МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ
Фізико-математичний факультет
Кафедра інформатики та прикладної математики

«Допущено до захисту»
Завідувач кафедри
_____Соловйов В.М.
«____»_____2020 р.

Реєстраційний № _____
«____»_____2020 р.

АНАЛІЗ СТІЙКОСТІ КОМПАНІЙ ВІДНОВЛЮВАЛЬНОЇ ЕНЕРГЕТИКИ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

Магістерська робота студента
групи Ім-15
ступінь вищої освіти «магістр»
спеціальності 014 Середня освіта (Інформатика)
Якимова Юрія Володимировича

Керівник: проф., д. ф.-м. н. Соловйов Володимир Миколайович

Оцінка:
Національна шкала _____
Шкала ECTS ____ Кількість балів ____

Голова ЕК _____ _____
Члени ЕК _____ _____
_____ _____
_____ _____

Кривий Ріг — 2020

ЗМІСТ

ВСТУП

В останні роки все частіше і частіше говорять про нейронні мережі. Вони представляють собою один з напрямків наукових досліджень у сфері створення штучного інтелекту. В основі цього напрямку лежить метод імітування нервової системи людини. У тому числі, здатність системи виправляти помилки і самонавчатися. Все це повинно дозволити зімітувати хоча б примітивну роботу людського мозку [28].

Існує декілька способів використання штучних нейронних мереж, у тому ж числі для класифікації інформації та прогнозування результатів. У ході того, як мережі оброблюють та навчаються на даних, вони можуть класифікувати даний набір даних у заздалегідь заданий клас. Нейронна мережа може бути навчена прогнозувати виходи, результати, котрі очікуються від вказаного входу [1].

Google використовує 30-ти шарову нейронну мережу, для аналізу та пошуку зображення, а також для формування списку рекомендаційних відео у YouTube. Facebook використовує штучні нейронні мережі для свого алгоритму DeepFace, котрий може розпізнавати фотографії людей з точністю 97%. Skype використовує штучні нейронні мережі аби виконувати переклад тексту і мови в режимі реального часу.

В останні роки нейронні мережі почали використовувати задля аналізу медичних знімків та прогнозування захворювань. При цьому якість і точність аналізу постійно вимагає знаходження нових методів і технологій для уточнення і підвищення їх ефективності.

В умовах сучасної економічної ситуації та різкого збільшення темпів розвитку науки й техніки, задля отримання прибутків на ринках, все більш актуальними постають питання планування і прийняття рішень на основі прогнозування. У зв’язку з цим, задача прогнозування часових рядів є актуальною, оскільки в умовах ринкової економіки у підприємства виникає

потреба дослідження даних про стан ринків в минулому з метою оцінки майбутніх умов та результатів роботи.

У зв’язку з цим **актуальними** є подальші теоретичні та практичні розробки у галузі використання сучасних нейронних мереж для аналізу та прогнозування часових рядів.

Об’єкт дослідження є фінансово-економічний сектор відновлювальної енергетики.

Предметом дослідження – методи і моделі прогнозування та аналізу часових рядів засобами машинного навчання.

Метою магістерської роботи є розробка та навчання нейронної мережі аналізу часових рядів з використанням мови програмування Python.

Для досягнення мети необхідно розв’язати такі **задачі:**

- проаналізувати способи прогнозування часових рядів;
- проаналізувати роботу нейронної мережі;
- порівняти наявні програмні засоби;
- висунути функціональні вимоги до власного програмного засобу;
- відповідно до висунутих вимог, навчити нейронну мережу прогнозувати часові ряди;

Практичне значення роботи полягає в тому, що досліджені методи та отримані результати можуть бути використана для аналізу фінансових часових рядів. Результати роботи доповідались на міжнародній науковій конференції SCI та опубліковані у виданні НВЦ SCI-CONF.com.ua.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Область використання штучних нейронних мереж з кожним роком все більш розширюється, та на сьогоднішній день вони успішно застосовуються в таких сферах як:

- Машинне навчання, яке представляє собою різновид штучного інтелекту, в основі якого лежить навчання штучного інтелекту на прикладі мільйонів однакових завдань. На приклад на основі мільйонів пошукових запитів, які ми кожен день вводимо в пошуковій системі Google, алгоритми компанії навчаються відображати нам найбільш слушну інформацію, аби ми могли відшукати саме те, що нам необхідно.
- В робототехніці нейронні мережі використовуються у розробленні алгоритмів поводження роботів.
- Архітектори комп’ютерів користуються штучними нейронними мережами для розв’язання проблеми обчислень.
- З допомогою нейронних мереж математики можуть вирішувати задачі різного рівня складності.

В цілому для вирішення кожного з завдань використовують різні види нейронних мереж, серед яких можна виокремити:

- згорткові нейронні мережі;
- рекурентні нейроні мережі;
- нейронна мережа Хопфілда;

Так само, як і ми навчаємося на досвіді нашого життя, нейронні мережі потребують даних для дослідження [29]. У більшості випадків, чим більше даних опрацює мережа, тим точніше вона буде працювати. Це схоже на те, як

коли людина аби навчитись чомусь, виконує необхідну задачу знов і знов, практикується. З часом людина виконує необхідні дії більш ефективно та з меншою кількістю помилок.

Коли дослідники чи вчені збираються навчати нейронну мережу, вони зазвичай ділять данні на три набори. Перший набір – навчальний, котрий допомагає мережі встановити розбіжність ваг між її вузлами. Після цього вони налаштовують її, використовуючи набір для перевірки. І нарешті, вони використовують тестовий набір, аби перевірити, чи зможе мережа вдало проаналізувати необхідну інформацію і видати правильні результати.

1.1. Аналіз області штучного інтелекту

Штучні нейронні мережі – це обчислювальні моделі, котрі працюють аналогічно до функціонування нейронної мережі людини. Це штучно змодельовані нейрони мозку. Нейрони можуть мати стан, зазвичай, представлений дійсними числами між 0 та 1. Комбінація цих двох станів серед мільярдів нейронів нашого мозку допомагає нам відчувати нашу реальність.

Нейронні мережі мають за мету змоделювати біологічний нейрон. Це особлива клітина, що складається з ядра, тіла і відростків. Також, ця клітина має тісний зв'язок з тисячами інших нейронів. Через цей зв'язок передаються електрохімічні імпульси, які призводять мережу в збуджений або спокійний стан [28]. При збудженні, як наслідок, нейронна мережа передасть сигнал до інших частин нашого тіла, що призводить до підвищення серцебиття, більш частого моргання очей, або якоїсь іншої реакції [1].

Існує декілька різновидів нейронної мережі. Всі вони реалізуються на основі математичних операцій та наборів параметрів, котрі необхідні задля визначення вихідних даних. Розглянемо деякі з мереж:

1. Рекурентна нейронна мережа (рисунок 1.1) працює за принципом збереження вихідних даних шару й передача його на вхідні дані, аби допомогти в прогнозуванні результату прошарку.

На цьому етапі перший прошарок формується аналогічно нейронній мережі з прямим зв’язком, з підрахуванням суми ваг та ознак. Рекурентний процес нейронної мережі починається після того, як відбулись всі підрахунки. Це означає, що з часу початку одного етапу, до наступного, кожний нейрон буде пам’ятати деяку інформацію, котру він мав на попередньому етапі. Це змушує кожний нейрон працювати як клітина пам’яті при виконанні розрахунків.

Якщо прогноз невірний, необхідно коригувати швидкість навчання та інші помилки. Необхідно вносити невеликі корегування, аби згодом працювати в напрямку правильного прогнозу під час зворотного розповсюдження [12].

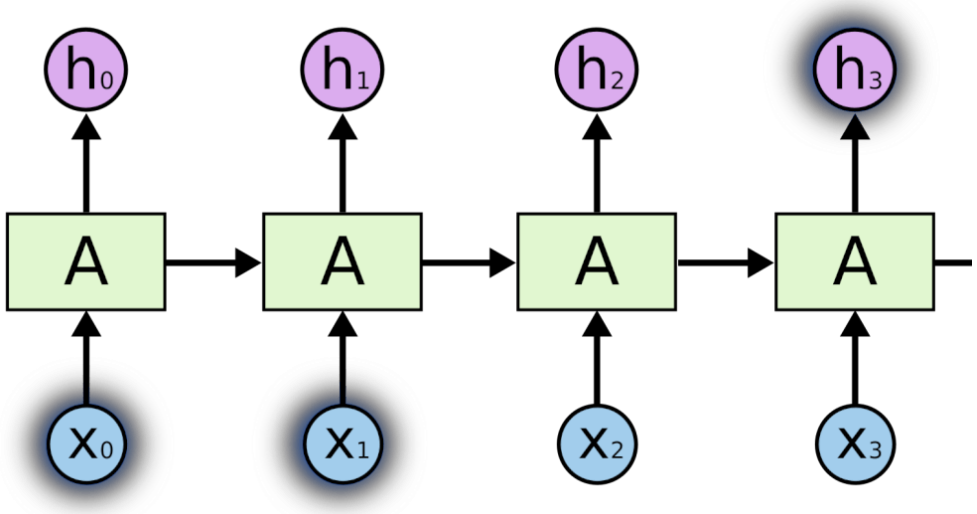

Рис 1.1 Рекурентна нейронна мережа

2. Згорткова нейронна мережа (рисунок 1.2) схожа на нейронну мережу з прямим зв’язком, в котрій нейрони мають здатність до навчання ваг та зміщення. Його застосування полягає у обробці сигналів та зображень. У цій нейронній мережі вхідні функції беруться в пакетному режимі мов фільтр. Це допоможе мережі запам’ятати зображення по шматкам та вирахувати необхідні операції. Ці розрахунки включають в себе перетворення

зображення з типу RGB або HIS в сірий. Після цього процесу можна, за допомогою вияву змін у піксельному значенні, виявити ребра, а саме зображення можна класифікувати за різними категоріями. Згорткові нейронні мережі застосовуються для вирішення таких завдань, як обробка сигналів і методи класифікації зображень. Методи комп'ютерного зору переважно використовують згорткові нейронні мережі через їх точність у класифікації зображень.

Згорткова нейронна мережа містить у собі один чи декілька загорткових шарів, які об’єднані чи повністю пов’язані між собою, та використовує варіацію багатошарових перцептронів. Згорткові шари використовують операцію згортання для виводу, передаючи результат наступному шару [13].

Згорткові нейронні мережі показують чудові результати у додатках, де використовуються необхідний аналіз зображень чи мови.

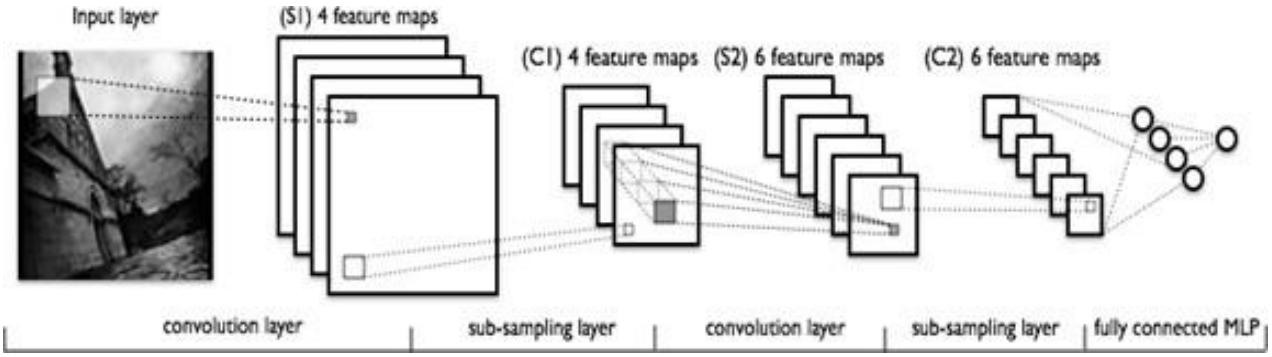

Рис 1.2 Згорткова нейронна мережа

3. Модульні нейронні мережі (рисунок 1.3) мають колекцію різноманітних мереж, які працюють незалежно та сприяють отриманню результату. Кожна нейронна мережа має набір вхідних даних, котрі є унікальними в порівнянні з іншими мережами, котрі створюють та виконують додаткові задачі. Ці мережі не взаємодіють та не повідомляють один одному про виконання завдань. Переваги модульної нейронної мережі у тому, що вона розбиває великий розрахунковий процес на більш менші компоненти, зменшуючи складність. Цей розподіл допоможе зменшити

кількість зв’язків та повністю прибрати необхідність їх взаємодії одна з одною, що збільшить швидкість розрахунків [6].

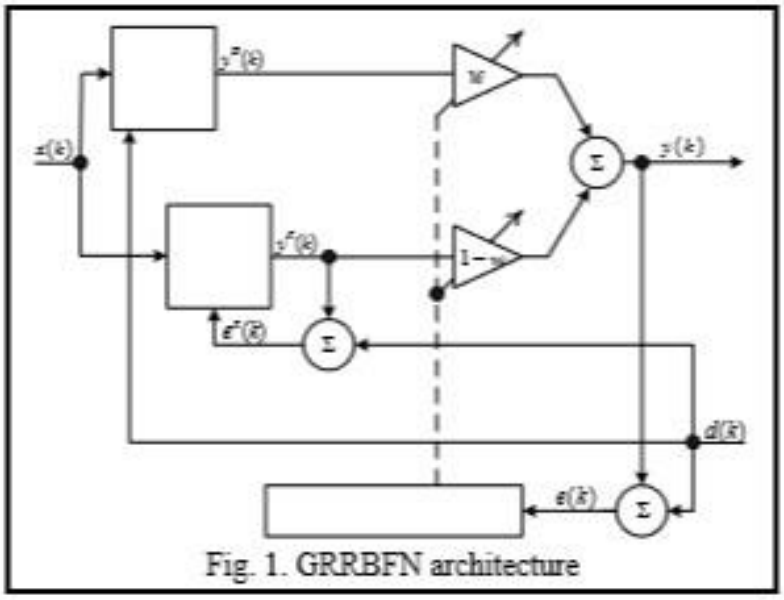

Fig. 1. GRRBFN architecture

Рис 1.3 Модульна нейронна мережа

4. Багатошаровий перцептрон (MLP)

Багатошаровий перцептрон (рисунок 1.4) має три або більше шарів. Він використовує нелінійну функцію активації, котра дозволяє класифікувати дані, котрі нелінійно розділені. Кожний вузол у шарі підключається до кожного вузла у наступному шарі, що робить кожну мережу повністю підключеною. Наприклад, багатошаровими додатками обробки природньої мови перцептронів є розпізнавання голосу та машинний переклад [14].

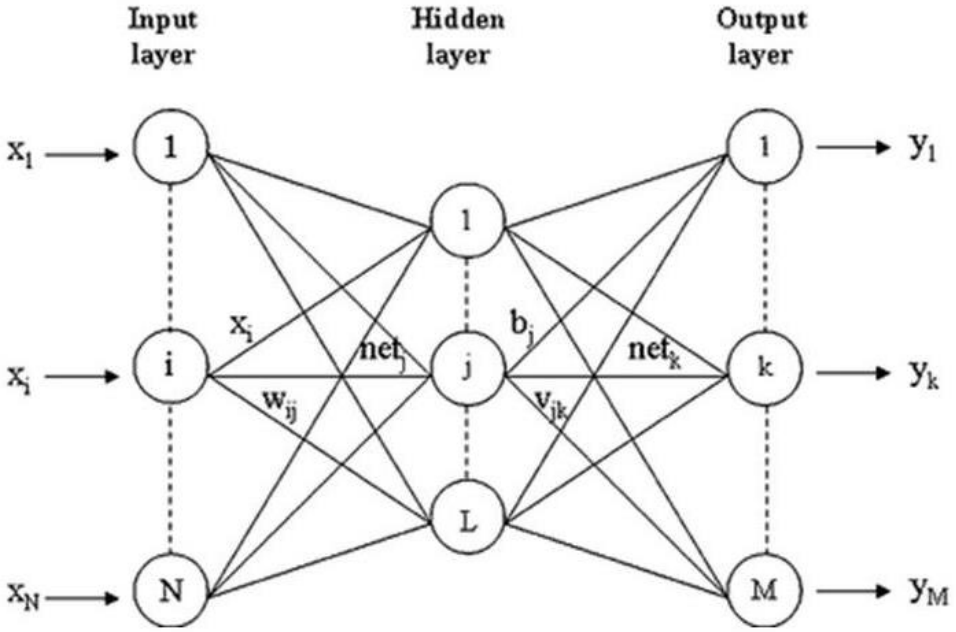

Рис. 1.4 Багатошаровий перцептрон

5. Рекурсивна нейронна мережа

Рекурсивна нейронна мережа (рисунок 1.5) – це тип глибокої нейронної мережі, сформований шляхом застосування одного й того ж набору ваг рекурсивно над структурою, щоб зробити структуроване передбачення над вхідними структурами змінного розміру чи скалярне передбачення на ньому, шляхом обходу даної структури у топологічному порядку. У простішій архітектурі нелінійність та вагова матриця, загальна для всієї мережі, використовуються для об’єднання батьківських вузлів [6].

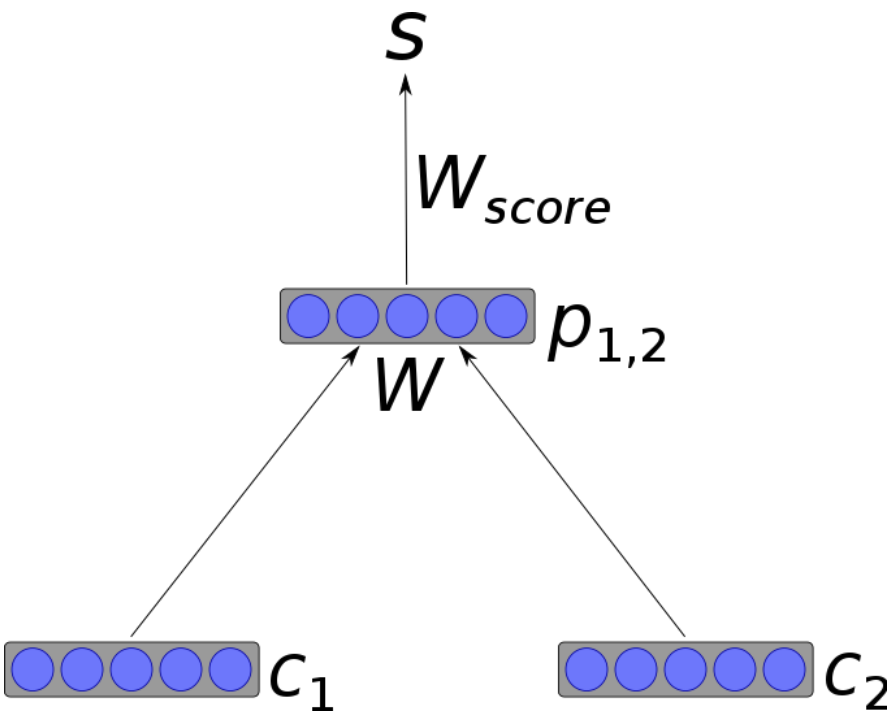

Рис 1.5 Рекурсивна нейронна мережа

6. Довготривала короткострокова пам’ять (LSTM)

Це специфічна архітектура (рисунок 1.6) рекурентної нейронної мережі, яка була розроблена для моделювання часових послідовностей та їх довгострокових залежностей більш точно, аніж звичайна рекурентна нейронна мережа. LSTM не використовує активаційну функцію у своїх рекурентних компонентах, збережені значення не змінюються, а градієнт не зникає під час навчання. Зазвичай блоки LSTM реалізуються у блоках з декількома частинами. Ці блоки мають чотири “двері”, котрі керують інформаційним потоком, опираючись на логістичну функцію [6].

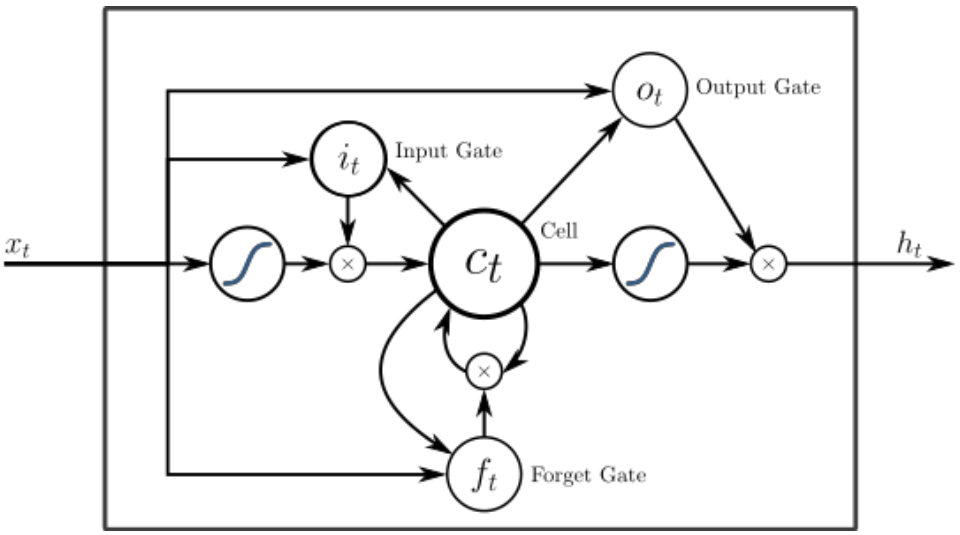

Рис 1.6 Архітектура довготривалої короткострокової пам’яті

7. Модель “послідовність - послідовність”

Зазвичай модель “послідовність – послідовність” складається з двох рекурентних нейронних мереж: кодеру, котрий оброблює вхід, та декодеру, обробляючого вихід. Шифратор та дешифратор можуть використовувати ті ж самі чи інші набори параметрів.

Моделі послідовності в основному використовуються у системах відповідей на питання, чат ботах та машинному перекладі. Такі багатошарові клітини були успішно використані у моделях “послідовність – послідовність” для перекладу у послідовність навчання з дослідженням нейронних мереж [6].

До нейронної мережі загалом можна віднести наступні складові:

Нейрон – це базова складова нейронної мережі. Він позначає кількість входів та значення зміщення. Коли нейрон отримує сигнал, значення, він помножується на значення ваг. Якщо нейрон має чотири входи, він має відповідно чотири значення ваг.

З’єднання (рисунок 1.7) – з’єднує один нейрон в одному прошарку з іншим нейроном в цьому ж прошарку. Зі з’єднанням завжди пов’язано значення ваги. Мета навчання – оновити значення ваг, аби зменшити похибку.

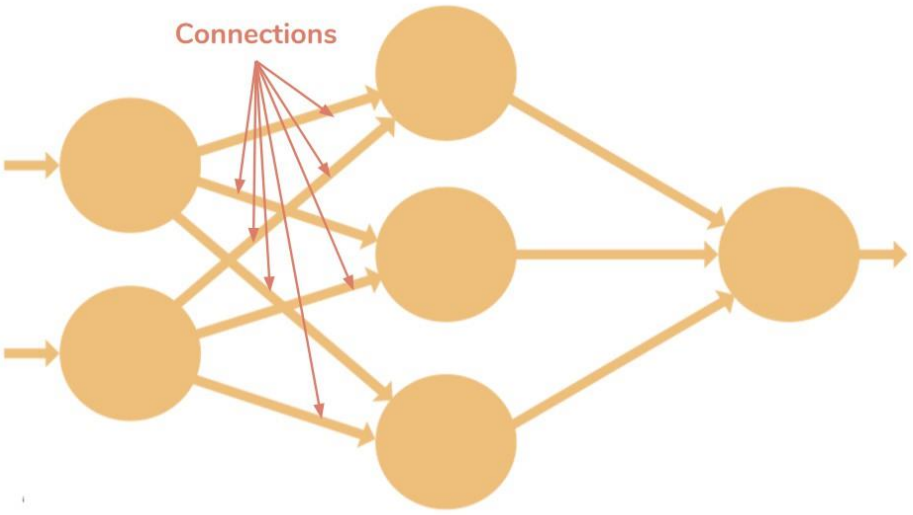

Рис 1.7 Схематичне зображення з’єднань нейронів

Зміщення (рисунок 1.8) – це додатковий вхід для нейронів, він завжди рівний одиниці та має власну вагу з’єднання. Це гарантує, що навіть коли всі входи не рівні нулю, в нейроні буде відбуватись активація.

Тоді як вхід має розмірність $w - h$, а ядро згортки мало розмірність $k_x \times k_y$, і якщо використовується зміщення s, то вихід буде мати розмірність:

$$\left\lfloor \frac{w - k_x}{s} + 1 \right\rfloor \times \left\lfloor \frac{h - k_y}{s} + 1 \right\rfloor$$

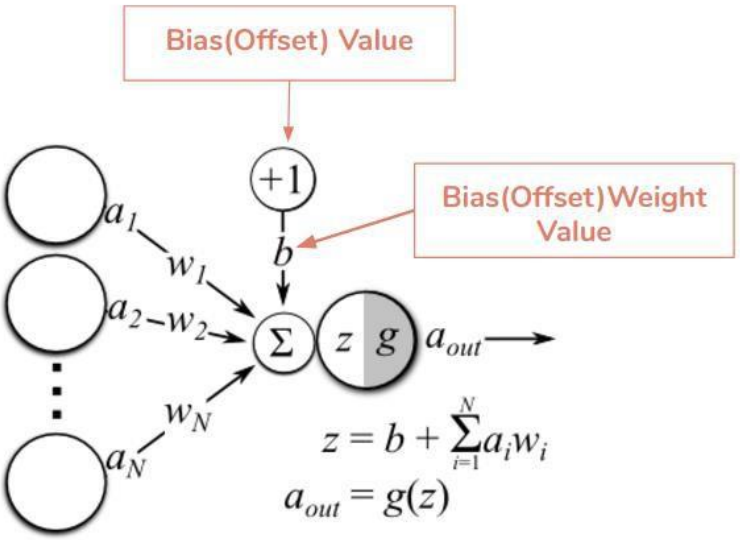

Рис 1.8 Схема зміщення

Функція активація (рисунок 1.9) використовується для введення не лінійності в нейронні мережі. Це розділяє значення на менші діапазони, а саме, функція активації зводить значення у діапазон від 0 до 1. Існує багато функцій активації у сфері машинного навчання.

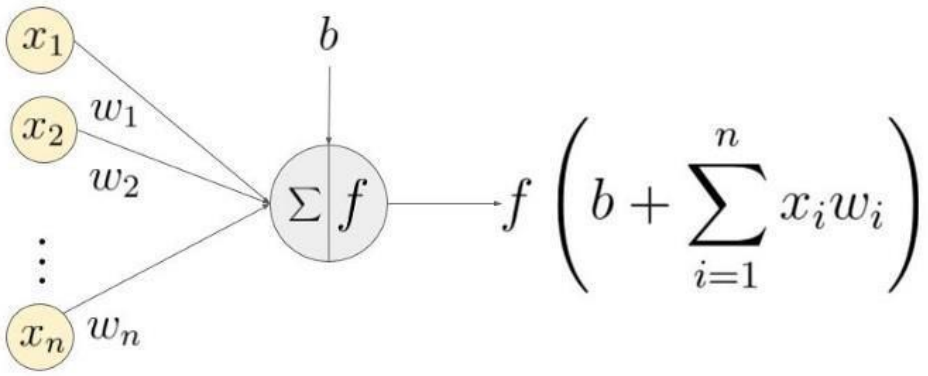

Рис 1.9 Схематичне зображення функції активації

Шар входу – це перший шар в нейронній мережі. Він приймає вхідні сигнали, значення, й передає їх на наступний рівень. Він не застосовує ніяких операцій до вхідних значень і не має пов’язаних значень ваг та зміщень.

Схований шар (рисунок 1.10) має нейрони, вузли, котрі застосовують різноманітні перетворення до вхідних даних. Один схований шар представляє собою набір нейронів. Останній схований шар передає значення до вихідного прошарку. Усі нейрони у схованому шарі пов’язані з кожним нейроном у наступному шарі, тому сховані шари повністю пов’язані.

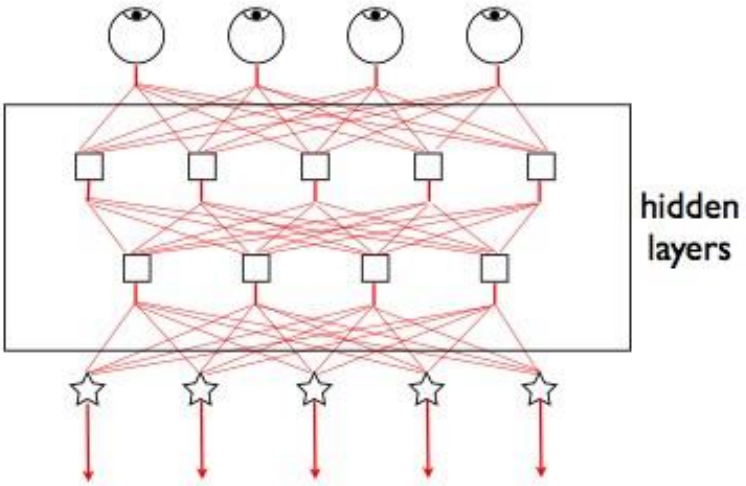

Рис 1.10 Схематичне зображення схованого шару

Вихідний шар (рисунок 1.11) – це шар, котрий являється останнім прошарком мережі й отримує данні від останнього схованого шару. За допомогою цього шару можна отримати бажану кількість значень у бажаному діапазоні.

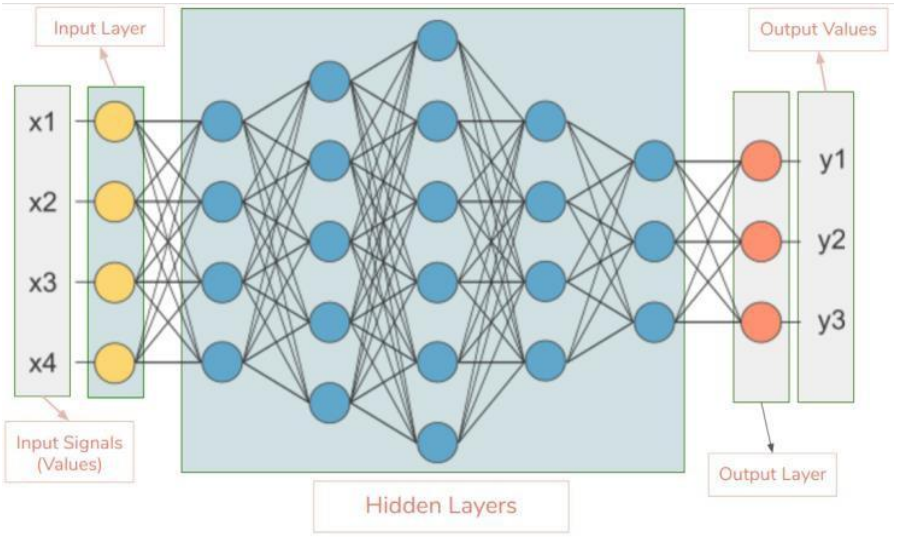

Рис 1.11 Схематичне зображення вихідного шару та нейронної мережі загалом

Вхідна форма – це форма вхідної матриці, котра передається для аналізу на вхідний шар.

Ваги (параметри) (рисунок. 1.12) – вага представляє собою силу зв’язку між одиницями. Якщо вага першого вузла до другого має більшу величину, це означає, що перший нейрон має більший вплив на другий нейрон. Вага знижує важність вхідного значення. Вага, котра близька до нуля, означає, що зміна цього входу не змінить вихід. Від’ємні ваги означають, що збільшення цього входу зменшить вихід. Вага вирішує, який вплив має вхід на вихід.

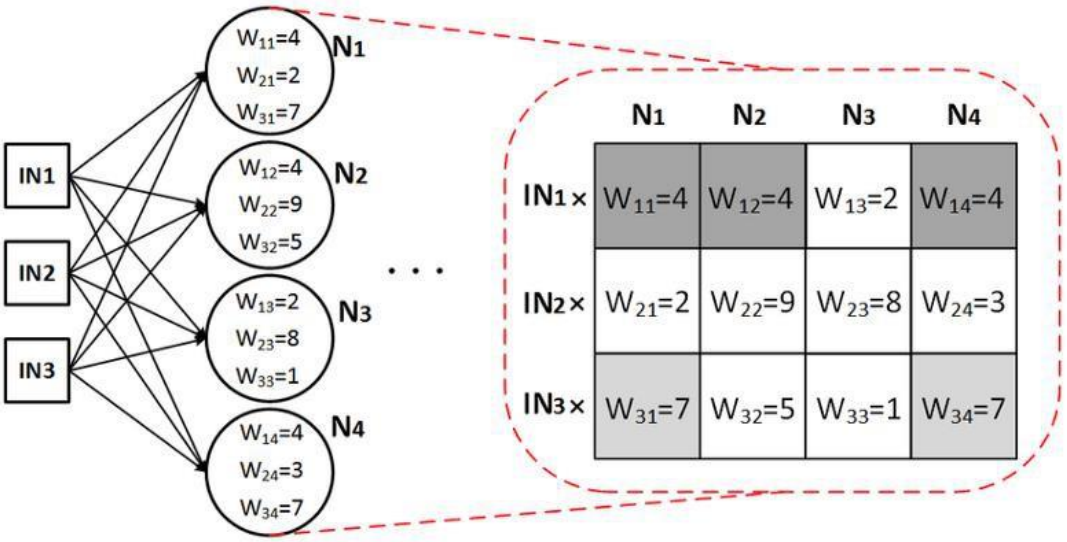

Рис 1.12 Схематичне зображення ваг

Пряме розповсюдження – це процес передачі вхідних даних в нейронну мережу та отримання вихідних даних, котрі називаються прогнозованими значеннями. Іноді пряме розповсюдження називають виводом. Коли передаються вхідні значення у перший шар нейронної мережі, він проходить без проведення операцій. Другий рівень приймає значення першого й застосовує операції множення, додавання та активації, після чого передає ці значення наступному рівню. Цей самий процес повторюється і для наступних шарів після чого отримується вихідне значення з останнього шару.[15]

Зворотне поширення (рисунок 1.13) – після прямого поширення отримується вихідне значення, котре являється прогнозованим. Аби розрахувати похибку, необхідно порівняти прогнозоване значення з фактичним вихідним значенням. Для цього використовують функцію втрат. Після цього розраховується похідна від значення похибки за кожною вагою в нейронній мережі. Зворотне розповсюдження використовує ланцюгове правило диференціального числення. За першим правилом ланцюга розраховується похідне значення похибки за значенням ваги останнього шару. Ці похідні називають градієнтом та використовують для розрахунку градієнту другого та останнього шару. Цей процес повторюється, доки не отримується градієнт для всіх ваг у мережі. Після цього значення градієнта віднімається від

значення ваг, щоб зменшити значення похибки. Таким чином досягаються мінімальні втрати.

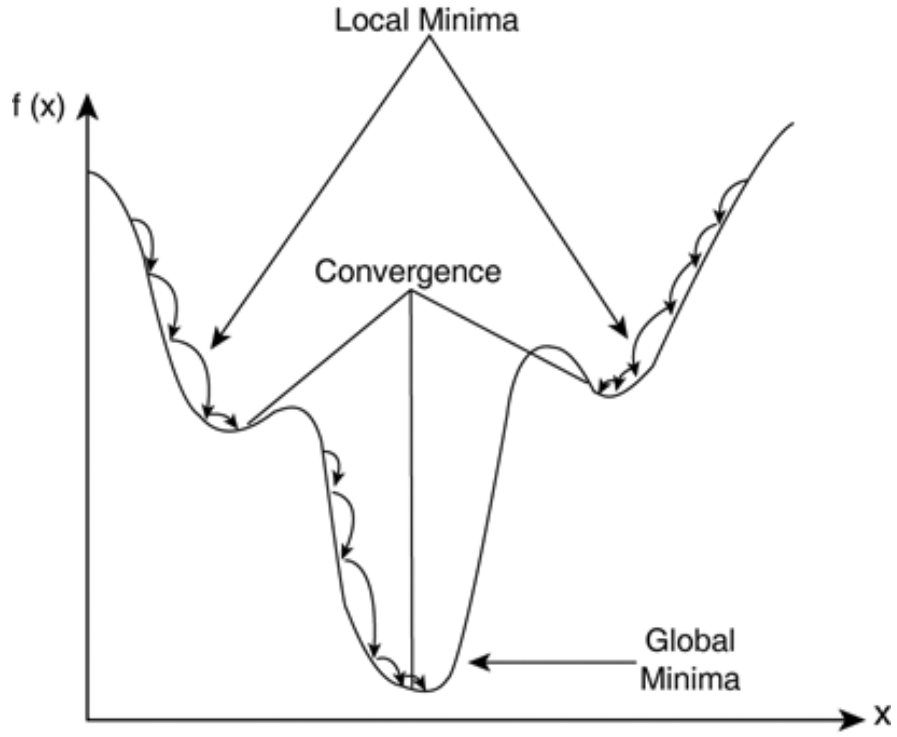

Рис 1.13 Графік зворотного поширення

Зазвичай, коли відбувається процес навчання нейронної мережі використовується градієнтний спуск, для оптимізації ваг. На кожній з ітерацій використовується зворотне розповсюдження, щоб розрахувати похідну функцію втрат за кожною вагою та відняти її значення. Швидкість навчання визначає наскільки швидко чи повільно можна оновити значення ваг, їх параметру. Швидкість навчання повинна бути достатньо високою, щоб не чекати роки, і вона повинна бути достатньо низькою, щоб встигала знаходити локальні мінімуми.

Точність відноситься до близькості вимірюваного значення до стандартного. Вона відображає близькість двох чи більше вимірів один до одного. Це частота або відтворюваність виміру.

Нормалізація – нормалізація даних – це процес зміни масштабу одного чи декількох атрибутів в діапазоні від нуля до одного. Нормалізація – це метод для того, щоб дізнатись як розподілені дані, чи якщо невідомо, чи

являється розподіл Гаусовським. Це може допомогти в прискоренні процесу навчання.

Повністю пов’язані шари (рисунок 1.14)– коли активація всіх вузлів у одному шарі переходять на кожен вузол у наступному шарі. Коли всі вузли у L-ому шарі поєднуються з усіма вузлами у (L + 1)-ому шарі – це називається повністю пов’язаними шарами [15].

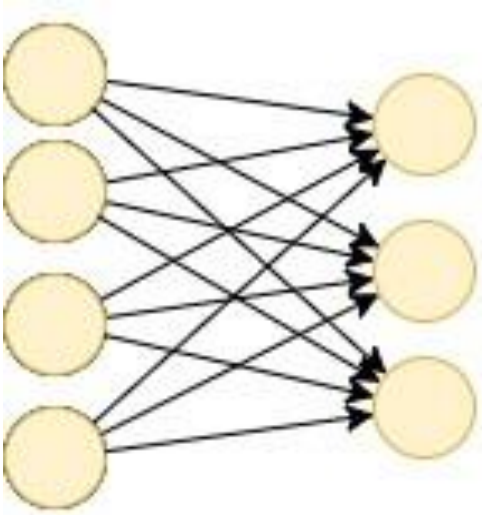

Рис 1.14 Повністю пов’язані шари

1.2. Представлення аналогів

Прогнозування міжнародного валютного ринку сьогодні ведеться на основі двох складових. Перша – фундаментальний аналіз, друга – технічний аналіз.

Фундаментальний аналіз враховує зовнішній вплив на котирування, будь то економічні і політичні події, важливі заяви глав держав і так далі.

Технічний же аналіз являє собою вивчення графіків, рівнів підтримки і опору. У неї закладено глибокий математичний апарат, тому проводити його вручну важко, а іноді і зовсім неможливо.

На просторах глобальної мережі їх безліч, розглянемо найбільш популярні, перевірені часом і практикою.

1. TRADER:

Програми для прогнозів Forex створюються, в першу чергу, для виявлення динаміки руху курсів валют, і TRADER справляється з цим завданням найбільш успішно. Він працює на основі історичних даних - максимальний і мінімальний рівні ціни, ціни закриття і відкриття, добові обсяги угод. Оскільки не завжди система має всіма перерахованими даними, передбачений аналіз на основі певної інформації. Наприклад, є тільки ціна закриття, програма вважає її і за мінімальну, і за максимальну. Що стосується алгоритмів аналізу, крім вбудованих в програму, користувач може створювати власні формули.

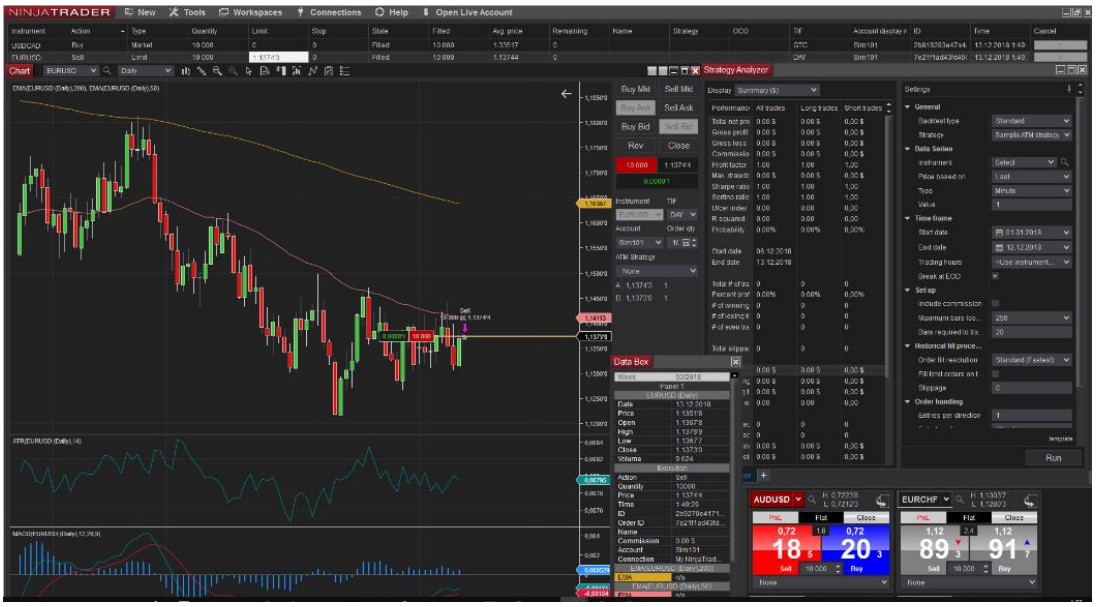

Рис 1.15 Інтерфейс програми TRADER

2. AmiBroker:

AmiBroker - це програма з дуже складним пристроєм і вражаючим набором функцій. Вона оснащена відразу декількома індикаторами, які можуть проводити технічний аналіз, а також створювати власні торгові системи, проводити їх тестування і корекцію.

Серед явних плюсів програми можна виділити:

- можливість коригувати код програми вручну, володіючи мовами програмування VBScript / JScript;
- сучасне оснащення пакетами для технічного аналізу;
- просте керівництво, можливість швидкого навчання роботи в програмі;
- кілька варіантів отримання котирувань;
- невисока вартість.

Програма спрямована на прогнозування котирувань з використанням різних джерел даних.

Рис 1.16 Інтерфейс програми AmiBroker

3. MetaStock:

Програма MetaStock на сьогодні, розроблена компанією Eguis, представляється найбільш затребуваним інструментом для технічного аналізу.

У програми MetaStock є особливість, яку впровадили розробники, передбачивши приблизно 160 вбудованих індикаторів і лінійних аналізаторів. Так само можливе написання своїх власних, які тільки можуть бути необхідні, індикаторів і надалі їх впровадження в середу MetaStock.

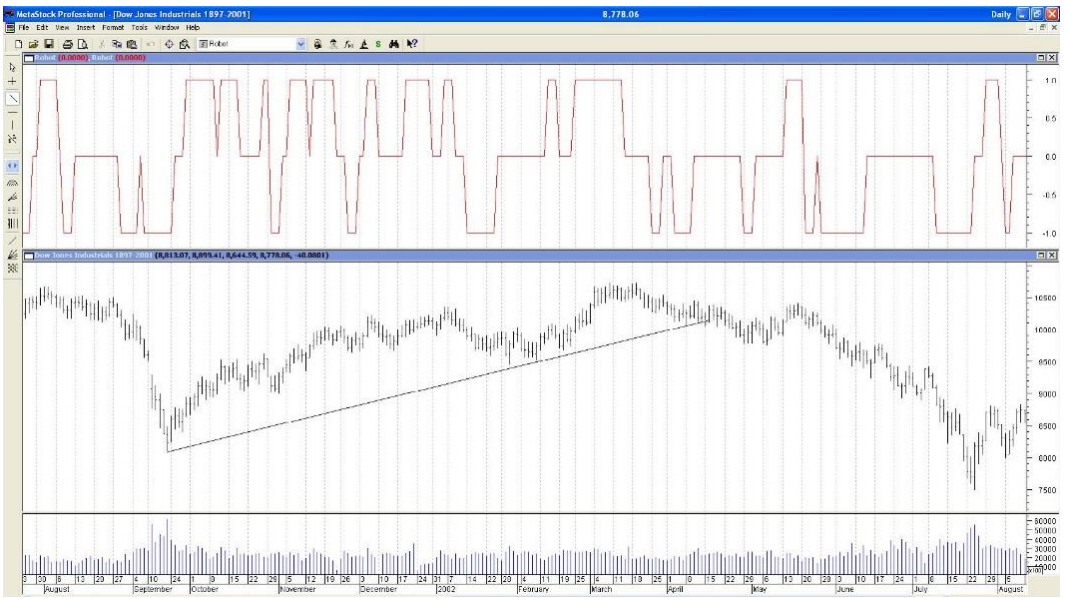

Рис 1.17 Інтерфейс програми MetaStock

4. Omega Research TradeStation 2000i:

Omega Research ProSuite 2000i – відмінна програма технічного аналізу, призначена для створення і тестування механічних торгових систем.

Розробники Omega передбачили безліч різних налаштувань для бірж, торгових сесій, переведення годинників, свят і т.д. Задумка хороша, але якщо не всі налаштування виставлені вірно - графіки можуть некоректно відображатися, а стратегії не працювати.

Рис 1.18 Інтерфейс програми Omega Research TradeStation 2000i

5. Wealth-Lab Developer:

Wealth-Lab Developer (WLD) є готовою платформою для розробки і бек-тестінгу торгових стратегій для акцій і ф'ючерсів, заснованих на технічному аналізі. Отримані торгові стратегії можуть бути використані як для роботи всередині дня, так і на більш довгих тимчасових інтервалах.

Торгова стратегія (або торговельна система) – це набір точно визначених правил, що генерують сигнали покупки (buy), продажу (sell), короткого продажу (sell short) і короткою покупки (cover). Торгова система призначена для надання Вам можливості отримання прибутку на ринках. Wealth-Lab Developer дозволяє розробляти і перевіряти торгові стратегії і також може бути використаний для застосування їх в реальній торгівлі. Програма має велику кількість дослідницьких інструментів і різних налаштувань.

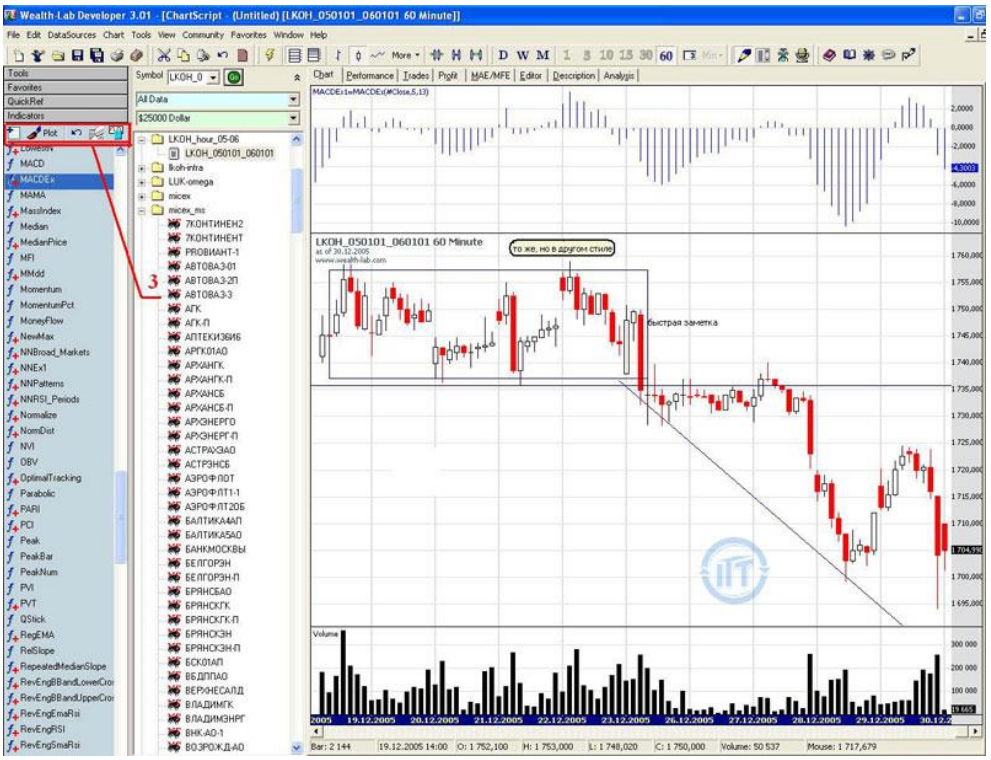

Рис 1.18 Інтерфейс програми Wealth-Lab Developer

1.3. Огляд багатошарового перцептрона

Обираючи тип нейронної мережі для створення магістерської роботи були прийняті до уваги наступні унікальні аспекти багатошарового перцептрона.

Багатошаровий перцептрон – нейронна мережа, що складається з шарів, кожен з яких складається з елементів - нейронів (точніше їх моделей). Ці елементи бувають трьох типів: сенсорні (вхідні, S), асоціативні (навчанні «приховані» шари, A) і реагують (вихідні, R). Багатошаровим цей тип перцептронів називається не тому, що складається з декількох шарів, адже вхідний і вихідний шари можна взагалі не оформляти в коді, а тому, що містить кілька (зазвичай, не більше двох - трьох) учнів (A) шарів.

Модель нейрона – це елемент мережі, який має кілька входів, кожен з яких має вагу. Нейрон, отримуючи сигнал, помножается сигнали на ваги і підсумовує отримані величини, після чого передає результат до іншого

нейрона або на вихід мережі. Тут теж багатошаровий перцептрон має відмінності. Його функція – сигмоид, вона видає значення на проміжку від 0 до 1.

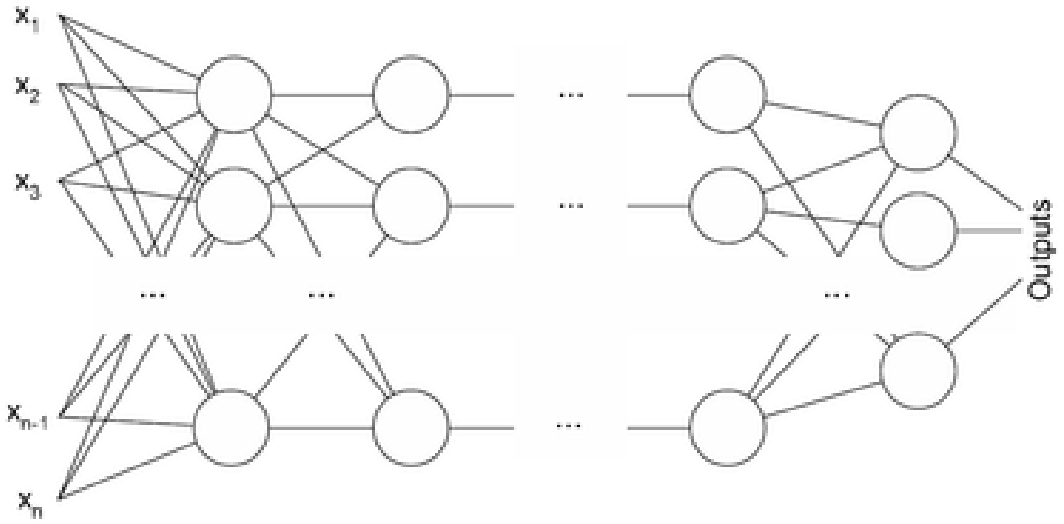

Рис 1.21 Архітектура багатошарового перцептрона

Багатошаровий перцептрон загалом може складатися з таких елементів, як:

- безліч вхідних вузлів, що разом утворюють вхідний шар;
- один або ж декілька прихованих шарів, які виконують функцію обчислювальних нейронів;
- один вихідний шар нейронів.

Багатошаровий перцептрон представляє собою узагальнення одношарового перцептрона Розенблатта. Наступна модель нейронної мережі є прикладом багатошарового перцептрона:

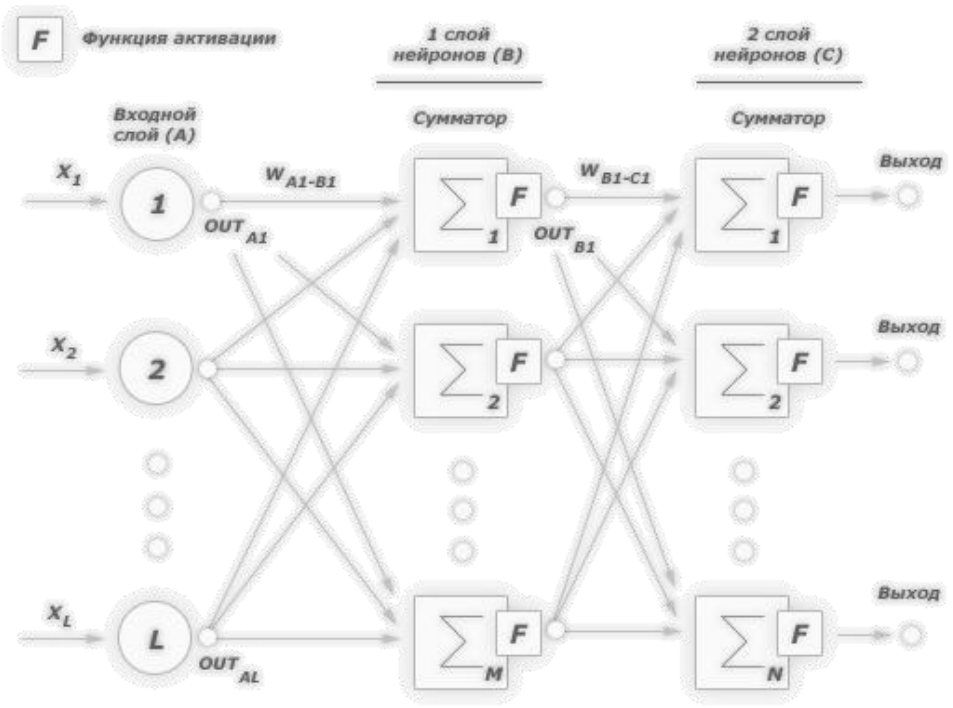

Рис 1.21 Приклад двошарового перцептрона

Багатошарові перцептрони успішно використовується для рішення різноманітних складних задач і мають наступних три відмінних особливості.

Властивість 1. Кожен нейрон мережі має нелінійну функцію активації

Важливо зазначити, що така нелінійна функція повинна бути гладкою (тобто функція диференціюється), на відміну від жорсткої порогової функції, що використовується у перцептроні Розенблатта. Самою популярною формою функції, що відповідає цій вимозі, є сигмоїдальна. Прикладом сигмоїдальної функції може бути логістична функція, що вказується наступним чином:

$$OUT = \frac{1}{1 + \exp(- \propto Y)}$$

де α – параметр нахилу відповідної функції. Змінюючи параметр нахилу, можна створити функції з різним степенем крутизни.

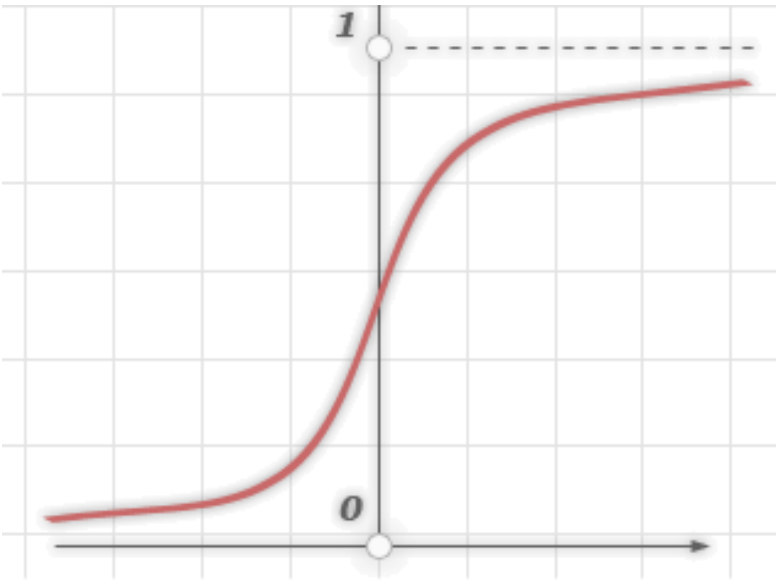

Рис 1.22 Сигмоїдна функція

Наявність нелінійності має дуже велике значення, так як в іншому випадку відображення «вхід-вихід» мережі можна привести до звичайного одношарового перцептрону.

Властивість 2. Кілька прихованих шарів

Багатошаровий перцептрон містить у собі один або декілька шарів прихованих нейронів, що не є частиною входу або виходу мережі. Ці нейрони дозволяють мережі навчатися вирішенню різних завдань, послідовно отримуючи найбільш важливі ознаки з образу входу.

Властивість 3. Висока зв'язність

Багатошаровий перцептрон має високу ступінь зв'язності, що реалізовується за допомогою синаптичних з'єднань. Зміна рівня зв'язності мережі вимагає також зміни великої кількості синаптичних з'єднань або їх вагових коефіцієнтів.

Поєднання всіх цих властивостей разом зі здатністю до навчання на власному досвіді забезпечує обчислювальну потужність багатошарового перцептрона. Але в той же час, ці ж якості є причиною неповноти сучасних знань про поведінку мереж такого роду: розподілена форма нелінійності і

висока зв'язність мережі помітно ускладнюють теоретичний аналіз багатошарового перцептрона [14].

Обов'язкові функції: завантаження часового ряду у форматі csv, вибір періоду часового ряду, що буде використовуватись для аналізу ринку компанії. Додаткові функції – можливість відображення вхідних даних та відображення результату у вигляді графіків.

ВИСНОВКИ ДО РОЗДІЛУ 1

Проаналізувавши предметну область, а саме, дослідження штучного інтелекту можна зробити наступні висновки:

- Нейронні мережі, з кожним роком, становляться все більш популярними.
- В останні роки популярність здобувається в області фінансів.
- Нейронні мережі допомагають робити прогнози щодо ситуацій на ринку компаній.
- Нейронні мережі мають дуже багато різновидів, наприклад такі як, рекурентні, згорткові, модульні та інші.
- Нейронні мережі складаються з певних елементів, до яких можна віднести нейрон, з’єднання, зміщення, функція активації, схований та вихідний шари, та інші.

Обрано оптимальну для магістерського проекту, нейронну мережу та описані її переваги. Поставлена здача дослідити обраний метод, та використати його для аналізу компаній відновлювальної енергетики.

РОЗДІЛ 2. МЕТОДИ ТА ЗАСОБИ ПРОЕКТУВАННЯ І РЕАЛІЗАЦІЇ ЗАДАЧІ

У нейронних мереж безліч важливих якостей, але ключові з них – це здатність до навчання. Навчання нейронної мережі у першу чергу полягає у зміні «сили» синоптичних зв’язків між нейронами. У класичному досвіді Павлова, кожний раз безпосередньо перед годуванням пса дзвонив дзвінок. Собака достатньо швидко навчилась асоціювати дзвінок з прийомом їжі. Це стало внаслідок того, що синоптичні зв’язки між зонами головного мозку, які відповідають слух та слинні залози, посилились. В наслідок цього, збудження нейронної мережі звуком дзвіночка призводило до більш сильного слиновиділенню собаки.

На сьогодні нейронні мережі є одним із пріоритетних напрямків досліджень в області штучного інтелекту.

Для створення програмного засобу було використано мову програмування Python та бібліотека Keras.

2.1. Вибір інструментів розробки

При плануванні програми виникло питання, яку мову, функції та бібліотеки краще використовувати.

Розглянемо для початку мови програмування, на яких можлива реалізація нейронної мережі і наскільки це можливо зробити з їх допомогою.

Загалом нейронну мережу можна створити за допомогою будь якої мови програмування. У мережі безліч книжок про машинне навчання з реалізаціями на Java, C#, JavaScript, Go, R, Erlang, Julia, Haskell і так далі.

Для порівняння було обрано дві крайності цього переліку, це такі мови програмування, як C++ та Python. Мова програмування C++ має перевагу в швидкості виконання. Можливо мова Go хоч якось наздоганяє її в цьому. Все це через те, що мова програмування C++ являється мовою низького рівня, що дозволяє використовувати усі особливості процесору, пам'яті і так далі. Можливо швидше виявиться хіба що мова Assembler. В цей же час, C++ - це мова, на якій важче всього і найдовше доведеться писати код. Іноді п'ять строк коду на мові Python рівноцінні сорокам строкам на мові C++. Також, слід зазначити, що Python має велику кількість вже створених бібліотек для роботи з нейронними мережами, що дозволяє ще ефективніше і швидше працювати над проектом.

Якщо необхідно швидко виконати дослідження, створити модель, провести її тестування, необхідно обрати мову Python. Якщо необхідно, аби процес прийняття рішень були максимально швидкими, обираємо C++. В деяких областях, наприклад, розпізнавання візуальних образів, відсвіжування працездатності приладів, збереження безпеки, розпізнавання голосу, торгуванні на біржі – швидкість критично необхідна. В інших випадках, в діагностиці медичних знімків, маркетингу, економічному аналізі, розпізнаванні текстів, сентиментному та агротехнічному аналізі швидкість значення не має.

Отже, програмувати або моделювати нейронну мережу можна на будь-якій мові програмуванні загального призначення. Вибір залежить від необхідній швидкості роботи мережі та наявності готових бібліотек для створення штучного інтелекту. Все залежить від поставленої мети. В нашому випадку обрана мова програмування Python, оскільки для написання магістерської роботи, швидкість значення не матиме а наявність додаткових бібліотек спростить процес виконання задачі.

Оскільки вибір мови програмування випав на Python, слід розглянути існуючі бібліотеки, котрі можуть знадобитись для машинного навчання.

1. Scikit-learn

Scikit-learn – це одна з найбільш популярних бібліотек для машинного навчання, яка може підтримувати велику кількість контрольованих та неконтрольованих алгоритмів навчання, таких як лінійні та логістичні регресії, дерева прийняття рішень, k-means, кластеризація та інші.

Дана бібліотека була створена на базі двох основних бібліотек Python – NumPy і SciPy. У Scikit-learn додано набори алгоритмів для розповсюдження задач машинного навчання та знаходження даних, що включає в себе кластеризацію, регресію та класифікацію. Навіть такі завдання, як перетворення даних та вибір функції можуть бути реалізовані за допомогою лише декількох строк.

Рис 2.1 Бібліотека Scikit-learn

2. TensorFlow

Цікаво, що при використанні даної бібліотеки можна скомпілювати та запускати програму як за допомогою CPU, так і за допомогою GPU. Таким чином для запуску на GPU не доведеться використовувати C++ чи CUDA.

Бібліотека використовує систему багаторівневих вузлів, котра дозволяє швидко налаштовувати та навчати штучні нейронні мережі з великими наборами даних. Саме це дозволяє наприклад Google визначати речі на фотографіях та розуміти слова у додатках для розпізнавання голосу.

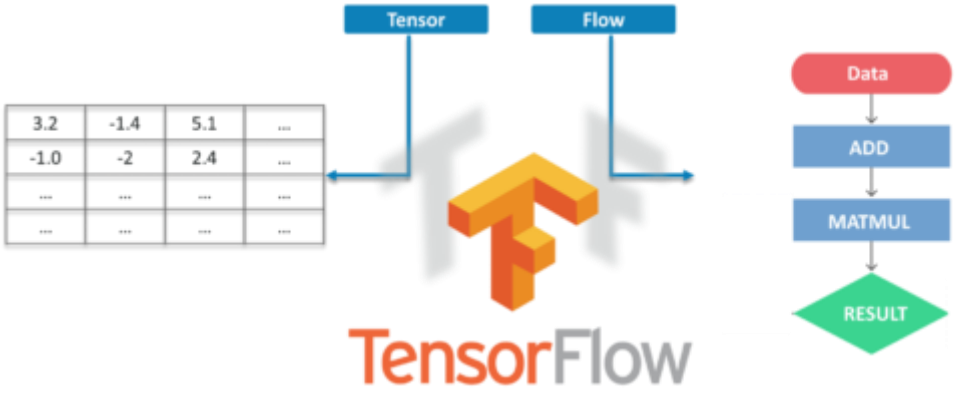

Рис 2.2 Бібліотека TensorFlow

3. Theano

Theano – бібліотека для алгоритму числових розрахунків, схожа на NumPy. Вона дозволяє ефективно визначати, оптимізувати та розраховувати математичні вирази з багатомірними масивами.

Цю бібліотеку виділяє використання виключно переваг GPU комп'ютера. Це дозволяє їй робити розрахунки з великими обсягами даних у сто разів швидше, ніж при запуску виключно на CPU. Швидкість Theano особливо цінна задля глибокого навчання та інших задач, пов'язаних з складанням розрахунків.

Рис 2.3 Бібліотека Theano

4. Pandas

Pandas являється дуже популярною бібліотекою, яка представляє собою багаторівневі структури даних, легкі в використанні й інтуїтивно зрозумілі.

В даній бібліотеці є велика кількість вбудованих методів для групування, комбінування даних та їх фільтрації, а також аналізу часових рядів. Pandas може з легкістю витягати данні з різноманітних джерел, таких як

бази даних SQL, файли CSV, Excel, JSON та маніпулювати цими даними для здійснення операцій з ними.

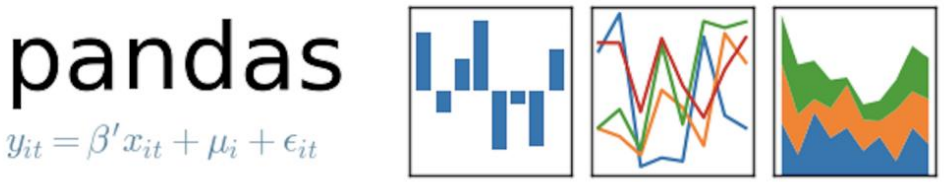

Рис 2.4 Бібліотека Pandas

5. Matplotlib

Найкраще і вигадливе машинне навчання не має сенсу, якщо ви не можете розповісти про нього іншим людям.

Matplotlib – це стандартна бібліотека Python, яку використовують для створення графіків. Вона достатньо низькорівнева, а це означає, що потребує більше команд для генерації графіків та фігур аніж інші бібліотеки.

Оперуючи достатньою кількістю команд, можна створити практично будь-який графік. Можна будувати різноманітні діаграми, гістограми, діаграми розсіювання, графіки з недекартовими координатами. Графіки можна експортувати у всіх розповсюджених форматах.

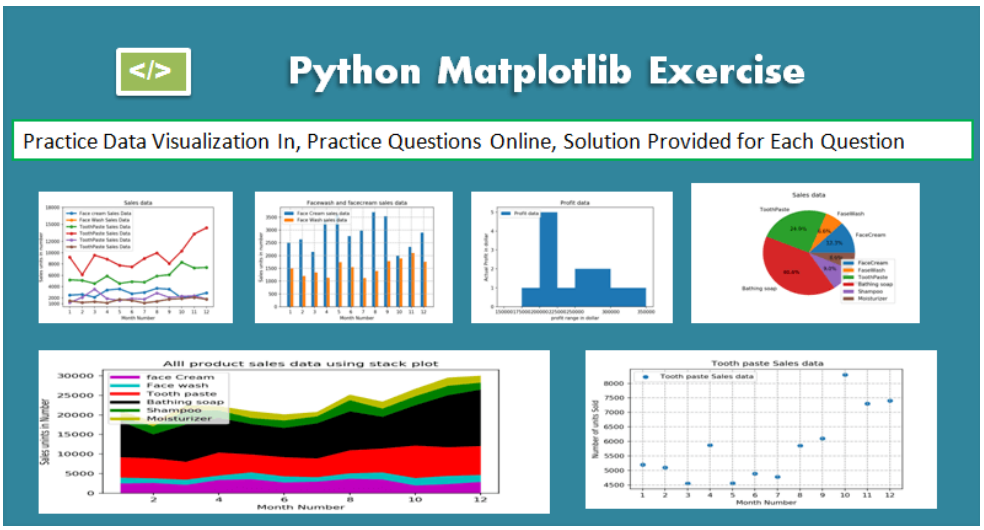

Рис 2.5 Бібліотека Matplotlib

6. Seaborn

Seaborn – це популярна бібліотека візуалізації, яка будує графіки на основі Matplotlib. Ця бібліотека більш високого рівня, а це означає, що з її допомогою легше генерувати певні види графіків, у тому числи теплові мапи, часові ряди та багато іншого.

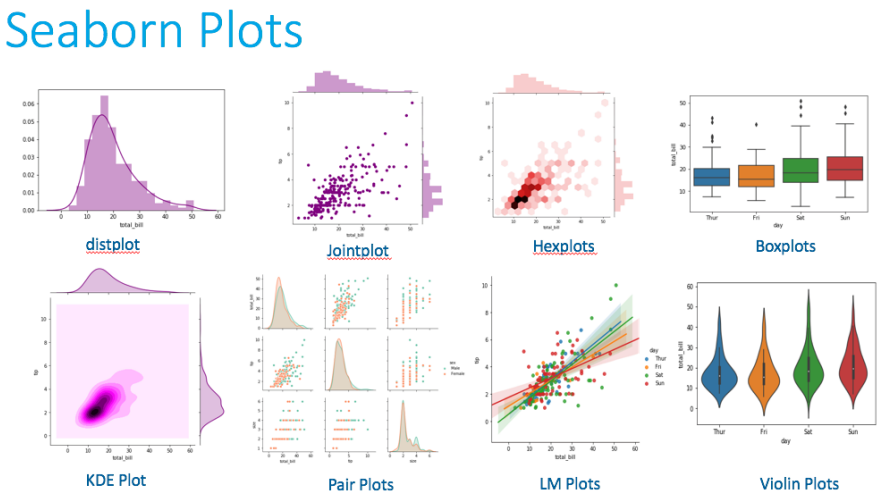

Рис 2.5 Бібліотека Seaborn

При огляді допоміжних бібліотек для створення та тестування нейронних мереж було обрано бібліотеки TenslorFlow та Pandas. Розглянемо бібліотеку TenslorFlow, і те, як вона працює, більш детально.

TenslorFlow дозволяє розробникам створювати графи потоків даних – структури, котрі описують, як данні переміщуються через граф чи серію вузлів обробки. Кожний вузол на графіку представляє математичну операцію, а кожний зв'язок чи ребро між вузлами – це багатовимірний масив даних або тензор.

TenslorFlow надає все для комфортного створення нейронної мережі на мові Python. На той же час Python легкий у вивченні та роботі.

TenslorFlow має при собі надбудовану бібліотеку під назвою Keras. Це відкрита бібліотека, написана на мові Python. Вона націлена на оперативну роботу з мережами глибокого навчання, при цьому спроектована так, аби бути компактною, модульною та розширяємою. Планувалось, що Google буде підтримувати Keras у основній бібліотеці TenslorFlow, але її розробник,

Франсуа Шолле, виокремив Keras у окрему надбудову, так відповідно концепції Keras виступає у якості інтерфейсу, аніж у якості системи машинного навчання. Keras представляє високорівневий, більш інтуїтивний набір абстракцій, котрий робить простим формування нейронних мереж, незалежно від використаних бібліотек дослідницьких розрахунків. Також, Microsoft працює над тим, аби додати до Keras низькорівневих бібліотек CNTK (Microsoft Cognitive Toolkit). Кожний програмний засіб з використанням нейронної мережі унікальний, але розробка мережі зазвичай виконується за наступними етапами:

- Доступ та підготовка даних;
- Створення нейронної мережі;
- Налаштування входів та виходів мережі;
- Налаштування параметрів мережі, а саме, ваг та зміщення, для оптимізації продуктивності;
- Тренування мережі;
- Перевірка результатів;

Також треба розглянути бібліотеку Pandas, а саме, її можливості.

Pandas — це бібліотека, яка написана на мові програмування Python задля керування, аналізу і взаємодії з даними. Данна бібліотека, пропонує можливості взаємодії зі структурами даних та засоби для маніпулювання числовими схемами і часовими рядами.

Розробник Pandas, Вес Маккінні розпочав роботу над бібліотекою у 2008 році в той час, коли працював у компанії AQR Capital Management, оскільки виникла потреба у продуктивному, гнучкому інструменті який допоможе здійснювати кількісний аналіз фінансових часових даних. Перед тим, як залишити компанію AQR, Весу вдалось умовити своє керівництво дати йому можливість поширити бібліотеку віддавши її у відкритий для всіх доступ [19].

Бібліотеку Pandas було обрано через те, що вона була створена саме для аналізу фінансових даних. Основні можливості бібліотеки Pandas:

- Об’єкт DataFrame використовується задля маніполювання проіндексованими двовимірними масивами;
- Інструменти, які можуть використовуватись для обміну даними між різними структурами у пам’яті та з файлами різноманітних форматів;
- Встановлені засоби для об’єднання даних та засоби, за допомогою яких можна обробляти відсутню інформацію;
- Переформатування наборів даних та створення вільних таблиць;
- Зріз даних зазначенням їх індексу;
- Вставка та видалення стовпців даних;
- Можливість групування, яка дозволяє виконувати операції типу "розьєднання, змінення, обьєднання";
- Поєднання наборів даних
- Ієрархічне індексування, яке дозволяє працювати з даними високої розмірності у структурах, що мають розмірність нижче;
- Можливість працювати з часовими рядами, а саме: формування часових періодів, змінення їх інтервалів та інше.

2.2. Огляд доступних джерел баз даних

Перед початком навчання нейронної мережі необхідно підготувати набір даних, які будуть використовуватись у процесі.

Наявність якісного джерела фінансових даних – це ключова умова створення рішень в області штучного інтелекту.

Аналіз даних фінансових часових рядів та створення систем для аналізу стійкості компаній – це один із найпопулярніших направлень застосування штучного інтелекту. Аби знайти фінансову базу компанії відновлювальної

енергетики, для дослідження методів аналізу, було розглянута певна кількість, популярних у цій сфері, джерел, наприклад:

finance.yahoo

Велика фінансова база даних різноманітних компаній. Yahoo! Finance - провайдер фінансової інформації, який належить Yahoo!

Один з головних постачальників подібної інформації в США з близько 70 мільйонами відвідувачів на травень 2018 року.

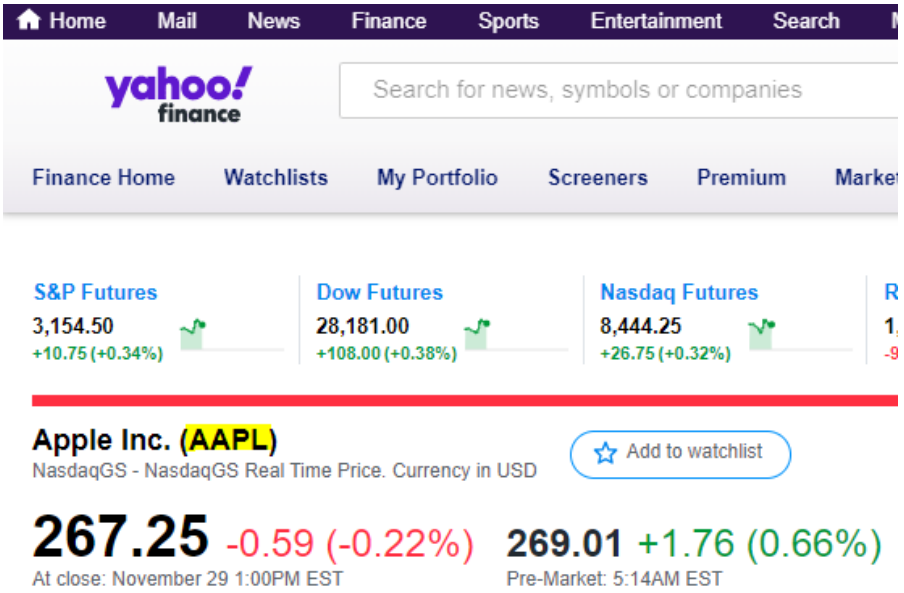

Рис 2.1 Інтерфейс сайту finance.yahoo

Google Finance

Google Finance - провайдер, який надає фінансову інформацію, що належить компанії Google Inc. Хмарний сервіс Google Finance надає фінансову інформацію про велику кількість транснаціональних компаній.

Інформація, яка надається - це доступ по котируваннях і рейтинги дорогих паперів, пресрелізи та фінансові звітності компаній.

Відображаються результати агрегаторів з Google News і Google Blog Search по кожній компанії. Історичні дані доступні у вигляді графіків, реалізованих за технологією Adobe Flash.

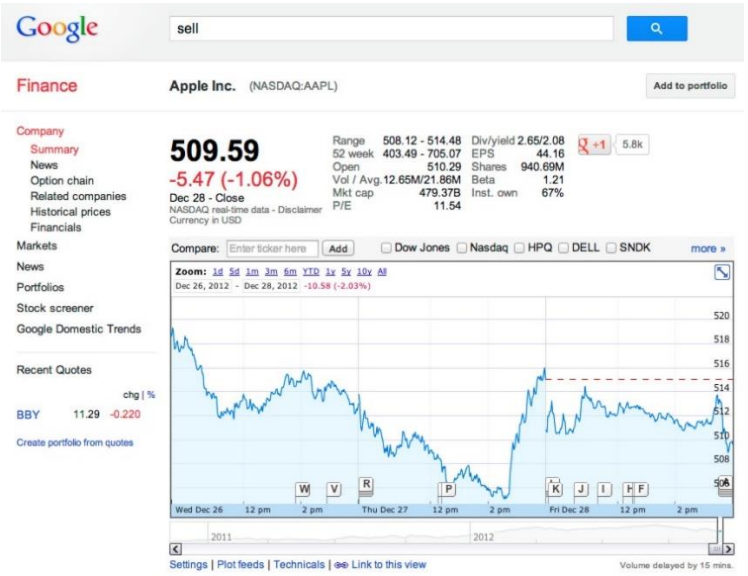

Рис 2.2 Інтерфейс сайту Google Finance

Bloomberg L.P.

Bloomberg L.P. - один з головних постачальників фінансової інформації для професійних учасників фінансових торгів.

Основним продуктом являється Bloomberg Terminal, за допомогою якого можна отримати інформацію щодо поточних і історичних цін відносно практично всіх світових бірж і більшості позабіржових торгових ринків, стрічок новин агентства Bloomberg та інших головних засобів масової інформації, систем торгівлі облігаціями та іншими важливими паперами.

Рис 2.3 Інтерфейс сайту Bloomberg Terminal

Reuters

Reuters - одне з найбільших у світі міжнародних агентств фінансових новин та фінансової інформації, існує з другої половини XIX століття.

У 2008 році агентство куплене корпорацією Thomson, яка після того стала називатися Thomson Reuters.

Reuters Group plc. є публічною компанією, акції обертаються на Лондонській фондовій біржі.

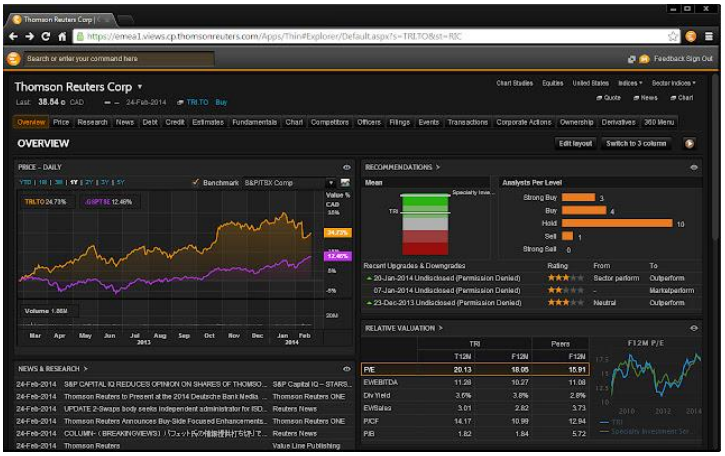

Рис 2.4 Інтерфейс сайту Thomson Reuters

Factiva

Factiva – підрозділ компанії Dow Jones, що займається наданням доступу до ділової та аналітичної інформації через свої інформаційно-аналітичні служби. У своїй основі служби Factiva мають більш ніж 30 тисяч первинних джерел з 157 країн світу на 23 мовах. Доступ в інформаційно-аналітичні служби здійснюється через Інтернет.

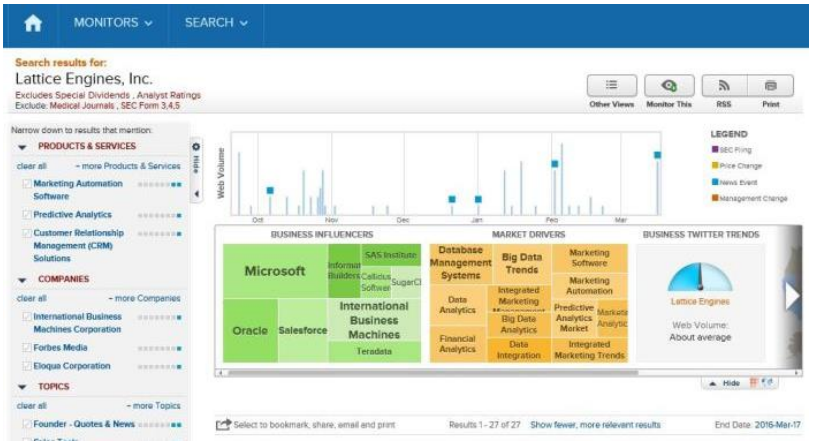

Рис 2.5 Інтерфейс сайту Factiva

ВИСНОВКИ ДО РОЗДІЛУ 2

Поглянувши методи проектування та реалізації нейронної мережі було порівняно такі мови програмування як C++ та Python, аби обрати з них найбільш оптимальний варіант для створення нейронної мережі, яка дозволить прогнозувати часові ряди. Обрано мову Python, через те, що вона має велику кількість бібліотек, які можуть зробити легше процес створення мережі, та більш простий синтаксис.

Після вибору мови програмування, було розглянуто існуючі бібліотеки для створення та аналізу нейронних мереж, а також, були розглянуті джерела, з яких можна завантажити часовий ряд компанії відновлювальної енергетики.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ АЛГОРИТМІВ ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ

Опираючись на попередні розділи необхідно спроектувати та реалізувати алгоритм MLP мережі. Розглянути основні функції, оглянути реалізацію проекту. Розглянути задачі регресії та класифікації методами MLP мережі.

Існує певна відмінність між проблемами класифікації та регресії. Класифікацією являється передбачення мітки даних, а регресія – передбачення кількостей.

Класифікаційно-прогнозуюче моделювання – це задача наближення функції відображення від вхідних змінних до дискретних вихідних змінних. Вихідні змінні також називають мітками або ж категоріями. Функція відображення передбачує клас або ж категорію для спостережень.

- Задача класифікації вимагає, аби значення були класифіковані в один або ж два класи.
- Проблема з двома класами часто називається проблемою двокласної або двійкової класифікації.
- Проблема з біль ніж двома класами часто називається проблемою класифікації декількох класів.
- Проблема, коли для прикладу назначається декілька класів, називається проблемою класифікації за кількома мітками.

Точність класифікації – це відсоток правильно класифікованих прикладів від загальної кількості прогнозів. Прогнозована вірогідність може бути перетворено в значення класу шляхом вибору мітки класу, що має найбільшу вірогідність.

Прогнозуючи-регресійне моделювання – це задача наближення функції відображення від вхідних змінних до неперервної вихідної змінної. Неперервна вихідна змінна – це дійсне значення, ціле або ж число з плаваючою точкою, яке часто являється значенням кількості.

- Задача регресії потребує передбачення конкретного значення, кількості.
- Проблема з декількома вхідними змінними часто називається проблемою багатовимірної регресії.
- Регресія може мати дійсні, або ж дискретні вхідні змінні.
- Задача регресії, в якій вхідні дані впорядковані за часом, називається задача прогнозування часових рядів.

Деякі алгоритми мають в іменуванні слово «регресія», наприклад, лінійна регресія або ж логістична регресія, що може привести до плутанини, оскільки лінійна регресія є алгоритмом регресії, тоді як логістична регресія являється алгоритмом класифікації.

3.1 Перевірка залежностей та аналіз авторегресійного методу прогнозування часових рядів

Скориставшись аналізом часових рядів, можна спробувати відповісти на питання, чи залежать ціни на вітроенергетику від ціни на нафту.

Для початку треба імпортувати необхідні модулі для роботи, а саме: pandas_datareader.data, numpy, pandas, statsmodels.api, matplotlib.pyplot.

Для завантаження даних з веб ресурсів скористаємось наступною бібліотекою:

```
import pandas_datareader.data as web
```

Для аналіза даних імпортуємо наступним чином відповідні бібліотеки:

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
```

Для побудови графіків і візуалізації результатів імпортуємо pyplot:

```
import matplotlib.pyplot as plt
```

Інформацію о цінах на нафту візьмемо з бази даних федерального резерву США, а індикатором стану компанії відновлювальної енергетики візьмемо дані компанії Vestas Wind Systems A/S, під індексом VWS.CO.

Для завантаження даних буде використаний модуль pandas_datareader, який дозволяє імпортувати дані з різних джерел.

Вкажемо діапазон, який ми візьмемо для аналізу часових рядів.

```
start = '2013-01-01'
end = '2020-10-31'
```

Тепер імпортуємо часовий ряд цін на нафту, вказавши індекс, початок та кінець часового ряду.

```
brent = web.DataReader('DCOILBRENTEU', 'fred', start=start, end=end)
```

Скориставшись сервісом Yahoo! Finance завантажимо часовий ряд компанії індексу VWS.CO:

```
vws = web.DataReader('VWS.CO', 'yahoo', start=start, end=end)
```

Згрупуємо завантажені дані по дням, на той випадок, якщо імпортовані часові ряди розсинхруються.

```
brent = brent.groupby(pd.Grouper(freq='1D')).aggregate("mean")
vws = vws.groupby(pd.Grouper(freq='1D')).aggregate("mean")
```

Створимо графік з відповідною назвою, на якому відобразимо зв'язок цін на нафту, та цін на вітроенергетику.

```
plt.figure(figsize=(15,5))
plt.title('Індекс VWS та ціна на нафть')
```

```
    ax_1 = df.vws_price.plot(color='blue', grid=True,
label='VWS.CO')
    ax_2 = df.brent_price.plot(color='red', grid=True,
secondary_y=True, label='Brent')
    h_1, l_1 = ax1.get_legend_handles_labels()
    h_2, l_2 = ax2.get_legend_handles_labels()
    plt.legend(h_1+h_2, l_1+l_2, loc=1)
    plt.show()
```

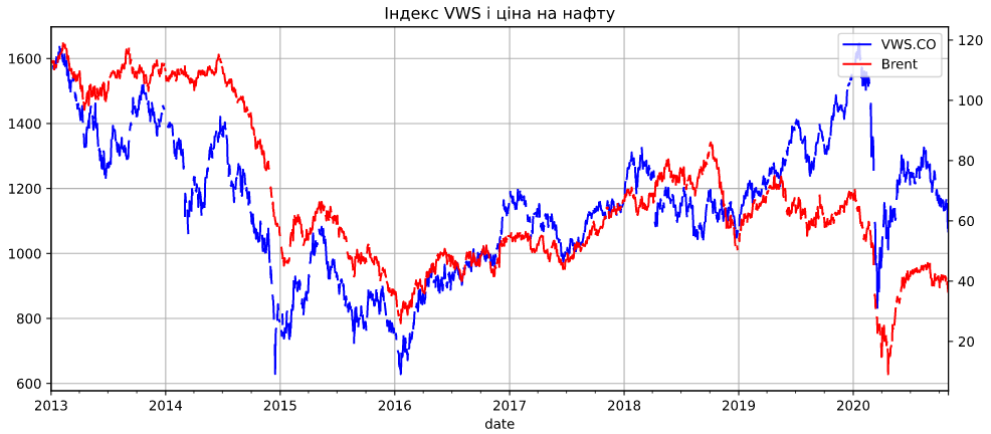

Рис 3.1 Графік зв’язку цін нафти на компанію відновлювальної енергетики

На перший погляд можна навіть сказати, що кореляція, взаємозвязок, дуже сильний, і що графіки співпадають. Розраховуючи кореляцію можна визначити степінь лінійного зв’язку двох часових рядів.

```
corr=df.dropna().vws_price.corr(df.dropna().brent_p
rice)"Correlation is {:.3}".format(corr)
```

Значення кореляції, що перевищує 0.66 зазвичай вважається сильною, а результатом розрахунку кореляції часового ряду VWS.CO та часового ряду цін на нафту рівний 0.628, що є значенням близьким до сильного.

Якщо побудувати графік, у котрому на одній осі відмітимо ціну нафти, а на іншій індекс VWS.CO, то зможемо наглядно побачити кореляцію.

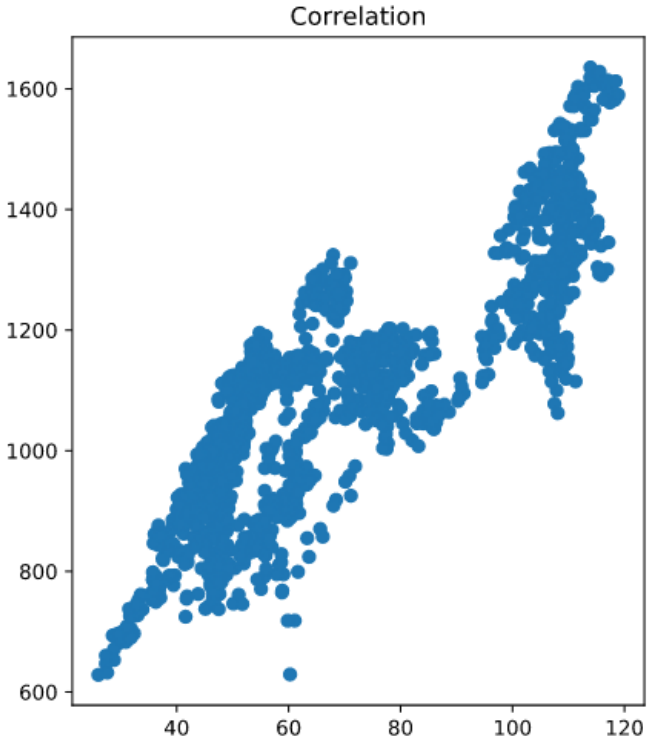

Рис 3.2 Графік кореляції

За допомогою авторегресіонних рівнянь можна спробувати змоделювати часовий ряд та спрогнозувати можливе майбутнє індексу VWS.CO. Авторегресіонну модель можна представити наступною формулою:

$$X_t = c + \sum_{i=1}^{p} a_i L^i X_t + \varepsilon_t$$

Для використання авторегресіонної моделі необхідно імпортувати відповідну модель ARMA:

```
from statsmodels.tsa.arima_model import ARMA
```

Наступним кроком вказуємо модель, ім’я часового ряду та використовуючи функцію `.dropna()` видаляємо пусті рядки в даних:

```
mod=ARMA(df_monthly['vws_price'].dropna(),
order=(1, 0));
res=mod.fit();
res.summary()
```

Після чого можемо відобразити графік з спрогнозованим значенням. Для цього у функції `plot_predict` вказуємо дату, з котрої буде будуватись

графік нашого часового ряду, та кінець. Завантаженні дані VWS.CO були з 01.01.2013 по 31.10.2020, ми ж спробуємо спрогнозувати, які показники будуть у 01.2021.

```
res.plot_predict(start=0, end="2021-1");
```

З чого отримуємо наступний часовий ряд:

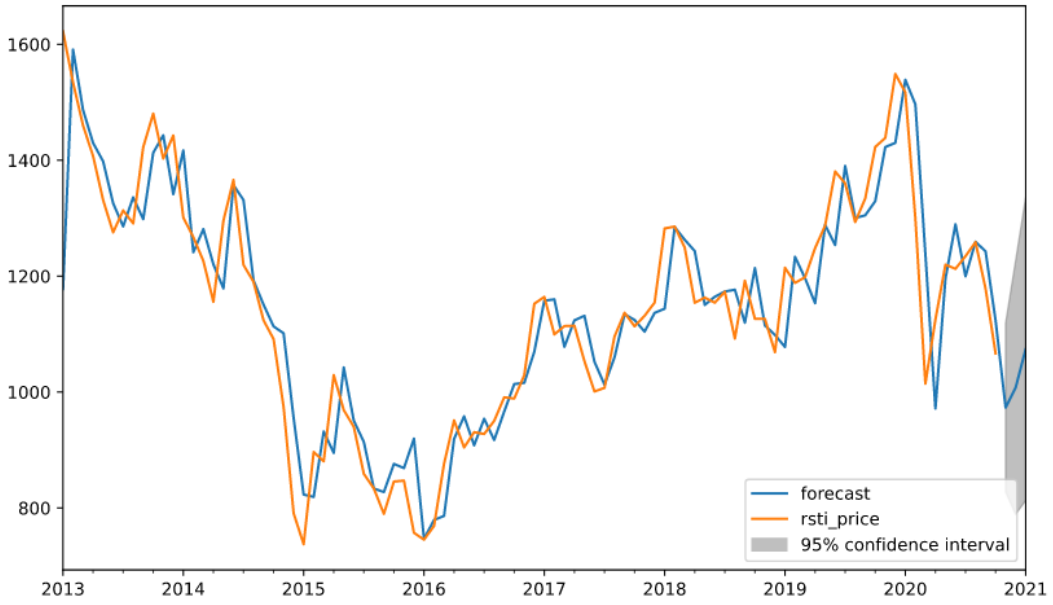

Рис 3.3 приклад використання авторегессії для прогнозування

На графіку можна побачити, що за перший місяць 2021 року прогнозується приріст значень часового ряду.

3.2 Теоретичні відомості

Для створення магістерської роботи необхідно якомога більше дізнатись про основні складові MLP мережі та більш детально розглянути алгоритм прогнозування.

Під часовим поряд розуміються послідовно змінені через деякі (частіше за все рівні) зміни часу дані. Наприклад, люди протягом довгого періоду вимірювання температуру повітря в деякому місці, та отримали часовий ряд. Основне, що можна робити з часовими рядами - це прогнозування їх подальшого значення. Прогноз – це завжди вірогідне, але при цьому науково обґрунтоване судження про можливі майбутні стани об'єкта чи явища, і в

ньому завжди є доля помилок. У кращому випадку ця помилка випадкова, в поганому – систематична.

Крім цього на тимчасових рядах можна тестувати різні гіпотези: чи є між двома часовими рядами зв'язок, чи є вони стаціонарними, чи ні. Стаціонарність – це властивість процесу не міняти свої характеристики з часом, тобто якщо в часовому ряду є якийсь тренд, він вже не стаціонарний.

Передбачення цільової змінної зазвичай ґрунтується не на сторонніх ознаках, а на значеннях самої себе за попередній часовий період.

Функції активації використовуються для визначення спрацювання нейронів у мережі. Враховуючи лінійну комбінацію вхідних даних та ваг з попереднього шару, функція активації керує тим, як ми передаємо цю інформацію на наступний шар. Існують такі типи функції активації: тотожна, функція двійкового кроку, логістична функція, TanH, ArcTan, Softsign, ISRU, випрямлена лінійна функція, ReLU, PReLU, RReLU та багато інших.

Обрана функція ReLU – тип функції активації, яка використовується для вирівнювання лінійного блоку. Математично функція (рисунок 3.4) визначається як $y = \max(0, x)$. Візуально це виглядає наступним чином:

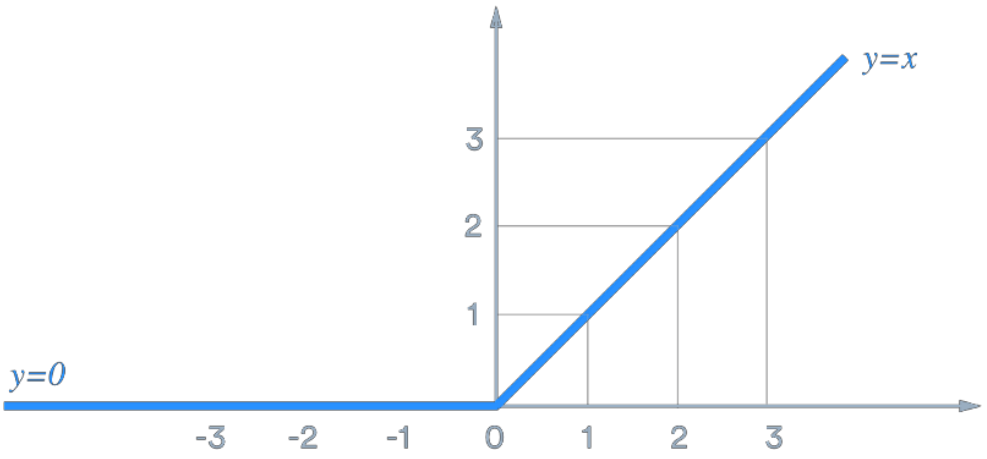

Рис 3.4 Графік функції ReLU

ReLU являється найбільш часто використовуваною функцією активації у нейронних мережах, особливо у загорткових нейронних мережах.

Існує дві варіації функції ReLU такі як, ReLU з витоком та параметричний ReLU. Варіант з витоком, замість нуля, має невеликий нахил для від'ємних значень. Наприклад, вона може мати $y = 0.01x$, тоді як $x < 0$.

У іншому випадку, параметричний ReLU (рисунок 3.5) – це тип ReLU з витоком, котрий замість вказаного нахилу, такого як 0.01, робить його параметром для нейронної мережі. Наприклад: $y = ax$, тоді як $x < 0$.

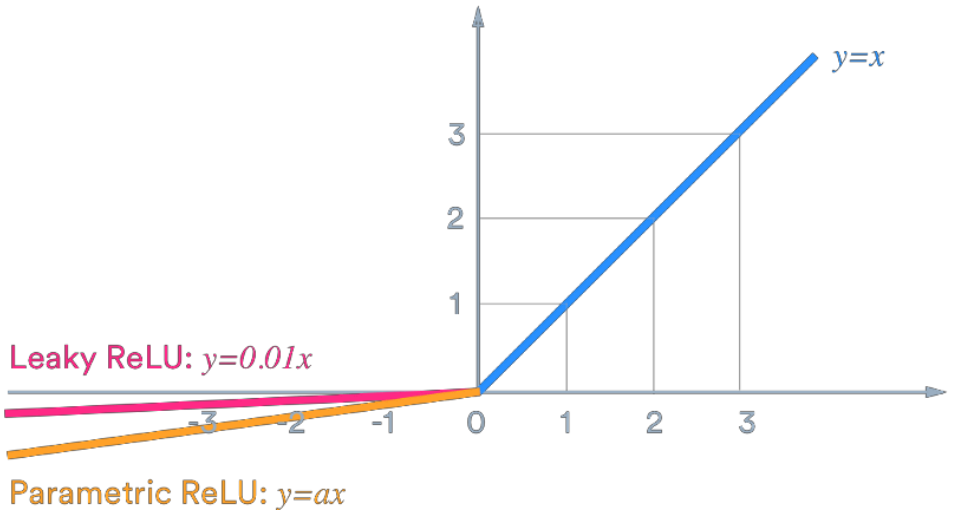

Рис 3.5 Графік функції параметричного ReLU

Функція ReLU з витоком має дві переваги:

- Вона виправляє проблему "вмираючого ReLU", так як у нього немає частин з нульовим нахилом.
- Це прискорює навчання. Існують докази того, що наявність коефіцієнту "середньої активації", близької до нуля, робить навчання швидше.

Навіть розглянувши всі ці фактори, ReLU з витоком не завжди є кращим варіантом і повинен розглядатися лише як можлива альтернатива звичайному ReLU.

Для побудови вектора виводу використовується Dense Layer. Це щільний шар, що представляє собою множення матриці на вектор. Значення в матриці – це навчальні параметри, котрі оновлюються під час оберненого зміщення:

$$u^T.W, W \in R^{n \times m}$$

Таким чином можна отримати m – вимірний вектор у якості виводу. Щільний шар використовується для зміни розмірів вектора. Він використовує обертання, масштабування та перетворення до вектору.

Шар відсіву Dropout Layer, що використовується для регуляризації, у якому необхідно випадково встановлювати деякі розміри вхідного вектора рівними нулю з вірогідністю keep_prob. Випадаючий шар не має навчальних параметрів, тобто нічого не оновлюється і не змінюється при зворотному проходженні зворотного розповсюдження. Аби гарантувати, що очікувана сума векторів, що подається до цього шару, залишиться незмінною, якщо відсів не було застосовано, ті розміри, що не встановлені на нуль, масштабуються на $\frac{1}{keep\_prob}$.

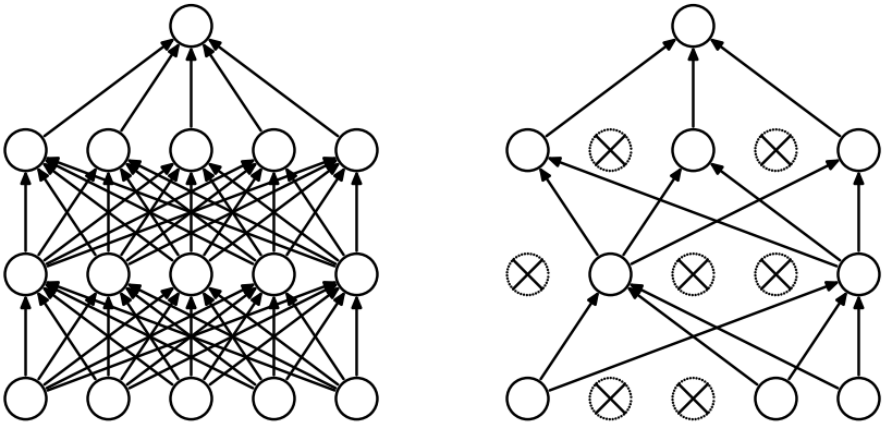

Рис 3.6 Візуалізація функції Dropout Layer

Метод нормалізації пакетів, BatchNormalization() – це метод, який використовується для того, щоб зробити штучні нейронні мережі швидшими та стабільнішими завдяки нормалізації вхідного рівня шляхом повторного центрування та повторного масштабування даних.

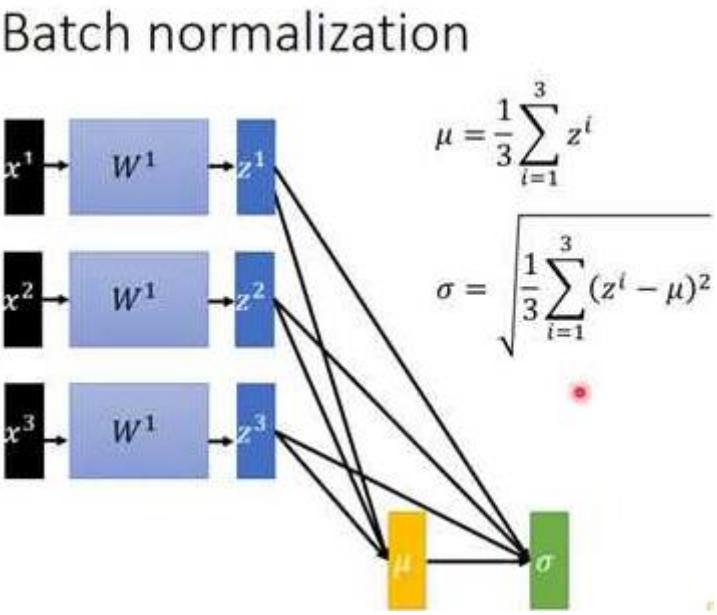

Рис 3.7 Метод нормалізації пакетів

Метод нормалцізації можна описати наступними формулами:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \text{ або } \mu_B = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu\beta)^2$$

В штучних нейронних мережах функція активації нейрона визначає вихідний сигнал, що визначається вхідним сигналом чи їх набором. Ця функція є двійковою, що означає, що нейрон або збуджується, або ж ні. Формула функції активації softmax, що використовується в даній роботі виглядає наступним чином:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

В поданій формулі активації softmax присутні такі складові:

σ – softmax;

z – вхідний вектор;

e^{z_i} – стандартна експоненціальна функція для вхідного вектора;

K – кількість класів у багатокласному класифікаторі;

e^{z_j} – стандартна експоненціальна функція для вихідного вектора;

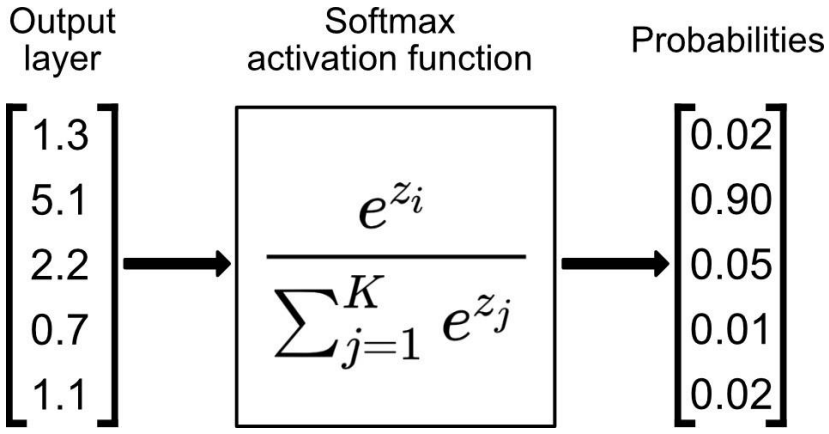

Рис 3.8 Функція активації Softmax

Функція активації Linear – функція, яка приймає вхідні дані, помножені на ваги кожного нейрона, та видає вихідний сигнал, який є пропорціональним вхідному. В деякому сенсі лінійна функція краще за ступінчату, оскільки вона допускає декілька виходів, а не лише так чи ні. Але попри всі переваги лінійна функція активації має два основних недоліка:

Неможливість використовувати градієнтний спуск для навчання моделі. Неможливо повернутися назад по алгоритму та дізнатись які ваги у нейронах можуть забезпечити кращий прогноз.

Всі шари нейронної мережі об’єднуються в один з лінійними функціями активації, незалежно від того, скільки шарів у нейронній мережі. Останній шар буде лінійною функцією першого. Нейронну мережу з лінійною функцією активації можна назвати моделлю лінійної регресії, яка здатна оброблювати складні непостійні вхідні дані.

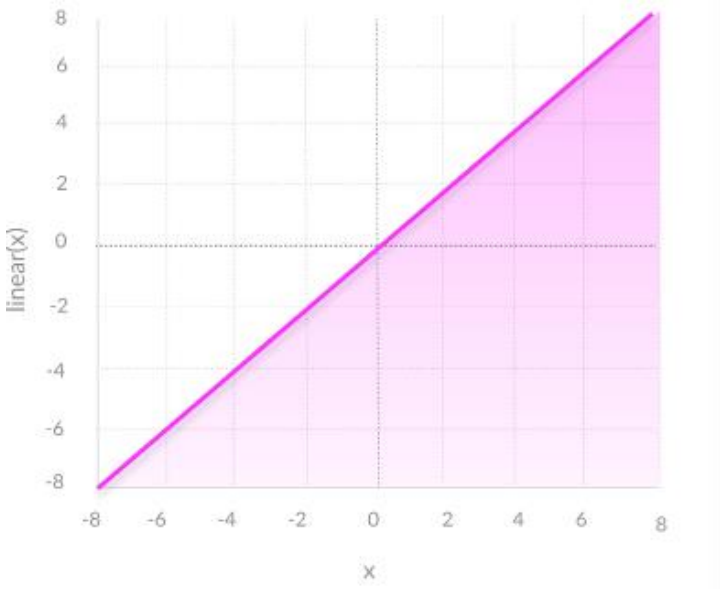

Рис 3.9 Графік лінійної функції активації

Опираючись на результати функцій для розрахунку точності навчання було встановлено, що мережа якісніше навчається на великих даних та на кількості епох, яка приблизно рівна 150-200.

3.3 Огляд функцій

Після вибору нейронної мережі та завантаження часового ряду у форматі csv, необхідно імпортувати данні. Для завантаження часового ряду використовуємо бібліотеку Pandas. Ця бібліотека дозволить привести дані у нормальний вид переконвертувавши їх у список, який необхідний для подальшого навчання мережі. Тож для початку слід імпотувати необхідні бібліотеки для побудови графіків та роботи з csv файлами:

```
import matplotlib.pylab as plt
import numpy as np
import pandas as pd
```

Присвоїмо змінній data значення, що містяться у файлі часового ряду, за допомогою функції бібліотеки Pandas, а саме pd.read_csv. Вкажемо, що будемо використовувати усі значення часового ряду, до значень AdjClose включно:

```
data = pd.read_csv('data/VWS.CO.csv')[::1]
data=data.loc[:,'AdjClose'].pct_changen().dropna().
tolist()
```

Перевіривши завантажені дані, у випадку, якщо вони віддзеркалені, слід змінити коефіцієнта pd.read_csv зі значення 1 на -1, аби розвернути значення у правильному порядку.

Аби відобразити часовий ряд на графіку, використовуємо функції бібліотеки matplotlib:

```
plt.plot(data)
plt.show()
```

Результатом функції plot є наступний графік, що має на вісі x значення дати часового ряду з 04.01.2010 по 29.11.2020, а на вісі y – значення зростання та падіння ціни компанії.

На графіку можна побачити піки падінь та приросту, які нам згодом треба буде спробувати передбачити.

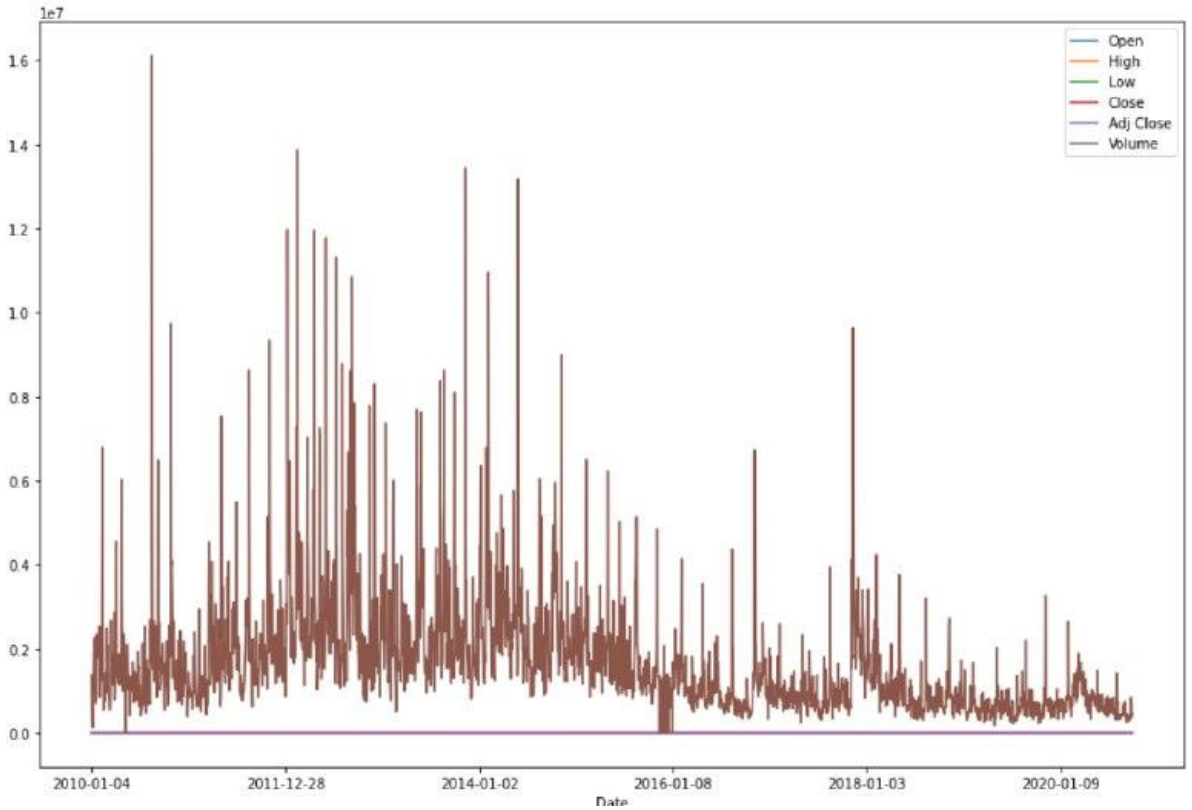

Рис 3.9 Графік часового ряду VWS.CO

Для розподілення на навчаючу та тренуючу вибірки буде взято 80% вікон задля навчання, та останні 20% – для тренування.

Задля навчання нейронної мережі буде використано дві пари X та Y, такі як ціна у момент закриття ринку та [1, 0] або [0, 1], залежно від того зросло чи впало значення ціни для бінарної класифікації.

Задача класифікації

В якості фреймворку візьмемо Keras, оскільки він відносно простий, інтуїтивно зрозумілий та у ньому можна реалізовувати доволі складні графи. Реалізуємо вхідний шар з 30 нейронами, перший прихований шар буде з 64 нейронами, після чого використаємо функцію BatchNormalization, та функцію активації LeakyReLU. На виході розміщено один нейрон, дві класифікації, котрий в залежності від задачі, класифікація чи регресія буде мати або ж softmax на виході, чи залишимо його без нелінійності, аби мати можливість прогнозувати будь яке значення. Класифікація виглядатиме наступним чином:

```
mlp_mod = Sequential()
mlp_mod.add(Dense(64, input_dim=30,
activ_regul=regularizers.l2(0.01)))
mlp_mod.add(BatchNormalization())
mlp_mod.add(LeakyReLU())

mlp_mod.add(Dense(16,
activ_regul=regularizers.l2(0.01)))
mlp_mod.add(BatchNormalization())
mlp_mod.add(LeakyReLU())
mlp_mod.add(Dense(1))
mlp_mod.add(Activation('softmax'))
```

У місці, де ми можемо отримати перенавчання, додається регулізація. При перенавчанні будується модель, яка “запам’ятовує” тренувальні дані й не дозволяє узагальнити знання на нові дані. Для цього використовується

найбільш популярний метод додаткового доданка з L2 нормою (евклідова норма). У Keras це робиться за допомогою keras.regularizers.activity_regularizer.

Аби збільшити точність без збільшення кількості похибок, необхідно скористатись популярною у останні роки технікою Dropout. Дана техніка є випадковим ігноруванням деяких ваг у процесі навчання, аби запобігти адаптацію нейронів, аби вони не навчались однаковим ознакам. Під час тестування метод Dropout не використовується.

Параметром для loss буде середня квадратична похибка – mse. Використання бібліотеки Keras дозволяє контролювати процес навчання. Непоганим рішенням можна вважати використання зменшення значення кроку градієнтного спуску в тому випадку, якщо результати не покращуються. Для цього використано ReduceLROnPlateau:

```
    reduc_lrop = ReduceLROnPlateau(monitor='val_loss',
factor=0.9,  patience=5,  min_lr=0.000001,  verbose=1)
mlp_mod.compile(optimizer=opt,
losses='categor_crossentrop',
metric=['accur'])
```

Навчання нейронної мережі відбувається після додавання до неї всіх необхідних параметрів:

```
    historys = model.fit(X_training, Y_training,
    nb_epochs = 50,
    batch_sizes = 128,
    verboses=1,
    validation_data=(X_testing, Y_testing),
    shuffle=True,
    callback=[reduce_lr])
```

У процесі навчання мережі виводиться детальна інформація відносно кількості похибок та точності навчання на кожній ітерації:

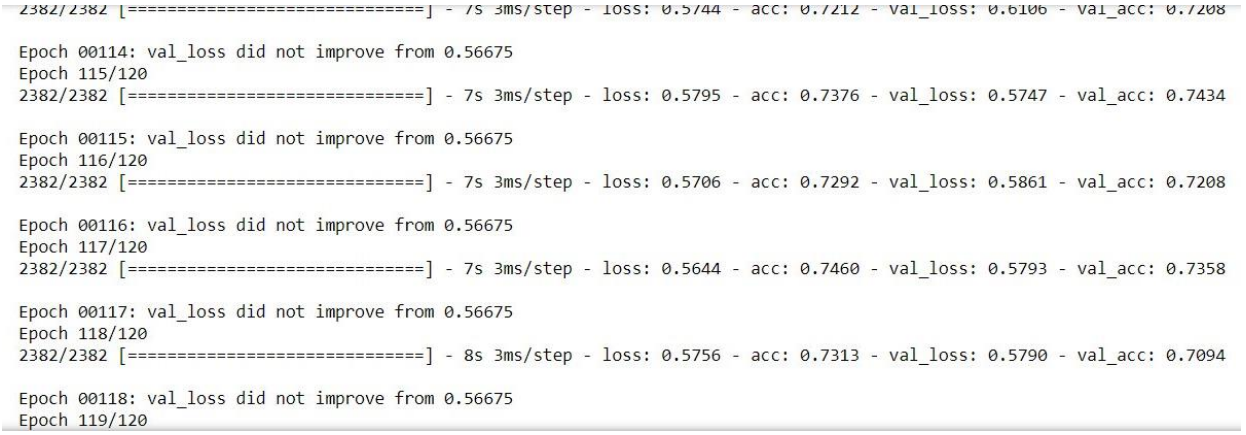

Рис 3.10 Процес навчання нейронної мережі

Після проходження всіх ітерацій навчання необхідно вивести графіки динаміки значень похибки та точності:

```
plt.figure()
plt.plot(history.history['losseing_s'])
plt.plot(history.history['vallidation_lossing'])
plt.title('m_losses')
plt.ylabel('losses')
plt.xlabel('epochs')
plt.legend(['t_dat', 't_dat'], loc='best')
plt.show()

plt.figure()
plt.plot(history.history['accuracy_s'])
plt.plot(history.history['vallidation_accuracy'])
plt.title('m_accuracy')
plt.ylabel('acc')
plt.xlabel('epochs')
plt.legend(['t_dat', 't_dat'], loc='best')
plt.show()
```

За допомогою графіків можна побачити етапи перенавчання, що вказуватиме на те, що параметри мережі треба краще налаштувати. Навчати

алогритми на таких даних необхідно робити приблизно на 50-100 епохах. Це пов’язано з тим, що якщо навчати мережу на 15-20 епохах, результатом буде, скоріш за все, точність у розмірі 55%, і це буде означати, що ми лише навчились знаходити паттерни, тобто елементи, які повторюються. Наприклад, в такому випадку, 55% паттернів будуть означати підвищення даних, а інші 45% - їх зниження.

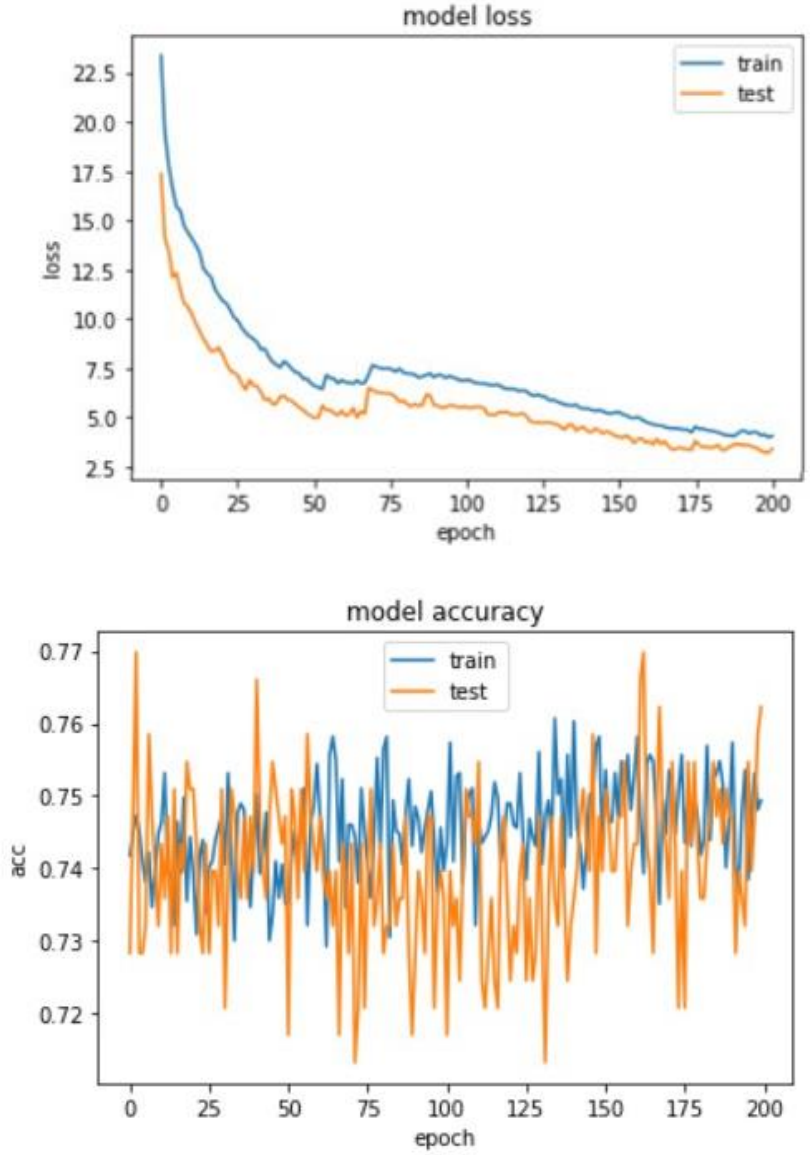

Рис 3.11 Графіки динаміки значень похибки (1) та точності (2)

З отриманих результатів можна зробити висновок, що якщо вчасно зупинити навчання, то можна отримати до 60% точності.

Задача регресії

Для задачі регресії можна взяти архітектуру, яка вже добре себе показала при класифікації, прибрати Dropout та навчити на великій кількості ітерацій. Також тепер ми можемо не просто спостерігати за результатами навчання у якості графіків похибок та точності, а й візуально оцінити якість прогнозування:

```
predict = mlp_mod.predict(np.array(test_X))
origin_d = test_Y
pred_d = predict

plt.plot(origin_d, color='black', label = 'Original')
plt.plot(pred_d, color='blue', label = 'Predicted')
plt.legend(loc='best')
plt.title('Predicted data')
plt.show()
```

Архітектура мережі виглядатиме наступним чином:

```
mlp_mod = Sequential()
mlp_mod.add(Dense(64, input_dim=30,
activ_regul=regularizers.l2(0.01)))
mlp_mod.add(BatchNormalization())
mlp_mod.add(LeakyReLU())
mlp_mod.add(Dense(16,
activity_regularizer=regularizers.l2(0.01)))
mlp_mod.add(BatchNormalization())
mlp_mod.add(LeakyReLU())
mlp_mod.add(Dense(1))
mlp_mod.add(Activation('linear'))
```

Результатом навчання мережі буде наступний графік прогнозування:

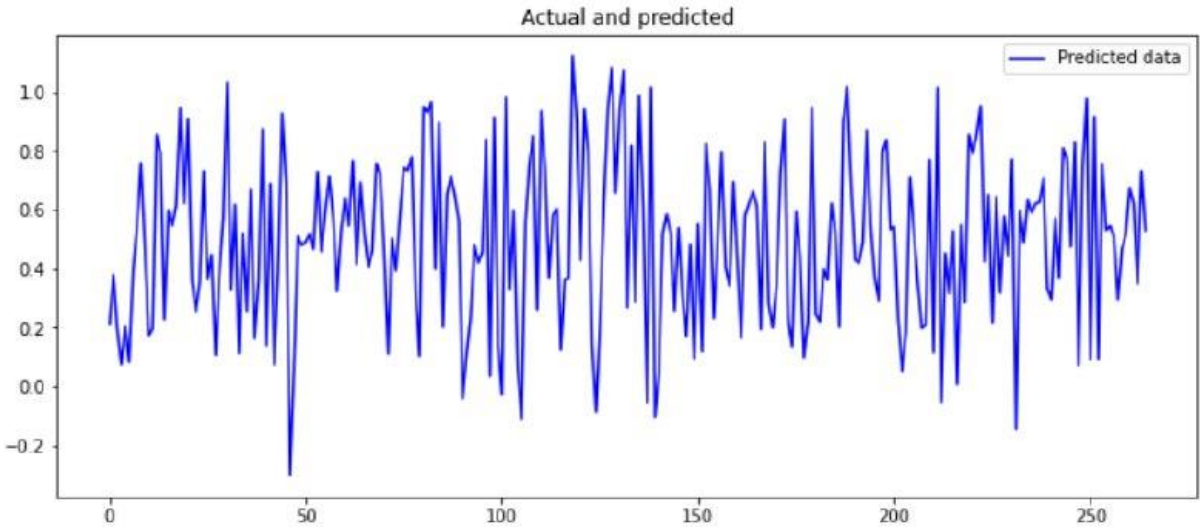

Рис 3.12 Графік прогнозування часового ряду використовуючи задачу регресії

ВИСНОВКИ ДО РОЗДІЛУ 3

Аби краще розібратися з аспектами роботи програмного засобу було розглянуто основні теоретичні відомості стосовно обраної нейронної мережі, а саме:

- ReLU
- Dense Layer
- Dropout Layer
- MaxPooling2D
- Сигмоїда (sigmoid)

Було розглянуто авторегресивний метод прогнозування часових рядів, і можна зробити висновок, що у випадку, коли необхідно спрогнозувати або проаналізувати невеликий проміжок часу на часовому ряді, цього методу може бути достатньо, оскільки для його реалізації не потрібно багато часу.

Також, був розглянутий та описаний код алгоритмів аналізу та прогнозування фінансових часових рядів за допомогою MLP мережі, починаючи від імпортування даних, закінчуючи створенням візуалізації результатів у вигляді графіків.

ВИСНОВКИ

Результатом магістерської роботи є навчена, прогнозувати динаміку часових рядів, нейронна мережа за допомогою мови програмування Python та засобу Keras. Проведена робота дає змогу зробити такі загальні висновки, що відображають вирішення завдань відповідно до поставленої мети:

1. Аби навчити нейронну мережу прогнозувати часові ряди необхідно було знайти та розглянути аналоги програм, які можуть аналізувати часові ряди та бази даних, з яких можна ці часові ряди завантажити. Були розглянуті доступні бази даних часових рядів, такі як:

- finance.yahoo
- Google Finance
- Bloomberg L.P.
- Reuters
- Factiva

Обрано базу finance.yahoo, в якій наявна велика кількість часових рядів, і взаємодія з якою значно зручніша ніж з іншими, оскільки finance.yahoo має навіть власну бібліотеку для Python.

2. Для досягнення мети було проаналізовано роботу нейронної мережі, порівняно наявні аналоги програмного засобу. Це дало змогу виділити такі переваги, як швидкість аналізу часових рядів та їх прогнозування, оптимізація процесу аналізу стійкості компаній і недоліки, як обмеженість у доступі, велика вартість. Це дозволило висунути вимоги до власного проекту з досліджень методів прогнозування часових рядів. Висунуто функціональні вимоги до власної нейронної мережі, досліджено алгоритми прогнозування та зроблені певні висновки.

3. Були проаналізовані способи прогнозування часових рядів методами штучного інтелекту та авторегресії.

4. Висунуто функціональні вимоги, які було реалізовано. Під час розрахунку акуратності обчислень, було виявлено максимальну точність

аналізу 68%, але цей відсоток можливо збільшити використовуючи більше змінних при прогнозуванні.

Досліджені у рамках магістерської роботи алгоритми прогнозування фінансових часових рядів можливо буде в майбутньому вдосконалити шляхом оптимізації та підвищення якості, швидкості, аналізу з допомогою використанні більшої кількості змінних при прогнозуванні. Практичне значення роботи полягає в тому, що проаналізовані алгоритми можливо буде використовувати як альтернатива вже наявним комерційним аналогам, які можна швидко відтворити з мінімальною витратою часу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bernard M. What Are Artificial Neural Networks [Електронний ресурс] / Marr Bernard – Режим доступу до ресурсу: https://www.forbes.com/sites/bernardmarr/2018/09/24/what-are-artificial-neural-networks-a-simple-explanation-for-absolutely-anyone/#7935cbe01245.

2. Babs T. The Mathematics of Neural Networks [Електронний ресурс] / Temi Babs – Режим доступу до ресурсу: https://medium.com/coinmonks/the mathematics-of-neural-network-60a112dd3e05.

3. Danqing L. A Practical Guide to ReLU [Електронний ресурс] / Liu Danqing – Режим доступу до ресурсу: https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7.

4. Глубокое обучение для новичков [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/wunderfund/blog/314872/.

5. Keras optimizers [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/residentmario/keras-optimizers#Adam.

6. 7 types of Artificial Neural Networks for Natural Language Processing [Електронний ресурс] – Режим доступу до ресурсу: https://www.digitalvidya.com/blog/types-of-neural-networks/

7. Франсуа Ш. Глибоке навчання на Python / Шолле Франсуа., 2018. – 400 с.

8. Нейронні мережі: їх застосування [Електронний ресурс] – Режим доступу до ресурсу: http://www.poznavayka.org/uk/nauka-i-tehnika-2/neyronni-merezhi-yih-zastosuvannya-robota/.

9. Особливості поширення збудження в ЦНС [Електронний ресурс] – Режим доступу до ресурсу: http://um.co.ua/9/9-19/9-194054.html.

10. Binary or binomial classification [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Binary_classification.

11. Geoffrey H. Improving neural networks by preventing co-adaptation of feature detectors / H. Geoffrey, K. Alex., 2012.

12. Samarasinghe S. Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition / Samarasinghe S., 2006.

13. Денисенко О. О. Решение задачи бинарной классификации при помощи свёрточных нейронных сетей [Електронний ресурс] / Олексій Олександрович Денисенко. – 2019 – Режим доступу до ресурсу: https://moluch.ru/conf/tech/archive/324/14834/

14. Khosravani H. R. Artificial neural network models: data selection and online adaptation / Khosravani H. R., 2017.

15. Nicholson C. A Beginner's Guide to Neural Networks and Deep Learning [Електронний ресурс] / Chris Nicholson. – 2018 – Режим доступу до ресурсу: http://wiki.pathmind.com/neural-network

16. Маккінні У. Python for Data Analysis. / У. Маккінні., 2015. – 482 с.

17. Бринк Х. Машинное обучение / Х. Бринк, Д. Ричардс, М. Феверолф., 2018. – 336 с.

18. Вандер Д. П. Python Data Science Handbook: Essential Tools for Working with Data / Дж. Плас Вандер., 2017. – 576 с.

19. Хейдт М. Learning pandas. / М. Хейдт., 2018. – 432 с.

20. Goodfellow I. Deep Learning / I. Goodfellow, Y. Bengio, A. Courville. – The MIT Press, 2016. – 775 с.

21. Isakov S. Как работает нейронная сеть: алгоритмы, обучение, функции активации и потери [Електронний ресурс] / Stanislav Isakov. – 2018. – Режим доступу до ресурсу: https://neurohive.io/ru/osnovy-data-science/osnovy-nejronnyh-setej-algoritmy-obuchenie-funkcii-aktivacii-i-poteri/.

22. Штучні нейронні мережі [Електронний ресурс] – Режим доступу до ресурсу: https://www.it.ua/ru/knowledge-base/technology-innovation/iskusstvennye-nejronnye-seti-ins.

23. Нейронні мережі [Електронний ресурс] – Режим доступу до ресурсу: https://neuralnet.info/book/.

24. Нейронные сети — математический аппарат [Електронний ресурс] – Режим доступу до ресурсу: https://basegroup.ru/community/articles/math.

25. Baestaens E. D. Neural Network Solution for Trading in Financial Markets / Emma Dirk Baestaens.

26. Hristev R. M. Artifical Neural Networks [Електронний ресурс] / R. M. Hristev – Режим доступу до ресурсу: https://www.twirpx.com/file/559386/.

27. Короткий С. Нейронные сети: Основные положения / С. Короткий., 2017. – 69 с.

28. Нейронні мережі програмування. Як працює нейронна мережа: алгоритми, навчання, функції активації і втрати. [Електронний ресурс] – Режим доступу до ресурсу: https://maylohack.ru/uk/windows-7/neironnye-seti-programmirovanie-kak-rabotaet-neironnaya-set.html.

29. Основи нейромереж. Бум нейромереж: Хто робить нейронні мережі, навіщо вони потрібні і скільки грошей можуть приносити [Електронний ресурс] – Режим доступу до ресурсу: https://passportbdd.ru/uk/tehnologii/osnovy-neirosetei-bum-neirosetei-kto-delaet-neironnye-seti/.

30. Нейронні мережі: їх застосування, робота [Електронний ресурс] – Режим доступу до ресурсу: https://www.poznavayka.org/uk/nauka-i-tehnika-2/neyronni-merezhi-yih-zastosuvannya-robota/.