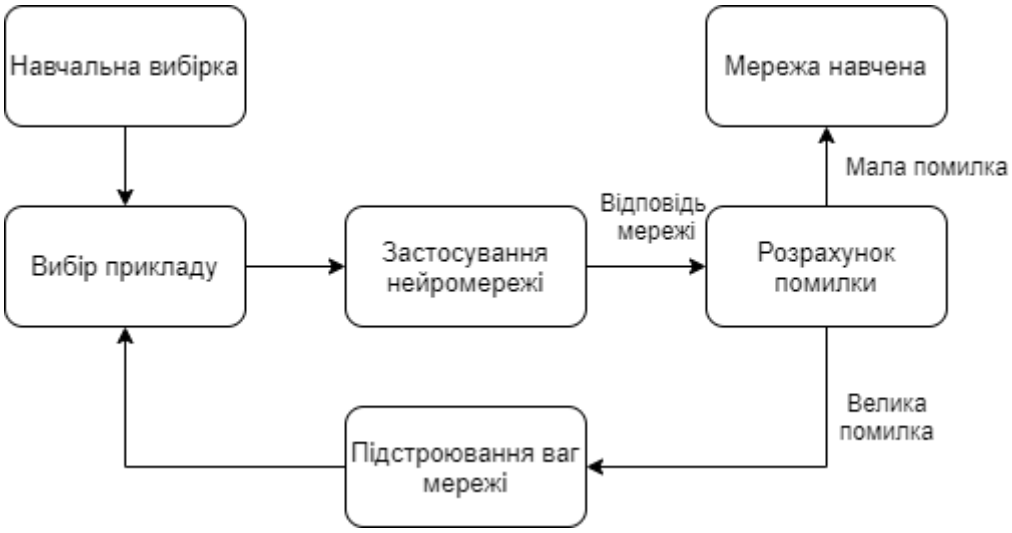

Рисунок 1.1 – Схема процесу навчання нейромережі з учителем

Коли в мережі тільки один шар, алгоритм її навчання з учителем досить очевидний, так як правильні вихідні стани нейронів єдиного шару відомі, і підстроювання синаптичних зв'язків йде в напрямку, що мінімізує помилку на виході мережі. За цим принципом будується, наприклад, алгоритм навчання одношарового персептрона[6].

В основному навчання з учителем застосовується для вирішення двох типів задач: класифікації і регресії.

У задачах класифікації алгоритм пророкує дискретні значення, які відповідають номерам класів, до яких належать об'єкти. Якість алгоритму оцінюється тим, наскільки точно він може правильно класифікувати об'єкти. А ось завдання регресії пов'язані з безперервними даними. Один із прикладів, лінійна регресія, обчислює очікуване значення змінної *y*, враховуючи конкретні значення *x*.

Отже, при навчанні мережі з учителем їй на входи подають сигнали, а потім порівнюють її вихід із заздалегідь відомим правильним значенням. Цей процес повторюють до тих пір, поки не буде досягнута необхідна точність відповідей.

Таким чином, навчання з учителем найбільше підходить для задач, коли є значний набір достовірних даних для навчання алгоритму.

1.3.2 Навчання без учителя

Ідеально помічені та чисті дані отримати непросто. Тому іноді завдання алгоритму – заздалегідь знайти невідомі відповіді. Якщо мережі тільки подають вхідні сигнали, без порівняння їх з готовими виходами, то мережа починає самостійну класифікацію цих вхідних сигналів. Іншими словами, вона виконує кластеризацію вхідних сигналів. Таке навчання називають навчанням без учителя.

Навчання без учителя – це тип мережевого навчання, в якому мережа незалежно класифікує вхідні сигнали. Правильні вихідні сигнали не показані.

Залежно від завдання модель систематизує дані по-різному.

До найбільш поширеного завдання для навчання без учителя відноситься кластеризація. Алгоритм підбирає схожі дані, знаходячи спільні ознаки, і групує їх разом. Також навчання без вчителя використовують для знаходження викидів в даних, іншими словами, аномалій в даних. Ще один алгоритм – виявлення асоціацій. Деякі характеристики об'єкта корелюють з іншими ознаками. Розглядаючи пару ключових ознак об'єкта, модель може передбачити інші, з якими існує зв'язок.

У навчанні без учителя є недолік – складно обчислити точність алгоритму, тому що в даних відсутні «правильні відповіді» або мітки. Але мічені дані часто є недостовірними або їх складно отримати. У таких випадках, надаючи моделі свободи дій для пошуку залежностей, можна отримати хороші результуючі виходи.

1.3.3 Навчання нейромережі, засноване на корекції помилок

Навчання, основане на корекції помилок, реалізує метод оптимальної фільтрації. Метод корекції помилок був запропонований Френком Розенблаттом.

Це метод навчання являє собою такий метод навчання, при якому вага зв'язку не змінюється до тих пір, поки поточна реакція перцептрона залишається правильною. При появі неправильної реакції вага змінюється на одиницю, а знак (+/-) визначається протилежним від знаку помилки.

На рисунку 1.2 зображена узагальнена схема нейронної мережі при навчанні, яке засноване на корекції помилок. Нейрон q працює під керівництвом вектору сигналу *x(n),* що виробляється одним або декількома прихованими шарами нейронів, які , у свою чергу, отримують інформацію від вхідного сигналу. Далі цей сигнал передається у вхідний шар нейронної мережі. Під літерою n мається на увазі номер ітеративного процесу налаштування синаптичних ваг нейрона. Вихідний сигнал нейрона позначається $y_q(n)$. Цей сигнал є єдиним виходом нейронної мережі. Він і буде порівнюватись з бажаним виходом $d_q(n)$. У кінцевому підсумку отримуємо сигнал помилки $e_q(n)$[7].

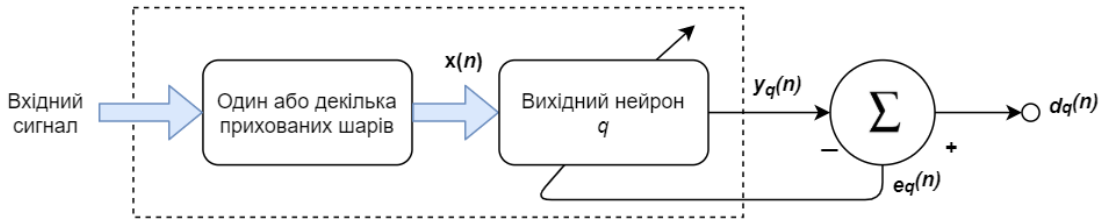

Рисунок 1.2 – Узагальнена схема нейронної мережі при навчанні, що засноване на корекції помилок

З цього випливає наступне рівняння:

$$e_q(n) = d_q(n) - y_q(n). \qquad (1.1)$$

Сигнал помилки призводить до стану готовності механізм управління, мета якого полягає в тому, щоб застосувати послідовність корекції до синаптичних ваг нейрона q. Ці зміни робляться для того, щоб поступово наблизити вихідний сигнал $y_q(n)$ до бажаного $d_q(n)$.

Надана модель використовує метод навчання з учителем, тобто навчальна множина складається із сукупності вхідних векторів, для кожного з яких вказано

вихідний вектор. Не звертаючи уваги на деякі обмеження, модель стала основою для безлічі передових алгоритмів навчання.

1.3.4 Конкурентне навчання

Як випливає із самої назви, в конкурентному навчанні вихідні нейрони нейронної мережі конкурують між собою за право бути активованими. Особливістю є те, що в конкурентній мережі в кожний момент часу у збудженому стані може знаходитися лише один нейрон. Завдяки цій властивості конкурентне навчання дуже зручно використовувати для дослідження статистичних якостей, що найчастіше використовуються в задачах класифікації вхідних образів.

Метод конкурентного навчання засновано на застосуванні трьох основних складових[8]:

1. велика кількість подібних нейронів з випадково розподіленими синаптичними вагами, що призводять до різної реакції нейронів на один і той же вхідний сигнал;
2. ліміт сили кожного нейрона;
3. механізм, що дозволяє нейронам конкурувати за право відгуку на дану кількість вхідних сигналів і визначає єдиний активний вихідний нейрон. Нейрон, що переможе в цьому змаганні, називають “нейроном-переможцем”. А принцип конкурентного навчання можна описати так – “переможець забирає все”.

Алгоритм “переможець забирає все” реалізує принцип навчання без учителя, оскільки нейрон-переможець обирається по максимальному рівню активності.

Метод конкурентного навчання можна комбінувати з іншими нейромережевими алгоритмами та архітектурами, і забезпечити тим самим більш складні моделі навчання.

1.3.5 Алгоритм зворотного поширення помилки

Метод зворотного поширення помилки – це метод навчання багатошарового перцептрона. Вперше цей метод був описаний у 1974 році. Основна ідея цього методу полягає в поширенні сигналів помилки від виходів мережі до її входів, в напрямку, зворотному прямому поширенню сигналів у звичайному режимі роботи.

Алгоритм зворотного поширення помилки забезпечує спосіб налаштування ваг з урахуванням багатошарової структури нейронної мережі. У рамках цього підходу помилка на виході мережі поширюється в зворотному напрямку до прихованих шарів (рис. 1.3).

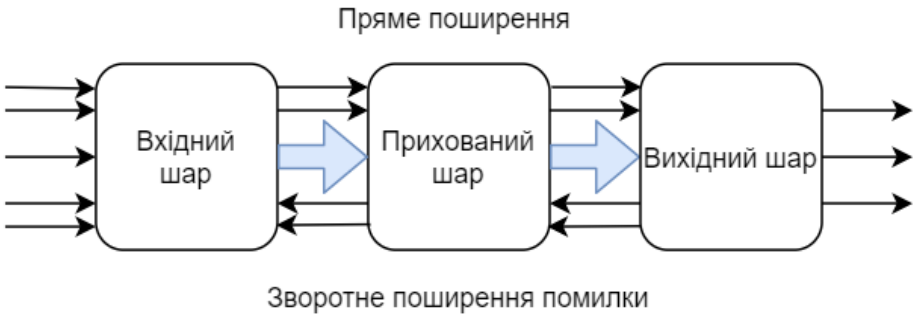

Рисунок 1.3 – Узагальнена схема зворотного поширення помилки в нейронній мережі з прихованим шаром

Навчання методом зворотного поширення помилки передбачає два проходи по всім шарам мережі: прямого і зворотного. При прямому проході вхідний вектор подається на сенсорні вузли мережі, після чого поширюється по мережі від шару до шару. В результаті генерується набір вихідних сигналів, який і є фактичною реакцією мережі на даний вхідний вектор. Під час прямого проходу всі синаптичні ваги мережі фіксовані. Під час зворотного проходу всі синаптичні ваги налаштовуються у відповідності з правилом корекції помилок, а саме: фактичний вихід мережі віднімається від бажаного відгуку, в результаті чого

формується сигнал помилки. Цей сигнал згодом поширюється по мережі в напрямку, протилежному напрямку синаптичних зв'язків.

1.3.6 Формування навчальної вибірки

Формування навчальної вибірки – це збір даних для навчання нейронної мережі. Формування навчальної вибірки є непростим завданням і потребує багатьох ресурсів.

Для формування правильної вибірки дані повинні задовольняти певним критеріям:

- дійсність даних – дані повинні показувати дійсне положення речей;
- узгодженість даних – суперечливі дані можуть призвести до поганої якості навчання нейромережі;
- обсяг – для того щоб мережа функціонувала правильно, необхідно щоб число записів у вибірці значно перевищувало кількість зв'язків між нейронами в цій мережі. В іншому випадку, мережа не зможе правильно обробити дані і виконати узагальнення.

Висновки по розділу I

У даному розділі було проаналізовано задачу керування інвестиційним портфелем та визначено, що алгоритми машинного навчання можуть стати гарним підґрунтям для її вирішення.

Нейронні мережі можна розділити на одношарові та багатошарові. Одношарова мережа – найпростіша нейронна мережа, у якій сигнали відразу подаються на вихідний рівень. Багатошарові нейронні мережі мають більші можливості, ніж одношарові. У багатошаровій мережі, між вхідним та вихідним шарами, розташовані приховані шари. Кількість шарів визначає складність мережі.

Однією з особливостей використання нейронних мереж є можливість навчання. Існує безліч різних алгоритмів навчання. У розділі розглянуто деякі найвідоміші з них. Алгоритми можна поділити на дві категорії: навчання з учителем та навчання без учителя. Вони відрізняються тим, що в першому варіанті синаптичні ваги змінюються так, щоб відповіді мережі мінімально відрізнятись від готових правильних відповідей. При навчанні без учителя правильні відповіді заздалегідь не відомі. Найчастіше навчання без учителя використовують для вирішення завдання кластеризації.

2. АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ. ВИЗНАЧЕННЯ ШЛЯХІВ ТА МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

2.1 Порівняльний аналіз інструментів для роботи з алгоритмами машинного навчання

Завдяки програмному забезпеченню з відкритим кодом, стало набагато легше впроваджувати машинне навчання з використанням найпопулярніших мов програмування в мережах та на окремих пристроях. До інструментів для навчання відносяться бібліотеки на мовах програмування C++, Java, JavaScript, Python, R та багато інших.

Розглянемо інструменти для роботи з машинними алгоритмами.

1. Scikit-learn – бібліотека для мови програмування Python.

Являє собою модуль Python для машинного навчання, побудований на версії SciPy. Проект був розпочатий у 2007 році Девідом Курнапо як проект Google Summer of Code, і відтоді багато волонтерів зробили свій внесок[9].

Переваги:

- найбільшими перевагами є доступність та простота використання;
- має прості та ефективні інструменти для аналізу даних;
- можна об’єднати з іншими бібліотеками, наприклад з NumPy, SciPy і Matplotlib, щоб створити інтерактивний додаток у середовищі розробки або вбудувати в інше програмне забезпечення і використовувати заново;
- пропонує безліч інструментів для маніпулювання даними та таких утиліт, як функції метрик, генерації штучних наборів даних тощо.

Недоліки:

- надає API моделі, що не володіють достатньою гнучкістю;
- деякі моделі, наприклад Random Forests, мають нестандартні або помилкові реалізації;
- не підходить для глибоких досліджень нейронних мереж;

- може працювати занадто повільно у зв’язку з тим, що не використовує апаратне прискорення;
- через простоту використання може спонукати молодших фахівців в області даних обходитися без розуміння основ.

2. Shogun[17] – це бібліотека машинного навчання з відкритим програмним кодом.

Пропонує широкий спектр ефективних і уніфікованих методів машинного навчання. Проект був створений в 1999 році і написаний на C ++, але він може використовуватися з мовами Java, Python, C #, Ruby, R, Lua, Octave і Matlab.

Переваги:

- широкий спектр стандартних та передових алгоритмів;
- швидке створення прототипів;
- зручно використовувати під час робочих процесів;
- ліцензія з відкритим програмним кодом;
- може одночасно обробляти велику кількість даних;
- однією з ключових особливостей є комбіноване ядро для побудови зважених лінійних комбінацій декількох ядер, які можуть бути визначені для різних вхідних доменів;
- надає інтерактивні інтерфейси користувача для більшості основних мов програмування.

3. TensorFlow – це безкоштовна бібліотека машинного навчання з відкритим програмним кодом, яка призначена для чисельних обчислень з використанням графіків потоку даних.

TensorFlow спочатку був розроблений дослідниками та інженерами, які працюють в команді Google Brain в рамках дослідницької організації Google Machine Intelligence для проведення машинного навчання та глибоких досліджень нейронних мереж, але його можна використовувати і для широкого спектру досліджень[11].

Переваги:

- призначений для глибокого навчання нейронних мереж;

- гнучка архітектура, яка дозволяє розгортати її на одному або декількох процесорах або графічних процесорах на настільному комп'ютері, сервері або мобільному пристрої з одним і тим же API.

Недоліки:

- недоцільно використовувати для вирішення простих задач;
- TensorFlow є надмірним для більшості практичних завдань з машинного навчання;
- важкий у використанні;
- не повністю відкритий код.

4. Weka – це програмне забезпечення для машинного навчання з відкритим кодом, до якого можна отримати доступ через графічний інтерфейс користувача, стандартні програми терміналів або Java API.

Програмне забезпечення широко використовується для викладання, досліджень та промислових застосувань, містить безліч вбудованих інструментів для стандартних завдань машинного навчання, а також надає прозорий доступ до відомих наборів інструментів, таких як scikit-learn, R та Deeplearning4j[12].

Переваги:

- надає онлайн курси для навчання;
- підходить для студентів;
- алгоритми роботи легко зрозуміти та освоїти.

Недоліки:

- не достатньо документації;
- не має онлайн підтримки.

5. PyTorch – фреймворк машинного навчання для мови Python з відкритим вихідним кодом, створена на базі Torch. Використовується для вирішення різних завдань: комп'ютерний зір, обробка природної мови. Розробляється переважно групою штучного інтелекту Facebook[16].

Переваги:

- є курси для навчання та практичних знань для роботи з фреймворком;
- легке внесення змін і експериментування;

- хороша документація.

Недоліки:

- немає системи для візуалізації (порівнюючи з Tensorflow);

Алгоритм машинного навчання потребує багато обчислювальних ресурсів для навчання моделі. Чим більше обчислювальна потужність комп'ютерної системи, тим швидше модель буде навчатися.

Але якщо потужності не вистачає, є хмарні сервіси, які пропонують безкоштовну допомогу у цій ситуації.

Одними з таких сервісів є:

- Google Colab;
- JupyterLab;
- Gradient by paperspace;

Google Colab – це безкоштовний хмарний сервіс на основі Jupyter Notebook. Основною особливістю Google Colab є те, що він надає все необхідне для машинного навчання прямо в браузері. Середовище розробки Colab розроблено на основі Jupyter Notebook – це інтерактивне обчислювальне середовище.

Jupyter Notebook поєднує у собі три компоненти:

1. Веб-додатки – це інтерактивна програма для написання та запуску кодів. Веб-додатки дозволяють користувачам редагувати та запускати код в браузері, переглядати результати обчислень з мультимедійним поданням (HTML, LaTeX, PNG, SVG, PDF тощо), створювати та використовувати інтерактивні віджети JavaScript та інше.
2. Ядра – окремі процеси, що виконують веб-додатки, запуск коду користувача та повернення вихідних даних. Також ядро може виконувати різні обчислення. Завдяки архітектурі ядер Jupyter Notebook дозволяє запускати код на різних мовах програмування. Для кожного документа, який відкриває користувач, веб-додаток запускає ядро. Кожне ядро здатне виконати код однією мовою програмування, за замовчуванням використовується мова програмування Python.

3. Документи блокнота – автономні документи, які містять представлення всього вмісту, видимого у веб-додатку, включаючи обчислення введення і виведення, текст, рівняння, зображення і мультимедійні представлення об'єктів. У кожного документа є своє ядро. Крім того, будь-який документ із блокнота, доступний із загальнодоступної URL-адреси або на GitHub.

Фактично Google Colab та Jupyter Notebook два дуже схожих середовища розробки. Можна виділити такі основні переваги використання Google Colab:

- платформа доступна безкоштовно;
- Google Colab надає вбудовану систему контролю версій з Git, і дозволяє легко створювати нотатки та документацію, зображення і електронні таблиці;
- дозволяє використовувати безкоштовний графічний процесор в середовищі Colab, незалежно від того, який персональний комп'ютер використовується;
- програмування моделей поглибленого навчання з використанням графічних процесорів;
- використовує переваги спільної роботи над Документами Google;
- працює на серверах Google з використанням віртуальних машин.

JupyterLab – це ще один безкоштовний хмарний сервіс який дає змогу гнучко, інтегровано та розширювано працювати з документами та такими видами діяльності, як блокноти Jupyter, текстові редактори, термінали та користувацькі компоненти.

Основними перевагами системи є:

- консолі коду забезпечують тимчасові блокноти для запуску коду в інтерактивному режимі, з повною підтримкою розширеного виводу. Наприклад, консоль коду може бути пов'язана з ядром блокнота як журнал обчислень;
- документи, що підтримуються ядром, дозволяють інтерактивно запускати код у будь-якому текстовому файлі (Markdown, Python, R, LaTeX тощо) у будь-якому ядрі Jupyter;

- кілька переглядів документів з різними редакторами або програмами перегляду дозволяють редагувати документи в реальному часі, відображені в інших програмах перегляду. Наприклад, легко отримати попередній перегляд документів Markdown.

Gradient by paperspace[18] - найбільш розвинутий хмарний сервіс для розробки й навчання машинних алгоритмів. Платформа дозволяє використовувати відеокарти різних рівнів, що дає змогу пришвидшити навчання моделі в рази.

Переваги використання Gradient:

- гнучке налаштування блокноту, де є вибір популярних фреймворків, які автоматично будуть підключені до середовища;
- використання відеокарт для навчання моделі;
- за додаткову плату можна збільшувати кількість та обчислювальну потужність для середовища.

2.2 Вибір інструментів роботи з алгоритмами машинного навчання

Технології машинного навчання розвиваються дуже стрімко, це призвело до появи багатьох інструментів для роботи з машинним навчанням.

Для реалізації алгоритму варто використати мову програмування Python. У якості основи буде використано Keras, який є висококласним API нейронної мережі, написаним на Python. Разом з Keras буде використано спеціальну бібліотеку машинного навчання – TensorFlow від Google. Як середовище розробки і тестування – Jupyter Notebook.

Мова програмування Python – це проста високоповерхова інтерпретована мова програмування, яку можна використовувати для будь-яких завдань. Також ця мова підходить для новачків в програмуванні, бо є легкою для вивчення у порівнянні з іншими мовами. Також, Python є найпопулярнішою мовою серед інженерів для машинного навчання та нейронних мереж[10].

Причин для використання Python:

1. підготовленні бібліотеки, такі як NumPy, SciPy та Pandas та інші, роблять наукові розрахунки легшими та швидшими, оскільки більшість цих бібліотек добре оптимізовані для загальних завдань машинного навчання.
2. Python не залежить від платформи та може працювати майже на будь-якому пристрої. Це означає, що Python легко сумісний на різних платформах і може бути розгорнутий практично в будь-якому місці, якщо його встановити на пристрій (як для мови Java).
3. Python має активну спільноту, яка надає низку документів та навчальних посібників для роботи з мовою. Більшість проблем з якими може стикнутись користувач є вирішеними.
4. легко компілюється у код іншої мови програмування, що дає змогу пришвидшити швидкість роботи коду (компіляція у мову програмування C).

Для роботи з нейронними мережами застосовуються допоміжні бібліотеки. Для розробки нейронної мережі, що представлена в даній роботі, використано бібліотеку під назвою Keras.

У машинному і глибокому навчанні, щоб забезпечити правильний результат, необхідно провести дуже складні розрахунки. З'явилися різні бібліотеки машинного навчання, які допомагають впоратися з цими труднощами.

Бібліотека Keras[13], яка написана мовою програмування Python, надає програмний інтерфейс високого рівня, що підходить для роботи зі штучними нейронними мережами. Цей продукт був створений для того, щоб полегшити розробку нейронних мереж. Компанія Google викупила цей інструмент у 2016 році та зробила його основним елементом верхнього рівня для бібліотеки TensorFlow.

Основні причини використання Keras засновані на його основних принципах, особливо на тому, що він зручний для користувача. На додаток до простого навчання і простого моделювання, Keras пропонує переваги широкого впровадження, підтримку багатьох платформ для впровадження, інтеграцію як мінімум з п'ятьма механізмами резервного копіювання (TensorFlow, CNTK, Theano, MXNet і PlaidML) і потужну підтримку декількох графічних процесорів і

розподіленого навчання. Крім того, Keras підтримує Google, Microsoft, Amazon, Apple, Nvidia, Uber та інші.

З Keras легко та зручно працювати. Ця бібліотека пропонує послідовний та простий програмний інтерфейс, мінімізує кількість необхідних дій, що робить код набагато меншим, та забезпечує чіткий і дієвий зворотний зв'язок. Оскільки Keras інтегрується з поглибленими мовами навчання (особливо TensorFlow), це дозволяє отримати більше можливостей. API Keras є офіційним інтерфейсом TensorFlow через модуль tf.keras.

Нейронні шари, функції втрат, оптимізатори, схеми ініціалізації, функції активації та схеми регуляризації – все це окремі модулі, які можна комбінувати для створення нових моделей. Нові модулі додаються дуже просто, як нові класи та функції.

Моделі Keras можна легко розгорнути на більш широкому діапазоні платформ, ніж інші фреймворки:

- на iOS через CoreML від Apple;
- на Android через модуль виконання TensorFlow для Android;
- у браузері використовуючи швидкі графічні процесори, такі як Keras.js та WebDNN;
- у Google Cloud через TensorFlow-Serving та інші.

Важливо, що моделі Keras можуть бути розроблені з використанням різноманітних програм глибокого навчання. Будь-яка модель Keras, яка використовує лише вбудовані шари, буде портативною. Тобто, може бути переміщена на різні бекенди. Наприклад, "The TensorFlow backend" (від Google), або "The Theano backend".

Перевага використання Keras[14] разом з TensorFlow полягає в тому, що, на відміну, від MXNet або Theano, велика увага приділяється реалізації TensorFlow. Тому більшість речей працюють без особливих проблем і досить добре оптимізовані.

TensorFlow – це бібліотека машинного навчання з відкритим програмним кодом, яка була розроблена компанією "Google" і випущена в 2015 році.

TensorFlow можна вважати новачком в машинному навчанні, але це не завадило цій бібліотеці отримати велику популярність, завдяки простоті використання в порівнянні з попередниками. Найбільш широко в машинному навчанні використовується TensorFlow для створення нейронних мереж, що можуть аналізувати почерк і розпізнавати обличчя. Хоча бібліотека була написана мовою програмування Python, завдяки популярності JavaScript, можна використовувати TensorFlow.js – це бібліотека для машинного навчання на JavaScript.

TensorFlow є гнучкою і може використовуватися для вираження широкого спектру алгоритмів, включаючи алгоритми навчання і логічного виводу для моделей глибоких нейронних мереж, також використовується для проведення досліджень і впровадження систем машинного навчання в виробництво в більш ніж десяти сферах інформатики та інших сферах. Наприклад: розпізнавання мови, робототехніка, пошук інформації та інші.

Щоб забезпечити більш високий рівень розгортання нейронної мережі, TensorFlow дозволяє клієнтам легко використовувати різні типи паралелізму за допомогою реплікацій та паралельної обробки задач між центральними та графічними процесорами GPU.

Розрахунки в TensorFlow подаються у вигляді орієнтованого графу, який складається з набору вузлів. Зазвичай граф обчислюється за допомогою однієї з підтримуваних мов інтерфейсу, наприклад Python. Значення, які розташовані уздовж нормальних ребер в графі (від виходів до входів), називаються тензорами, масивами довільної розмірності, де тип базового елементу вказується або виводиться під час побудови графа. Тобто тензор – це збірний багатовимірний масив[15].

Клієнтські програми взаємодіють із TensorFlow, створюючи сесію (Session). Щоб створити обчислювальний граф, інтерфейс сесії підтримує метод Extend для збільшення поточного графа, керованого сеансом, додатковими вузлами та ребрами (початковий граф, коли сеанс створюється, порожній).

Розрізняють два типи сесій: звичайні та інтерактивні. Більшість тензорів зникає після одного використання графа.

Кожна операція представляє абстрактний розрахунок. Операція може мати атрибути. Ядро – це конкретна реалізація операції, яку можна виконати на певному типі пристрою, наприклад графічний процесор.

Змінні (Variable) – це особливий вид операції, який повертає дескриптор постійного змінюваного тензора, який зберігається при використанні графа. Змінна не зникає після одного використання графа. Для машинного навчання TensorFlow параметри моделі, як правило, зберігаються в тензорах, що містяться в змінних, які оновлюються на кожному кроці навчання.

Одним з основних компонентів в системі TensorFlow є клієнт, який використовує сесію для зв'язку з одним або декількома робочими процесами, причому кожен робочий процес відповідає за доступ до одного або кількох обчислювальних пристроїв і для виконання вузлів графа на цих пристроях відповідно до вказівок майстра.

Існують як локальні, так і розподілені реалізації інтерфейсу TensorFlow. Локальна реалізація використовується, коли клієнт, майстер і робітник працюють на одному комп'ютері в контексті одного процесу операційної системи. Розподілена реалізація розділяє більшу частину коду з локальної реалізацією, але розширює її підтримкою середовища, в якій клієнт, майстер і робітники можуть перебувати в різних процесах на різних машинах.

Інтерфейс TensorFlow і еталонна реалізація відкриті за ліцензією Apache 2.0, система доступна для завантаження на сайті www.tensorflow.org. Система включає в себе детальну документацію, ряд навчальних посібників і ряд прикладів, які демонструють, як використовувати TensorFlow для різних цілей.

До переваг TensorFlow можна віднести:

- розширена функціональність. TensorFlow більш вдосконалений, особливо для операцій високого рівня, таких як багатопотоковість і черги;
- посилення контролю. Повний контроль не завжди необхідний, але деякі нейронні мережі можуть вимагати більш глибокого розуміння, особливо при роботі з такими операціями, як ваги або градієнти.

Звичайно є і недоліки, TensorFlow важко використовувати без попередньої підготовки. Завдяки бібліотеці Keras можна спростити та полегшити використання TensorFlow.

У даній роботі використовувався Jupyter Notebook для налаштування та тестування моделі машинного навчання.

2.3 Алгоритми машинного навчання у прогнозуванні часових рядів

Основні алгоритми для прогнозування часових рядів це:

- Autoregressive (AR);
- Moving Average (MA);
- Exponential Smoothing (ES);
- LSTM (Long-Short Term Memory).

Розглянемо кожен з них окремо.

Авторегресія (Autoregressive) – це модель регресії, така як лінійна регресія, моделює вихідне значення на основі лінійної комбінації вхідних значень. Цей прийом може бути використаний у часових рядах, де вхідні змінні приймаються як спостереження на попередніх етапах часу, які називаються змінними відставання. Наприклад, ми можемо передбачити значення для наступного часового кроку (t + 1), враховуючи спостереження на останніх двох часових кроках (t-1 і t-2). Як модель регресії це буде виглядати наступним чином:

$$X(t+1) = b0 + b1 * X(t-1) + b2 * X(t-2) \qquad (2.1)$$

Оскільки модель регресії використовує дані з тієї самої вхідної змінної на попередніх етапах часу, вона називається авторегресією (регресія самості).

Метод ковзного середнього (Moving Average) моделює наступний крок у послідовності як лінійну функцію залишкових помилок середнього процесу на попередніх етапах часу.

Ковзне середнє допомагає зменшити кількість "шуму" на графіку цін. Якщо подивитись на напрямок ковзної середньої, то можна отримати базове уявлення про те, в який бік рухається ціна. Якщо вона під кутом, ціна в цілому рухається

вгору, під кутом вниз, і ціна в цілому рухається вниз, рухається вбік, і ціна, ймовірно, коливається.

Експоненціальне згладжування (Exponential Smoothing) – це метод прогнозування часових рядів для одновимірних даних. Методи прогнозування експоненціального згладжування схожі за тим, що прогнозування є зваженою сумою минулих спостережень, але модель явно використовує експоненційно зменшувальну вагу для минулих спостережень.

Для аналізу та знаходження закономірностей в часових рядах підходить довга короткочасна пам'ять. ДКП – це архітектура штучної рекурентної нейронної мережі, що використовується в галузі глибокого навчання. На відміну від стандартних нейронних мереж прямого зв'язку, ДКП має з'єднання зворотного зв'язку. Він може не тільки обробляти окремі точки даних, але й цілі послідовності даних. Наприклад, ДКП застосовується до таких завдань, як несегментоване, підключене розпізнавання рукописного вводу, розпізнавання мови та виявлення аномалій у мережевому трафіку або системи виявлення вторгнень.

Шар ДКП складається з безлічі періодично з'єднаних блоків, відомих як блоки пам'яті. Ці блоки можна сприймати як диференційовану версію мікросхем пам'яті в цифровому комп’ютері. Кожна з них містить одну або кілька рекурентно підключених комірок пам'яті та три мультиплікативні блоки – вхідні, вихідні та забутні ворота – які забезпечують безперервні аналоги операцій запису, читання та скидання для комірок.

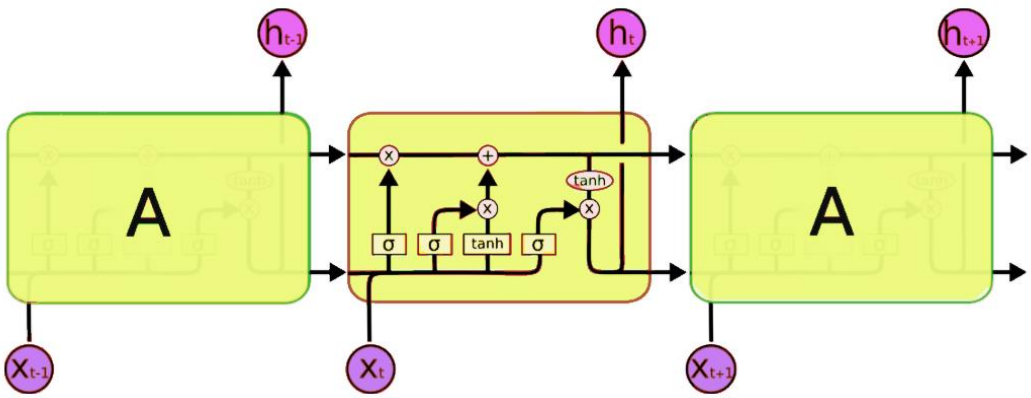

Рисунок 2.1 – Структура блоку LSTM [2]

Перевагою ДКП є те, що результат після навчання моделі зберігається, що дає можливість використовувати навчені дані для поліпшення та прискорення повторного навчання.

Для навчання штучної нейронної мережі може знадобитись багато часу. Проте, нейронні мережі досягли успіху в цьому типі завдань.

На відміну від рекурентних нейронних мереж, ДКП оновлює та зберігає стан, через що прискорює навчання. Використовуючи правильні налаштування та параметри для моделі машинного навчання, можна досягти оптимальної швидкості у навчанні. Не менш вагомим є кількість вибіркових даних, які використовуються для навчання. Чим більше та якісніше будуть ці дані, тим ефективніше буде робота самої моделі на тестових даних.

Коли модель ДКП навчається на даних, вона шукає особливості або закономірності, по яких можна передбачити подальшу динаміку ціни. Наприклад, якщо по даним, графік йде вверх, то висновок, що він буде рости й далі.

Алгоритм довгої короткострокової пам’яті був розроблений з метою виявлення закономірностей на часових рядах, що дає можливість передбачити подальший рух показника або, у нашому випадку, ціни акції. Алгоритм може використовуватись для будь-якої компанії, яка має інформацію про ціну акції та історичні дані про динаміку ціни. Найбільш ефективно алгоритм буде працювати, коли компанія знаходиться вже великий проміжок часу, що дає можливість мати велику кількість даних.

Використання та правильний вибір алгоритму є головним для розв'язання будь-якої задачі. Використання ДКП є ефективним, коли у завданні є часові рамки та суттєвий набір даних. З недавніми проривами, що відбулися в науці про дані, виявлено, що майже для всіх цих проблем прогнозування послідовності нейромережі ДКП, або ж LSTM, можна визначити як найбільш ефективне рішення [3].

Висновки по розділу II

Існує багато різних інструментів для роботи зі штучними нейронними мережами. При виборі інструменту слід звернути увагу на мову програмування, доступність ресурсу, функції, якими володіє система, та завдання, яке необхідно виконати.

Для роботи зі штучною нейронною мережею було обрано наступні інструменти. Як основу використано бібліотеку машинного навчання Keras, синтаксис якої написаний мовою програмування Python, ця бібліотека надає програмний інтерфейс високого рівня та підходить для роботи з нейронними мережами. Разом з Keras використано TensorFlow – відкриту програмну бібліотеку для машинного навчання, що розроблена компанією Google для побудови і тренування нейронних мереж. Код задокументовано за допомогою середовища розробки Jupyter Notebook.

Є багато алгоритмів для прогнозування часових рядів на основі попередніх даних. Для даної роботи більш всього підходить LSTM алгоритм, завдяки якому можна створити ефективний сервіс по прогнозуванню вартості акцій для інвесторів. Основною перевагою є збереження даних навчання, що дає змогу їх використання без повторного навчання моделі.

3. РОЗРОБКА МОДЕЛІ ДЛЯ ПРОГНОЗУВАННЯ ЦІНИ АКЦІЇ З ВИКОРИСТАННЯМ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ

3.1 Розробка архітектури нейромережі

Машинне навчання залежить від наявності хороших даних для навчання нейромережі. Навчальна вибірка використовується для виконання початкового навчання моделі, для ініціалізації ваг нейронної мережі.

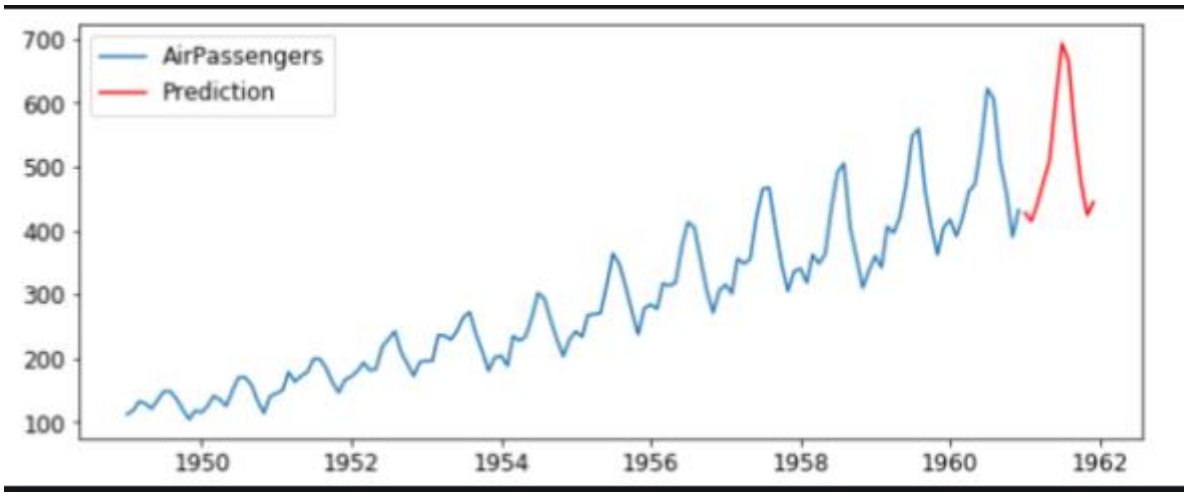

Рисунок 3.1 – Приклад прогнозу часових рядів за допомогою машинного навчання [1]

У роботі виконана оцінка даних акцій компаній. За допомогою даних ціни за проміжок часу, алгоритм визначає ціну акції на короткий проміжок часу.

Тестова вибірка використовується після навчання нейронної мережі. Вона використовується для налаштування гіперпараметрів мережі і порівняння їх змін з передбачуваною точністю моделі. З огляду на те, що навчальну вибірку можна розглядати як набір, який використано для побудови ваг нейронної мережі, це дозволяє точно налаштувати параметри або архітектуру моделі нейронної мережі. Користь цього полягає у тому, що це дозволяє повторно порівняти різні параметри з одними і тими ж даними і вагою мережі, щоб побачити, як зміни параметрів впливають на прогнозуючу здатність мережі.

Після цього, тестова вибірка використовується тільки для перевірки точності прогнозу навченої нейронної мережі на раніше небачених даних після навчання і вибору параметрів та архітектури з використанням наборів даних навчання і перевірки.

Набір даних для навчання та тестування містить інформацію про дату початку торгів акції та ціну закриття. Як приклад, візьмемо компанію Тесла, яка на біржі знаходиться з 2010 року. Для того, щоб переконатися, що дані ті, що потрібні, їх треба переглянути.

```
tesla = pd.read_csv('TSLA.csv')
```

```
tesla.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2010-06-29	19.000000	25.00	17.540001	23.889999	23.889999	18766300
1	2010-06-30	25.790001	30.42	23.299999	23.830000	23.830000	17187100
2	2010-07-01	25.000000	25.92	20.270000	21.959999	21.959999	8218800
3	2010-07-02	23.000000	23.10	18.709999	19.200001	19.200001	5139800
4	2010-07-06	20.000000	20.00	15.830000	16.110001	16.110001	6866900

Рисунок 3.2 – Таблиця з переліком даних для тренування

Найважливіші показники для навчання є дата та ціна закриття ринку акцій[5]. Тому в коді виділяємо головні рядки й перевіряємо нові дані та дивимось на часовий ряд, по якому буде відбуватися навчання.

```
tesla = tesla[['Date', 'Close']]
```

```
tesla.head()
```

	Date	Close
0	2010-06-29	23.889999
1	2010-06-30	23.830000
2	2010-07-01	21.959999
3	2010-07-02	19.200001
4	2010-07-06	16.110001

Рисунок 3.3 – Таблиця з основними даними для тренування

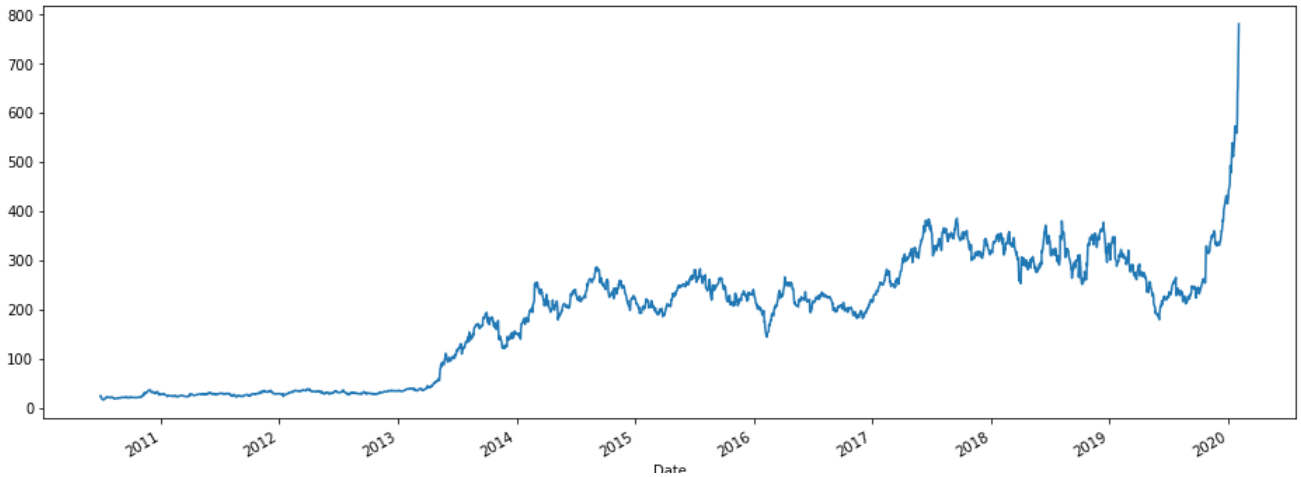

Рисунок 3.4 – Графік ціни акції компанії Tesla

Для ефективного навчання потрібно багато даних, за якими мережа зможе визначити ключові моменти, що дозволяють передбачити подальший хід ціни. Тому найкраще брати компанії, які знаходяться на фондовому ринку великий проміжок часу.

Головний момент у навчанні алгоритму є те, щоб у даних про вартість не було різких спадів, коли компанія роздрібнює свої акції для того, щоб люди, які не можуть дозволити собі її придбати, мали таку можливість. Приклад графіку з роздрібленням показаний нижче, рисунок 3.5.

Рисунок 3.5 – Приклад роздріблення акцій компанії Apple

На даному етапі потрібно визначити, на якому проміжку хочемо перевірити навчання нашої мережі. На рисунку 3.4 можна побачити, що найкращі періоди без різких коливань є періоди з 2014 по 2016 або з 2017 по 2019. Кожний період знаходиться у певному індексі, так, як інформація збережена у списку, можна відділити початок та кінець періоду. За допомогою функцій head() та tail() можна визначити початок та кінець списку. За підрахунками, початок знаходиться за індексом 884, а кінець – 1639.

```
new_tesla = tesla.loc[884:1639]
```

```
new_tesla.head()
```

	Date	Close
884	2014-01-02	150.100006
885	2014-01-03	149.559998
886	2014-01-06	147.000000
887	2014-01-07	149.360001
888	2014-01-08	151.279999

```
new_tesla.tail()
```

	Date	Close
1635	2016-12-23	213.339996
1636	2016-12-27	219.529999
1637	2016-12-28	219.740005
1638	2016-12-29	214.679993
1639	2016-12-30	213.690002

Рисунок 3.6 – Розділення списку на навчальну вибірку

3.2 Навчання нейромережі

Для побудови моделі, треба описати особливості з даних часових рядів, де аргументами є дані та кількість днів, за якими буде відбуватися обчислення. Після цього, треба переконатися, що дані для навчання не потрапили у тестову вибірку.

Фрагмент коду розділення та описання особливостей виглядає наступним чином:

```
def create_features(data, window_size):
    X, Y = [], []
    for i in range(len(data) - window_size - 1):
        window = data[i:(i + window_size), 0]
        X.append(window)
        Y.append(data[i + window_size, 0])
    return np.array(X), np.array(Y)
window_size = 20
X_train, Y_train = create_features(train, window_size)
X_test, Y_test = create_features(test, window_size)
X_train = np.reshape(X_train, (X_train.shape[0], 1, X_train.shape[1]))
X_test = np.reshape(X_test, (X_test.shape[0], 1, X_test.shape[1]))
```

Нижче будуємо модель LSTM, яка складається з одного шару LSTM з щільністю 50 нейронів та функцією активації RELU, а також шару регуляризації Dropout. Схематично архітектуру мережі представлено на рисунку 3.7.

Додаємо контрольну точку моделі, яка прагне мінімізувати втрати набору перевірки. Зберігаємо кожну модель, яка складається з менших втрат під час перевірки у порівнянні з будь-якою моделлю, яка була раніше до неї.

```
model = Sequential()

model.add(LSTM(units = 50, activation = 'relu',
               input_shape = (X_train.shape[1], window_size)))
model.add(Dropout(0.2))

model.add(Dense(1, activation = 'linear'))
model.compile(loss = 'mean_squared_error', optimizer = 'adam')
```

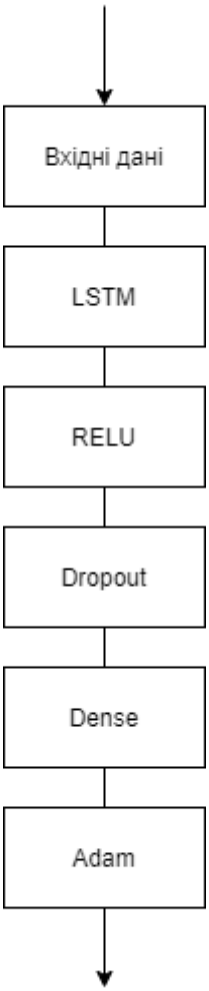

Рисунок 3.7 – Графічне зображення архітектури розробленої нейронної мережі

Далі налаштовуємо навчання нашого алгоритму, де задаємо аргументи про кількість проходів та зберігання навчених даних.

Параметри «steps_per_epoch» і «validation_steps» задаються в цьому фрагменті коду. Параметр «steps_per_epochs» або кількість ітерацій показує загальну кількість кроків, які використовуються для оголошення однієї епохи як завершеної і початку іншої епохи. Параметр «validation_steps» – це загальна кількість кроків (пакетів вибірок), які повинні бути перевірені перед зупинкою.

Фрагмент коду навчання виглядає наступним чином:

```
checkpoint = ModelCheckpoint(filepath = filepath,
                             monitor = 'val_loss',
                             verbose = 1,
                             save_best_only = True,
```

```
                        mode ='min'
                    )
history = model.fit(X_train, Y_train, epochs = 100, batch_size = 20, validation_data = (X_test, Y_test),
                    callbacks = [checkpoint],
                    verbose = 1, shuffle = False)

model.summary()
```

3.3 Аналіз показників якості моделі

Під час виконання коду прогрес видно на консолі. Після навчання моделі, можна перейти до тестування моделі. На цьому етапі порівнюємо другий набір даних. Цей набір даних ніколи не був помічений моделлю. Наприкінці навчання та тестування можна оцінити результати, яких досягла модель.

Втрати та точність – це дві різні метрики для оцінки продуктивності моделі нейронної мережі, які зазвичай використовуються на різних етапах.

Класифікаційна точність – це відношення кількості правильних прогнозів до загальної кількості вхідних вибірок.

Метрики точності використовуються для інтерпретації продуктивності алгоритму. Точність моделі зазвичай визначається параметрами моделі і розраховується у відсотках. Це міра точності прогнозу моделі з фактичними даними.

На рисунку 3.8 позначено, що реальна ціна акції позначена синім кольором, а прогнозована – оранжевим. Таким чином, можна побачити, як відпрацював алгоритм, що маючи невелику похибку був дуже близько до реальної ціни. Це означає, що модель прогнозування надає якісний прогноз.

Таблиця 3.1 – Результати навчання моделі

Результати навчання			
Навчальна похибка (Train RMSE)	**Тестова похибка (Test RMSE)**	**Реальна ціна (Actual)**	**Прогнозована (Predicted)**
7.53353	4.8405	181.41	190.03
		177.11	190.09
		178.72	187.17
		174.42	187.17
		178.38	186.33
		186.53	186.29
		196.56	190.54
		196.61	195.60
		195.32	199.25
		199.63	200.73

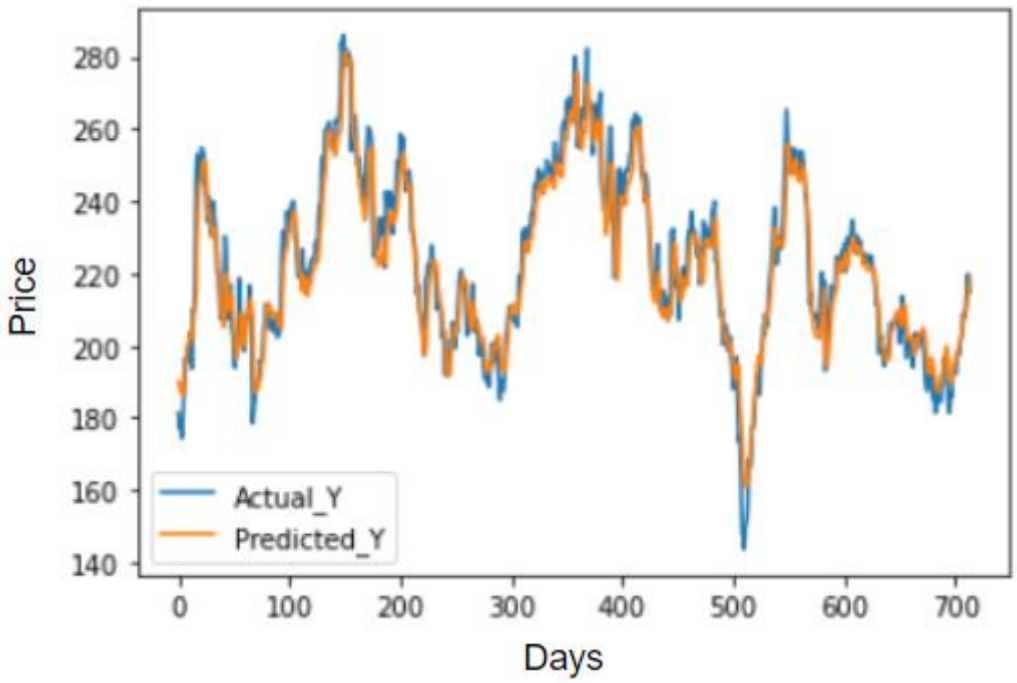

Рисунок 3.8 – Графік порівняння прогнозу моделі мережі з реальними показниками

Висновки по розділу III

У розділі розглянуто реалізацію процесу розробки та навчання нейронної мережі.

Дані для навчання були взяті з сервісу Alpha Vantage, де за допомогою тікеру можна отримати інформацію про обіг акцій за весь період існування компанії у .csv файлі.

Розроблено архітектуру нейронної мережі для передбачення ціни акції компанії, протестовано її роботу на прикладі компанії Tesla. Створено та навчено модель ДКП для вирішення завдання прогнозування подальшої ціни акції.

За результатами навчання та тестування модель має середню різницю у ціні на 4.4% меншу на навчальному наборі та 2.1% – на тестовому.

Для покращення результату можна збільшити кроки або вибірку даних, але це збільшить час навчання моделі.

4. ШЛЯХИ ПРАКТИЧНОГО ВИКОРИСТАННЯ ЗАПРОПОНОВАНОЇ МОДЕЛІ. РОЗРОБКА ТЕЛЕГРАМ-БОТА

4.1 Створення бота у телеграмі для прогнозування ціни акції

Для того, щоб використовувати розроблену модель на практиці, треба створити систему, яка буде отримувати від користувача тікер компанії, яка йому цікава, та при цьому видавати результат, отриманий з моделі. Найкращий варіант для цього – це створення бота.

Python є основою для реалізації алгоритму машинного навчання і розробки моделі, тому використаємо його для налаштування бота. Середовище застосування бота використаємо Telegram. Для отримання API для бота, у телеграмі є “батько-бот”, де можна зареєструватись на створення свого бота. Записуємо назву нашого бота та отримаємо API.

У Python є бібліотека під назвою pytelegrambotapi, за допомогою якої можна легко налаштувати та запустити бота. Все що потрібно – це API, який вже маємо.

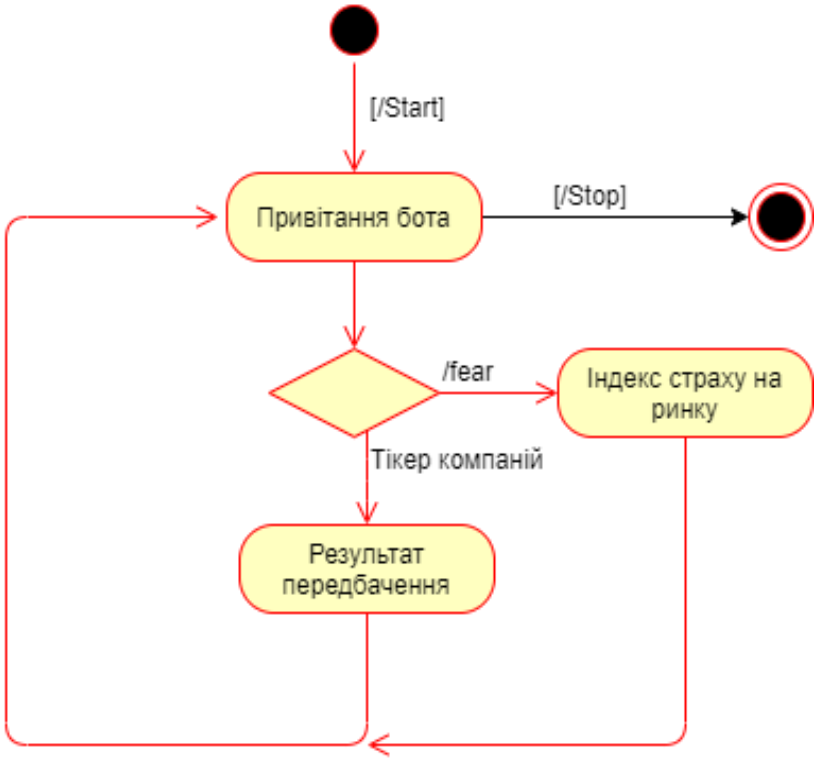

Рисунок 4.1 – Діаграма станів телеграм-бота

Весь алгоритм записуємо у функцію, яку будемо використовувати для запиту. Аргументом буде тікер акції, який хочемо дізнатися подальшу ціну. Для автоматизації отримання інформації про тікер компаній, використовуємо сервіс під назвою alphavantage, де по запиту можна отримати інформацію про ціну акції компанії у .csv файлі. Результатом функції буде передбачувана ціна, яка буде виводитись користувачу.

Розглянемо приклади коду для отримання та виведення інформації користувачу. Перша функція позначає початок роботи з телеграм-ботом, у ній передбачено виведення текстового привітання. Далі користувач має ввести наступну команду боту у вигляді текстового повідомлення. Код цієї функції має вигляд:

```
@bot.message_handler(commands=['start'])
def start_message(message):
    bot.send_message(
        message.chat.id, 'Hello, I am invest bot. I can help you with your investments', reply_markup=keyboard1)
```

Друга функція описує аналіз та обробку повідомлень від користувача, зокрема н введення тікеру. Код цієї функції має вигляд:

```
@bot.message_handler(content_types=['text'])
def send_text(message):
    if message.text:
        bot.send_message(message.chat.id, f'Checking for {message.text}. Please wait')
        price = round(float(tickerfromuser(message.text)), 2)
        bot.send_message(message.chat.id, f'I think, {message.text} will be at {price} in 5 days')
    elif message.text.lower() == 'end':
        bot.send_message(message.chat.id, 'Good bye')

bot.polling()
```

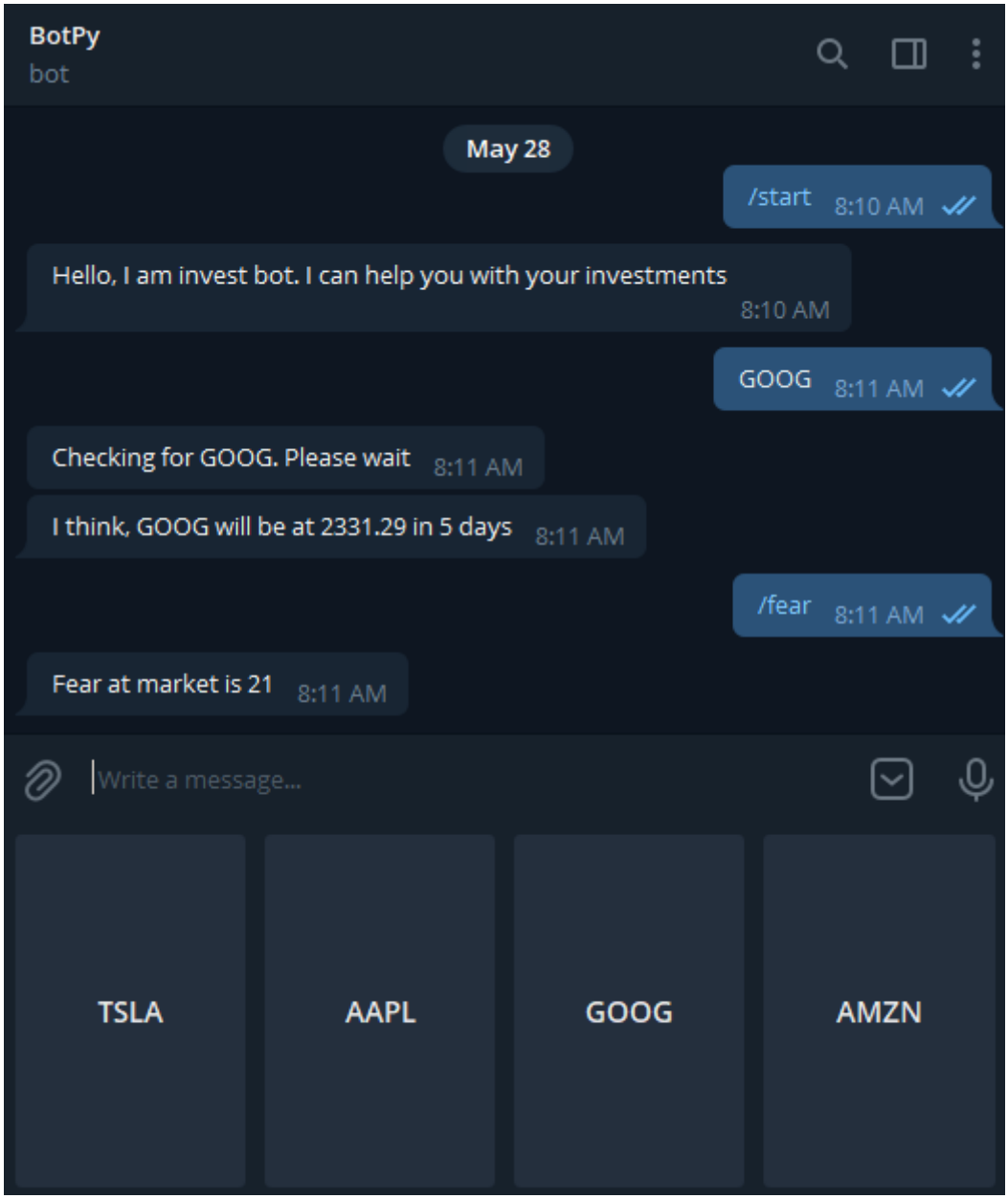

Рисунок 4.2 – Вигляд бота у телеграмі

На даному рисунку видно, що після отримання тікеру компанії, бот починає свою роботу та видає результат, отриманий від розробленої моделі. Знизу бота показані популярні тікері компаній, які після кліку автоматично записує боту повідомлення з вибраним тікером.

Нижче показана діаграма кооперації бота.

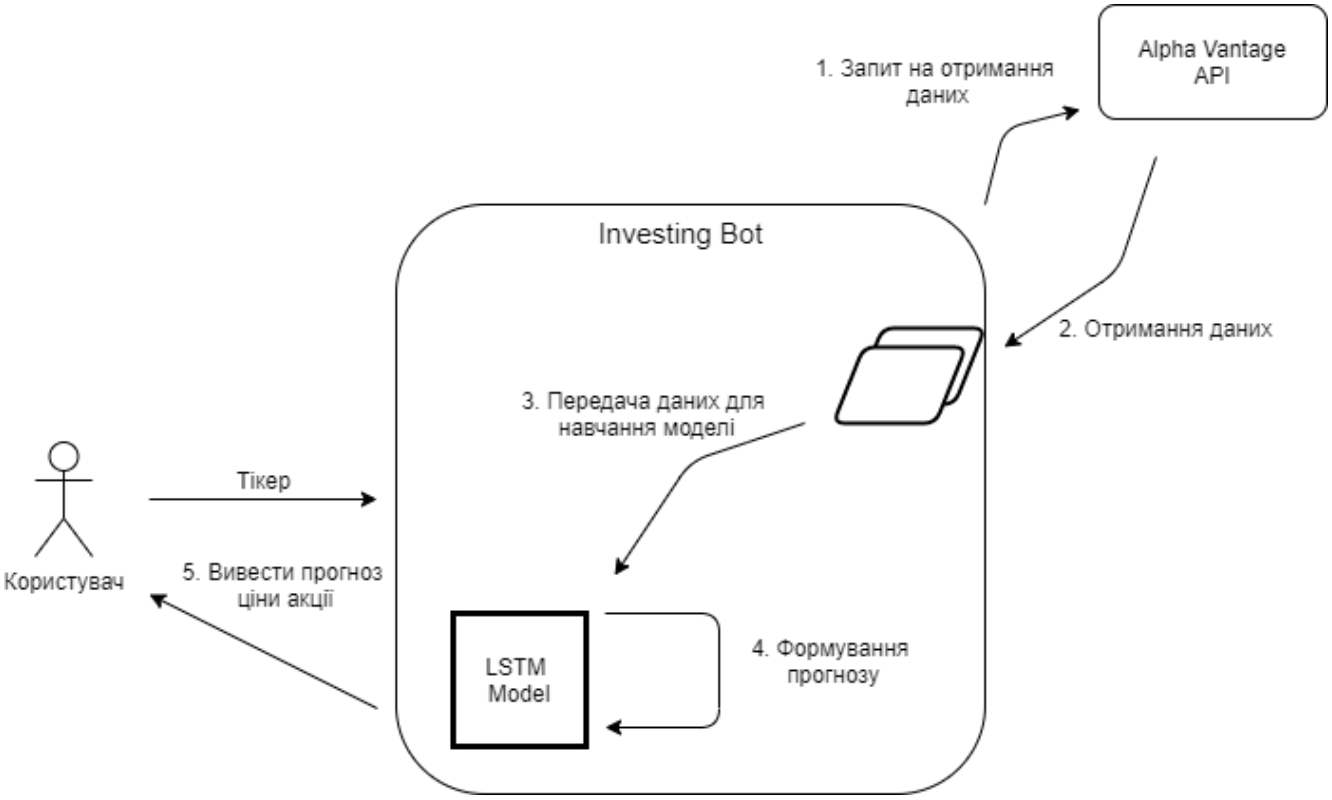

Рисунок 4.3 – Діаграма кооперації телеграм-бота

На діаграмі видно, що після того, як користувач відправив тікер, бот починає свою працю від запиту до сервісу Alpha Vantage та закінчуючи виведенням результату ціни акції користувачу.

4.2 Шляхи подальшого вдосконалення запропонованої моделі

Для того, щоб покращити результати точності моделі, необхідно експериментувати з параметрами та використовувати для кожної нової компанії більш якісний набір даних, без різких змін у графіку. Це допоможе значно покращити якість прогнозу моделі.

Створений алгоритм працює так, що прогнозує ціну акції на основі ряду особливостей, такі як дата відкриття та вартість. Одним зі шляхів вдосконалення моделі є додавання більшої кількості особливостей, які можуть як поліпшити модель, так і погіршити.

Особливості, які можна додати до моделі:

1. кількість акцій – це один з головних критеріїв, по якій можна визначити капіталізацію компанії, що дає можливість зрозуміти ціну однієї акції. Також це критерій допоможе по визначенню можливої ціни криптовалюти;
2. ціна / прибуток – критерій, по якому визначається, що ринок вважає за цю компанію. Оптимальне значення для цієї особливості є 10[4];
3. дивіденди – деякі компанії виплачують відсоток від прибутку, що дає змогу отримувати гроші, просто тримаючи акцію при собі. Через це ціна акції йде вниз. Виплачуються у кожної компанії можуть бути в один квартал, два або чотири.

Такий розподіл допоможе оптимізувати прогнозування та вивести його на інший рівень. Якщо перебільшити з особливостями, це може погіршити прогнозування вартості. Основне завдання у створенні моделі – знайти оптимальну кількість аргументів, що на даний час виходить за рамки даної роботи і може бути виконано у подальшому.

Висновки по розділу IV

У розділі 4 розглянуто напрямки практичного використання запропонованої моделі машинного навчання. У цій роботі алгоритм використаний для прогнозування або виявлення закономірностей на часовому проміжку. Головним критерієм для ефективної праці алгоритму є правильно згруповані дані, де не буде різких змін даних, що може погіршити прогнозування.

Для практичного використання моделі, краще всього підходить створення боту, який буде використовуватись як сервіс для допомоги інвесторам у вирішені питань з приводу покупки або продажу акцій.

Середовище для створення боту було вибрано Телеграм месенджер. На мові Python є бібліотека для полегшення його створення.

З моделі було створено функцію, яка отримувала за аргумент тікер акції, після чого модель починала навчання та видавала результат, який бот видає користувачеві.

Завдяки боту, алгоритм працює автоматизовано та видає результат користувачу через деякий час, після отримання запиту.

ВИСНОВКИ

Машинне навчання має великі перспективи у всіх сферах життя людини, та вже показує дивовижні можливості. Алгоритми машинного навчання можуть розв'язувати складні задачі, але вимагають великої обчислювальної потужності. Також їх можна застосовувати при виникненні проблем, які потребують вирішення в реальному часі або в прийнятні терміни. Важливою властивістю алгоритмів машинного навчання є здатність до навчання. Вони навчаються або з учителем, де мережі заздалегідь надається правильна відповідь, або без учителя, де мережа самостійно знаходить шаблони з представлених даних. Загалом, навчання алгоритму виконується за допомогою корекції синаптичних ваг нейронів мережі. У роботі розглянуто існуючі підходи до навчання нейромереж та їх характеристики.

Серед існуючої кількості архітектур нейромереж особливе місце займає довга короткострокова пам’ять. Виявлення закономірностей на часових рядах є основною сферою застосування ДКП. Багато відомих компаній використовують ці технології. Довга короткострокова пам’ять має ряд унікальних особливостей, що дає можливість для розв'язання різних типів задачі. ДКП мають велику кількість параметрів для налаштування. Вони здатні до узагальнення та вивчення ознак і особливостей на часових даних. У даній роботі запропоновано використати довгу короткострокову пам’ять для передбачення вартості акції на ринку.

Результатом роботи стала модель ДКП, призначена для передбачення вартості акції. Програмна модель алгоритму була реалізована з використанням бібліотеки Keras мовою програмування Python, а бібліотека TensorFlow використовувалася для виконання складних математичних обчислень.

У роботі отримано набір даних з сервісу Alpha Vantage для навчання і тестування алгоритму та проведено навчання моделі на цьому наборі даних. За результатами навчання та тестування модель має розбіжність у 7 доларів США на

навчальному наборі та 4 долари США на тестовому у порівнянні з оригінальною ціною.

Створена модель ДКП може бути використана для прогнозування ціни акції будь-якої компанії, яка має інформацію про динаміку ціни акції. Використання алгоритму допомагає швидко знайти та оцінити особливості, щоб прийняти рішення про покупку або продаж і автоматизувати цей процес.

Створення боту дає можливість використання моделі нейронної мережі як продукт, який може використовувати будь-який користувач, у якого є аккаунт у соціальній мережі Телеграм.

ПЕРЕЛІК ПОСИЛАНЬ

1. A Quick Example of Time-Series Prediction Using Long Short-Term Memory (LSTM) Networks [Електронний ресурс]. – Режим доступу: https://medium.com/swlh/a-quick-example-of-time-series-forecasting-using-long-short-term-memory-lstm-networks-ddc10dc1467d
2. Understanding LSTM Networks by Example using Torch [Електронний ресурс]. – Режим доступу: https://medium.com/@aadhilr92/understanding-lstm-networks-by-example-using-torch-c63dba7bbb3c
3. Using a Keras Long Short-Term Memory (LSTM) Model to Predict Stock Prices [Електронний ресурс]. – Режим доступу: https://www.kdnuggets.com/2018/11/keras-long-short-term-memory-lstm-model-predict-stock-prices.html
4. Что показывает мультипликатор P/E? [Електронний ресурс]. – Режим доступу: https://finrange.com/dictionary/p/pe
5. Stock Prices Prediction Using Machine Learning and Deep Learning Techniques (with Python codes) [Електронний ресурс]. – Режим доступу: https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/
6. Інтелектуальні інформаційні системи [Електронний ресурс]. – Режим доступу: http://libraryno.ru/iis/
7. А.М. Семенов, Н.А. Соловьев, Е.Н. Чернопрудова, А.С. Цыганков. Интеллектуальные системы: учебное пособие – Оренбургский гос. ун-т. – Оренбург: ОГУ, 2013. – 236 с.
8. Саймон Хайкин. Нейронные сети: полный курс, 2-е издание. Пер. с англ. – Издательский дом «Вильямс», 2006. – 1104 с.
9. Scikit-learn. Machine Learning in Python [Електронний ресурс]. – Режим доступу: https://scikit-learn.org/stable/
10. Manohar Swamynathan. Mastering Machine Learning with Python in Six Steps, Second Edition, 2019. – 384 с.

11. TensorFlow.js | Machine Learning for Javascript Developers [Електронний ресурс]. – Режим доступу: https://www.tensorflow.org/js/
12. Weka 3 – Data Mining with Open Source Machine Learning Software in Java [Електронний ресурс]. – Режим доступу: https://www.cs.waikato.ac.nz/ml/weka/
13. Keras: The Python Deep Learning library [Електронний ресурс]. – Режим доступу: https://keras.io/
14. Keras API reference / Data preprocessing / Image data preprocessing [Електронний ресурс]. – Режим доступу: https://keras.io/api/preprocessing/image/
15. M. Abadi, A. Agarwal et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," Mar. 2016 [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1603.04467
16. Туториал по PyTorch: от установки до готовой нейронной сети [Електронний ресурс]. – Режим доступу: https://neurohive.io/ru/tutorial/glubokoe-obuchenie-s-pytorch/
17. Shogun. Unified and efficient machine learning library [Електронний ресурс]. – Режим доступу: https://www.shogun-toolbox.org/
18. Paperspace Brings Gradient ML Platform To Hybrid And Multi-Cloud Environments [Електронний ресурс]. – Режим доступу: https://www.forbes.com/sites/janakirammsv/2019/09/30/paperspace-brings-gradient-ml-platform-to-hybrid-and-multi-cloud-environments/?sh=9cbaf72608a6

ДОДАТОК А
ВМІСТ ФАЙЛУ MAIN.PY

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import urllib.request, json
import datetime
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.callbacks import ModelCheckpoint
from keras.models import load_model
from sklearn.metrics import mean_squared_error



def tickerfromuser(userTicker):
   data_source = 'alphavantage'

   if data_source == 'alphavantage':

       api_key = 'Your api key for alpha'

       ticker = userTicker

       url_string =
"https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol
=%s&outputsize=full&apikey=%s"%(ticker,api_key)

       file_to_save = 'stock_market_data-%s.csv'%ticker
```

```
        with urllib.request.urlopen(url_string) as url:
            data = json.loads(url.read().decode())
            data = data['Time Series (Daily)']
            df =
pd.DataFrame(columns=['Date','Low','High','Close','Open'])
            for k,v in data.items():
                date = datetime.datetime.strptime(k, '%Y-%m-%d')
                data_row = [date.date(),float(v['3.
low']),float(v['2. high']),
                            float(v['4. close']),float(v['1. open'])]
                df.loc[-1,:] = data_row
                df.index = df.index + 1
        print('Data saved to : %s'%file_to_save)
        df.to_csv(file_to_save)


    ticker = pd.read_csv('stock_market_data-%s.csv'%ticker)


    ticker = ticker[['Date', 'Close']]


    ticker.Date = pd.to_datetime(ticker.Date, format ='%Y/%m/%d')


    ticker_length = len(ticker)
    new_ticker = ticker.loc[0:ticker_length//2]


    new_ticker = new_ticker.drop('Date', axis = 1)
    new_ticker = new_ticker.reset_index(drop = True)


    T = new_ticker.values
    T = T.astype('float32')
    T = np.reshape(T, (-1, 1))
```

```
    scaler = MinMaxScaler(feature_range = (0, 1))
    T = scaler.fit_transform(T)

    train_size = int(len(T) * 0.80)
    test_size = int(len(T) - train_size)
    train, test = T[0:train_size,:], T[train_size:len(T),:]



    def create_features(data, window_size):
        X, Y = [], []
        for i in range(len(data) - window_size - 1):
            window = data[i:(i + window_size), 0]
            X.append(window)
            Y.append(data[i + window_size, 0])
        return np.array(X), np.array(Y)

    window_size = 20
    X_train, Y_train = create_features(train, window_size)

    X_test, Y_test = create_features(test, window_size)

    X_train = np.reshape(X_train, (X_train.shape[0], 1,
X_train.shape[1]))

    X_test = np.reshape(X_test, (X_test.shape[0], 1,
X_test.shape[1]))

    T_shape = T.shape
    train_shape = train.shape
    test_shape = test.shape
```

```
def isLeak(T_shape, train_shape, test_shape):
    return not(T_shape[0] == (train_shape[0] + test_shape[0]))

print(isLeak(T_shape, train_shape, test_shape))

tf.random.set_seed(11)
np.random.seed(11)

model = Sequential()

model.add(LSTM(units = 50, activation = 'relu',
               input_shape = (X_train.shape[1], window_size)))
model.add(Dropout(0.2))

model.add(Dense(1, activation = 'linear'))
model.compile(loss = 'mean_squared_error', optimizer = 'adam')

filepath = 'saved_models/model_epoch_{epoch:02d}.hdf5'

checkpoint = ModelCheckpoint(filepath = filepath,
                             monitor = 'val_loss',
                             verbose = 1,
                             save_best_only = True,
                             mode ='min'
                            )

history = model.fit(X_train, Y_train, epochs = 100, batch_size =
20, validation_data = (X_test, Y_test),
                    callbacks = [checkpoint],
                    verbose = 1, shuffle = False)
```

```
    model.summary()

    best_model = load_model('saved_models/model_epoch_93.hdf5')

    train_predict = best_model.predict(X_train)

    Y_hat_train = scaler.inverse_transform(train_predict)

    test_predict = best_model.predict(X_test)

    Y_hat_test = scaler.inverse_transform(test_predict)

    Y_test = scaler.inverse_transform([Y_test])
    Y_train = scaler.inverse_transform([Y_train])
    Y_hat_train = np.reshape(Y_hat_train, newshape =
len(Y_hat_train))
    Y_hat_test = np.reshape(Y_hat_test, newshape = len(Y_hat_test))

    Y_train = np.reshape(Y_train, newshape = len(Y_hat_train))
    Y_test = np.reshape(Y_test, newshape = len(Y_hat_test))

    train_RMSE = np.sqrt(mean_squared_error(Y_train, Y_hat_train))

    test_RMSE = np.sqrt(mean_squared_error(Y_test, Y_hat_test))

    print('Train RMSE is: ')
    print(train_RMSE, '\n')
    print('Test RMSE is: ')
    print(test_RMSE)
```

```
Y = np.append(Y_train, Y_test)
Y_hat = np.append(Y_hat_train, Y_hat_test)

result_df = pd.DataFrame()

result_df['Actual_Y'] = Y
result_df['Predicted_Y'] = Y_hat

return round(result_df['Predicted_Y'][0], 3)
```

ДОДАТОК Б

ВМІСТ ФАЙЛУ BOT.PY

```
from main import tickerfromuser

import telebot

from feargreed import getfear


bot = telebot.TeleBot('API key for telegram')

keyboard1 = telebot.types.ReplyKeyboardMarkup()

keyboard1.row('TSLA', 'AAPL', 'GOOG', 'AMZN')



@bot.message_handler(commands=['start'])

def start_message(message):

    bot.send_message(

        message.chat.id, 'Hello, I am invest bot. I can help you
with your investments', reply_markup=keyboard1)



@bot.message_handler(commands=['fear'])

def start_message(message):

    bot.send_message(

        message.chat.id, f'Fear at market is {getfear()}',
reply_markup=keyboard1)



@bot.message_handler(content_types=['text'])

def send_text(message):

    if message.text:

        bot.send_message(message.chat.id, f'Checking for
{message.text}. Please wait')
```

```
        price = round(float(tickerfromuser(message.text)), 2)

        bot.send_message(message.chat.id, f'I think, {message.text}
will be at {price} in 5 days')

    elif message.text.lower() == 'end':

        bot.send_message(message.chat.id, 'Good bye')



bot.polling()
```

ДОДАТОК В

Вміст файлу feargreed.py

```
import requests
import json


def getfear():
    url = "https://api.alternative.me/fng/?limit=2"
    response = requests.request("GET", url)
    res = response.text
    res_dict = json.loads(res)
    return res_dict['data'][0]['value']
```