ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка системи прогнозування та аналізу фінансового ринку з використанням нейромереж»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Єфрем АЛОРДЕЙ*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувача вищої освіти гр. КІД-42
Єфрем АЛОРДЕЙ
Ім'я, ПРІЗВИЩЕ
Керівник: Ігор БУЧЕНКО
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 Комп'ютерної інженерії

Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
_____Ім'я, ПРІЗВИЩЕ
«_____» _____ 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Алордей Єфрем Бенедіктус
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розробка системи прогнозування та аналізу фінансового ринку з використанням нейромереж
керівник кваліфікаційної роботи Ігор БУЧЕНКО,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024р. № 36

2. Строк подання кваліфікаційної роботи «3» червня 2024 р.

3. Вихідні дані до кваліфікаційної роботи:
1) теза за дослідженням на тему: « Використання нейромереж у розпізнаванні та діагностиці навчальних здібностей здобувачів освіти »;
2) теза за дослідженням на тему: «Використання нейромереж у процесі інтелектуального (кластерного) аналізу даних».
.
Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Яка математична теорія лежить в основі нейронної системи зворотного поширення?
2. Чи можуть нейронні мережі точно прогнозувати індекс фондового ринку?
3. Чи можна використовувати нейронні мережі як практичний інструмент прогнозування?
1. Перелік ілюстративного матеріалу: *презентація*
2. Дата видачі завдання «27» лютого 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	28.02-22.03.2024	Виконано
2	Обробка матеріалу	23.03-05.04.2024	Виконано
3	Розробка теоретичної частини	06.04-10.04.2024	Виконано
4	Розробка формул моделі	11.04-30.04.2024	Виконано
5	Розробка нейронної мережі	01.05-03.05.2024	Виконано
6	Висновки за результатами аналізу	04.05-08.05.2024	Виконано
7	Розробка презентації	09.05-11.05.2024	Виконано
8	Передзахист	14.05-17.05.2024	Виконано
9	Здача в деканат	25.05-3.06.2024	Виконано

Здобувач(ка) вищої освіти _____ Єфрем АЛОРДЕЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник бакалаврської роботи _____ Ігор БУЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 45 стор., 35 рис., 30 джерел.

Мета роботи – створити модель машинного навчання, яка може передбачити завтрашню ціну індексу S&P 500, враховуючи історичні дані.

Об'єкт дослідження – розробка системи прогнозування та аналізу фінансових ринків з використанням нейромереж.

Предмет дослідження – прогнозування індексу S&P500 за допомогою нейромережі.

Короткий зміст роботи: Це дослідження вивчає та аналізує використання нейронних мереж як інструменту прогнозування.

Зокрема, здатність нейронної мережі передбачати майбутні тенденції індексів фондового ринку. Точність порівнюється з традиційним методом прогнозування, багатолінійним регресійний аналіз. Це дослідження визначає доцільність і практичність використання нейронних мереж як інструмент прогнозування для індивідуального інвестора. Це дослідження базується на роботі, виконаній Едвардом Гейтлі у своїй книзі «Нейронні мережі для фінансового прогнозування». Це дослідження описує розробку нейронної мережі, яка була створена за допомогою мови програвання Python на платформі Jupyter Notebook. Точність цієї моделі може досягти 95,3 відсотка ймовірності прогнозування зростання ринку та 92,7 відсотка ймовірність прогнозування падіння ринку S&P500.

КЛЮЧОВІ СЛОВА: S&P500, НЕЙРОН, МЕРЕЖА, НЕЙРОМЕРЕЖА, МОДЕЛЬ, АНАЛІЗ, ДАНІ.

ЗМІСТ

ВСТУП

Це дослідження вивчає та аналізує використання нейронних мереж як інструменту прогнозування.

Зокрема, здатність нейронної мережі передбачати майбутні тенденції індексів фондового ринку. Точність порівнюється з традиційним методом прогнозування, багатолінійним регресійний аналіз. Нарешті, ймовірність того, що прогноз моделі буде правильним розраховується з використанням умовних ймовірностей.

Це дослідження визначає доцільність і практичність використання нейронних мереж як інструмент прогнозування для індивідуального інвестора. Це дослідження базується на роботі, виконаній Едвардом Гейтлі у своїй книзі «Нейронні мережі для фінансового прогнозування». Це дослідження описує розробку нейронної мережі, яка була створена за допомогою мови програвання Python на платформі Jupyter Notebook.

Точність цієї моделі може досягти 95,3 відсотка ймовірності прогнозування зростання ринку та 92,7 відсотка ймовірність прогнозування падіння ринку S&P500.

РОЗДІЛ 1. НЕЙРОННА МЕРЕЖА

1.1 Біологічній нейрон

Люди робили кілька спроб імітувати біологічні системи, і однією з них є штучні нейронні мережі, натхненні біологічними нейронними мережами живих організмів. Однак вони дуже відрізняються кількома способами. Наприклад, птахи надихнули людей на створення літаків, а чотирилапі надихнули нас на створення автомобілів. Штучні аналоги, безумовно, потужніші й покращують наше життя. Персептрони, які є попередниками штучних нейронів, були створені для імітації певних частин біологічного нейрона, таких як дендрит, аксон і тіло клітини, за допомогою математичних моделей, електроніки та будь-якої обмеженої інформації, яку ми маємо про біологічні нейронні мережі. Для створення математичних моделей для штучних нейронних мереж теоретичний аналіз біологічних нейронних мереж є важливим, оскільки вони мають дуже тісний зв’язок. І це розуміння нейронних мереж мозку відкрило горизонти для розробки систем штучних нейронних мереж і адаптивних систем, призначених для навчання та адаптації до ситуацій і вхідних даних.

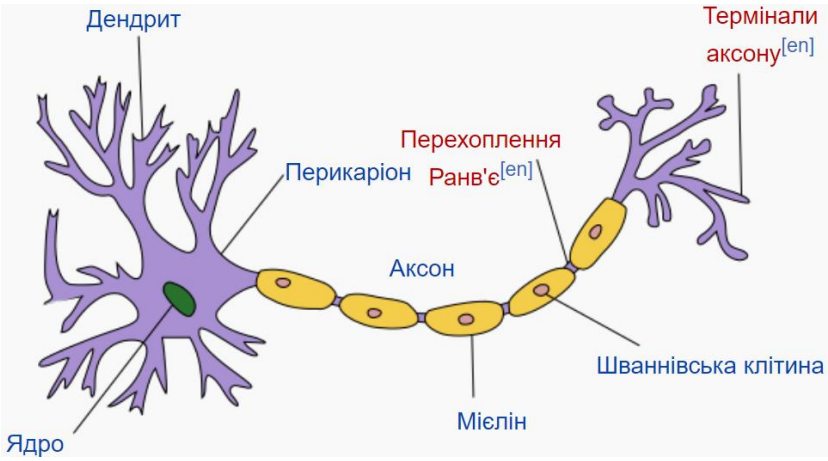


Рисунок 1.1 – Схема біологічного нейрону

Біологічна нейронна мережа — це мережа нейронів, з’єднаних між собою аксонами та дендритами. Зв’язки між нейронами здійснюються за допомогою синапсів. Аксони транспортують хімічні речовини, які спричиняють вивільнення нейромедіаторів на дендрити, де нейромедіатори потім здатні збуджувати або гальмувати сусідній нейрон. Нейронна мережа здатна вивчати і запам'ятовувати інформацію, що дозволяє їй вирішувати проблеми або приймати рішення.

У живих організмах мозок є керуючою одиницею нейронної мережі, і він має різні субодиниці, які відповідають за зір, почуття, рух і слух. Мозок пов’язаний густою мережею нервів з іншими сенсорами та акторами тіла. У мозку є приблизно 10^{11} нейронів, і вони є будівельними блоками повної центральної нервової системи живого організму.

Нейрон є основним будівельним блоком нейронних мереж. У біологічних системах нейрон — це клітина, як і будь-яка інша клітина організму, яка має код ДНК і генерується так само, як і інші клітини. Хоча він може мати різну ДНК, функція подібна в усіх організмів. Нейрон складається з трьох основних частин: тіла клітини (також називається Сома), дендритів і аксона. Дендрити схожі на волокна, розгалужені в різних напрямках і з’єднані з багатьма клітинами цього скупчення.

Дендрити отримують сигнали від навколишніх нейронів, а аксон передає сигнал до інших нейронів. На кінцевій терміналі аксона контакт з дендритом здійснюється через синапс.

Аксон — це довге волокно, яке транспортує вихідний сигнал у вигляді електричних імпульсів уздовж своєї довжини. Кожен нейрон має один аксон. Аксони передають імпульси від одного нейрона до іншого, як ефект доміно. Людський мозок складається з приблизно 86 мільярдів нейронів і понад 100 трильйонів синапсів. У штучних нейронних мережах кількість нейронів становить приблизно від 10 до 1000.

Але ми не можемо порівнювати можливості біологічних і штучних нейронних мереж лише за кількістю нейронів. Існують також інші фактори, які

необхідно враховувати. У штучних нейронних мережах багато рівнів, і вони з’єднані між собою для вирішення проблем класифікації. Біологічні нейронні мережі терплять велику кількість неоднозначності даних. Однак штучні нейронні мережі вимагають певної міри точних, структурованих і відформатованих даних, щоб допускати неоднозначність. Біологічні нейронні мережі до певного рівня відмовостійкі, і незначні збої не завжди призводять до втрати пам’яті.

1.2 Історія ідеї нейронної мережі

Теорія нейронної мережі вийшла з дослідження штучного інтелекту або дослідження у проектуванні машин з когнітивними можливостями.

Нейронна мережа — це математична модель що працює запринципом людского мозгу, вона навчается шляхом опрацювання великого набору данних не вимагаючи написання окремого коду під конкретне завданння.

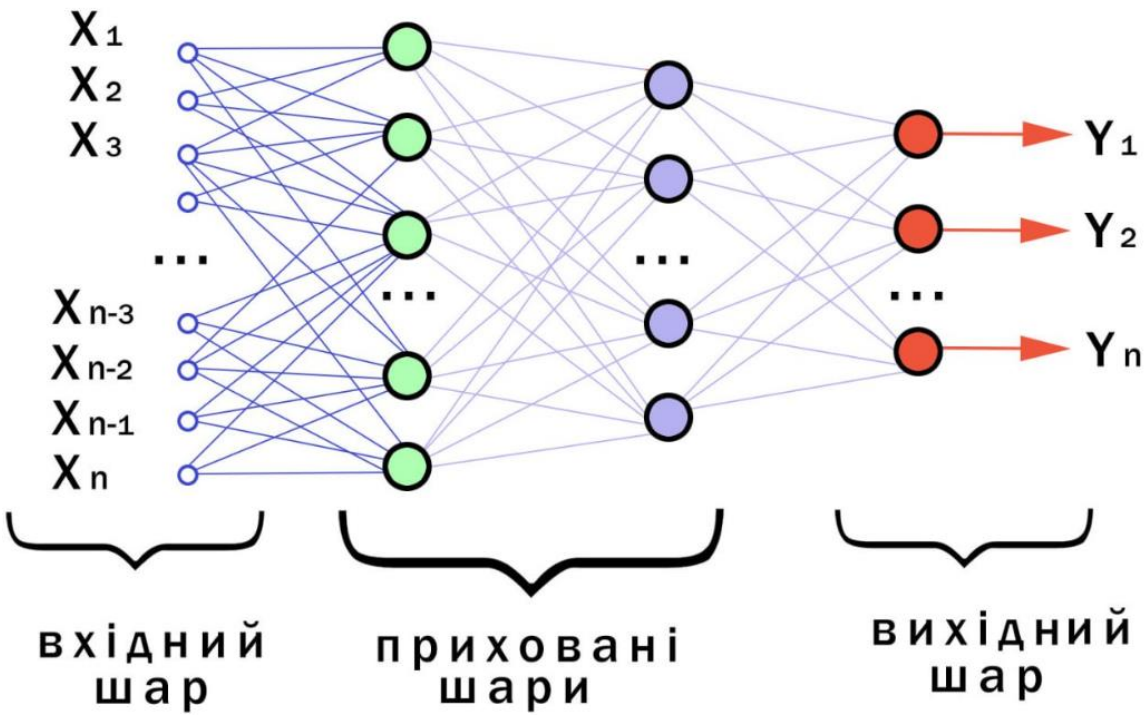


Рисунок 1.2 – Схема нейронної мережі

Френк Розенблат (1928 – 1971) був американським психологом помітний у галузі штучного інтелекту. В 1957 рік він почав щось справді велике. Він розробив перцептрон програму, на комп'ютері IBM 704 в аеронавігаційній лабораторії Cornell.

Вчені виявили, що клітини мозку (нейрони) отримують вхідні дані від наших органів чуття електричними сигналами. Тоді нейрони знову використовують електричні сигнали для зберігання інформації та для прийняття рішень на основі попереднього введення.

У Френка була ідея, що перцептрони можуть імітувати принципи мозку, здатність вчитися та приймати рішення. Таким чином перцептрони стали попередником нейромережі та штучного інтелекту.

Спочатку перцептрон був розроблений для того, щоб взяти ряд двійкових входів щоб виробляти один двійковий вихід нуль або один.

Ідея полягала в тому, щоб використовуючи різні ваги представляти важливість кожного входу, і що сума значень повинна бути більшою, ніж поріг значення перед тим, як зробити рішення, так або ні, справжній або помилковий.

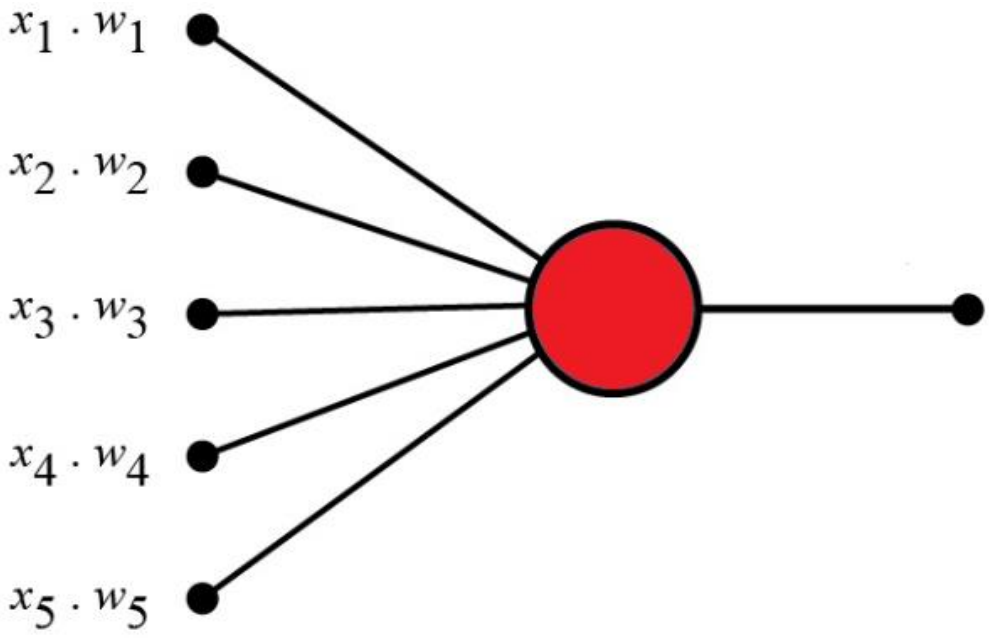


Рисунок 1.3 – Схема перцептору

Перцептрон був створений, щоб функціонувати як окрема нервова клітина, яка працює як простий підсумовувач і компаратор це якраз і означає що він отримує вхідний набір числа, множить їх на деякий коефіцієнти, які також називають вагами сумує все разом і порівнює результат з пороговим значенням, якщо результуючу значення перевищує порогове значення перцептрон посилає номер один як виведення до сусідів.

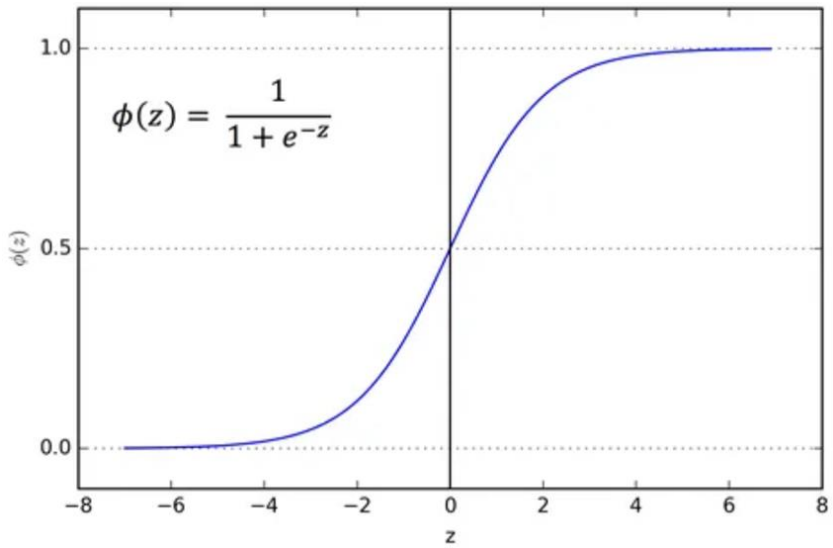


Рисунок 1.4 – Функція логістичної активації

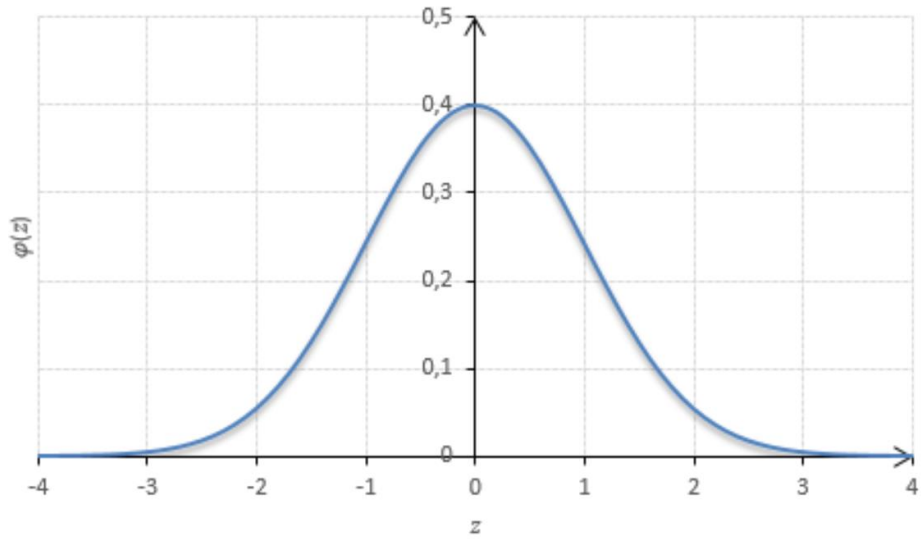


Рисунок 1.5 – Функція активації Гауса

Дві загальні функції активації, які також використовувалися мережі, перевіреної під час цього дослідження. Перша – логістична або сигмоїдна функція

f(x) =l/(l+exp(-*X*)). Друга — функція Гауса f(x) = exp (-X^2).

Остаточний вихід нейрона являє собою вихід функції активації. Великий ряд взаємопов'язаних штучних нейронів мають здатність навчатися або зберігати знання у їхніх синаптичних вагах. Здатність до навчання штучної нейронної мережі буде обговорюватиметься пізніше, однак ця властивість робить його ідеальним для спроб ідентифікувати сигнали в межах та потоку даних. Одним із прикладів такого потоку даних може бути ціна закриття S&P500.

1.3 Типи нейронних мереж та якими функціями вони працюють

Нейронні мережі можна класифікувати на різні типи, які використовуються для різних цілей. Хоча це не вичерпний список типів, наведений нижче буде репрезентативним для найпоширеніших типів нейронних мереж, які ви зустрінете для типових випадків використання.

Згорткові нейронні мережі схожі на мережі прямого зв’язку, але вони зазвичай використовуються для розпізнавання зображень, розпізнавання образів і комп’ютерного зору. Ці мережі використовують принципи лінійної алгебри, зокрема множення матриць, щоб ідентифікувати шаблони в зображенні.

Рекурентні нейронні мережі ідентифікуються за їх петлями зворотного зв’язку. Ці алгоритми навчання в основному використовуються під час використання даних часових рядів для прогнозування майбутніх результатів, наприклад прогнозування фондового ринку чи прогнозування продажів.

Алгоритм зворотного поширення прагне мінімізувати помилковий термін між входом нейронної мережі та фактичним бажаним вихідним значенням.

Обчислюється Термін обчислюється шляхом порівняння помилки чистого з бажаним результатом, а потім подається назад через мережу, що спричиняє зміну синаптичних ваг, щоб мінімізувати помилку. Процес повторюється, поки помилка не досягне мінімального значення.

$$W_{ij(t+1)} = W_{ij,t} + (\lambda)(\varepsilon W_{ij}(N_i) \qquad (1.1)$$

Мережа використовує рівняння для оновлення ваги *Wij* від заданого вузла *Ni* до поточного вузла *Nj;* де t відноситься до кількості λ разів, коли мережа була оновлена, і λ відноситься до навчального параметру. Параметр навчання, або швидкість навчання, контролює швидкість ваги яка змінюється в міру швидкості навчання. Чутливість вузла *Nj* до зміни ваги *Wij* відображається в εWij.

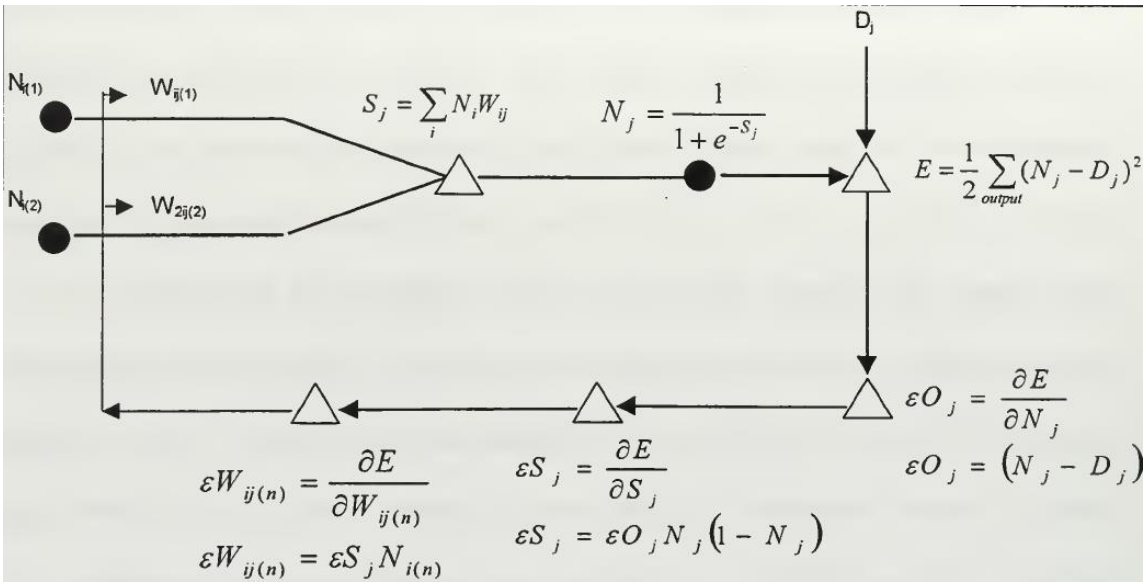


Рисунок 1.6 – Штучний нейрон з функцією зворотного навчання

Загальний вхід для вузла описується у другому рівнянні для подальшої обробки інформації.

$$S_j = \sum_i N_i W_{ij} \qquad (1.2)$$

де *Sj* — сума всіх входів у вузол, *N* — вихід попереднього вузла;
Wij — вагове з'єднання між вузлом попереднього шару.
Цей вихід потім перетворюється за допомогою функції логістичної активації, описаної вище. Загальний вихід вузла *j* представлений *Nj*.

$$N_j = \frac{1}{1+e^{-Sj}} \tag{1.3}$$

де *Nj* — Загальний вихід вузла.

$$\boldsymbol{E} = \frac{\mathbf{1}}{\mathbf{2}}\sum(N_j - D_j)^{\mathbf{2}} \tag{1.4}$$

Загальна помилка для одного проходу нейронної мережі представлена рівняння 4, де D - це бажаний вихід вихідного вузла j. Тепер, коли помилковий термін для всієї мережі обчислено, ця інформація передається через мережу для зменшення помилок. Термін помилки для кожного вихідного вузла повинні бути розраховані, по суті ми намагаємось визначити, наскільки змінюється термін помилки відносно зміни кожного вихідного вузла. Таким чином помилка мережі поширюється назад рекурсивно по всій мережі, і всі ваги налаштовані таким чином, щоб мінімізувати загальну помилку мережі.

РОЗДІЛ 2. МЕТОДОЛОГІЯ РОЗРОБКИ НЕКЙРОМЕРЕЖІ

2.1 Індекс S&P 500

Спочатку була визначена інформація, яку потрібно прогнозувати. Оскільки це дослідження розширено, отримано аналогічну модель вибрано принаймні для однієї моделі з метою порівняння. Два окремих модельних виходи.

S&P 500 — це індекс фондового ринку, який розглядається як показник того, наскільки успішно працює фондовий ринок у цілому. До нього входять близько 500 найбільших компаній США. Щоб мати право на індекс, компанії повинні відповідати певним критеріям. Серед іншого, компанії повинні:

— мати ринкову капіталізацію — яка відноситься до загальної вартості акцій компанії — щонайменше 8,2 мільярда доларів;

— перебувати в США;

— мати структуру як корпорацію та пропонувати звичайні акції;

— бути зареєстрованим на відповідній біржі США. (Інвестиційні фонди;нерухомості, відомі як REITs, мають право на включення.)

— мати позитивні звітні прибутки за останній квартал на додаток до чотирьох останніх кварталів разом.

S&P 500 відстежує ринкову капіталізацію приблизно 500 компаній, включених до індексу, вимірюючи вартість акцій цих компаній.

Ринкова капіталізація обчислюється шляхом множення кількості акцій компанії в обігу на поточну ціну акцій. Отже, якщо компанія має 2 мільйони акцій, які зараз належать акціонерам, а поточна ціна акції становить 5 доларів, тоді ринкова капіталізація компанії становить 10 мільйонів доларів. Простіше кажучи, вартість компанії становить 10 мільйонів доларів.

Значення S&P 500 розраховується на основі ринкової капіталізації кожної компанії, скоригованої з урахуванням лише кількості акцій, які торгуються публічно. Проте кожній компанії в S&P 500 надається конкретна вага, отримана шляхом ділення індивідуальної ринкової капіталізації компанії на загальну ринкову капіталізацію S&P 500.

Таким чином, компанії з більшою ринковою капіталізацією мають більшу вагу, ніж компанії з меншою ринковою капіталізацією. Оскільки ціни на акції компаній S&P 500 змінюються протягом дня, кожна зміна впливає на значення індексу, хоча компанії, що знаходяться у верхній частині списку, мають значно більший вплив, ніж ті, що знаходяться в нижній частині.

Dow Jones Industrial Average, або просто Dow, — це ще один індекс фондового ринку, який включає великі відомі компанії. Однак є кілька ключових відмінностей. Він складається лише з 30 компаній, кожна з яких вважається лідером у своїй галузі. Індекс Dow зважується на основі ціни акцій кожної компанії, а не ринкової капіталізації, що означає, що компанії з вищими цінами на акції надають більшу вагу. Індекс розраховується шляхом додавання цін акцій усіх 30 компаній, коригування ваги, а потім ділення на заздалегідь визначену константу, яка називається дільником Dow.

Індекс Dow представляє дев'ять секторів, у порівнянні з 11 в індексі S&P 500. І S&P 500, і Dow включають компанії, які вважаються найздоровішими корпораціями країни.

Першим було значення закриття індексу S&P 500 через десять днів у майбутньому, а другим був майбутній десятиденний відсоток S&P500. Набагато точніший прогноз можливий при прогнозуванні відсоткової зміни порівняно з фактичною вартістю закриття акції чи індексу. Наприклад, якщо індекс S&P500 був оцінений в 1400 доларів і модель, яка передбачила значення закриття була помилкова на десять відсотків, потенційна похибка результату становитиме 140 доларів. Якщо так самоіндекс зробив десятивідсоткову зміну ціни та модель, яка передбачила цей відсоток зміна ціни мала десятивідсоткову похибку, потенційна похибка результату становитиме один відсоток. За десятивідсоткову

зміну ціни, з 1272 доларів до 1400 доларів, ця неточність призведе до помилки приблизно в дванадцять доларів.

Перша модель нейронної мережі під назвою Close Network була обрана як перевірка роботи мережі і передбачання індексу S&P500 через десять днів майбутнє. Вхідні дані для моделі були розроблені шляхом попереднього завантаження наступних необробленого даних з Dial Data за допомогою програмного забезпечення Data Downloader від Omega Research:

- індекс S&P 500 *SPX;
- транспортний індекс Dow Jones *DWT X;
- індустріальний індекс Dow Jones *DWI X;
- індекс Dow Jones Utilities *DWU;
- товарознавче бюро *CRB;
- індекс нафти AMEX *XOI;
- індекс видобутку золота та срібла *XAU.

Наступні індекси були експортовані та збережені як текстовий файл із Wall Street Analyst.

2.2 Збір та обробка даних

Цілісність інформації надзвичайно важлива для нейронної мережі. Гарантуючи цілісність де кожен стовпець дати індексу перевірявся на цілісність дат S&P500 за допомогою простого правила в Excel. Якщо дати збігаются, ставиться нуль підряд, якщо дати не збігаються, Excel розміщує один у рядку. Ця функція скопіювала весь набір даних, а потім підсумувала. Сума нуль не містить розбіжностей щодо цілісності дати. Сума чогось більшого за нуль вказує, що існує розбіжність у даті. Сума була нульовою для всіх індексів за виняток *CRB. У *CRB було виявлено, що один торговий день був відсутній. Замість видалення індексу додано торговий день. Середнє значення закриття

день, наступний і наступний за відсутнім днем, був обраний для представлення зниклого день.

Останній стовпець у електронній таблиці мав назву «Майбутній (10-денний) S&P500 C». Це представляє майбутню ціну закриття індексу S&P500 і є фактичним значенням мережа, яка використовується для порівняння з прогнозом під час навчання. Це «майбутня» інформація було створено шляхом простого копіювання ціни закриття S&P 500 в останній стовпець електронна таблиця видаляє дані за перші 10 днів і переміщує решту даних на 10 рядків вище.

Після завершення всі стовпці дат було видалено, за винятком одного пов'язані з даними S&P 500. Файл було збережено як аркуш Excel 4, щоб він міг імпортувати в Neuroshell 2 Professional. Наступні дані було імпортовано в програмний пакет нейронної мережі як файл шаблону:

— S&P500 H;
— S&P500 L;
— S&P500 C;
— транспортний індекс Dow C;
— індекс Dow Utilities C;
— індекс нафти Amex C;
— промисловий індекс CRB CDow V;
— видобуток золота та срібла C;
— майбутнє (10 днів) S&P500 C.

Потім файл шаблону було змінено за допомогою програмного модуля Market Indicator. Наступні технічні показники були додані:

— MvAvg (30) промислового індексу Dow V;
— лаг (10) CRB C;
— відставання (10) від Amex Oil Index C;
— відставання (10) транспортного індексу Dow C.

Ці технічні індикатори відображають середній тридцятиденний обіг Dow Industrial. Обсяг індексу та десятиденне відставання різних необроблених даних. У модулі Define Inputs Neuroshell 2, кожен стовпець файлу даних було визначено як невикористаний, вхід або фактичний вихід. Дата та необроблений обсяг інформації було виключено, визначивши її як невикористану. Майбутнє (10 днів) S&P500 Close було визначено як фактичний вихід, а всі інші були визначені як входи. Мінімум і максимальні значення для кожного стовпця даних були автоматично розраховані для використання мережою. Далі з набору даних було вилучено тестовий набір або набір даних "поза зразком".

Кожен десятий набір даних був обраний для представлення тестових даних. Було 1700 навчальних рядів і 188 тестових рядків. У модулі Проектування мережі рекомендоване програмне забезпечення Ward Network.

Мережа Ward — це мережа зворотного поширення з одним входом шар, три приховані шари та один вихідний шар. Кожен шар називається плитою. Кожна плита складається від одного до шістнадцяти окремих нейронів залежно від розташування всередині мережі. Кожна плита також має різні функції активації, як описано нижче:

— плита 1 (вхід): Лінійний [-1,1], 13 нейронів;
— плита 2 (прихована): Гаусс, 16 нейронів;
— плита 3 (прихована): tanh, 16 нейронів;
— плита 4 (прихована): порівняння Гауса, 16 нейронів;
— плита 5 (вихід): логістика, 1 нейрон.

Швидкість навчання була встановлена на 0,05, а імпульс був встановлений на 0,5. В межах модулю Training Criteria, вибір шаблону встановлено у випадковий спосіб, встановлено інтервал калібрування на кожні 200 шаблонів, а критерії зупинки були встановлені для припинення навчання на 40 000 подіях мінімальна середня помилка в межах тестового набору. У рамках навчального модуля мережа була налаштована на автоматично зберігати ваги для найкращого тестового набору. Потім мережа була навчена всередині навчальний модуль. Потім навчена мережа була застосована до 188 зразків. Мережевий

вихід – це файл, який містить три стовпці: Фактичний (1), Мережа (1) і Act (l)-Net (1), що представляє фактичну вартість закриття S&P500, мережа передбачення та помилка мережі. Цей файл було відкрито, і фактичні та прогнозовані діаграми були створені в Excel.

Друга модель мережі під назвою Percent Network була розроблена для прогнозування майбутні десятиденні відсоткові зміни S&P500, використовуючи переваги можливого зниження помилка передбаченна. Мережа отримала ті самі необроблені дані, що й закрита мережа.

Однак у файлі необроблених даних було створено два нових стовпці. Перша колонка була визначена як десятиденна відсоткова зміна ціни закриття S&P500. Починаючи з 10-го дня, це було розраховано за такою формулою (комірка 1-комірка 1 1)/комірка 1*100. Другий було визначено як майбутні десятиденні процентні зміни. По суті, відсоткову десятиденну зміну стовпця було скопійовано в новий стовпець, а значення перенесено на десять днів назад.

Майбутня десятиденна відсоткова зміна була фактично бажаним виходом мережі та була використана під час навчання обчислювати похибку. Файл було імпортовано в Neuroshell і попередньо оброблено. Під час попередньої обробки, до файлу даних додано такі технічні показники:

- MvAvg (30) промислового індексу Dow V;
- відставання (10) CRBC;
- відставання (10) від Amex Oil Index C;
- відставання (10) транспортного індексу Доу C;
- LinRegChange (10,5) S&P500 C;
- LinRegChange (10,10) S&P500 C.

Технічний індикатор LinRegChange (x,n) — це зображення прогнозованої зміни між поточним значенням і значенням часу. Прогноз базується на лінії лінійної регресії, обчисленій за останні n періодів часу. Вилучення тестового набору, дизайн мережі та налаштування навчання були ідентичний Close Network. По суті, Percent Network виглядатиме так само як і необроблені дані з доданими індикаторами, як закрита мережа, але прогнозують відсоткову зміну

проти ціни закриття. Використовувалися ті самі дані вибірки. Коли відсоток Мережа було застосована до вихідних даних вибірки, коли вона працювала відносно погано порівняно з Close Network. Щоб підвищити точність відсоткової мережі, чисті входи були змінені. LinRegChange (10,5) S&P500 C було видалено, а LinRegPredict (10,10) S&P 500 10 днів відсоткових змін додано. Технічний індикатор LinRegPredict (x,n), являє собою прогнозоване значення x періодів часу в майбутнє. Прогноз базується на лінії лінійної регресії, обчисленій за останнім n періоди часу. Після навчання та застосування до вихідних даних вибірки Percent Network показали лише незначне покращення точності прогнозу.

Мережні входи були знову скориговані, щоб спробувати підвищити точність. Технічний індикатор LinRegChange (10,10) S&P500 C видалено як вхід. Після навчання та застосування до вихідних даних, відсоткова точність мережі . зменшилася. На цьому етапі було прийнято рішення збільшити обіг середнього обсягу із необробленими даними в мережі. Відсоткова мережа виконана тільки трохи краще після включення сирого обсягу.

Ретельний огляд необроблених даних, використаних у моделі Гейтлі та Close Network показав незначний внесок технологічного сектору. Технологічна складова була додана в мережу для підвищення точності. Ця мережа називалася Percent Network і була такою на основі найточнішої мережі на той момент, відсоткової мережі . Він включав доданий індикатор, значення закриття GSM технологічний індикатор Goldman Sachs Індекс для напівпровідників.

2.3 Результати прогнозування

Результати показали незначне зниження точності, але повністю 100 відсотків прогнозів були в межах п'ять відсотків від фактичної вартості закриття. Цю мережу було обрано для остаточного закриття мережі.

Прогнозований показник закриття індексу S&P500 склав розраховано з використанням передбаченої відсоткової зміни, наданої Percent Network. Це дозволило порівняння одного типу даних між обома мережами. Двоє передбачили закриття значення порівнювали графічно та статистично. На цьому нейромережа завершена частина дослідження.

Більш традиційним інструментом статистичного прогнозування є регресійний аналіз. Цей метод використовує суму найменших квадратів помилок для підгонки кривої до набору даних. Рішення було зроблено, щоб спробувати передбачити вартість закриття S&P500, використовуючи ті самі дані, що використовуються в Close Мережа. Залежна змінна була позначена як майбутній (10 днів) S&P500 Сколонка. Наступні стовпці були перераховані як незалежні змінні: S&P500 H, S&P500 L, S&P500 C, Dow Transportation Index C, Dow Utilities Index C, Amex Oil Індекс C, CRB C, Gold and Silver Mining C, GSM C, MvAvg(30) Dow Industrial Індекс V, відставання (10) від CRB C, відставання (10) від Amex Oil індекс C, відставання (10) від Dow Транспортний індекс C і Lag(lO) GSM C. Використання інструменту аналізу даних усередині Excel, для набору даних було виконано множинний лінійний регресійний аналіз.

При проведенні множинного лінійного регресійного аналізу використовуються наступні припущення, що мають виконуватися, щоб модель була правильною:

— нормальність;
— гомоскедастичність;
— незалежність від помилок;
— лінійність.

Нормальність передбачає значення *Y* (залежна змінна) повинна бути нормально розподілена для кожного значення *X* (незалежна змінна). Регресійний аналіз є досить стійкий проти відхилень від припущення нормальності. Один із способів перевірки припущення нормальності полягає в тому, щоб побудувати та дослідити графік нормальної ймовірності для залежної змінної.

Здавалося, що точки на графіку відхиляються від прямої лінії в а випадковим чином. Це свідчить про нормальність. Якби лінія спочатку піднімалася крутіше і потім збільшувався зі швидкістю зменшення, це вказувало б на перекіс лівого набору даних.

Гомоскедастичність передбачає варіацію або помилку навколо лінії регресії аналогічні для низьких і високих значень незалежної змінної. У цьому можна переконатися дослідження залишкових графіків для кожної незалежної змінної. Для кожної змінної є Здається, немає великих відмінностей у мінливості залишку для різних значень незалежна змінна. Отже, припущення про гомоскедастичність було дійсним. Автокореляція або ймовірність того, що певний тип помилки передує або слідує за нею інший тип помилки, порушує незалежність припущення помилки. Якщо помилки є корельованими, існуватиме шаблон позитивних помилок, що слідують за позитивними помилками та негативними помилки, наступні за негативними помилками.

Найпростіший спосіб виключити автокореляцію – побудувати графік залишки з часом. Діаграма залишку від часу не показала закономірності та вказувала на відсутність автокореляції. Тому здавалося припущення про незалежність від помилок дійсний. Діаграма відповідна до лінії регресії вказують на ймовірний лінійний зв’язок варіювання ступеня між залежною змінною та кожною з незалежних змінних. Це було перевірено, оскільки не було закономірності на залишкових графіках для кожної незалежної змінної.

Таким чином, припущення про лінійність здавалося вірним. Усі чотири припущення регресійного аналізу були перевірені таким чином, що лінійна регресія модель здалася підходящою. При використанні множинної регресії метою є використання лише тих змінних, які мають істотний зв'язок із залежною змінною.

Перший крок у визначенні значущого зв'язку між залежним і незалежним змінна полягала в проведенні тесту *F*. Тест *F* використовувався для визначення наявності значних зв’язків між залежною змінною та обраною незалежною змінною. Нульова гіпотеза полягала в тому, що принаймні один коефіцієнт

регресії не дорівнював нулю. . Нульова гіпотеза відхилена в а певний рівень значущості, якщо оцінене значення *F* більше критичного значення *F*. Розрахунок дисперсійного аналізу надав значення *F*; це було 1416. Для 95 відсотків впевненого інтервалу.

Отже, *F* було більше критичного значення *F* і можна зробити висновок, що принаймні одна з незалежних змінних була пов’язана із залежною змінною. Наступним кроком було перевірити окремі частини моделі множинної регресії. Якщо окремі змінні не мають істотного впливу на модель, їх слід видалити і почати обчислювати нову регресію. Для визначення незалежності особи використовувався t-тест змінна мала значний вплив на залежну змінну, беручи до уваги інші незалежні змінні. Нульова гіпотеза полягала в тому, що зв’язку не було між незалежною та залежною змінними; альтернативна гіпотеза полягала в тому, що там був зв’язок. Правило прийняття рішення полягало у відхиленні нульової гіпотези, якщо оцінене t було менше від'ємного t критичного або більше ніж t критичне. Для 95 відсотків впевненого інтервалу, критичні значення *t* (0,025,14) становили -2,1448 і 2,1448. Вісім із 14 незалежних змінних не пройшли *t*-тест і не мали значущого зв’язку із залежною змінна. Регресія була перерахована з використанням лише тих незалежних змінних, які пройшов тест t. На цьому етапі всі припущення були переоцінені та визнані все ще дійсними. Потім тест було повторено на другій моделі регресії, де критичне значення *F*(6,477) становить приблизно 2,1. Модель пройшла тест F. Тому принаймні один із пояснювальних змінних була пов'язана із залежною змінною.

Потім тест t був повторений на другій моделі регресії. Критичне значення t(0,025,6) становить 2,4469. При розгляді значення t, було визначено, що існує значний зв'язок між залежна змінна та всі незалежні змінні, за винятком золота та срібла Інтелектуальний аналіз С. Цей вхідний параметр було вилучено та обчислено третьою регресію. Знову всі регресійні припущення були переоцінені та визнані дійсними. Потім тест був повторений на третій регресійній моделі. Критичне значення *F* (5,477) становить приблизно 2,21. Модель пройшла тест *F*. Тому принаймні один із пояснюючих змінних пов'язана із залежною змінною.

Потім тест t був повторений на друга регресійна модель. Критичне значення *t* (0,025,6) становить 2,5706. При розгляді *t* значення, існував значний зв'язок між залежною змінною та всіма незалежними змінними. Останній тест на достовірність моделі множинної регресії полягає в перевірці наявності відсутність мультиколінеарності між незалежними змінними. Коли дві незалежні змінні дуже колінеарні, вони можуть викликати регресію коефіцієнти різко коливаються, якщо один або обидва включені в модель. Важко розділити вплив двох колінеарних незалежних змінних на залежну змінну.

Мірою колінеарної є фактор інфляції дисперсії. Для кожного з незалежних змінних, фактор інфляції дисперсії було розраховано. Якщо набори змінних є не корельований, фактор інфляції дисперсії дорівнюватиме 1. Для сильно взаємокорельованих змінних фактор інфляції дисперсії може перевищувати 10. Якщо фактор інфляції дисперсії більше 10 це означає, велику кореляцію між незалежними змінними. У третій регресійній моделі всі значення фактор інфляції дисперсії були менше 10. Таким чином, третя модель регресії була повністю оптимізована перевірено та обрано як остаточну модель для порівняння з двома нейронними мережами.

З трьома робочими моделями прогнозування наступним кроком був розрахунок ймовірність того, що передбачення кожної моделі буде точним. Розрахунок базувався за теоремою Байєса, або умовною ймовірністю. Теорема Байєса стверджує, що ймовірність залежить від середовища, в якому вона заснована. Умовна ймовірність визначається як дане *X*, яка ймовірність Y або P(Y\X)? Спочатку дослідник мав знайти те, що можна легко впізнати в навколишньому середовищі. Це було б утомливо спробуйте обчислити ймовірності для окремих значень закриття S&P500 або окремих відсотків змінної. Однак можна підрахувати, скільки разів ринок підвищувався або падав; це потенційно достатньо інформації для інвестора. Зростання або падіння ринку відповідає відсоткова зміна, яка є результатом відсоткової мережі. Відсоткові зміни також можуть бути розраховано на основі результатів Close Network і регресійної моделі.

Тому питання використовується для обчислення ймовірності того, що кожна модель прогнозу є точною: враховуючи певну кількість історичних щоденних підйомів ринку, коли модель прогнозує зростання ринку, яка ймовірність того, що це дійсно відбудеться?

Математично це було б сформульовано як (ForecastedRise \ HistoricalRise). Скоріше ніж використовуючи математичну форму теореми Байєса, умовна ймовірність була розрахована.

Якщо твердження було використано для визначення відсоткового зростання тоді генерує одиницю, якщо відсоткова зміна більша за нуль або нуль, якщо відсоткова зміна менша за нуль.

Зменшення цього стовпця дає загальну кількість днів із відсотковим збільшенням. Воно віднімається із загальної кількості днів у наборі даних, щоб визначити падіння ринку. Таким чином було розраховано, скільки разів ринок піднімався та падав.

Далі була розрахована точність кожної моделі. Вихід кожної моделі для використовувалися дані поза вибіркою. Метод «якщо-тоді» використовувався, щоб визначити, коли обидва прогнози моделі та фактичні дані узгоджені або неузгоджені щодо зростання чи падіння ринку. Ці дані можна ввести в теорему Байєса, щоб отримати умовну ймовірність. Проте точність моделі була скоригована з використанням історичних даних навколишнього середовища. Обчислення ймовірності точного передбачення а зростання чи падіння індексу S&P500 стало простим питанням поділу точного зростання чи падіння прогнозу загального зростання або падіння.

РОЗДІЛ 3. РОЗРОБКИ ТА НАВЧАННЯ НЕЙРОМЕРЕЖІ

3.1 Мова програмування Python

Python — це інтерпретована, інтерактивна, об’єктно-орієнтована мова програмування. Він містить модулі, винятки, динамічну типізацію, динамічні типи даних дуже високого рівня та класи. Він підтримує кілька парадигм програмування, окрім об’єктно-орієнтованого програмування, наприклад процедурне та функціональне програмування.

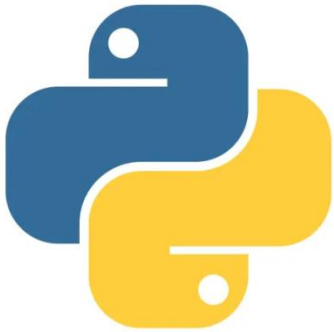


Рисунок 3.1 – Логотип мови програмування Python

Python поєднує в собі надзвичайну потужність із дуже чітким синтаксисом. Він має інтерфейси для багатьох системних викликів і бібліотек, а також для різних віконних систем і розширюється на C або C++. Його також можна використовувати як мову розширення для програм, яким потрібен програмований інтерфейс. Нарешті, Python портативний, він працює на багатьох варіантах Unix, включаючи Linux і macOS, а також на Windows.

Python був створений Гвідо ван Россумом і вперше випущений 20 лютого 1991 року. Хоча ви знаєте пітона як велику змію, назва мови програмування Python походить від старого комедійного серіалу BBC під назвою «Летючий цирк Монті Пайтона».

Однією з дивовижних особливостей Python є той факт, що це фактично робота однієї людини. Зазвичай нові мови програмування розробляють і публікують великі компанії, в яких працює багато професіоналів, і через правила авторського права дуже важко назвати когось із людей, які беруть участь у проекті. Python є винятком.

Звичайно, Гвідо ван Россум не розробляв і розвивав усі компоненти Python самостійно. Швидкість, з якою Python поширився по всьому світу, є результатом безперервної роботи тисяч (дуже часто анонімних) програмістів, тестувальників, користувачів (багато з них не IT-фахівці) та ентузіастів, але слід сказати, що саме Перша ідея (зерно, з якого пророс Python) прийшла в голову одній – Гвідо.

Python підтримується Python Software Foundation, некомерційною членською організацією та спільнотою, яка займається розробкою, вдосконаленням, розширенням і популяризацією мови Python та її середовища.

Python є усюдисущим, і люди щодня використовують численні пристрої на платформі Python, усвідомлюють вони це чи ні. На Python написані мільярди рядків коду, що означає майже необмежені можливості для повторного використання коду та навчання на добре розроблених прикладах. Більше того, існує велика й дуже активна спільнота Python, яка завжди рада допомогти. Існує також кілька факторів, які роблять Python чудовим для вивчення:

— його легко вивчити – час, необхідний для вивчення Python, менший, ніж для багатьох інших мов; це означає, що можна швидше розпочати фактичне програмування;

— він простий у використанні для написання нового програмного забезпечення – часто можна писати код швидше, використовуючи Python;

— його легко отримати, встановити та розгорнути – Python є безкоштовним, відкритим і багато платформним; не всі мови можуть цим похвалитися.

Навички програмування підготують вас до кар’єри майже в будь-якій галузі та є обов’язковими, якщо ви хочете продовжувати працювати на більш просунутих і високооплачуваних посадах у розробці програмного забезпечення та інженерії. Python — це мова програмування, яка відкриває більше дверей, ніж будь-яка інша. Маючи міцні знання Python, ви можете працювати на багатьох роботах і в багатьох галузях. І чим більше ви розумієте Python, тим більше ви можете зробити в 21 столітті. Навіть якщо це вам не потрібно для роботи, вам буде корисно знати. Мова постачається з великою стандартною бібліотекою, яка охоплює такі області, як обробка рядків (регулярні вирази, Unicode, обчислення відмінностей між файлами), Інтернет-протоколи (HTTP, FTP, SMTP, XML-RPC, POP, IMAP), розробка програмного забезпечення (модульне тестування). , журналювання, профілювання, розбір коду Python) та інтерфейси операційної системи (системні виклики, файлові системи, сокети TCP/IP).

Досі прийнято починати вивчати програмування із процедурної та статично типізованої мови, такої як Pascal, C або підмножини C++ чи Java. Особі краще буде вивчати Python як рідну мову. Python має дуже простий і послідовний синтаксис і велику стандартну бібліотеку, і, що найважливіше, використання Python на початковому вивчені програмування дозволяє людині зосередитися на важливих навичках програмування, таких як декомпозиція проблеми та проектування типів даних. За допомогою Python будь хто може швидко познайомитися з основними поняттями, такими як цикли та процедури. Ймовірно, вони навіть можуть працювати з об’єктами, визначеними користувачем.

Для особи, яка ніколи раніше не програмувала, використання статично типізованої мови виглядає неприродним. Це створює додаткову складність, яку студент повинен освоїти, і уповільнює темп курсу. Студенти намагаються навчитися мислити як комп’ютер, декомпонувати проблеми, проектувати послідовні інтерфейси та інкапсулювати дані. Хоча навчитися використовувати статично типізовану мову є важливим у довгостроковій перспективі, це не

обов’язково найкраща тема для вивчення в першому курсі програмування студентів.

Багато інших аспектів Python роблять його хорошою першою мовою. Подібно до Java, Python має велику стандартну бібліотеку, тому студентам можна призначати проекти програмування на початку курсу, які щось роблять. Завдання не обмежуються стандартним функціональним калькулятором та програмами перевірки балансу. Використовуючи стандартну бібліотеку, студенти можуть отримати задоволення від роботи над реалістичними програмами, вивчаючи основи програмування. Використання стандартної бібліотеки також навчає студентів повторному використанню коду.

3.2 Платформа розробки Jupyter Notebook

Jupyter Notebook (раніше відомий як IPython Notebook) — це інтерактивна веб-програма для створення та обміну обчислювальними документами. Спочатку проект був названий IPython, а пізніше перейменований на Jupyter у 2014 році. Це повністю відкритий продукт, і користувачі можуть використовувати всі доступні функції безкоштовно. Він підтримує понад 40 мов, включаючи Python, R і Scala.

Блокнот — це змінний файл, збережений у форматі ipynb. Jupyter Notebook має інформаційну панель блокнота, яка допомагає користувачам керувати різними блокнотами. Ядра також є частиною блокнотів Jupyter. Ядра — це процеси, які виконують інтерактивний код на певній мові програмування та повертають вихідні дані користувачеві. Ядра також відповідають на запити завершення табуляції та самоаналізу. Блокноти Jupyter можна конвертувати у відкритий стандартний формат, наприклад HTML LaTeX, PDF, Markdown і Python, за допомогою функції «Завантажити як» у веб-інтерфейсі. Процес

перетворення також можна автоматизувати за допомогою таких інструментів, як nbconvert.

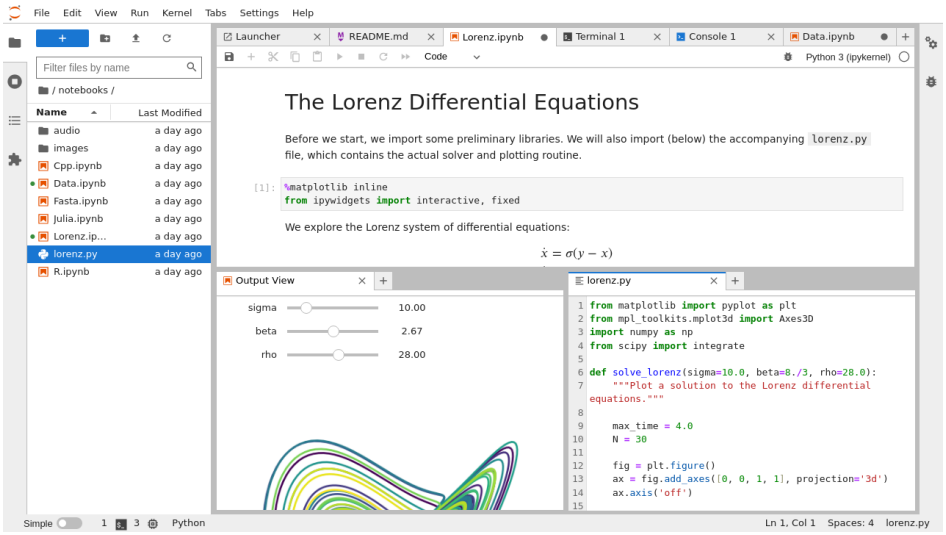


Рисунок 3.2 – Інтерфейс програми

Блокноти Jupyter використовуються для різних цілей. Він працює як інтерактивне обчислювальне середовище, в якому користувачі можуть виконувати певний фрагмент коду, спостерігати за виходом і вносити зміни в код, щоб привести його до бажаного результату або дослідити більше. Блокноти Jupyter широко використовуються для дослідження даних, оскільки це передбачає багато повторень. Він також використовується в інших процесах обробки даних, таких як експерименти з машинним навчанням і моделювання. Його також можна використовувати для документування зразків коду. Блокнот Jupyter має незалежні комірки виконуваного коду, які користувачі можуть запускати в будь-якому порядку. Документування можна виконувати, чергуючи клітинки коду та розцінки. Ноутбук Jupyter складається з двох компонентів: зовнішньої веб-сторінки та внутрішнього ядра. Інтерфейсна веб-сторінка дозволяє дослідникам даних вводити програмний код або текст у прямокутні «комірки». Браузер потім передає код до внутрішнього ядра, яке запускає код і повертає результати.

JupyterLab пов’язаний із створенням програм під егідою Project Jupyter, таких як Jupyter Notebook і Jupyter Desktop. Хоча і JupyterLab, і Notebook підтримують такі мови програмування, як Python, Julia, Scala та R, підтримка всіх цих мов попередньо встановлена в JupyterLab. Jupyter Notebook пропонує лише дуже простий інтерфейс, за допомогою якого користувачі можуть відкривати блокноти, термінали та текстові файли. JupyterLab пропонує дуже інтерактивний інтерфейс, який включає блокноти, консолі, термінали, редактори CSV, редактори уцінок, інтерактивні карти тощо. У лабораторії Jupyter ноутбуки також мають кілька вдосконалень, наприклад функцію перетягування клітинок, яка недоступна в блокноті Jupyter. Загалом JupyterLab можна розглядати як розширення Jupyter Notebook для розширення його масштабу.

3.3 Перша модель нейронної мережі

Прогнозування фондового ринку за допомогою лінійної регресії. У цій моделі мережі було використано історичні дані про ціну індексу S&P500, щоб створити прогностичну модель і прогнозувати майбутні ціни. Спочатку я імпортую необхідні бібліотеки.

In [1]:
```
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib.style as style


from datetime import datetime
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

Рисунок 3.3 – Імпорт бібліотек

Працюючи з файлом CSV, який містить індексні ціни. Кожен рядок у файлі містить щоденний запис ціни індексу S&P500 з 1950 по 2015 рік. Набір даних зберігається у sphist.csv. Стовпці набору даних:

— Date – дата запису;

— Open - ціна відкриття дня (коли починається торгівля) ;

— High - найвища торгова ціна протягом дня;

— Low - найнижча торгова ціна протягом дня;

— Volume - ціна закриття за день (коли торгівля завершена) ;

— Обсяг - кількість акцій, що продаються;

— Adj Close - щоденна ціна закриття, скоригована заднім числом, щоб включити будь-які корпоративні дії.

Використаємо вищеперелічені дані для розробки прогнозної моделі.

In [2]:
```
#Read in data
data = pd.read_csv('sphist.csv')
data.head()
```

Out[2]:

	Date	Open	High	Low	Close	Volume	Adj Close
0	2015-12-07	2090.419922	2090.419922	2066.780029	2077.070068	4.043820e+09	2077.070068
1	2015-12-04	2051.239990	2093.840088	2051.239990	2091.689941	4.214910e+09	2091.689941
2	2015-12-03	2080.709961	2085.000000	2042.349976	2049.620117	4.306490e+09	2049.620117
3	2015-12-02	2101.709961	2104.270020	2077.110107	2079.510010	3.950640e+09	2079.510010
4	2015-12-01	2082.929932	2103.370117	2082.929932	2102.629883	3.712120e+09	2102.629883

In [3]:
```
#Get Info
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16590 entries, 0 to 16589
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       16590 non-null  object
 1   Open       16590 non-null  float64
 2   High       16590 non-null  float64
 3   Low        16590 non-null  float64
 4   Close      16590 non-null  float64
 5   Volume     16590 non-null  float64
 6   Adj Close  16590 non-null  float64
dtypes: float64(6), object(1)
memory usage: 907.4+ KB
```

Рисунок 3.4 – Перша частина коду

У звичайній вправі машинного навчання кожен рядок обробляється незалежно. Дані фондового ринку є послідовними, і кожне спостереження відбувається через день після попереднього спостереження. Таким чином, не всі спостереження є незалежними, і їх не можна вважати такими.

Це означає, що треба бути дуже обережним, щоб не вводити «майбутні» знання в минулі рядки, коли я тренуюся та прогнозую. Впровадження майбутніх знань зробить нашу модель добре зображеною, коли ви навчаєтесь і тестуєте її, але зробить її невдалою в реальному світі.

In [4]:
```
# Fromat Date Column as datetime
data['Date'] = pd.to_datetime(data['Date'])
```

In [5]:
```
#Sanity Check date column
data.info()
```

Рисунок 3.5 – Друга частина коду

In [6]:
```
#Sort in oldest to newest
data.sort_values('Date', inplace=True)
data.head()
```

Out[6]:

	Date	Open	High	Low	Close	Volume	Adj Close
16589	1950-01-03	16.66	16.66	16.66	16.66	1260000.0	16.66
16588	1950-01-04	16.85	16.85	16.85	16.85	1890000.0	16.85
16587	1950-01-05	16.93	16.93	16.93	16.93	2550000.0	16.93
16586	1950-01-06	16.98	16.98	16.98	16.98	2010000.0	16.98
16585	1950-01-09	17.08	17.08	17.08	17.08	2520000.0	17.08

Рисунок 3.6 – Оновлення впроваджених даних

```
In [7]:
#Plot Closing Value
plt.figure(figsize=(40,15))
plt.scatter(data['Date'], data['Close'], c='b')
plt.title('S&P Closing Value', fontsize=60)
plt.xlabel('Date', fontsize=40)
plt.ylabel('Price', fontsize=40)

Out[7]: Text(0, 0.5, 'Price')
```

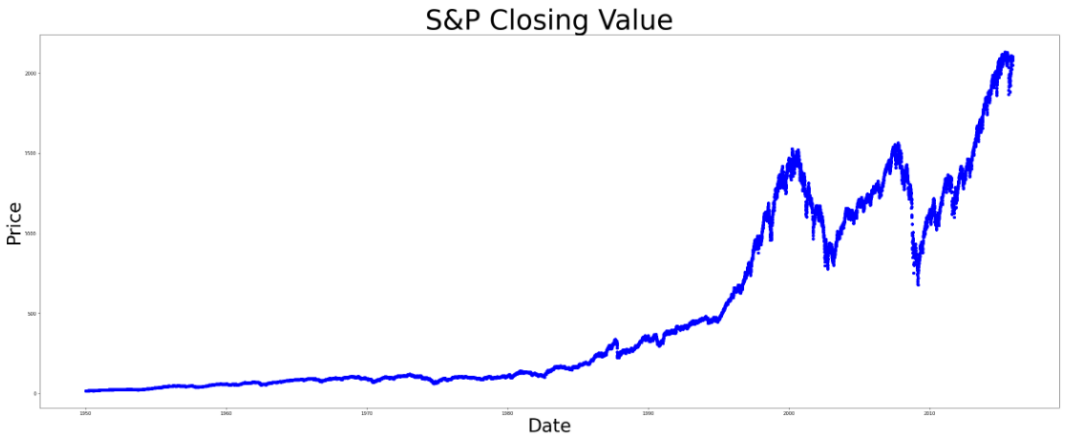


Рисунок 3.7 – Графічне зображення оброблених даних

Характер часових рядів даних означає, що модель може генерувати показники, щоб зробити нашу модель більш точною. Це об’єднає інформацію з кількох попередніх рядків в один і зробить прогнози набагато точнішими. Я створю такі функції:

— 5_day_avg - середня ціна за останні 5 днів;

— 30_day_avg - середня ціна за останні 30 днів;

— year_avg - середня ціна за останні 365 днів;

— avg_ratio - співвідношення між середньою ціною за останні 5 днів і середньою ціною за останні 365 днів;

— 5_day_std - стандартне відхилення ціни за останні 5 днів;

— year_std - стандартне відхилення ціни за останні 365 днів;

— std_ratio - співвідношення між стандартним відхиленням за останні 5 днів і стандартним відхиленням за останні 365 днів.

Середнє значення та стандартне відхилення використовуватимуть ціну поточного дня. Використовуючи метод "he shift", щоб переіндексувати отриманий ряд, щоб перемістити всі значення "вперед" на один день. Наприклад, середнє змінне, розраховане для 01.03.1950, потрібно буде призначити 01.04.1950 і так далі.

In [8]:
```
# The rolling mean and standard deviation will use the current day's price.
# I use .he shift method to reindex the resulting series to shift all the values "forward" one day.
# For example, the rolling mean calculated for 1950-01-03 will need to be assigned to 1950-01-04, and so on.
data['5_day_avg'] = data['Close'].rolling(window=5).mean().shift(1)
data['30_day_avg']= data['Close'].rolling(window=5).mean().shift(1)
data['year_avg'] = data['Close'].rolling(window=365).mean().shift(1)
data['avg_ratio'] = data['5_day_avg']/data['year_avg']

data['5_day_std'] = data['Close'].rolling(window=5).std().shift(1)
data['year_std'] = data['Close'].rolling(window=365).mean().shift(1)
data['std_ratio'] = data['5_day_std']/data['year_std']
```

In [9]:
```
data.head(20)
```

Рисунок 3.8 – Код для розрахунку

Out[9]:

	Date	Open	High	Low	Close	Volume	Adj Close	5_day_avg	30_day_avg	year_avg	avg_ratio
16589	1950-01-03	16.660000	16.660000	16.660000	16.660000	1260000.0	16.660000	NaN	NaN	NaN	NaN
16588	1950-01-04	16.850000	16.850000	16.850000	16.850000	1890000.0	16.850000	NaN	NaN	NaN	NaN
16587	1950-01-05	16.930000	16.930000	16.930000	16.930000	2550000.0	16.930000	NaN	NaN	NaN	NaN
16586	1950-01-06	16.980000	16.980000	16.980000	16.980000	2010000.0	16.980000	NaN	NaN	NaN	NaN
16585	1950-01-09	17.080000	17.080000	17.080000	17.080000	2520000.0	17.080000	NaN	NaN	NaN	NaN
16584	1950-01-10	17.030001	17.030001	17.030001	17.030001	2160000.0	17.030001	16.900	16.900	NaN	NaN

Рисунок 3.9 – Вихідні значення розрахунків

16583	1950-01-11	17.090000	17.090000	17.090000	17.090000	2630000.0	17.090000	16.974	16.974	NaN	NaN
16582	1950-01-12	16.760000	16.760000	16.760000	16.760000	2970000.0	16.760000	17.022	17.022	NaN	NaN
16581	1950-01-13	16.670000	16.670000	16.670000	16.670000	3330000.0	16.670000	16.988	16.988	NaN	NaN
16580	1950-01-16	16.719999	16.719999	16.719999	16.719999	1460000.0	16.719999	16.926	16.926	NaN	NaN
16579	1950-01-17	16.860001	16.860001	16.860001	16.860001	1790000.0	16.860001	16.854	16.854	NaN	NaN

Рисунок 3.10 – Вихідні значення розрахунків

Деякі індикатори використовують історичні дані за 365 днів, а набір даних починається з 01.03.1950. Таким чином, перші 365 рядків не мають достатньо історичних даних для обчислення всіх показників. Ми видаляємо ці рядки, перш ніж розділити дані на набір для навчання та тестування. Ми

тренуємо модель з даними за 1950 по 2012 рік і намагаємося робити прогнози з 2013 по 2015 рік.

```
In [10]: data = data.dropna(axis=0)
         data.isnull().sum()

Out[10]: Date          0
         Open          0
         High          0
         Low           0
         Close         0
         Volume        0
```

Рисунок 3.7 – Вхідні значення розрахунків

```
Adj Close     0
5_day_avg     0
30_day_avg    0
year_avg      0
avg_ratio     0
5_day_std     0
year_std      0
std_ratio     0
dtype: int64

In [11]: train = data[data['Date']< datetime(year=2013, month=1, day=1)]
         train
```

Рисунок 3.11 – Вхідні значення розрахунків

Out[11]:

	Date	Open	High	Low	Close	Volume	Adj Close	5_day_avg	30_day_avg	year_avg
16224	1951-06-19	22.020000	22.020000	22.020000	22.020000	1.100000e+06	22.020000	21.800000	21.800000	19.447726
16223	1951-06-20	21.910000	21.910000	21.910000	21.910000	1.120000e+06	21.910000	21.900000	21.900000	19.462411
16222	1951-06-21	21.780001	21.780001	21.780001	21.780001	1.100000e+06	21.780001	21.972000	21.972000	19.476274
16221	1951-06-22	21.549999	21.549999	21.549999	21.549999	1.340000e+06	21.549999	21.960000	21.960000	19.489562
16220	1951-06-25	21.290001	21.290001	21.290001	21.290001	2.440000e+06	21.290001	21.862000	21.862000	19.502082

Рисунок 3.12 – Вихідні значення розрахунків

743	2012-12-24	1430.150024	1430.150024	1424.660034	1426.660034	1.248960e+09	1426.660034	1437.360010	1437.360010	1326.114028
742	2012-12-26	1426.660034	1429.420044	1416.430054	1419.829956	2.285030e+09	1419.829956	1436.620019	1436.620019	1326.412494
741	2012-12-27	1419.829956	1422.800049	1401.800049	1418.099976	2.830180e+09	1418.099976	1431.228003	1431.228003	1326.716494
740	2012-12-28	1418.099976	1418.099976	1401.579956	1402.430054	2.426680e+09	1402.430054	1427.685986	1427.685986	1326.995836
739	2012-12-31	1402.430054	1426.739990	1398.109985	1426.189941	3.204330e+09	1426.189941	1419.434009	1419.434009	1327.261562

Рисунок 3.13 – Вихідні значення розрахунків

```
In [12]: test = data[data["Date"] > datetime(year=2013, month=1, day=1)]
         test
```

Out[12]:

	Date	Open	High	Low	Close	Volume	Adj Close	5_day_avg	30_day_avg	year_avg
738	2013-01-02	1426.189941	1462.430054	1426.189941	1462.420044	4.202600e+09	1462.420044	1418.641992	1418.641992	1327.534055
737	2013-01-03	1462.420044	1465.469971	1455.530029	1459.369995	3.829730e+09	1459.369995	1425.793994	1425.793994	1327.908247
736	2013-01-04	1459.369995	1467.939941	1458.989990	1466.469971	3.424290e+09	1466.469971	1433.702002	1433.702002	1328.224877
735	2013-01-07	1466.469971	1466.469971	1456.619995	1461.890015	3.304970e+09	1461.890015	1443.376001	1443.376001	1328.557617
734	2013-01-08	1461.890015	1461.890015	1451.640015	1457.150024	3.601600e+09	1457.150024	1455.267993	1455.267993	1328.898603

Рисунок 3.14 – Вихідні значення розрахунків

4	2015-12-01	2082.929932	2103.370117	2082.929932	2102.629883	3.712120e+09	2102.629883	2087.024023	2087.024023	2035.531178
3	2015-12-02	2101.709961	2104.270020	2077.110107	2079.510010	3.950640e+09	2079.510010	2090.231982	2090.231982	2035.914082
2	2015-12-03	2080.709961	2085.000000	2042.349976	2049.620117	4.306490e+09	2049.620117	2088.306006	2088.306006	2036.234356
1	2015-12-04	2051.239990	2093.840088	2051.239990	2091.689941	4.214910e+09	2091.689941	2080.456006	2080.456006	2036.507343
0	2015-12-07	2090.419922	2090.419922	2066.780029	2077.070068	4.043820e+09	2077.070068	2080.771973	2080.771973	2036.869425

Рисунок 3.15 – Вихідні значення розрахунків

Тепер розробимо модель і зробимо прогнози. Використовуючи середню абсолютну похибку як метрику похибки, оскільки вона покаже, наскільки «наближені» прогнози до ціни в інтуїтивно зрозумілих термінах.

```
In [13]: data.columns

Out[13]: Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close',
               '5_day_avg', '30_day_avg', 'year_avg', 'avg_ratio', '5_day_std',
               'year_std', 'std_ratio'],
              dtype='object')

In [14]: features = ['5_day_avg', '30_day_avg', 'year_avg', 'avg_ratio', '5_day_std',
                     'year_std', 'std_ratio']
         y_train = train['Close']
         y_test = test['Close']

         model = LinearRegression()
         model.fit(train[features], y_train)
         predictions = model.predict(test[features])
         mae = mean_absolute_error(test['Close'], predictions)
         mae
```

Рисунок 3.16 – Код моделі

```
In [15]: plt.figure(figsize=(40,15))
         plt.scatter(test['Date'], test['Close'], c='b')
         plt.scatter(test['Date'], predictions, c='r')
         plt.title('Stock Predictions', fontsize=60)
         plt.xlabel('Date', fontsize=40)
         plt.ylabel('Stock Price', fontsize=40)
```

Рисунок 3.17 – Код моделі

```
Out[15]: Text(0, 0.5, 'Stock Price')
```

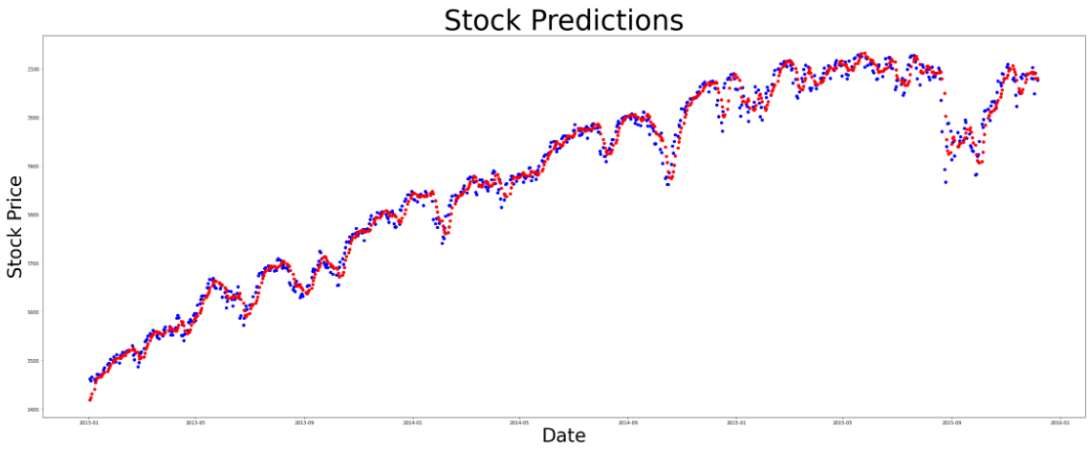


Рисунок 3.18 – Графічне зображення моделі

Модель відхиляється на 16 балів, що непогано для першого проходу. Але, безумовно, є місце для вдосконалення як індикатора, так і структури.

3.4 Друга модель нейронної мережі

Повторювана нейронна мережа (RNN) — це тип штучної нейронної мережі, яка використовує послідовні дані або дані часових рядів. Ці алгоритми глибокого навчання зазвичай використовуються для порядкових або часових проблем, таких як переклад мови, обробка природної мови (nlp), розпізнавання мовлення та підписи до зображень; вони включені в такі популярні програми, як Siri, голосовий пошук і Google Translate. Подібно до нейронних мереж прямого зв'язку та згорткових нейронних мереж (CNN), рекурентні нейронні мережі використовують впроваджені дані для навчання. Вони відрізняються своєю пам'яттю, оскільки вони беруть інформацію з попередніх вхідних даних, щоб впливати на поточні вхідні та вихідні дані. Хоча традиційні глибокі нейронні мережі припускають, що входи та виходи незалежні один від одного, вихід рекурентних нейронних мереж залежить від попередніх елементів у послідовності. Хоча майбутні події також були б корисними для визначення результату даної послідовності, односпрямовані рекурентні нейронні мережі не можуть врахувати ці події у своїх прогнозах.

Давайте візьмемо таку ідіому, як «почуття під погодою», яка зазвичай використовується, коли хтось хворий, щоб допомогти нам у поясненні RNN. Щоб ідіома мала сенс, вона має бути виражена в такому конкретному порядку. Як наслідок, рекурентним мережам потрібно враховувати позицію кожного слова в ідіомі, і вони використовують цю інформацію, щоб передбачити наступне слово в послідовності.

Іншою відмінною характеристикою рекурентних мереж є те, що вони спільно використовують параметри на кожному рівні мережі. У той час як мережі прямого зв'язку мають різні ваги для кожного вузла, рекурентні нейронні мережі мають однаковий параметр ваги на кожному рівні мережі. Тим

не менш, ці ваги все ще коригуються за допомогою процесів зворотного поширення та градієнтного спуску, щоб полегшити навчання з підкріпленням.

Повторювані нейронні мережі використовують алгоритм зворотного поширення в часі (BPTT) для визначення градієнтів, який дещо відрізняється від традиційного зворотного поширення, оскільки він специфічний для даних послідовності. Принципи BPTT такі ж, як і традиційного зворотного поширення, коли модель навчається, обчислюючи помилки від вихідного до вхідного рівня. Ці розрахунки дозволяють нам правильно налаштувати та підігнати параметри моделі. BPTT відрізняється від традиційного підходу тим, що BPTT підсумовує помилки на кожному кроці часу, тоді як упереджених мережах не потрібно підсумовувати помилки, оскільки вони не поділяють параметри на кожному рівні. Через цей процес RNN, як правило, стикаються з двома проблемами, відомими як вибухові градієнти та зникнення градієнтів. Ці проблеми визначаються розміром градієнта, який є нахилом функції втрат уздовж кривої помилок. Коли градієнт занадто малий, він продовжує зменшуватися, оновлюючи вагові параметри, поки вони не стануть незначними, тобто. 0. Коли це відбувається, алгоритм більше не навчається. Вибухові градієнти виникають, коли градієнт занадто великий, створюючи нестабільну модель. У цьому випадку ваги моделі виростуть занадто великими, і в кінцевому підсумку вони будуть представлені як NaN. Одним із рішень цих проблем є зменшення кількості прихованих шарів у нейронній мережі, усунення частини складності в моделі RNN. Тепер на основі цієї мережі ми збудуємо свою другу модель для прогнозування фінансового ринку.

In [1]:

```
# Recurrent Neural Network

# Part 1 - Data Preprocessing

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Рисунок 3.19 – Імпорт бібліотек

```
# Importing the training set
dataset_train = pd.read_csv('https://raw.githubusercontent.com/AMoazeni/Machine-Learning-Stock-Market-Prediction/master/Data
# '.values' need the 2nd Column Opening Price as a Numpy array (not vector)
# '1:2' is used because the upper bound is ignored
training_set = dataset_train.iloc[:, 1:2].values

# Feature Scaling
# Use Normalization (versus Standardization) for RNNs with Sigmoid Activation Functions
# 'MinMaxScalar' is a Normalization Library
from sklearn.preprocessing import MinMaxScaler
# 'feature_range = (0,1)' makes sure that training data is scaled to have values between 0 and 1
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_set)
```

Рисунок 3.20 – Перша частина коду моделі

```
# Creating a data structure with 60 timesteps (Look back 60 days) and 1 output
# This tells the RNN what to remember (Number of timesteps) when predicting the next Stock Price
# The wrong number of timesteps can lead to Overfitting or bogus results
# 'X_train' Input with 60 previous days' stock prices
X_train = []
# 'y_train' Output with next day's stock price
y_train = []
for i in range(60, 1258):
    X_train.append(training_set_scaled[i-60:i, 0])
    y_train.append(training_set_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)

# Reshaping (add more dimensions)
# This lets you add more indicators that may potentially have corelation with Stock Prices
# Keras RNNs expects an input shape (Batch Size, Timesteps, input_dim)
# '.shape[0]' is the number of Rows (Batch Size)
# '.shape[1]' is the number of Columns (timesteps)
# 'input_dim' is the number of factors that may affect stock prices
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))

# Show the dataset we're working with
display(dataset_train)
```

Рисунок 3.21 – Друга частина коду моделі

	Date	Open	High	Low	Close	Volume
0	1/3/2012	325.25	332.83	324.97	663.59	7,380,500
1	1/4/2012	331.27	333.87	329.08	666.45	5,749,400
2	1/5/2012	329.83	330.75	326.89	657.21	6,590,300
3	1/6/2012	328.34	328.77	323.68	648.24	5,405,900
4	1/9/2012	322.04	322.29	309.46	620.76	11,688,800
5	1/10/2012	313.70	315.72	307.30	621.43	8,824,000
6	1/11/2012	310.59	313.52	309.40	624.25	4,817,800

Рисунок 3.22 – Вихідні значення розрахунків

7	1/12/2012	314.43	315.26	312.08	627.92	3,764,400
8	1/13/2012	311.96	312.30	309.37	623.28	4,631,800
9	1/17/2012	314.81	314.81	311.67	626.86	3,832,800
10	1/18/2012	312.14	315.82	309.90	631.18	5,544,000
11	1/19/2012	319.30	319.30	314.55	637.82	12,657,800
12	1/20/2012	294.16	294.40	289.76	584.39	21,231,800
13	1/23/2012	291.91	293.23	290.49	583.92	6,851,300
14	1/24/2012	292.07	292.74	287.92	579.34	6,134,400
15	1/25/2012	287.68	288.27	282.13	567.93	10,012,700
16	1/26/2012	284.92	286.17	281.22	566.54	6,476,500
17	1/27/2012	284.32	289.08	283.60	578.39	7,262,000
18	1/30/2012	287.95	288.92	285.63	576.11	4,678,400
19	1/31/2012	290.41	290.91	286.50	578.52	4,300,700
20	2/1/2012	291.38	291.66	288.49	579.24	4,658,700
21	2/2/2012	291.34	292.11	289.95	583.51	4,847,400
22	2/3/2012	294.23	297.42	292.93	594.7	6,360,700
23	2/6/2012	296.39	304.27	295.90	607.42	7,386,700
24	2/7/2012	302.44	303.56	300.75	605.11	4,199,700
25	2/8/2012	303.18	304.53	301.24	608.18	3,686,400

Рисунок 3.23 – Вихідні значення розрахунків

26	2/9/2012	304.87	306.10	303.36	609.79	4,546,300
27	2/10/2012	302.81	302.93	300.87	604.25	4,667,700
28	2/13/2012	304.11	305.77	303.87	610.52	3,646,100
29	2/14/2012	304.63	304.86	301.25	608.09	3,620,900

Рисунок 3.24 – Вихідні значення розрахунків

1228	11/17/2016	766.92	772.70	764.23	771.23	1,304,000
1229	11/18/2016	771.37	775.00	760.00	760.54	1,547,100
1230	11/21/2016	762.61	769.70	760.60	769.2	1,330,600
1231	11/22/2016	772.63	776.96	767.00	768.27	1,593,100
1232	11/23/2016	767.73	768.28	755.25	760.99	1,478,400
1233	11/25/2016	764.26	765.00	760.52	761.68	587,400
1234	11/28/2016	760.00	779.53	759.80	768.24	2,188,200
1235	11/29/2016	771.53	778.50	768.24	770.84	1,616,600
1236	11/30/2016	770.07	772.99	754.83	758.04	2,392,900
1237	12/1/2016	757.44	759.85	737.03	747.92	3,017,900
1238	12/2/2016	744.59	754.00	743.10	750.5	1,452,500
1239	12/5/2016	757.71	763.90	752.90	762.52	1,394,200
1240	12/6/2016	764.73	768.83	757.34	759.11	1,690,700
1241	12/7/2016	761.00	771.36	755.80	771.19	1,761,000
1242	12/8/2016	772.48	778.18	767.23	776.42	1,488,100
1243	12/9/2016	780.00	789.43	779.02	789.29	1,821,900
1244	12/12/2016	785.04	791.25	784.35	789.27	2,104,100
1245	12/13/2016	793.90	804.38	793.34	796.1	2,145,200
1246	12/14/2016	797.40	804.00	794.01	797.07	1,704,200

Рисунок 3.25 – Вихідні значення розрахунків

1247	12/15/2016	797.34	803.00	792.92	797.85	1,626,500
1248	12/16/2016	800.40	800.86	790.29	790.8	2,443,800
1249	12/19/2016	790.22	797.66	786.27	794.2	1,232,100

Рисунок 3.26 – Вихідні значення розрахунків

1250	12/20/2016	796.76	798.65	793.27	796.42	951,000
1251	12/21/2016	795.84	796.68	787.10	794.56	1,211,300
1252	12/22/2016	792.36	793.32	788.58	791.26	972,200
1253	12/23/2016	790.90	792.74	787.28	789.91	623,400
1254	12/27/2016	790.68	797.86	787.66	791.55	789,100
1255	12/28/2016	793.70	794.23	783.20	785.05	1,153,800
1256	12/29/2016	783.33	785.93	778.92	782.79	744,300
1257	12/30/2016	782.75	782.78	770.41	771.82	1,770,000

Рисунок 3.27 – Вихідні значення розрахунків

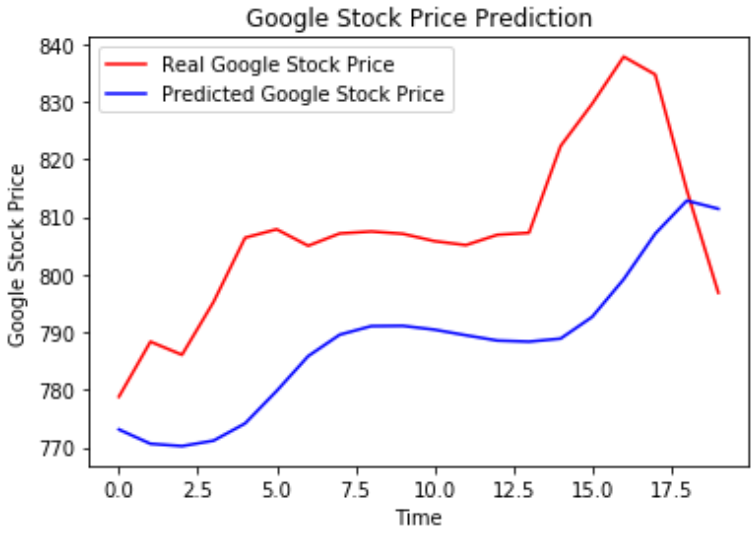


Рисунок 3.28 – Графічне зображення моделі

Із графічного зображення ми бачимо що повторювана нейронна мережа не настільки точна як модель із лінійною регресією.

ВИСНОВКИ

Було зроблено висновок, що нейромережі справді мають можливість прогнозувати фінансові ринки та, якщо їх належним чином навчити,окремий інвестор може отримати вигоду від використання цього інструменту прогнозування. Також ми можемо заначити, що модель із лінійною регресією найбільш ефективніша у прогнозуванні.

При використанні множинної лінійної регресії керуюча регресія припущення повинна відповідати дійсності. Лінійне припушення саме по собі може не витримати багато випадків. Нейронні мережі можуть моделювати як лінійні, так і криволінійні системи. При використанні множинного лінійного регресійного аналізу інвестор повинен мати глибоке розуміння статистики для забезпечення лише необхідної незалежної зміної данних. Це вірно лише в обмеженій мірі з нейронними мережами і може бути виправлено методом систематичного видалення. Для моделей, які вивчаються в цьому дослідженні, нейронні мережі мають значно більшу точнічть, ніж множинний лінійний регресійний аналіз.

Однак все ще є труднощі з визначенням якісних необроблених даних та попередньою обробкою цих даних, тому слід надавати системі якомога більше даних для аналізу і повторювати цей процес, доки не буде розроблена хороша модель. Індивідуального інвестора слід попередити, що точність моделі важко досягти і може зайняти місяці або роки розслідування**.**

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gately, E. (1996). Neural networks for financial forecasting. London: Taylor and Francis.

2. Haykin, S. (1999). Neural networks: A comprehensive foundation. Upper Saddle River, NJ: Prentice Hall.

3. Diebold, F. X. (2001). Financial forecasting with neural networks: What have we learned? [Working Paper]. University of Pennsylvania, Department of Economics.

4. Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. Sebastopol, CA: O'Reilly Media.

5. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. Cambridge, MA: MIT Press.

6. Zhang, G. P., et al. (1990). Forecasting stock indices using neural networks. In Proceedings of the International Joint Conference on Neural Networks (pp. 856-861). IEEE.

7. LeBaron, B. (1999). Neural networks and exchange rates. In Handbook of international finance (pp. 617-642). Elsevier.

8. Prokhorov, D. P. (2018). Sistema prognozuvannya ta analizu finansovykh rinkiv z vykorystannyam neyromerezh [Master's thesis]. Kyiv Polytechnic Institute.

9. Beale, M. H., & Jackson, J. (1990). Neural networks: A tutorial. IEEE Transactions on Systems, Man, and Cybernetics, 20(3), 463-476.

10. Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507. 11. Jackson, J., Burian, J., & Chau, L. P. (1990). A tutorial on neural networks for control and signal processing. IEEE Transactions on Neural Networks, 1(3), 388-408.

11. Karatapanis, D. I. (1996). Neural networks and fuzzy logic systems. Boston, MA: Kluwer Academic Publishers.

12. Khashei, M., & Bijari, M. (2019). A novel hybrid ANN-ARIMA approach for short-term load forecasting. IEEE Transactions on Power Systems, 34(4), 3138-3149.

13. Lin, T., Zhou, R., & Chen, Q. (2017). Stock price forecasting using machine learning and deep learning techniques. In International conference on data mining and knowledge discovery (pp. 202-213). Springer.

14. McMillan, S. (2000). Neural networks in finance: A survey. Journal of Financial Management, 49(1), 74-127.

15. Medeiros, K. A., & Terzi, G. C. A. (2011). A hybrid neural network and genetic algorithm approach for stock market forecasting. Neurocomputing, 74(1-3), 1-17.

16. Narendra, K., & Parthasarathy, K. (1997). Learning automata and neural networks. IEEE Transactions on Neural Networks, 8(5), 1137-1150.

17. Panait, L., & Durovic, I. (2012). Application of neural networks for trading on the stock exchange. Expert Systems with Applications, 39(1), 1-6.

18. Werbos, P. J. (1974). Beyond regression: New computational approaches to discrimination and classification. PhD thesis, Harvard University.

19. Application of neural networks for trading on the stock exchange. (2012). Expert Systems with Applications, 39(1), 1-6.

20. A hybrid ANN-ARIMA approach for short-term load forecasting. (2019). IEEE Transactions on Power Systems, 34(4), 3138-3149.

21. Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing – Journal of Cloud Computing. SpringerOpen. URL:

https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00453-3 (дата звернення: 15.04.2024).

22. Load Balancing Algorithms – GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/load-balancing-algorithms/ (дата звернення: 15.04.2024).

23. COMPARATIVE STUDY ON LOAD BALANCING TECHNIQUES IN DISTRIBUTED SYSTEMS. ResearchGate. URL: https://www.researchgate.net/figure/Comparison-of-Basic-Static-Load-Balancing-Algorithms_tbl1_266560116 (дата звернення: 15.04.2024).

24. Load Balancing Algorithms. EnjoyAlgorithms. URL: https://www.enjoyalgorithms.com/blog/types-of-load-balancing-algorithms (дата звернення: 15.04.2024).

25. Glushach R. Kubernetes Networking: Load Balancing Techniques and Algorithms. Medium. URL: https://romanglushach.medium.com/kubernetes-networking-load-balancing-techniques-and-algorithms-5da85c5c7253 (дата звернення: 15.04.2024).

26. Experimental Setup for Investigating the Efficient Load Balancing Algorithms on Virtual Cloud. Sensors. 2020. Т. 20, № 24. С. 20–24.

27. Enhanced Load Balanced Min-min Algorithm for Static Meta Task Scheduling in Cloud Computing. Procedia Computer Science. 2015. Т. 57, № 1. С. 545–553.

28. Benefits of Software-Based Hybrid Load Balancing | NGINX. NGINX. URL: https://www.nginx.com/resources/glossary/hybrid-load-balancing/ (дата звернення: 15.04.2024).

29. Combining Multiple Load Balancing Strategies – FasterCapital. FasterCapital. URL: https://fastercapital.com/keyword/combining-multiple-load-balancing-strategies.html (дата звернення: 15.04.2024).

30. Source IP Hash load balancing – Glossary – Kemp. Load Balancer For Always-On Application Experience – Kemp. URL: https://kemptechnologies.com/es/resources/glossary/source-ip-hash-load-balancing (дата звернення: 15.04.2024).

Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут Інформаційних технологій
Кафедра Комп'ютерної інженерії

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

на тему: «Розробка системи прогнозування та аналізу фінансових ринків з використанням нейромереж.»

Виконав студент групи КІД-42
Алордей Єфрем Бендіктус

Керівник кваліфікаційної роботи:
Бученко Ігор Анатолійович,
старший викладач кафедри КІ

Київ – 2024

Мета роботи – створити модель машинного навчання, яка може передбачити завтрашню ціну індексу S&P 500, враховуючи історичні дані

Об'єкт дослідження – розробка системи прогнозування та аналізу фінансових ринків з використанням нейромереж

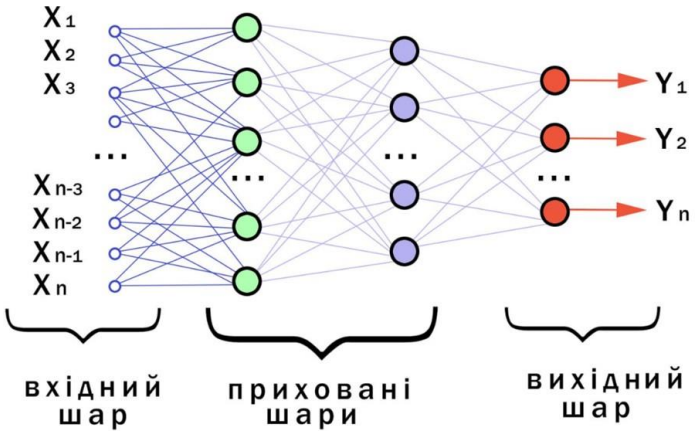
Предмет – прогнозування індексу S&P500 за допомогою нейромережі

Актуальність обраної теми

Це дослідження вивчає та аналізує використання нейронних мереж як інструменту прогнозування, завдяки чому можна буде дізнатися ціна на той чи інший продукт буде у майбутньому.

Також, з'являється все більше товарів та послуг які потребують швидкої та ефективної віртуалізації информації про те коли и як швидко зміниця ціна на той чи інший продукт або послугу. Віртуалізація системи прогнозування та аналізу фінансових ринків з використанням нейромереж може забеспечіти швидке розуміння сетуації на фінансовому ринкут.

Нейромережа – це математична модель щопрацює запринципом людского мозгу, вона навчается шляхом опрацювання великого набору данних не вимагаючи написання окремого коду під конкретне завданння.

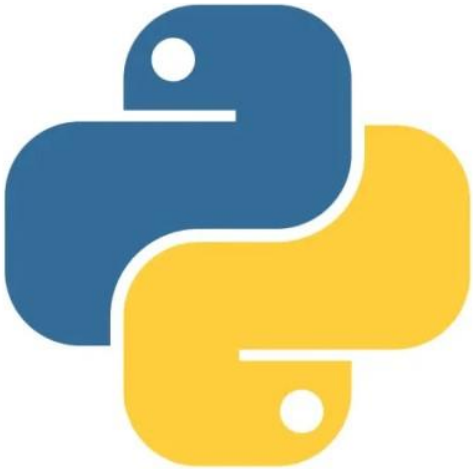


Індекс S&P 500 — це скорочене найменування американського індексу Standard & Poor's 500, який включає найбільші по капіталізації компанії США.

Ранг	Назва	Вид діяльності	Логотип
1	Apple	Технології	
2	Microsoft	Software / Technologie	Microsoft
3	ExxonMobil	Нафта і газ	ExxonMobil
4	Amazon.com	Einzelhandel	amazon
5	Johnson & Johnson	Pharmazie und Konsumgüter	Johnson & Johnson
6	Meta Platforms Inc. (A)	ЗМІ	Meta
7	Berkshire Hathaway (B)	Beteiligungsgesellschaft	Berkshire Hathaway Inc.
8	General Electric	електрика	
9	AT&T	телекомунікації	AT&T
10	JPMorgan Chase	банк	JPMorgan Chase & Co.

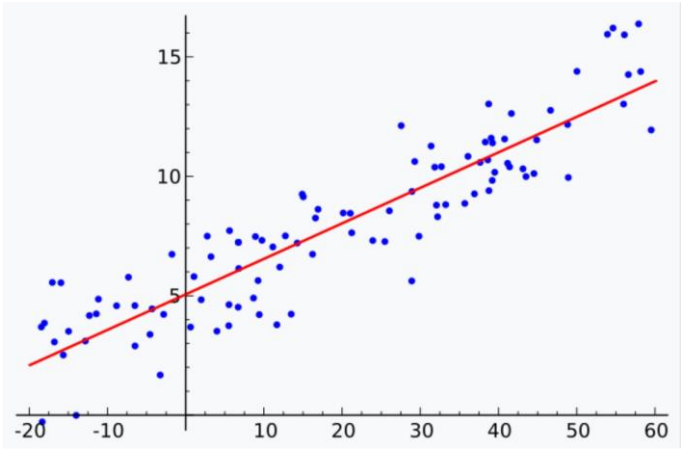
5

Python — це інтерпретована, інтерактивна, об’єктно -орієнтована мова програмування. В она містить модулі, винятки, динамічну типізацію, динамічні типи даних дуже високого рівня та класи.



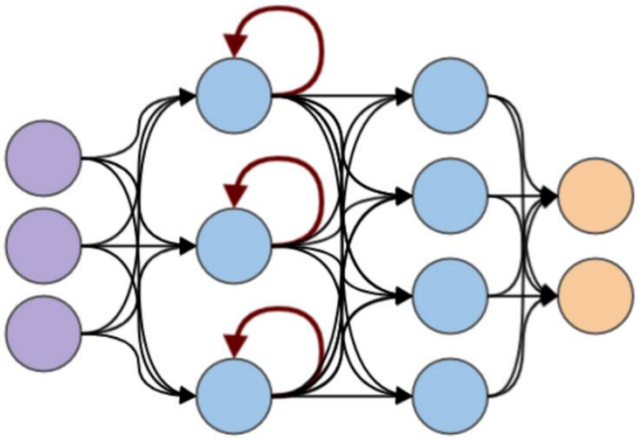
6

Лінійна регресія — це метод моделювання залежності між скалярною змінною *y* та векторною (у загальному випадку) змінною *X*. У разі, якщо змінна *X* також є скаляром, регресію називають простою.



7

Рекурентні нейронні мережі — це клас штучних нейронних мереж, у якому з'єднання між вузлами утворюють орієнтовний граф у часі.



8

Підключення до індексу S&P 500

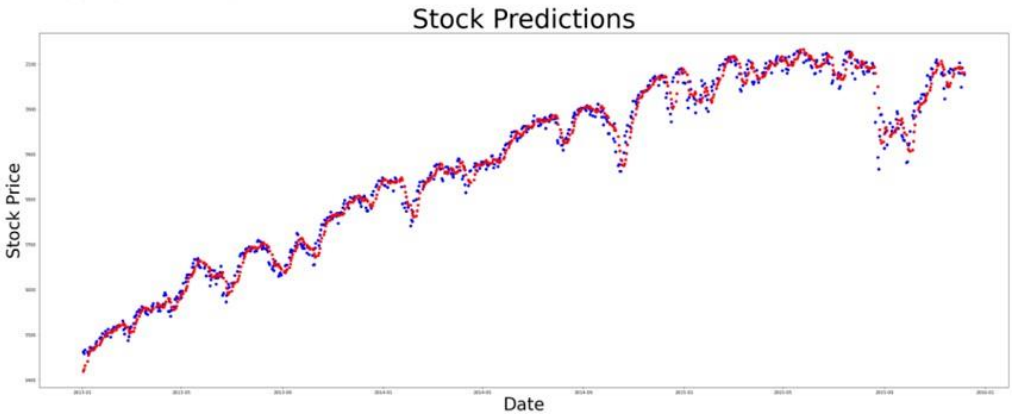
In [5]:
```
if os.path.exists("sp500.csv"):
    sp500 = pd.read_csv("sp500.csv", index_col=0)
else:
    sp500 = yf.Ticker("^GSPC")
    sp500 = sp500.history(period="max")
    sp500.to_csv("sp500.csv")
```

In [6]:
```
sp500.index = pd.to_datetime(sp500.index)
```

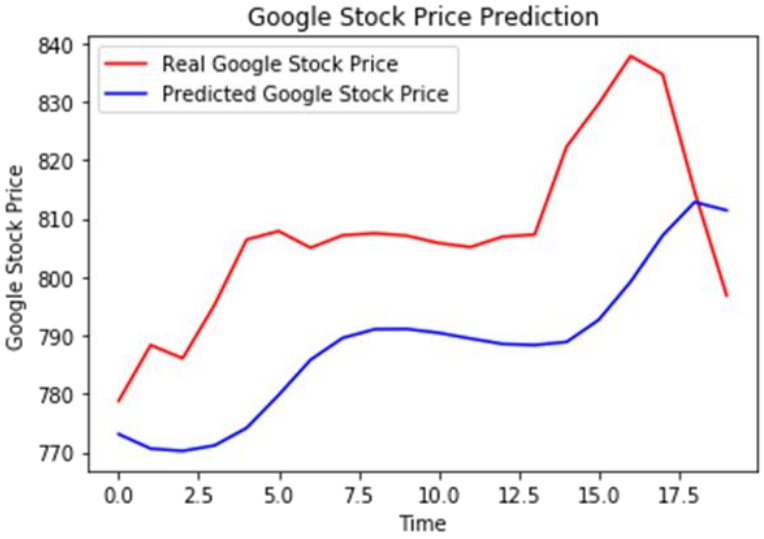
In [7]:
```
sp500
```


Графік вихідних даних першої моделі

Out[15]: Text(0, 0.5, 'Stock Price')



Графік вихідних даних другої моделі



ВИСНОВКИ

Було зроблено висновок, що нейромережі справді мають можливість прогнозувати фінансові ринки та, якщо їх належним чином навчити,окремий інвестор може отримати вигоду від використання цього інструменту прогнозування. Також ми можемо заначити, що модель із лінійною регресією найбільш ефективніша у прогнозуванні.

При використанні множинної лінійної регресії керуюча регресія припущення повинна відповідати дійсності. Лінійне припушення саме по собі може не витримати багато випадків. Нейронні мережі можуть моделювати як лінійні, так і криволінійні системи. При використанні множинного лінійного регресійного аналізу інвестор повинен мати глибоке розуміння статистики для забезпечення лише необхідної незалежної зміної данних.

ПРОДОВЖЕННЯ ВИСНОВКІВ

Це вірно лише в обмеженій мірі з нейронними мережами і може бути виправлено методом систематичного видалення. Для моделей, які вивчаються в цьому дослідженні, нейронні мережі мають значно більшу точнічть, ніж множинний лінійний регресійний аналіз.

Однак все ще є труднощі з визначенням якісних необроблених даних та попередньою обробкою цих даних, тому слід надавати системі якомога більше даних для аналізу і повторювати цей процес, доки не буде розроблена хороша модель. Індивідуального інвестора слід попередити, що точність моделі важко досягти і може зайняти місяці або роки розслідування.