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.

Методи глибокого навчання в дослідженні алгоритмів стійкого розвитку фінансового ринку

Магістерська робота студента
групи Ім-15
ступінь вищої освіти «магістр»
спеціальності 014 Середня освіта (Інформатика)
Тарасенка Андрія Олексійовича

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

За останні роки технології нейронних мереж стрімко завойовують своє місце в повсякденному житті людей, починаючи від розумних систем взаємодії з користувачами, закінчуючи дослідженнями в різних наукових сферах. Якщо ще років 5 – 10 назад можна було говорити про успіхи в сфері досліджень штучного інтелекту, то на сьогоднішній день нейронні мережі обслуговують мільйони користувачів в мережі інтернет чи займають місце в додатках смартфонів та інших пристроях.

Нейронні мережі вже вміють аналізувати людську мову, виділяти ознаки об’єктів на зображеннях чи відео, реконструювати пошкоджені дані та в певній мірі робити прогнози на майбутнє виходячи з минулих явищ.

Прогнозування явищ у майбутньому являє собою одну з найцікавіших сфер дослідження машинного навчання, адже, нейронні мережі моделюють певну роботу головного мозку людини, а люди здатні, іноді вірно, зробити певний прогноз явищ, якщо добре в ньому розбираються, отже, можливо змоделювати штучний інтелект подібно людському, щоб в мати змогу програмно зробити прогнозування.

В даній роботі проводиться дослідження можливості прогнозування фінансових часових рядів методами машинного навчання.

Часовий ряд – це ряд даних, які проіндексовано в хронологічному порядку.

Отже, актуальність задач прогнозування часових рядів має високу цінність в діяльності компаній різних предметних областей, починаючи від прогнозування погоди, закінчуючи цінами на метали, рідини тощо

Об’єкт дослідження: алгоритми машинного навчання.

Предмет дослідження: прогнозування ціни на фондовому ринку.

Метою даної роботи є аналіз алгоритмів машинного навчання для оцінки та прогнозу майбутніх значень фондових індексів.

Для досягнення мети слід проаналізувати та розв’язати такі задачі:

- Проаналізувати принцип роботи нейронних мереж;
- Порівняти існуючі реалізації програмних засобів;
- Спроектувати структуру нейронної мережі;
- Навчити спроектовану структуру прогнозувати майбутню ціну;
- Здійснити аналіз результатів.

Практичне значення роботи полягає в оцінюванні існуючих алгоритмів прогнозування та їх застосування в прикладному програмному забезпеченні.

РОЗДІЛ 1. АНАЛІЗ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ ТА ЧАСОВИХ РЯДІВ.

В 50-х роках минулого століття, коли область інформатики тільки зароджувалася, люди все задавалися питанням, а чи можливо змусити комп'ютери «думати»? І хоча, від моменту зародження самої ідеї пройшло відносно багато часу, лише завдяки науковому прориву та збільшення потужностей комп'ютерів ми можемо побачити та використовувати плоди зароджених ідей майже століття тому. Коротко можна розуміти область штучного інтелекту(ШІ), як автоматизація інтелектуальних задач.

Спочатку інженери думали, що ШІ можна створити, якщо дати програмісту достатню кількість правил, такий підхід був названий ***символічним штучним інтелектом.*** Проте, такий ШІ міг вирішувати лише певні чіткі, нескладні задачі, проте зовсім не підходив для складніших задач.

Пізніше, символічний ШІ замінив більш повий підхід – *машинне навчання.*

1.1 Основні поняття машинного навчання та штучних нейронних мереж.

Ідея машинного навчання виникла тоді, коли постало питання: чи можливо без програмування навчити комп'ютер? Чи можливо не задавати правила для вирішення задачі, а самостійно їх знаходити без чіткого програмування?

Якщо підхід символічного програмування базується на тому, що людина задає правила вирішення задачі, то підхід машинного навчання вирішує саме проблему навчання, де замість правил програміст дає комп'ютеру дані та «вірну відповідь», тобто, очікуваний вірний набір даних. Тоді, комп'ютер

здатен на знаходження певних ознак та правил, що вирішує питання про самонавчання комп’ютера.

Нейронна мережа - математична модель, а також її Програмне або апаратне втілення, побудована за принципом організації та функціонування біологічних нейронних мереж — мереж нервових клітин живого організму.

Машинне навчання - клас методів штучного інтелекту, характерною рисою яких є не пряме рішення задачі, а навчання в процесі застосування рішень безлічі подібних завдань.

Машинному навчанню потрібні такі складові даних:

- Дані для тренування, тобто, дані на яких система має навчатися, виділяти ознаки тощо;
- Приклади вірних результатів, тоді ознаки виділені системою будуть перевірятися на вірних даних;
- Способи оцінки якості моделі, тобто, для оцінки роботи та відхилення результатів роботи системи від вірного результату.

Глибоке навчання – розділ машинного навчання, що грунтується на алгоритмах високорівневих абстракцій представлення даних.

В глибокому навчанні головною ідеєю створення програми для вирішення задач є моделювання роботи людського мозку. Так, як і людський мозок має систему з шарів(рис. 1.2) зв’язаних між собою нейронних клітин(рис. 1.1), так і моделі глибокого навчання спроектовані, як послідовність з ***штучних моделей людських нейронів***(рис.1.3), які об’єднуються в певні шари клітин і утворюють собою ***глибину*** нейронних мереж.

Отже, глибоке навчання має на меті моделювання складних природних нейронних систем в їх штучні математичні моделі.

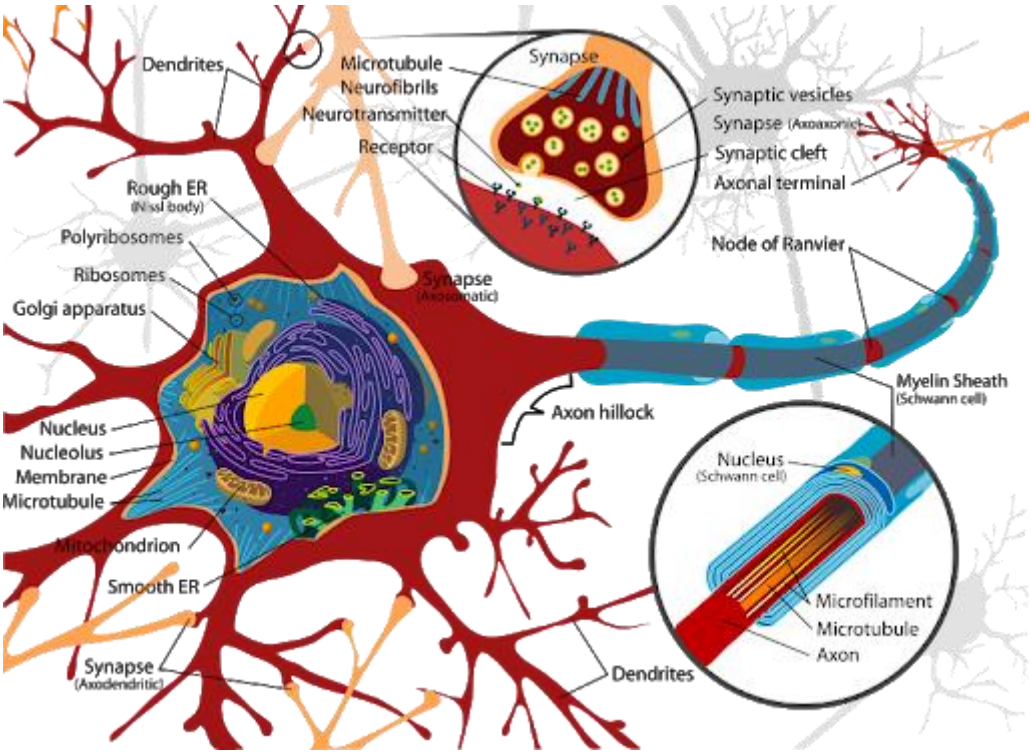

Рис. 1.1 Природна клітина мозку людини.

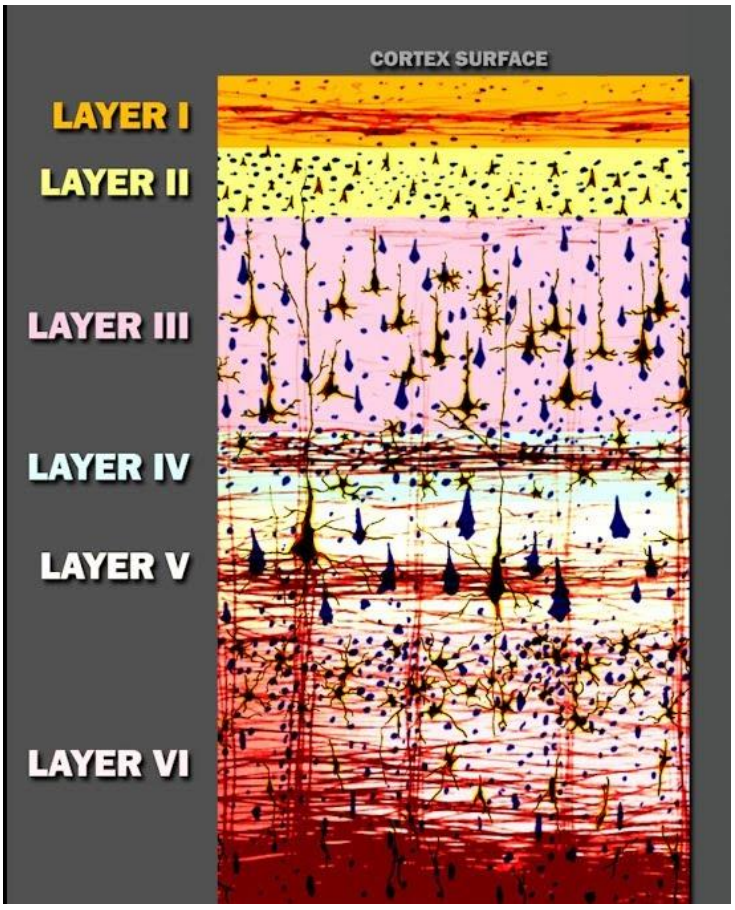

Рис. 1.2 Представлення шарів нейронів людського мозку.

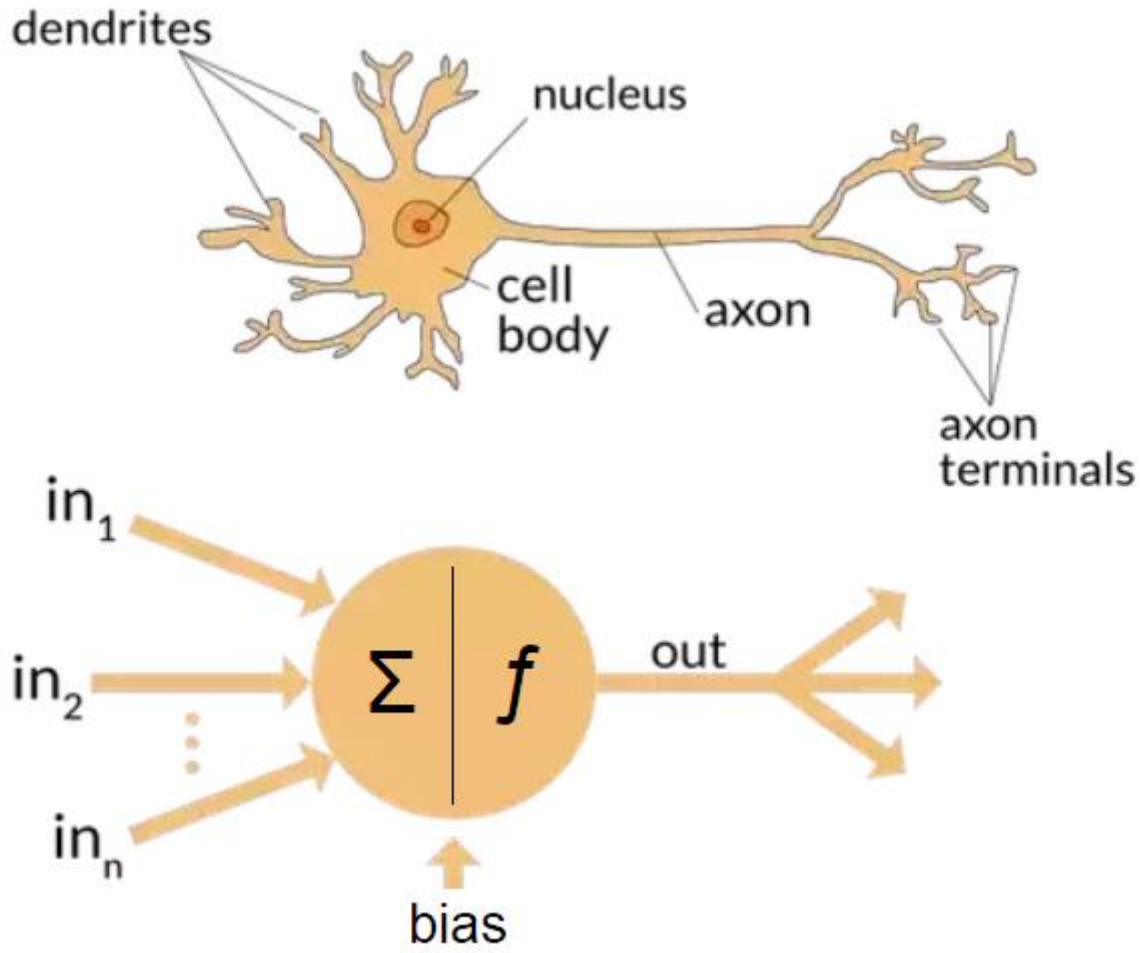

Рис. 1.3 Порівняння моделі штучного нейрону з природним.

$In_1 \ldots in_n$ – являє собою інформацію, яка потрапляє до нейрону. Якщо в корі головного мозку така інформація представлена електричними імпульсами, то штучні моделі можуть працювати лише з числовим представленням даних.

∑ - модель способу нейронів приймати інформацію, представляє собою математичний запис:

$$\sum = w_i * in_i + b,$$

де w_i – ваги входу даних для in_i

f – функція активації, що моделює збудження природного нейрону

b –показник зміщення.

Якщо люди здатні до навчання, а області штучного інтелекту досягли успіхів у моделювання природних систем мозку, тоді й навчання штучних моделей можливо представити аналогічно навчанню природних моделей.

Для підготовки мережі до навчання потрібно налаштувати параметри:

- Функція втрат – задає як саме мережа має оцінювати якість своєї роботи на навчальних даних, після чого корегувати параметри;
- Оптимізатор – механізм за допомогою якого мережа буде оновлювати свої параметри опираючись на функцію втрат та власні спостереження;

Без функції активації, такої як relu (рис. 1.4) (також званої фактором нелінійності), sigmoid(рис. 1.5), шар Dense буде складатися з двох лінійних операцій-скалярного добутку і складання.

Такий шар зможе навчатися тільки на лінійних (афінних) перетвореннях вхідних даних: простір гіпотез шару було б сукупністю всіх можливих лінійних перетворень вхідних даних в 16-мірний простір. Такий простір гіпотез занадто обмежено, і накладення декількох шарів уявлень один

на одного не приносило б ніякої вигоди, тому що глибокий стек лінійних шарів все одно реалізує лінійну операцію: додавання нових шарів не розширює простору гіпотез. Щоб отримати доступ до більш великого простору гіпотез, що дає додаткові вигоди від збільшення глибини уявлень, необхідно застосувати нелінійну функцію, або функцію активації. Функція активації relu-найпопулярніша в глибокому навчанні, однак на вибір є ще кілька функцій активації з трохи дивними на перший погляд іменами: prelu, elu і т. д.

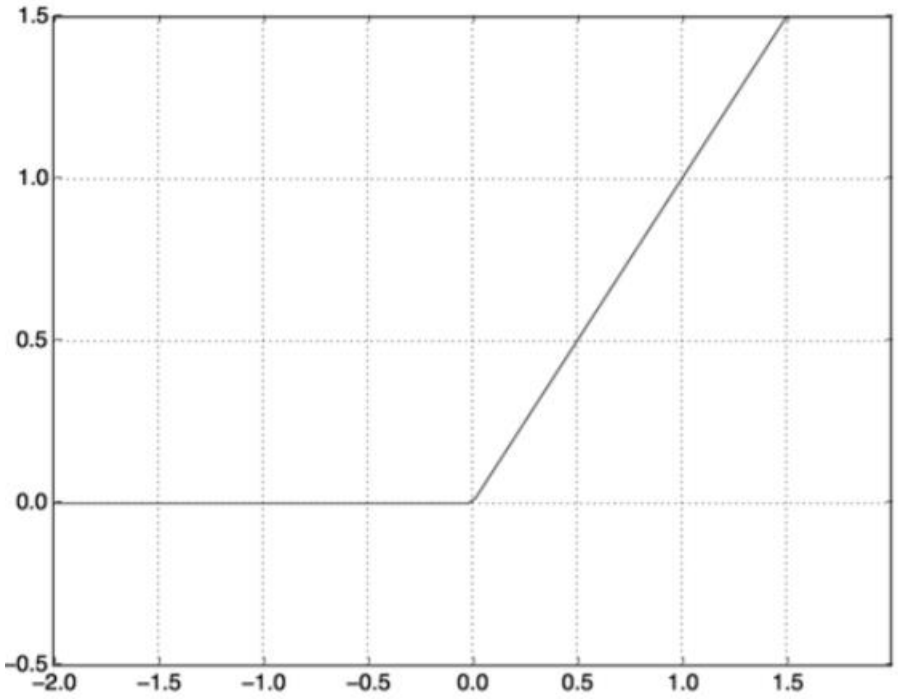

Рис. 1.4 Функція активації ReLu.

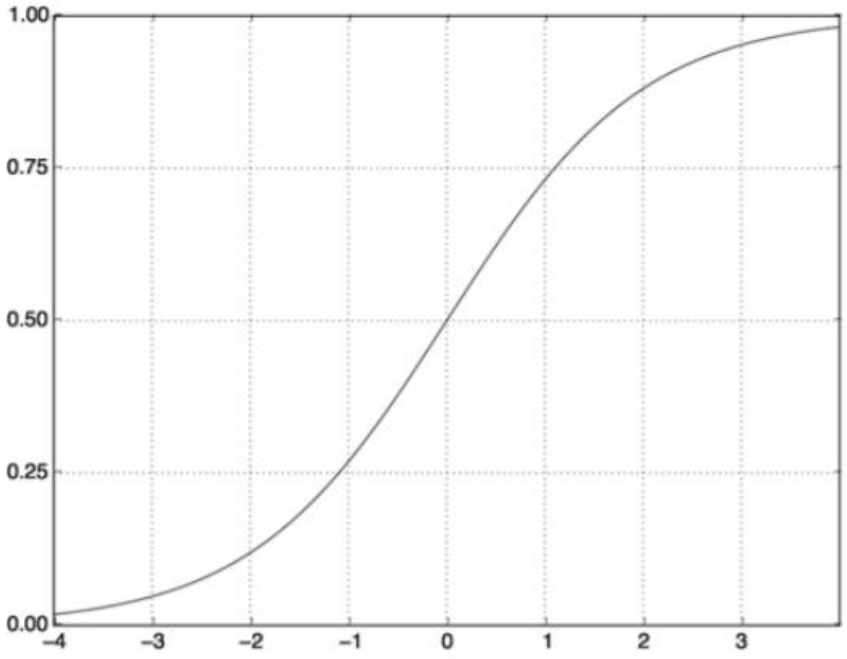

Рис. 1.5 Сигмоїдальна функція.

Представлення даних для нейронних мереж.

Одиницею даних для нейронних мереж називають ***тензором.***

Тензор в простому розумінні – це масив числових даних. Від вимірності масиву залежить поняття ***порядку тензору***. Якщо масив даних нульовий, а саме – число(скаляр), тоді й тензор нульового порядку.

Вектор – послідовність скалярів, утворюють тензор першого рангу.

Двовимірна матриця – тензор другого порядку.

Тензор визначається трьома ключовими атрибутами:

- Кількість осей — (ранг) — наприклад, тривимірний тензор має три осі, а матриця-дві. У бібліотеках для Python, таких як Numpy, цей атрибут тензорів має ім'я ndim.
- Форма-кортеж цілих чисел, що описують кількість вимірювань на кожній осі тензора. Наприклад, матриця в попередньому прикладі має форму (3, 5), а тривимірний тензор має форму (3, 3, 5). Вектор має форму з єдиним елементом, наприклад (5,), тоді як скаляр має порожню форму ().
- Тип даних (зазвичай в бібліотеках для Python йому дається ім'я dtype) - це тип даних, що містяться в тензорі; наприклад, тензор може мати тип float32, uint8, float64 і ін. У рідкісних випадках можна зустріти тензори типу char. Зверніть увагу, що в Numpy (і в більшості інших бібліотек) відсутні рядкові тензори, тому що тензори зберігаються в заздалегідь виділених, безперервних сегментах пам'яті і рядки, будучи сутностями з мінливою довжиною, перешкоджають використанню такої реалізації.

Щоб було зрозуміліше, перерахуємо кілька прикладів тензорів з даними,

які можуть зустрітися вам в майбутньому. Дані, якими вам доведеться маніпулювати, майже завжди будуть ставитися до однієї з наступних категорій:

- векторні дані — двовимірні тензори з формою (зразки, ознаки);
- часові ряди або послідовності-Тривимірні тензори з формою (зразки, мітки, ознаки);
- зображення-чотиривимірні тензори з формою (зразки, висота, ширина,

- колір) або з формою (зразки, колір, висота, ширина);
- відео-п'ятимірні тензори з формою (зразки, кадри, висота, ширина, колір) або з формою (зразки, кадри, колір, висота, ширина).

Часові ряди або послідовності

Кожен раз, коли час (або поняття послідовної впорядкованості) грає важливу роль у ваших даних, такі дані краще зберігати в тривимірному тензорі з явною віссю часу. Кожен зразок може бути представлений як послідовність векторів (двовимірних тензорів), а сам пакет даних-як тривимірний тензор.

Відповідно до угод, вісь часу завжди є другою віссю (віссю з індексом 1). Розглянемо кілька прикладів:

- Набір даних з цінами акцій. Кожну хвилину ми зберігаємо поточну ціну акцій, а також найбільшу і найменшу ціни за минулу хвилину. Тобто кожна хвилина представлена тривимірним вектором, весь торговий день-двовимірним тензором з формою (390, 3) (де 390-тривалість торгового дня у хвилинах), а дані за 250 днів-тривимірним тензором з формою (250, 390, 3). В даному випадку кожен зразок представляє дані за один торговий день.
- Набір даних з твітами, де кожен твіт кодується послідовністю з 280 символів з алфавіту зі 128 унікальними символами. В даному випадку кожен символ можна закодувати як двійковий вектор зі 128 елементами (містить нулі у всіх елементах, крім елемента з індексом, відповідним номеру символу в алфавіті, в який записується 1). При такій організації кожен твіт можна представити як двовимірний тензор з формою (280,128), а набір з мільйоном твітів — як тензор з формою (1000000, 280, 128).

Навчання з учителем(рис.1.6)**.**

Спосіб навчання нейронних мереж, коли дані мають мітку правильного результату, нейронна мережа виділяє ознаки на вірних даних та застосовує їх вже на реальних задачах.

Якщо результат не вірний, то нейронна мережа корегує свої значення. Цей процес продовжується до закінчення навчання.

Прикладом такого виду навчання є нейронні мережі, що розпізнають об’єкти. Вони застосовують виділені ознаки під час навчання на відео чи зображеннях, які раніше не потрапляли до мережі.

Навчання без вчителя(рис.1.7).

Вид навчання, коли нейронна мережа вчиться самостійно виділяти ознаки на даних без визначеної структури. Наприклад, прогнозування покупок в магазинах.

Навчання з підкріпленням(рис.1.8).

Один із алгоритмів глибокого навчання, в ході якого випробувана система навчається, коли взаємодіє з деяким середовищем.

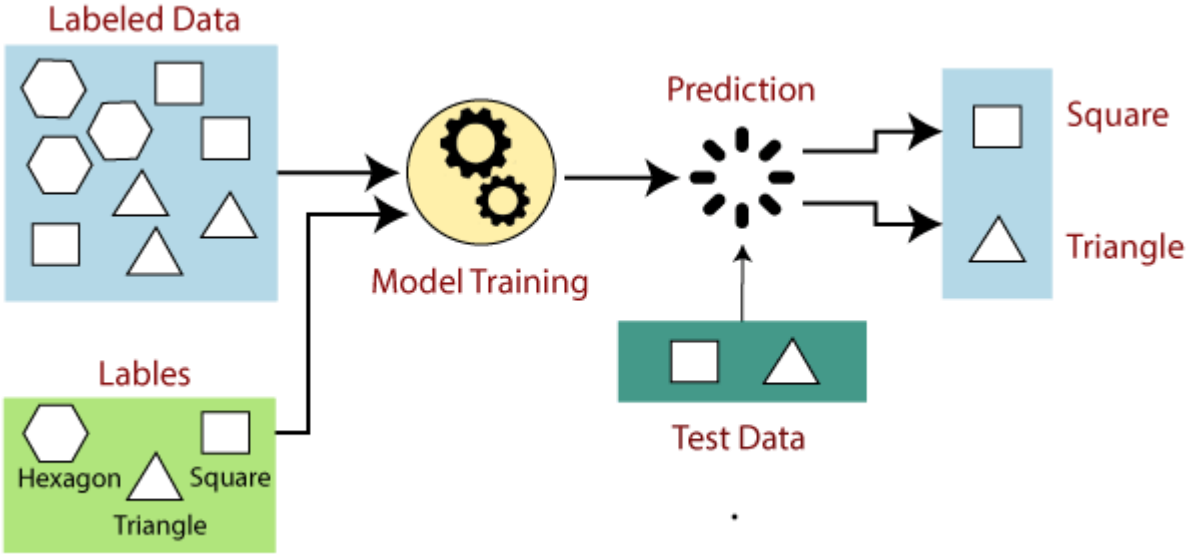

Рис.1.6 Принцип навчання з вчителем.

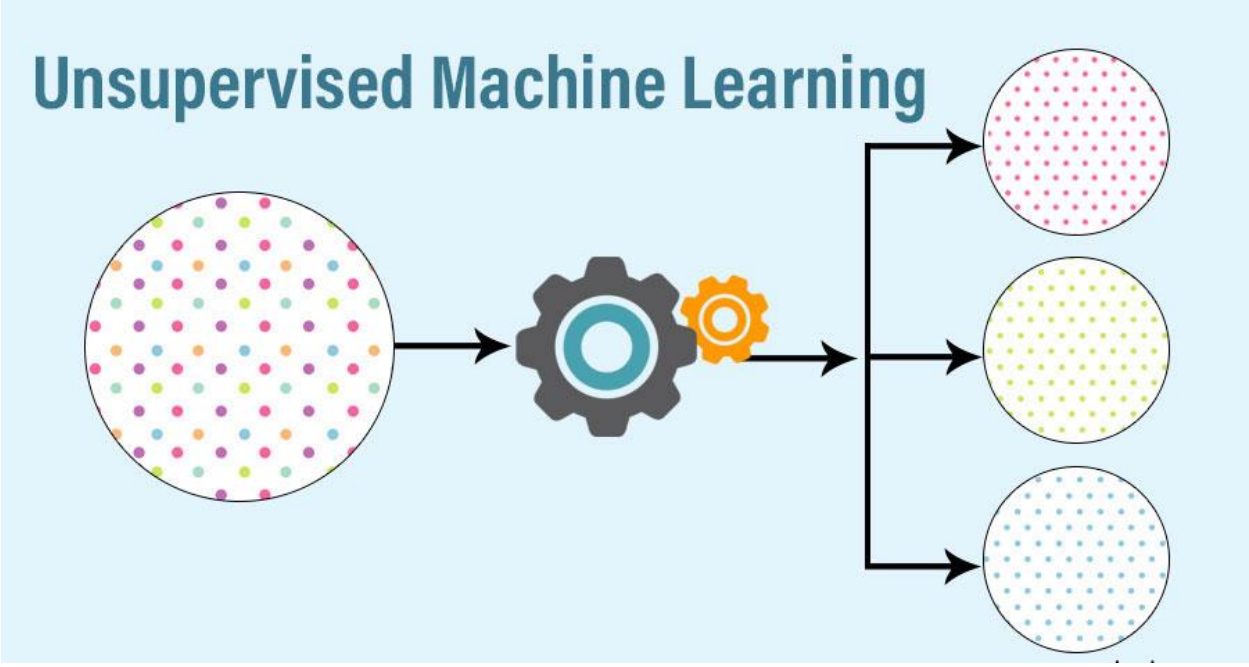

Рис.1.7 Навчання без вчителя.

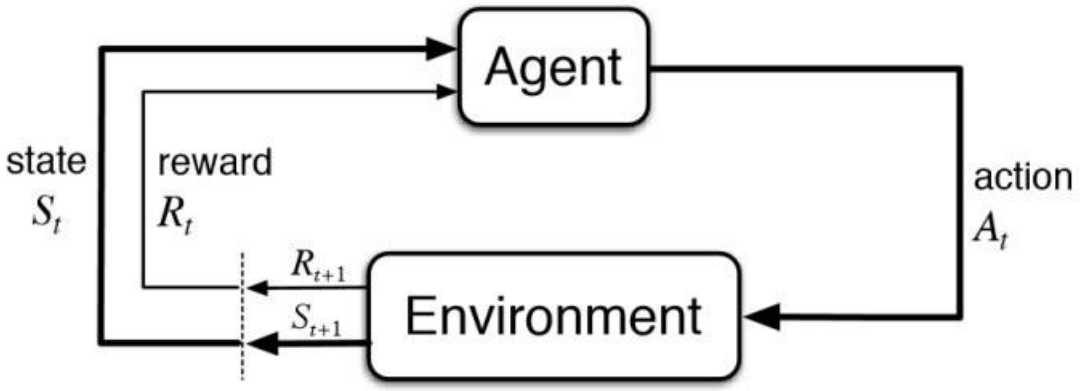

Рис.1.8 Навчання з підкріпленням.

1.2 Основні поняття часових рядів.

Часові ряди – це послідовність даних у хронологічному порядку. В часових рядах порядок спостереження є ключовим джерелом додаткової інформації яку слід обробити та проаналізувати.

Часові ряди можуть мати декілька значень, що міняються з часом. Такі ряди називаються одновимірним часовим рядом, якщо більше, то – багатовимірним часовим рядом.

Часові ряди використовуються в багатьох сферах таких, як фінанси, дані про певні події чи у будь-якій області, де спостерігаються зміни певних процесів у часі.

Під час аналізу часового ряду, отримується певний набір характеристик для того, щоб його зрозуміти. Проводячи аналіз часового ряду можна зробити біль точний прогноз його поведінки. Сам процес аналізу часового ряду зводиться до використання методів аналізу часового ряду, які намагаються зрозуміти саме природу даного ряду.

Прогнозування часового ряду включає в себе підбирання моделі на основі історичних даних, що являє собою навчальний набір даних та використання його для прогнозування майбутніх спостережень.

Детермінований часовий ряд.

Детермінований часовий ряд - це той, який може бути явно виражений аналітичним виразом. У нього немає випадкових або ймовірнісних аспектів. У математичному плані це можна описати точно за весь час в термінах розкладання в ряд Тейлора за умови, що всі його похідні відомі в деякий довільний момент часу. Його минуле і майбутнє повністю визначаються цінностями цих похідних в той час. Якщо це так, то ми завжди можемо передбачити його майбутню поведінку і вказати, як він поводився в минулому.

Недетермінований часовий ряд.

Недетермінований часовий ряд-це той, який не може бути описаний аналітичним виразом. У нього є певний випадковий аспект, який перешкоджає його поведінці бути описаним явно. Часовий ряд може бути недетермінованим, тому що:

Вся інформація, необхідна для її явного опису, недоступна, хоча в принципі може бути.

Природа процесу генерації за своєю природою випадкова.

Оскільки недетерміновані часові ряди мають випадковий аспект, він слідує імовірнісним законам Таким чином, дані визначаються статистичними термінами, тобто розподілами ймовірностей і середніми значеннями різних форм, таких як середні значення і дисперсії.

Стаціонарні часові ряди.

Стаціонарний часовий ряд-це такий, статистичні властивості якого, такі як середнє значення, дисперсія, автокореляція і т.д., не залежать від часу. Стаціонарний ряд відносно легко передбачити: ви просто прогнозуєте, що його статистичні властивості в майбутньому будуть такими ж, як і в минулому. Таким чином, більшість статистичних методів прогнозування засновані на припущенні, що часові ряди є приблизно стаціонарними.

Більшість статистичних методів прогнозування припускають, що ряди можуть бути (приблизно) стаціонарними за допомогою математичних перетворень.

Нестаціонарні часові ряди.

Нестаціонарний ряд - це той, статистичні властивості якого змінюються з часом. Існує безліч способів нестаціонарності часових рядів, таких як зміна дисперсії, зрушення рівнів, сезонність в 6-й момент і т. д. ось найбільш поширені моделі нестаціонарності:

Трендовий компонент: тенденція показує загальну тенденцію збільшення або зменшення даних протягом тривалого періоду часу. Тенденція-це плавна, загальна, Довгострокова, Середня тенденція. Не завжди необхідно, щоб збільшення або зменшення було в одному і тому ж напрямку протягом даного періоду часу. Якщо часовий ряд не показує зростаючу або спадаючу модель, то ряд в середньому є стаціонарним.

Циклічний компонент: будь-який патерн, що показує рух вгору і вниз навколо даного тренда, ідентифікується як циклічний патерн. У циклічному

патерні руху вгору і вниз не відбуваються через постійні проміжки часу, їх неможливо передбачити.

Сезонна складова: якщо піки і провали серії відбуваються через рівні проміжки часу, модель називається сезонної (наприклад, продажу морозива).

Випадковий компонент: залишок-це те, що залишилося, коли всі шаблони були видалені. Залишки-це випадкові коливання. Ви можете думати про них як про компонент шуму.

1.3 Визначення моделей нейронних мереж.

Для розв’язання різноманітних задач існує свій алгоритм, який підходить найкраще. Такі алгоритми були реалізовані в таких структурах нейронних мереж:

- Згорткова нейронна мережа(рис. 1.9). Нейронна мережа згорткового типу представляє собою послідовність з шарів прихованих нейронів, які знаходяться в певній кількості в кожному з шарів. Основною метою застосування згорткових мереж є задачі розпізнавання зображень, класифікація;

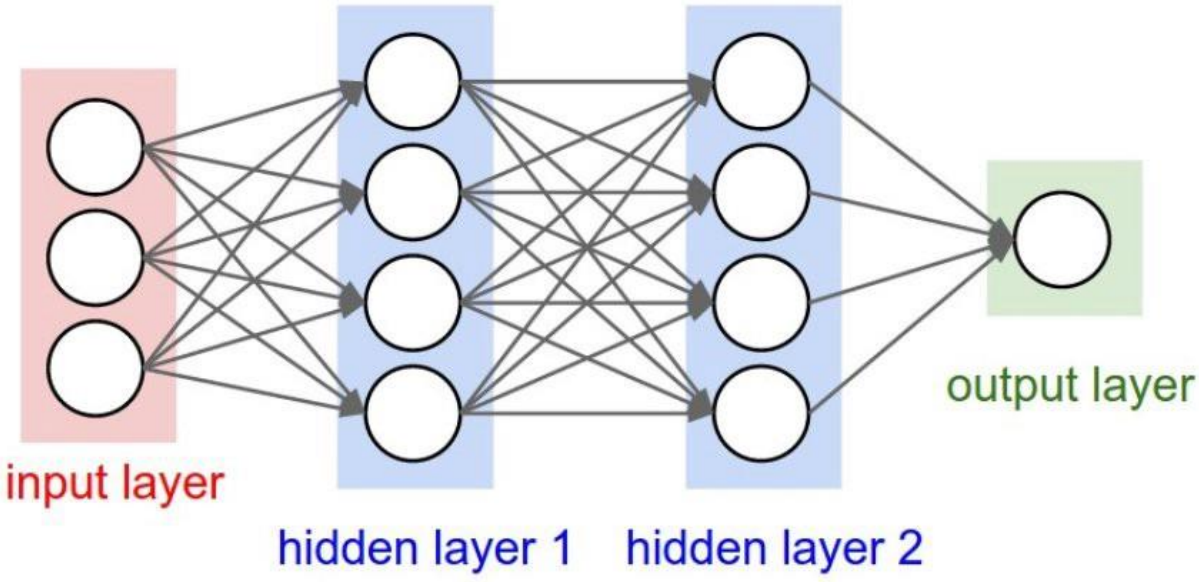

Рис.1.9 Схема згорткової нейронної мережі.

- Рекурентна нейронна мережа(рис.1.10). Клас рекурентних нейронних мереж представляє собою послідовність нейронів які мають властивість «пам’яті». Кожен нейрон на етапі навчання приймає а вхід не лише певне вагове число, а й вихід та ваги з попереднього нейрону і так від початку до кінця;

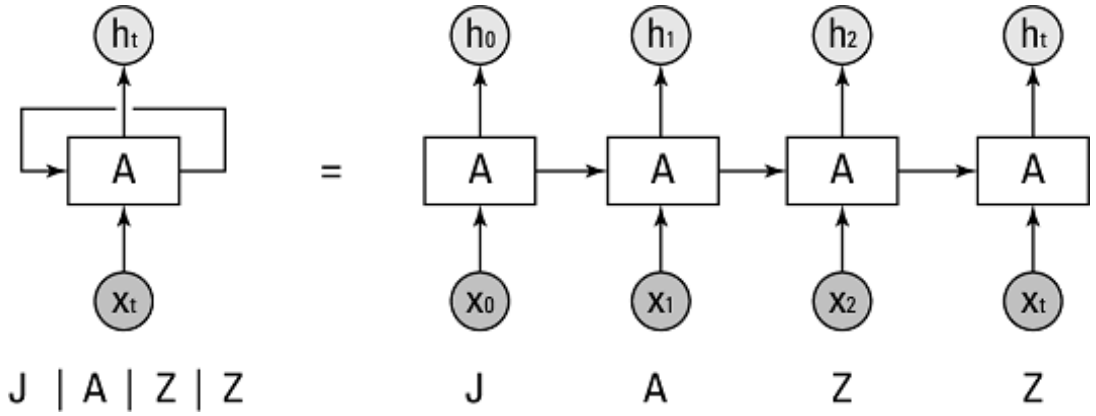

Рис.1.10 Схема рекурентної нейронної мережі.

- Багатошаровий персептрон (MLP)(рис.1.11). За архітектурою схожий на згорткові нейронні мережі, проте дозволяє класифікувати не лінійно розподілені дані. Добре підходить для задач розпізнавання голосу;

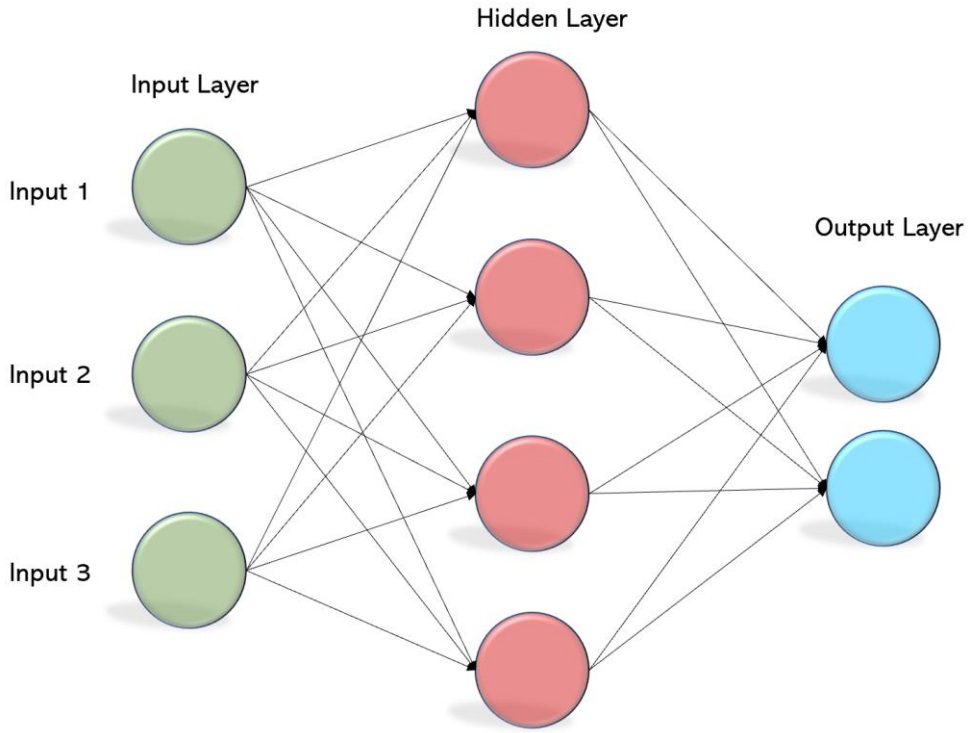

Рис.1.11 Багатошаровий персептрон.

- Модель рекурентної мережі з довготривалою-короткостроковою пам’яттю(LSTM)(рис.1.12). Спеціально розроблена для прогнозування часових рядів з досить великою кількістю точок даних.

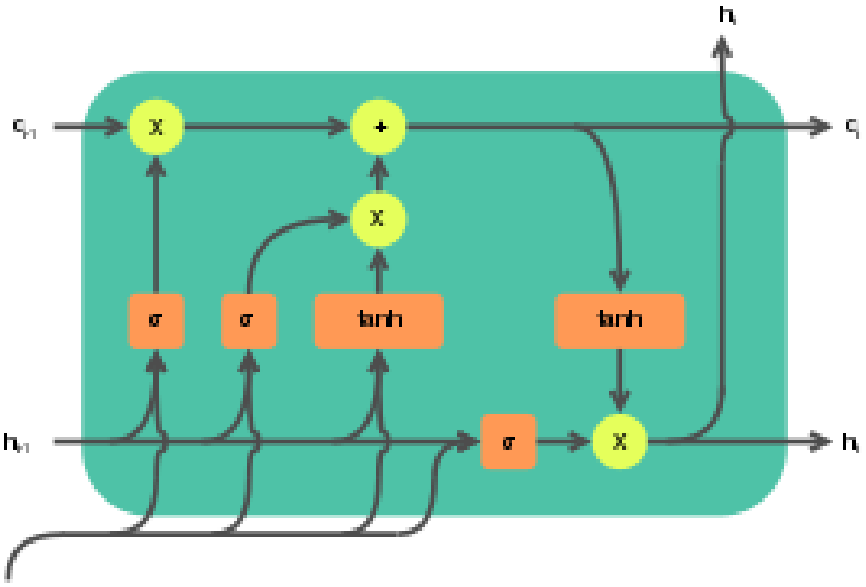

Рис.1.12 Модель LSTM.

1.4 Аналіз існуючих програмних реалізацій.

Option-lab(рис.1.13) - це система інтернет-трейдингу, справжня лабораторія по створенню і торгівлі опціонними стратегіями. Система працює на основі клієнт-серверної технології. Сервер знаходиться в дата-центрі Московської біржі. Клієнтський термінал встановлюється на комп'ютер трейдера, що дозволяє торгувати зі свого рахунку через Інтернет, використовуючи всі технічні переваги DMA доступу на терміновий ринок Московської біржі. Option-lab допоможе:

- легко автоматизувати свою торговельну систему;
- успішно торгувати опціонами і волатильністю;
- формулювати опціонний портфель будь-якої складності за допомогою конструктора опціонних стратегій, який дає можливість

створювати, аналізувати, розробляти варіанти управління опціонною конструкцією.

Рис.1.13 Логотип програми Option-lab.

AmiBroker(рис.1.14)-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.1.14 Скріншот програми AmiBroker.

MoneyWiz(рис.1.15) - Дуже функціональний додаток для управління фінансами. Воно має не тільки мобільні, але і десктопні версії для Windows і Mac. MoneyWiz - саме наворочений додаток в цьому списку: в ньому близько 600 функцій! Є підтримка інтернет-банкінгу, Синхронізація торгів і облік вкладень в цінні папери і криптовалюта, можливість створення звітів і фінансових прогнозів — загалом, це повноцінний інструмент бухобліку. Правда, за все це пишність доведеться платити, оскільки можливості безкоштовної версії Програми сильно урізані.

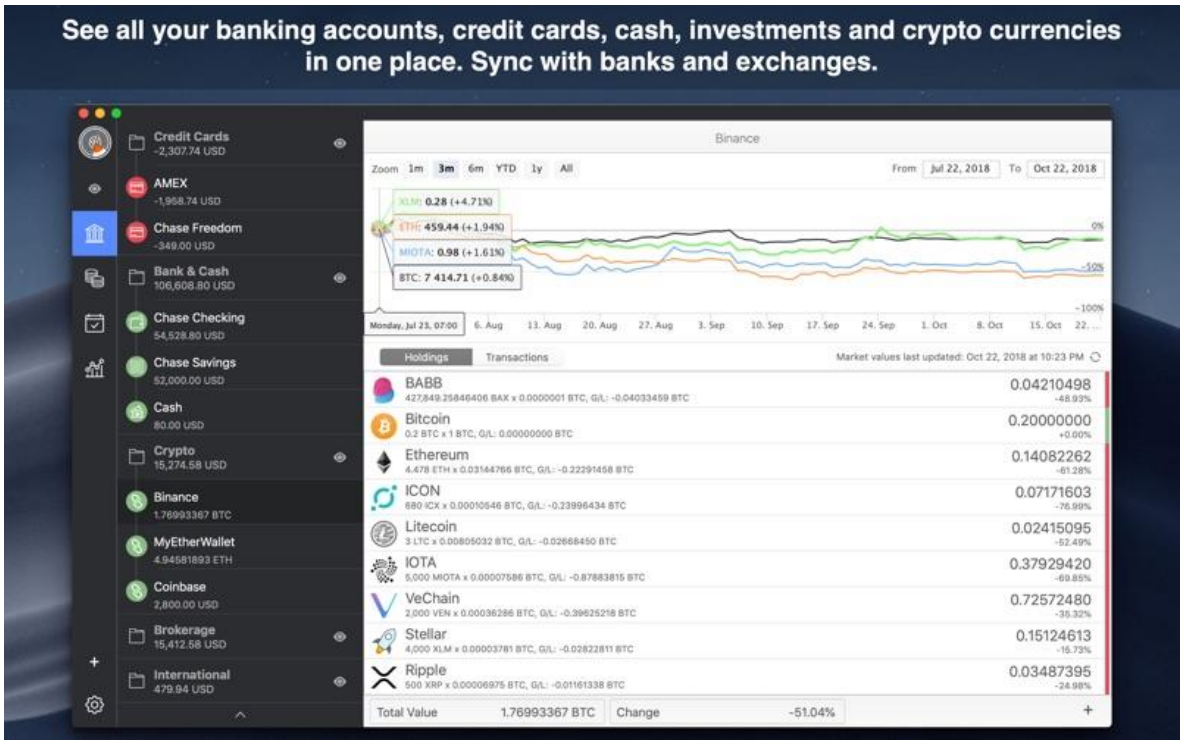

Рис.1.15 Скріншот програми MoneyWiz.

MetaTrader 4 одна з найпоширеніших серед трейдерів і брокерів платформ. Admiral Markets була однією з перших компаній, що запропонували цю платформу трейдерам, і з тих пір жодного разу не пошкодувала про це! MetaTrader 4 перевершують інші платформи за багатьма параметрами, проте головні його переваги-швидкість, надійність і зручність.

На відміну від інших платформ, які існували всього пару років і мали невелику аудиторію, MT4 на ринку вже більше 10 років і має величезну кількість користувачів. Так що платформа витримала випробування часом і тепер регулярно вдосконалюється.

MetaTrader 4 дуже популярний серед користувачів Windows, він доступний на безлічі версій: від XP до Windows 10. Давайте дізнаємося більше про особливості цієї чудової платформи.

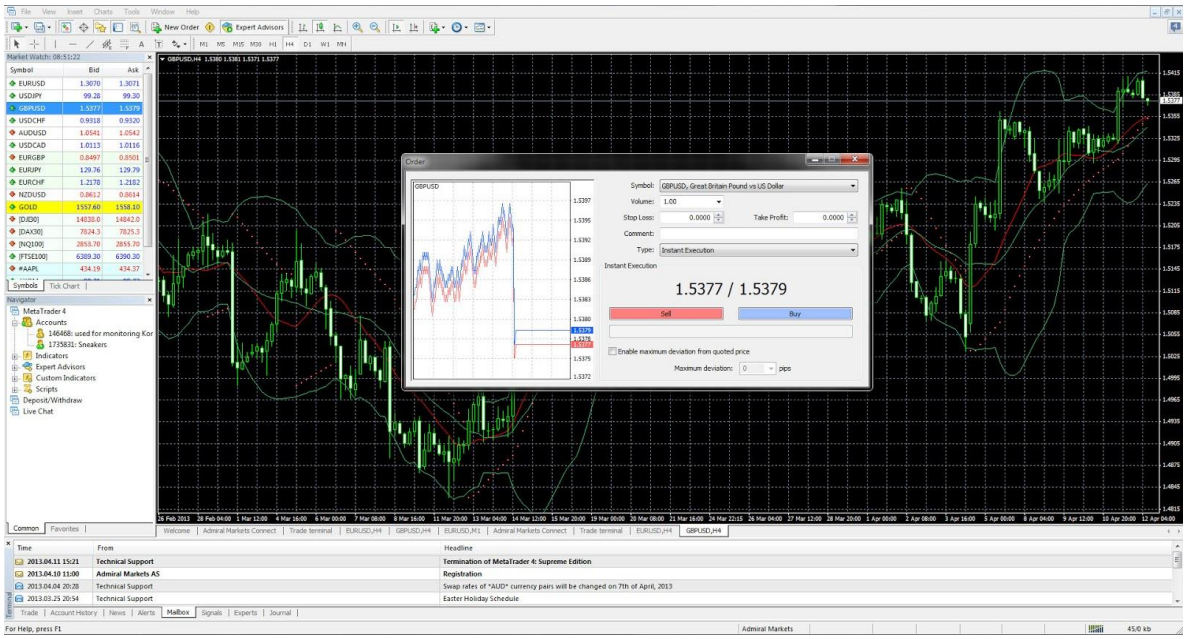

Рис. 1.16 Скріншот програми MetaTraider4.

ВИСНОВКИ ДО РОЗДІЛУ 1

Починаючи за 50-х років минулого століття, ідея навчання комп’ютера перейшла від риторичного питання, до використання в наших гаджетах різного розміру та призначення. Нейронні мережі знайшли своє місце в багатьох сферах людського життя, починаючи від пошуку інформації в мережі інтернет, до застосування як інструмент для дослідників у синтезі речовин, розпізнавання різних видів захворювань та лікування.

Нейронні мережі – спрощені моделі людського мозку. Окремі класи нейронних мереж здатні вирішувати задачі які не виходять за рамки самої моделі, тому для різних задач ми потребуємо застосування різних видів штучних нейронних мереж.

Нейронні мережі добре підходять для задач:

- Класифікації;
- Машинного зору;
- Розпізнавання мови;
- Обробка текстових даних;
- Оптимізація обчислень;
- Прогнозування явищ майбутнього.

До задач прогнозування відносять алгоритми рекурентних нейронних мереж, а саме їх різновиди – LSTM або GRU.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ЗАДАЧІ ТА ОБГРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТІВ ДОСЛІДЖЕННЯ.

З самого початку зародження ідей штучного інтелекту було розроблено досить багато алгоритмів навчання, моделей мереж, оптимізації, проте, через не достатню кількість розрахункових потужностей та ресурсів, реалізувати алгоритми в повній мірі вдається лише на протязі останніх років десяти. Алгоритми, які було розроблено ще в минулому столітті пройшли випробування часом та якістю і в наш час вони потерпіли мінімальні зміни і широко застосовуються в різних програмних продуктах.

2.1 Вибір засобів розробки.

Обираючи мову програмування для реалізації нейронних мереж слід вернути увагу на такі аспекти:

- Призначення мови програмування. Кожна мова програмування використовується різних сферах, отже, набір можливостей кожної з мов в певній мірі відрізняється від інших;
- Інструментарій. Реалізувати нейронні мережі, зв’язки, оптимізацію можливо майже в будь-якій мові програмування, але набір можливостей конкретної мови може бути більш зручним та ефективним ніж в інших;
- Світова спільнота. Навколо кожної з мов програмування утворюється своя спільнота розробників, розмір спільноти та досвід задає швидкість та якість розвитку самої мови та інструментів розробки.

З самого піднесення розвитку машинного навчання, біля десяти років тому, мова програмування Python(рис. 2.1) була обрана світовою спільнотою для дослідження штучних нейронних мереж, реалізації алгоритмів моделей мереж, навчання, оптимізації.

Рис.2.1 Офіційний логотип мови Python.

Для мови python світова open source спільнота розробила багато інструментів для дослідження роботи та поведінки алгоритмів нейронних мереж та штучного інтелекту.

З початку активного розвитку глибокого навчання було розроблене спеціальне середовище для досліджень нейронних мереж – IPython notebooks серед яких Juputer notebook(рис. 2.2) та Google Colaboratory(рис.2.3).

Середовище активно використовується для швидкого програмування, виводу результатів та побудови різних зображень, графіків, будь-якого графічного представлення даних.

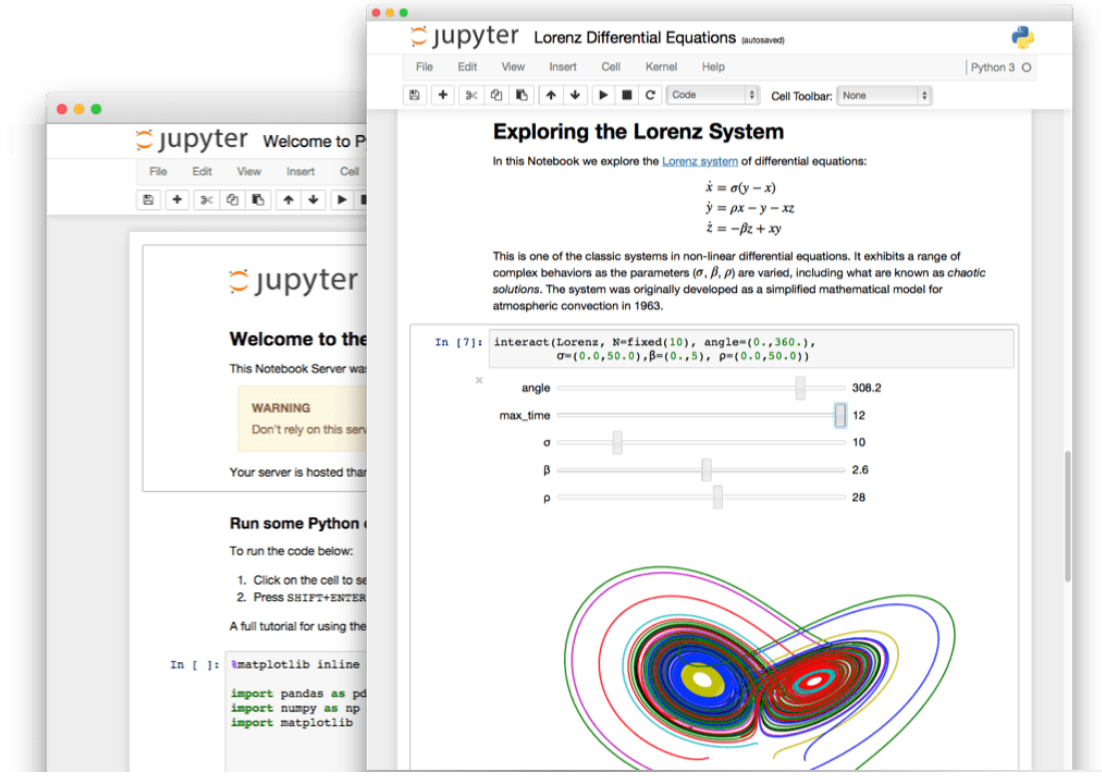

Рис.2.2 Скріншоти Jupiter Notebook.

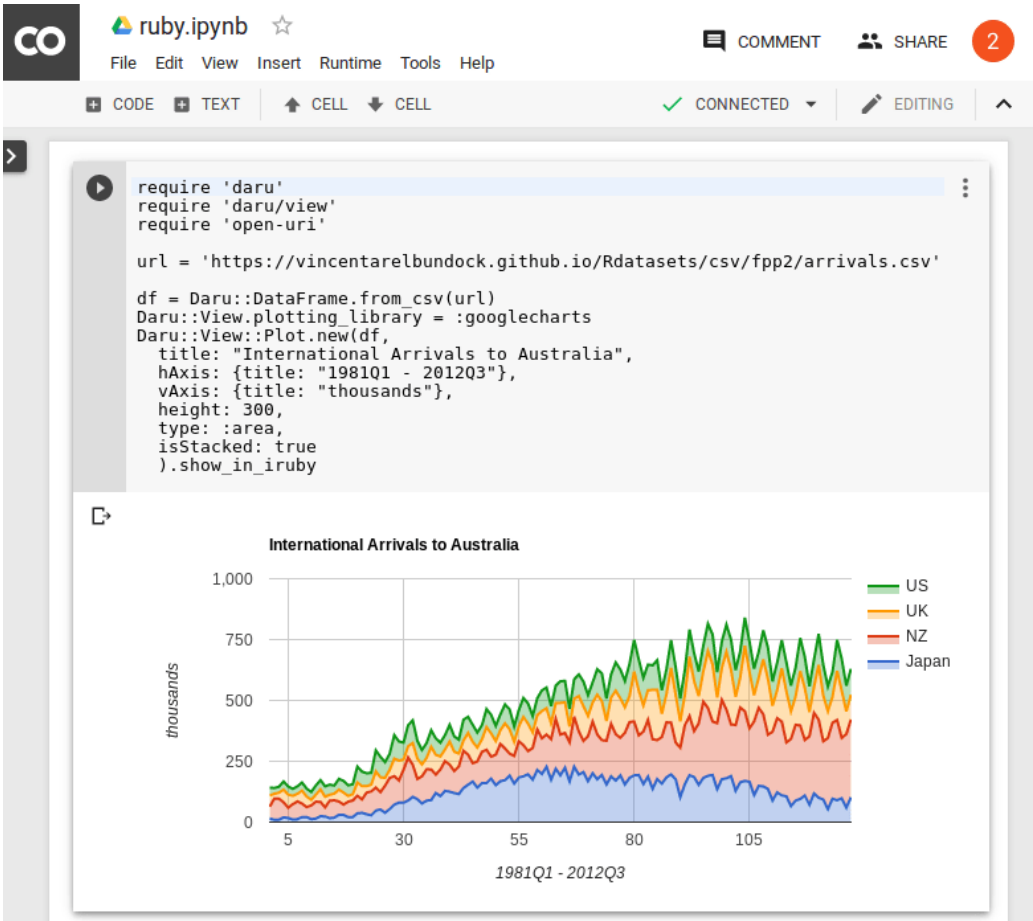

Рис.2.3 Скрішнот Google Colaboratory.

Для Python реалізовано велика кількість алгоритмів машинного навчання в таких програмних інструментах як:

- TensorFlow(рис.2.4) – безкоштовний набір для розробки будь-яких продуктів в області штучного інтелекту. Бібліотека вважається найкращим в світі інструментом для машинного та глибокого навчання;

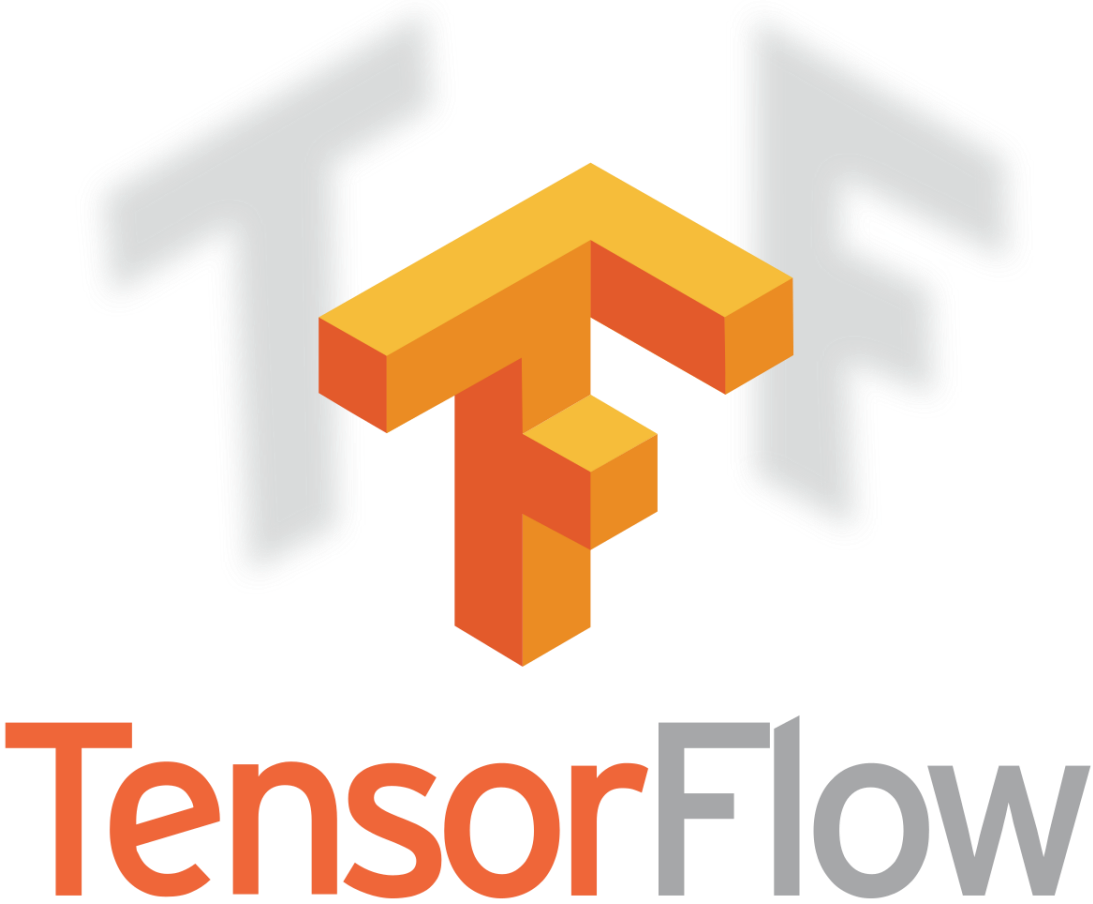

Рис.2.4 Логотип TensorFlow.

- Keras(рис.2.5) – відкрита нейромережева бібліотека, написана на мові Python. Вона являє собою надбудову над фреймворками Deeplearning4j, TensorFlow і Theano. націлена на оперативну роботу з мережами глибинного навчання, при цьому спроектована так, щоб бути компактною, модульною і розширюваною. Вона була створена як частина дослідницьких зусиль проекту ONEIROS (англ. Open-ended Neuro-Electronic Intelligent Robot Operating System), а її основним автором і підтримуючим є Франсуа Шолле (фр. François Chollet), інженер Google. Планувалося що Google буде підтримувати Keras в

основній бібліотеці TensorFlow, проте Шолле виділив Keras в окрему надбудову, так як згідно концепції Keras є скоріше інтерфейсом, ніж наскрізною системою машинного навчання. Keras надає високорівневий, більш інтуїтивний набір абстракцій, який робить простим формування нейронних мереж, незалежно від використовуваної в якості обчислювального бекенда бібліотеки наукових обчислень.

Рис.2.5 Keras як інструмент TensorFlow.

Обрано бібліотеку Keras як засіб реалізації нейронної мережі досягнення мети у прогнозуванні часового ряду.

Використаємо також допоміжні бібліотеки для роботи з даними:

- Matpljtlib(рис.2.6) – бібліотека на мові програмування Python для візуалізації даних двовимірної (2D) графікою (3D графіка також підтримується). Одержувані зображення можуть бути використані в якості ілюстрацій в публікаціях. Matplotlib написаний і підтримувався в основному Джоном Хантером (англ. John Hunter) і поширюється на умовах BSD-подібної ліцензії. Зображення, що генеруються в різних форматах, можуть бути використані в інтерактивній графіці, в наукових публікаціях, графічному інтерфейсі користувача, веб-додатках, де потрібна побудова діаграм (англ. plotting).

Рис.2.6 Популярний інструмент візуалізації даних.

- Pandas(рис.2.7) – програмна бібліотека на мові Python для обробки та аналізу даних. Робота pandas з даними будується поверх бібліотеки NumPy, що є інструментом нижчого рівня. Надає спеціальні структури даних і операції для маніпулювання числовими таблицями і часовими рядами. Назва бібліотеки походить від економетричного терміна “панельні дані”, що використовується для опису багатовимірних структурованих наборів інформації. pandas поширюється під новою ліцензією BSD. Основна область застосування-забезпечення роботи в рамках середовища Python не тільки для збору і очищення даних, але для завдань аналізу і моделювання даних, без перемикання на більш специфічні для статобробки мови (такі, як R і Octave). Також активно ведеться робота з реалізації "рідних" категоріальних типів даних. Пакет насамперед призначений для очищення і первинної оцінки даних за загальними показниками, наприклад середнім значенням, квантилям і так далі; статистичним пакетом він в повному сенсі не є, проте набори даних типів DataFrame і Series застосовуються в якості вхідних в більшості модулів аналізу даних і машинного навчання (SciPy, Scikit-Learn та інших).

Рис.2.7 Популярний засіб для обробки даних.

- NumPy(2.8) - бібліотека з відкритим вихідним кодом для мови програмування Python. Можливості: підтримка багатовимірних масивів (включаючи матриці); підтримка високорівневих математичних функцій, призначених для роботи з багатовимірними масивами. Математичні алгоритми, реалізовані на інтерпретованих мовах (наприклад, Python), часто працюють набагато повільніше тих же алгоритмів, реалізованих на компільованих мовах (наприклад, Фортран, C, Java). Бібліотека NumPy надає реалізації обчислювальних алгоритмів (у вигляді функцій і операторів), оптимізовані для роботи з багатовимірними масивами. В результаті будь-який алгоритм, який може бути виражений у вигляді послідовності операцій над масивами (матрицями) і реалізований з використанням NumPy, працює так само швидко, як еквівалентний код, що виконується в MATLAB.

Рис.2.8 Визнаний в світі засіб для роботи з масивами в Python.

2.2 Джерела та представлення даних.

Для навчання нейронних мереж застосовують набори даних, чим більші такі набори, тим краще навчиться нейронна мережа.

Одним з найбільш популярних та зручних є онлайн платформа Yahoo Finance(рис.2.9). За допомогою спеціального API дозволяє отримати будь-який наявний документ з даними цін на акції, матеріали, рідини тощо.

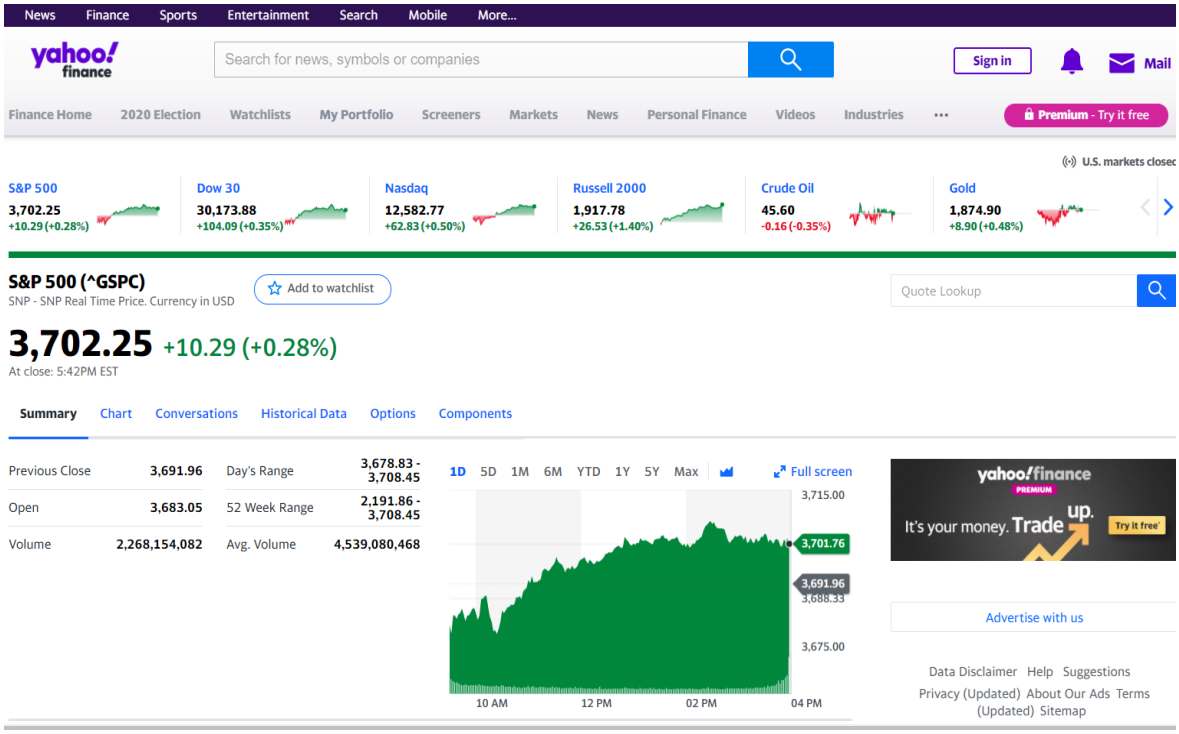

Рис.2.9 Веб сайт Yahoo Finance.

Сервіс Google Finance(рис.2.10) надає доступ до фінансової інформації про більшість транснаціональних компаній. Доступна інформація по котируваннях і рейтинги цінних паперів, прес-релізи та фінансові звіти компаній. По кожній компанії відображаються результати агрегаторів з Google News і Google Blog Search. Історичні дані доступні у вигляді графіків, реалізованих за технологією Adobe Flash.

Сайт пропонує ряд сервісів для управління персональною фінансовою інформацією.

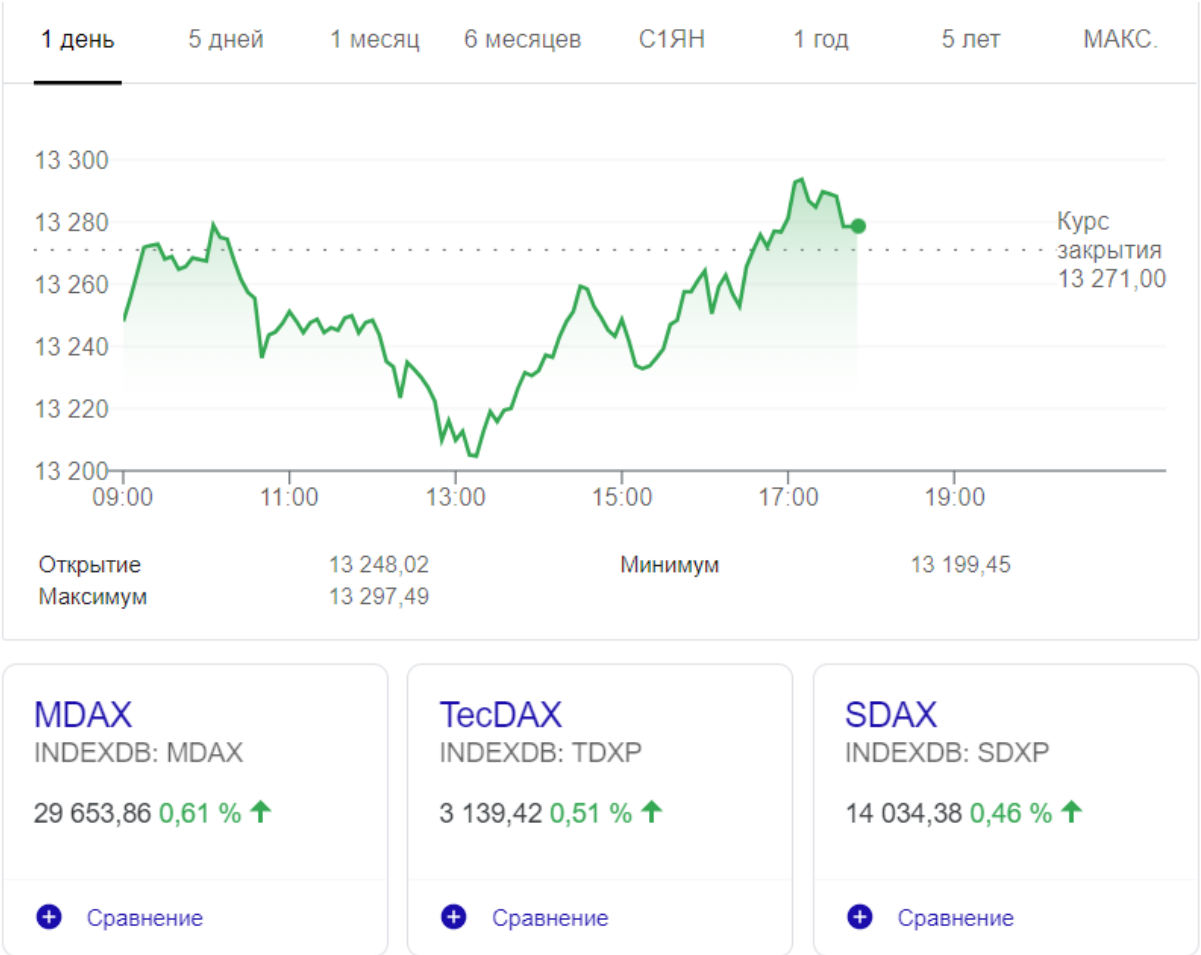

Рис.2.10 інформаційна платформа від Google.

В базах даних застосовують набір даних(рис.2.11), що представлений в текстовому вигляді формату CSV(Comma-Separated Values).

	A	B	C	D	E	F	G	H	I
1	Date,Oper	,High,Low,Close,Adj Close,Volume							
2	2010-01-07,1136.270020,1142.459961,1131.319946,1141.689941,1141.689941,5270680000								
3	2010-01-08,1140.520020,1145.390015,1136.219971,1144.979980,1144.979980,4389590000								
4	2010-01-11,1145.959961,1149.739990,1142.020020,1146.979980,1146.979980,4255780000								
5	2010-01-12,1143.810059,1143.810059,1131.770020,1136.219971,1136.219971,4716160000								
6	2010-01-13,1137.310059,1148.400024,1133.180054,1145.680054,1145.680054,4170360000								
7	2010-01-14,1145.680054,1150.410034,1143.800049,1148.459961,1148.459961,3915200000								
8	2010-01-15,1147.719971,1147.770020,1131.390015,1136.030029,1136.030029,4758730000								
9	2010-01-19,1136.030029,1150.449951,1135.770020,1150.229980,1150.229980,4724830000								
10	2010-01-20,1147.949951,1147.949951,1129.250000,1138.040039,1138.040039,4810560000								
11	2010-01-21,1138.680054,1141.579956,1114.839966,1116.479980,1116.479980,6874290000								
12	2010-01-22,1115.489990,1115.489990,1090.180054,1091.760010,1091.760010,6208650000								
13	2010-01-25,1092.400024,1102.969971,1092.400024,1096.780029,1096.780029,4481390000								
14	2010-01-26,1095.800049,1103.689941,1089.859985,1092.170044,1092.170044,4731910000								
15	2010-01-27,1091.939941,1099.510010,1083.109985,1097.500000,1097.500000,5319120000								
16	2010-01-28,1096.930054,1100.219971,1078.459961,1084.530029,1084.530029,5452400000								
17	2010-01-29,1087.609985,1096.449951,1071.589966,1073.869995,1073.869995,5412850000								
18	2010-02-01,1073.890015,1089.380005,1073.890015,1089.189941,1089.189941,4077610000								
19	2010-02-02,1090.050049,1104.729980,1087.959961,1103.319946,1103.319946,4749540000								
20	2010-02-03,1100.670044,1102.719971,1093.969971,1097.280029,1097.280029,4285450000								
21	2010-02-04,1097.250000,1097.250000,1062.780029,1063.109985,1063.109985,5859690000								
22	2010-02-05,1064.119995,1067.130005,1044.500000,1066.189941,1066.189941,6438900000								
23	2010-02-08,1065.510010,1071.199951,1056.510010,1056.739990,1056.739990,4089820000								
24	2010-02-09,1060.060059,1079.280029,1060.060059,1070.520020,1070.520020,5114260000								
25	2010-02-10,1069.680054,1073.670044,1059.339966,1068.130005,1068.130005,4251450000								
26	2010-02-11,1067.099976,1080.040039,1060.589966,1078.469971,1078.469971,4400870000								
27	2010-02-12,1075.949951,1077.810059,1062.969971,1075.510010,1075.510010,4160680000								
28	2010-02-16,1079.130005,1095.670044,1079.130005,1094.869995,1094.869995,4080770000								
29	2010-02-17,1096.140015,1101.030029,1094.719971,1099.510010,1099.510010,4259230000								
30	2010-02-18,1099.030029,1108.239990,1097.479980,1106.750000,1106.750000,3878620000								

^GSPC

Рис.2.11 Файл з даними фондового індексу S & P 500.

S & P 500 - фондовий індекс, в кошик якого включено 505 обраних торгованих на фондових біржах США публічних компаній, що мають найбільшу капіталізацію. Список належить компанії Standard & Poor's і нею ж складається.

2.3 Опис моделі нейронної мережі.

Як було сказано в першій частині, для різних задач існують свої алгоритми нейронних мереж. Мета даної роботи полягає в прогнозуванні, тому для задач прогнозування застосовують моделі рекурентних нейронних мереж, а саме модель що має так звану «пам’ять» - LSTM(Long Short-Term Memory)(рис.2.12), тобто, враховує попередні значення.

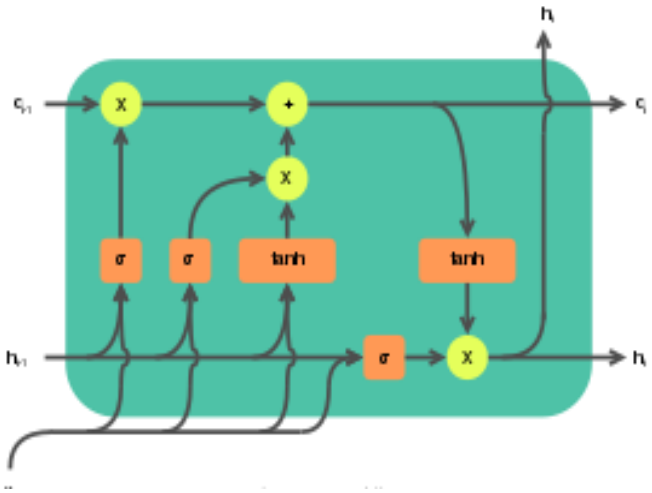

Рис.2.12 Комірка LSTM.

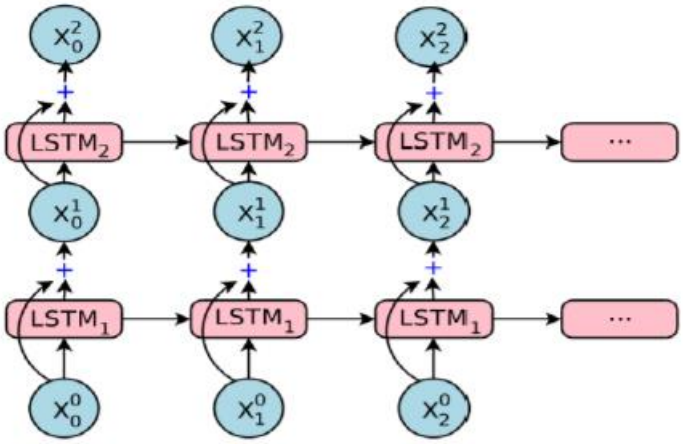

Рис.2.13 Послідовність LSTM комірок.

LSTM мережа вирішує проблему присутня в простих рекурентних мережах, а саме – затухання градієнта. В такій мережі підтримується передача інформації через довгі інтервали часу.

Етап проектування нейронної мережі зводиться до послідовності дій з обробки даних так, щоб нейронна мережа могла їх зчитати та обробити.

Реалізовані нейронні мережі з LSTM комірками в бібліотеці Keras являють собою програмну функцію, яку можна додавати до шаблону моделі.

Як було описано в першому розділі, нейронна мережа має в собі пряму послідовність нейронних комірок, тому досить обрати шаблон з послідовною архітектурою мережі(рис.2.14).

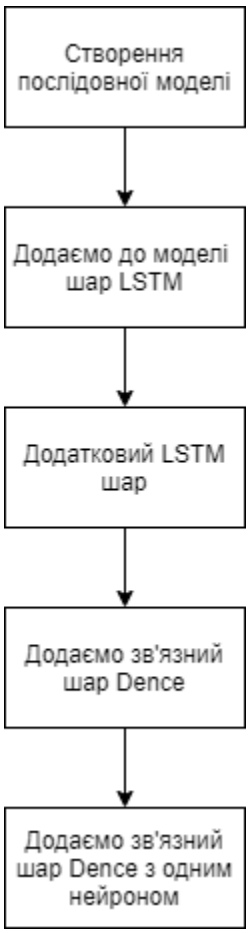

Рис.2.14 Послідовна архітектура мережі.

Вибір оцінки моделі можна покласти на середньоквадратичну похибку – MSE. Це одна з простіших способів оцінки якості роботи.

$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2$, де

y – вірна відповідь, y᾽- прогноз.

Функція активації LSTM комірки – Гіперболічний тангенс(рис.2.15).

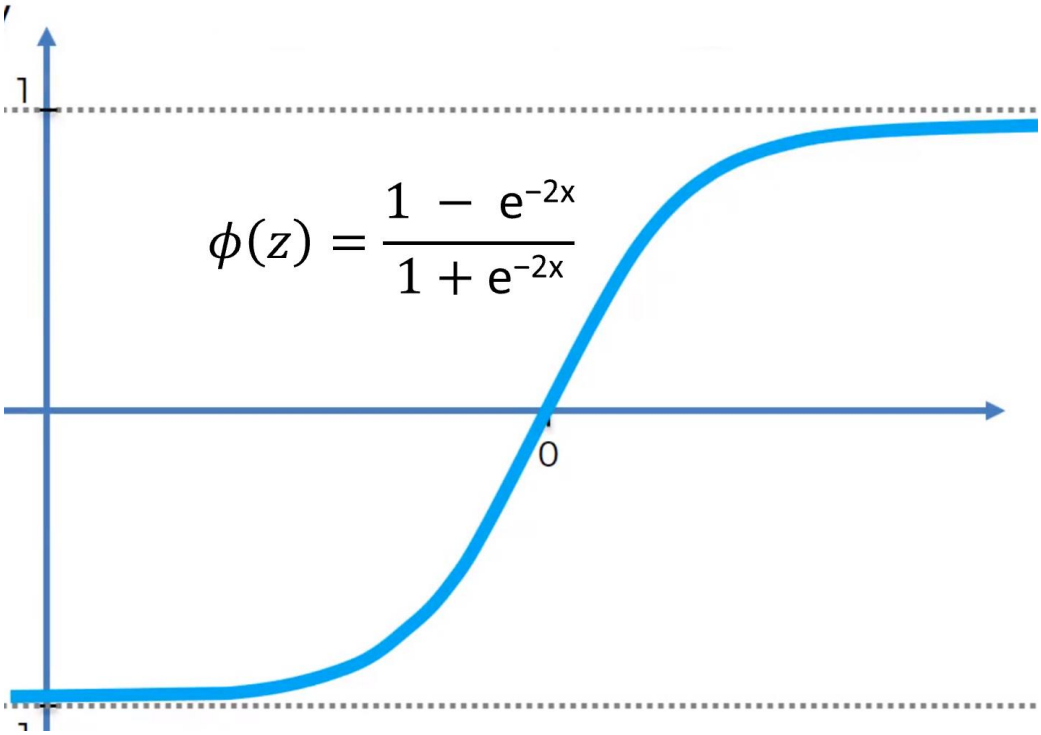

Рис.2.15 Функція гіперболічного тангенсу.

ВИСНОВКИ ДО РОЗДІЛУ 2

Було розглянуто історичні часові послідовності як база даних для тренування рекурентних нейронних мереж.

Дані представляються послідовністю з 6 значень:

- Open;
- High;
- Low;
- Close
- Adj Close;
- Volume.

Формат зберігання даних CSV для роботи з будь-яким редактором.

Було обрано Yahoo API як джерело даних. Серед доступних фінансових індексів, було обрано S&P 500.

Для вирішення задач прогнозування фінансових часових рядів було обрано спеціально розроблений алгоритм нейронних мереж LSTM, якому притаманна властивість довгої короткострокові пам’яті. Задача прогнозування в певній мірі базується на історичних даних будь-якого часового ряду, тому враховувати попередні значення критично необхідно для подальшого прогнозування.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ НЕЙРОННОЇ МЕРЕЖІ ТА ОЦІНКА РЕЗУЛЬТАТІВ.

3.1 Підготовка даних.

Підключення необхідних модулів до проекту.

Підключаємо модуль математичних функцій

```
import math
```

data_reader знадобиться нам, щоб віддалено завантажувати CSV бази даних.

```
import pandas_datareader as data_reader
```

Підключаємо бібліотеки для роботи з масивами.

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

Імпортуємо засіб для зведення всіх даних до чисел в діапазоні від 0 до 1.

```
from sklearn.preprocessing import MinMaxScaler
```

Імпортуємо послідовну Архітектуру мережі.

```
from keras.models import Sequential
from keras.layers import Dense, LSTM
from keras import layers
```

Для побудови графіків.

```
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

За допомогою Yahoo API отримаємо часовий ряд фондового індексу S&P 500(рис.3.1).

```
data_frame = data_reader.DataReader('^GSPC', data_source='yahoo', start='2012-01-01', end='2020-12-01')
```

Початок інтервалу було обрано за 01.01.2012, кінцеве значення 01.12.2020.

Date	High	Low	Open	Close	Volume	Adj Close
2012-01-03	1284.619995	1258.859985	1258.859985	1277.060059	3943710000	1277.060059
2012-01-04	1278.729980	1268.099976	1277.030029	1277.300049	3592580000	1277.300049
2012-01-05	1283.050049	1265.260010	1277.300049	1281.060059	4315950000	1281.060059
2012-01-06	1281.839966	1273.339966	1280.930054	1277.810059	3656830000	1277.810059
2012-01-09	1281.989990	1274.550049	1277.829956	1280.699951	3371600000	1280.699951
...	...	...	...	...	...	...
2020-11-24	3642.310059	3594.520020	3594.520020	3635.409912	6267570000	3635.409912
2020-11-25	3635.500000	3617.760010	3635.500000	3629.649902	4902560000	3629.649902
2020-11-27	3644.310059	3629.330078	3638.550049	3638.350098	2778450000	3638.350098
2020-11-30	3634.179932	3594.389893	3634.179932	3621.629883	6291400000	3621.629883
2020-12-01	3678.449951	3645.870117	3645.870117	3662.449951	5403660000	3662.449951

2244 rows × 6 columns

Рис.3.1 Таблиця завантажених даних індексу.

Через надлишок кількості даних, можемо побачити перші п’ять значень та останні значення.

Часовий ряд було завантажено починаючи з 2012-01-01 Та закінчуючи 2020-12-01.

Всього таблиця має 2244 дні виміру за даними з 6 стовбців.

Для дослідження нейронної мережі нам знадобиться поле дати та поле Close, в якому записана остаточна ціна на даний день.

За обраними параметрами побудовано часовий ряд у вигляді графіку(рис.3.2).

Побудуємо графік за даними цін закриття Close за датами.

```
plt.figure(figsize=(16,8))
plt.title('S&P 500 Close price history')
plt.plot(data_frame['Close'])
```

```
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close prise usd', fontsize=18)
plt.show
data_frame.shape
```

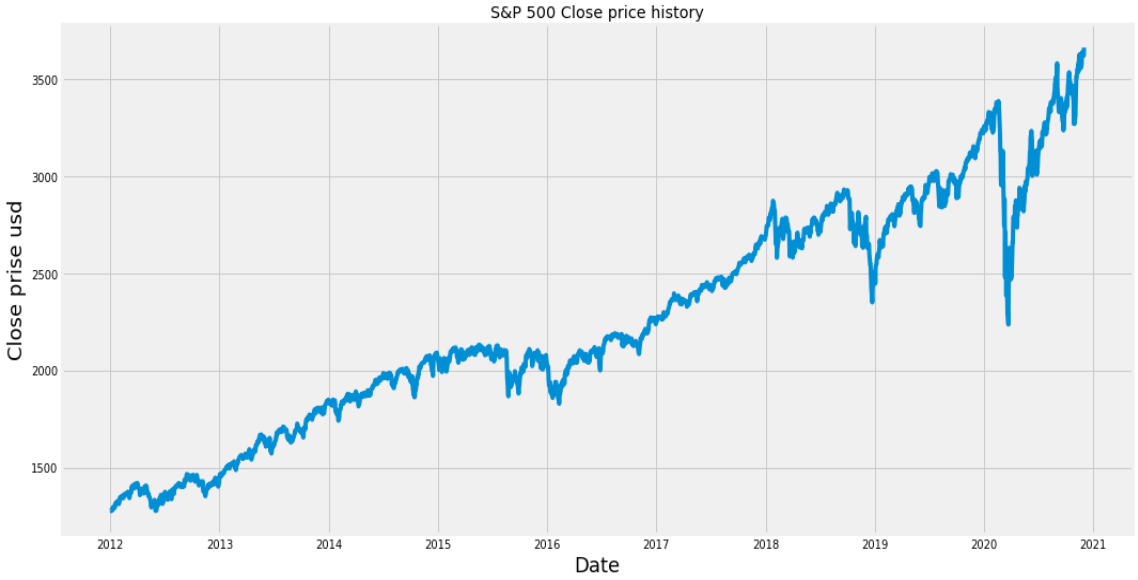

Рис.3.2 Графік індексу S&P 500 з 2012 року.

За даними графіком добре видно, що ціна фонду росте, та тенденцію періодичного спаду цін.

Отриманий часовий ряд було вирішено розділити на частини, де:

- 90% даних піде на навчання нейронної мережі;
- 10% даних нейронна мережа повинна прогнозувати.

Прогнозувати будемо за даними ціни закриття Close.

Розділивши дані, отримали, що 2020 значень нейронна мережа застосує для навчання, останні 224 спробуємо спрогнозувати.

```
data = data_frame.filter(['Close'])
dataset = data.values
train_data_len = math.ceil(len(dataset) * 0.9)
train_data_len
2020
```

Мережа на вхід має отримати певний масив даних, а самі дані являються числовими. Ціна закриття виявляється досить великою величиною, тому скористаємося функцією MinMaxScaler для того, що перетворити реальні дані в числа в 0 до 1(рис.3.3).

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

```
array([[0.00000000e+00],
       [1.00608389e-04],
       [1.67687472e-03],
       ...,
       [9.89896891e-01],
       [9.82887465e-01],
       [1.00000000e+00]])
```

Рис.3.3 Отримані числові значення від 0 до 1.

Перетворимо виділені масиви до виду, готового для нейронної мережі у вигляді тензору третього порядку(рис.3.4):

```
[array([0.        , 0.00010061, 0.00167687, 0.00031441, 0.00152591,
        0.00629662, 0.00646432, 0.00773037, 0.00504316, 0.00696322,
        0.01298739, 0.01569552, 0.01606444, 0.01632435, 0.01575842,
        0.02054172, 0.01734307, 0.01646267, 0.01507089, 0.01481937,
        0.01971582, 0.02032371, 0.02843978, 0.0282008 , 0.02934111,
        0.030561  , 0.03139524, 0.02749234, 0.03131981, 0.0307874 ]),
 array([0.00010061, 0.00167687, 0.00031441, 0.00152591, 0.00629662,
        0.00646432, 0.00773037, 0.00504316, 0.00696322, 0.01298739,
        0.01569552, 0.01606444, 0.01632435, 0.01575842, 0.02054172,
        0.01734307, 0.01646267, 0.01507089, 0.01481937, 0.01971582,
        0.02032371, 0.02843978, 0.0282008 , 0.02934111, 0.030561  ,
        0.03139524, 0.02749234, 0.03131981, 0.0307874 , 0.02773967]),
 array([0.00167687, 0.00031441, 0.00152591, 0.00629662, 0.00646432,
        0.00773037, 0.00504316, 0.00696322, 0.01298739, 0.01569552,
        0.01606444, 0.01632435, 0.01575842, 0.02054172, 0.01734307,
        0.01646267, 0.01507089, 0.01481937, 0.01971582, 0.02032371,
        0.02843978, 0.0282008 , 0.02934111, 0.030561  , 0.03139524,
        0.02749234, 0.03131981, 0.0307874 , 0.02773967, 0.03394832]),
 array([0.00031441, 0.00152591, 0.00629662, 0.00646432, 0.00773037,
        0.00504316, 0.00696322, 0.01298739, 0.01569552, 0.01606444,
        0.01632435, 0.01575842, 0.02054172, 0.01734307, 0.01646267,
        0.01507089, 0.01481937, 0.01971582, 0.02032371, 0.02843978,
        0.0282008 , 0.02934111, 0.030561  , 0.03139524, 0.02749234,
        0.03131981, 0.0307874 , 0.02773967, 0.03394832, 0.0352856 ]),
 array([0.00152591, 0.00629662, 0.00646432, 0.00773037, 0.00504316,
        0.00696322, 0.01298739, 0.01569552, 0.01606444, 0.01632435,
        0.01575842, 0.02054172, 0.01734307, 0.01646267, 0.01507089,
        0.01481937, 0.01971582, 0.02032371, 0.02843978, 0.0282008 ,
        0.02934111, 0.030561  , 0.03139524, 0.02749234, 0.03131981,
        0.0307874 , 0.02773967, 0.03394832, 0.0352856 , 0.03569643]),
```

Рис.3.4 Маємо навчальний набір даних для нейронної мережі.

Приведемо дані до 3 – вимірного тензору.

```
x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train,(x_train.shape[0], x_train.shape[1], 1))
x_train.shape
```

3.2 Реалізація архітектури мережі та оцінка прогнозування.

Архітектуру нейронної мережі було реалізовано в такому вигляді(рис.3.5):

- Послідовна архітектура нейронної мережі;

- Мережа має 2 послідовних шари з комірок LSTM;
- Кожен шар налічує 40 комірок нейронів;
- Повнозв’язний шар Dense з 25 нейронів;
- Вихідний шар Dense з одним нейроном.

```
model = Sequential()
model.add(LSTM(40,
return_sequences=True,input_shape=(x_train.shape[1],
1)))

model.add(LSTM(40, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 30, 40)            6720
_________________________________________________________________
lstm_3 (LSTM)                (None, 40)                12960
_________________________________________________________________
dense_2 (Dense)              (None, 25)                1025
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 26
=================================================================
Total params: 20,731
Trainable params: 20,731
Non-trainable params: 0
_________________________________________________________________
```

Рис.3.5 Вивід архітектури мережі.

В якості оптимізатора навчання обрано Adam.

Розрахунок похибки Mean Squared Error.

Навчання нейронної мережі відбувалося на кожному послідовному значенні часового ряду, тому оптимально для навчання достатньо однієї епохи.

```
    model.compile(optimizer='adam',
loss='mean_squared_error')
    history      =      model.fit(x_train,      y_train,
epochs=1,batch_size=1)
```

Далі для розрахунку якості проектованої моделі застосовано алгоритм RMSE, що добуває квадратний корінь з середньої квадратичної помилки(рис.3.6).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

Рис.3.6 Формула RMSE.

```
    rmse = np.sqrt(np.mean(predictions - y_test)**2)
    rmse
```

Отримане значення можна оцінити як задовільне – 12.659767150878906.

Це означає, що модель досить непогано виділила ознаки з історичних даних і змогла зробити майбутній прогноз(рис.3.7).

```
    train = data[: train_data_len]
    validation = data[train_data_len:]
    validation['Predictions'] = predictions

    plt.figure(figsize=(16,8))
    plt.title('Model')
    plt.xlabel('Date', fontsize=18)
    plt.ylabel('Close price', fontsize=18)
    plt.plot(train['Close'])
    plt.plot(validation[['Close', 'Predictions']])
```

```
    plt.legend(['Training',  'Value',  'Predictions'],
loc='lower right')
    plt.show()
```

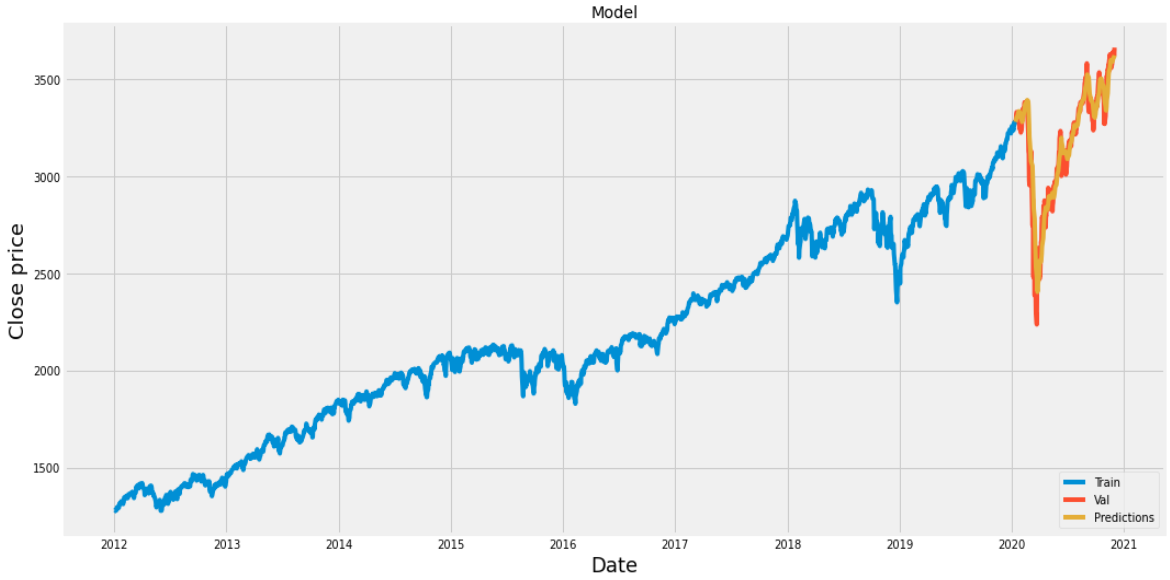

Рис.3.7 Прогнозовані значення моделі нейронної мережі на даних для перевірки.

Для оцінки числових даних виведемо у вигляді таблиці результат прогнозування поряд з реальними значеннями(рис.3.8).

Date	Close	Predictions
2020-01-14	3283.149902	3279.736816
2020-01-15	3289.290039	3286.354736
2020-01-16	3316.810059	3292.596191
2020-01-17	3329.620117	3302.162354
2020-01-21	3320.790039	3313.433594
...	...	...
2020-11-24	3635.409912	3589.983398
2020-11-25	3629.649902	3597.535889
2020-11-27	3638.350098	3606.547852
2020-11-30	3621.629883	3616.593994
2020-12-01	3662.449951	3622.562012

Рис.3.8 Порівняння реальної ціни з прогнозованою.

Отже, маємо, що нейронна мережа на початку має високу точність, а результат прогнозування відхиляється від ціни менш ніж на 10 USD. Проте, в подальших днях спостерігається все більша розбіжність у результатах, що свідчить про меншу якість прогнозу більше, ніж на 5-й день.

3.3 Порівняння різних алгоритмів рекурентним мереж.

В теоретичній частині роботи та в розділі проектування було обрано алгоритм LSTM нейронної мережі, як найкращим засобом досягнення мети роботи. Про існують інші алгоритми рекурентних мереж, які варто розглянути та порівняти результати навчання з фаворитним LSTM.

SimpleRNN – алгоритм простої рекурентної мережі, яка оброблю послідовності даних.

Для тестування не будемо змінювати файл з даними та інше, достатньо змінити архітектуру мережі на SimpleRNN(рис.3.9).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn (SimpleRNN)       (None, 30, 40)            1680
_________________________________________________________________
simple_rnn_1 (SimpleRNN)     (None, 40)                3240
_________________________________________________________________
dense (Dense)                (None, 25)                1025
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 26
=================================================================
Total params: 5,971
Trainable params: 5,971
Non-trainable params: 0
_________________________________________________________________
```

Рис.3.9 Структура мережі з простих RNN.

Мережа досить непогано впоралася з завданням прогнозування.

Після навчання оцінка похибки RMSE становить 19.267691476004465, що не дуже відрізняється від похибки LSTM(рис.3.10).

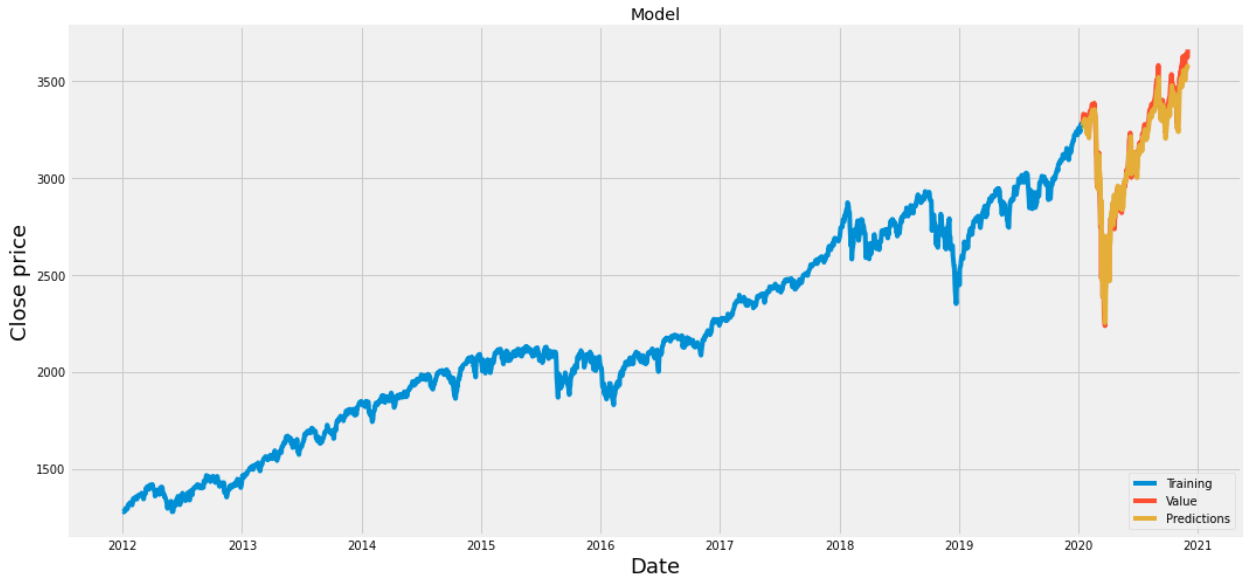

Рис.3.10 Прогноз на основі простої RNN.

Date	Close	Predictions
2020-01-14	3283.149902	3264.242432
2020-01-15	3289.290039	3263.541016
2020-01-16	3316.810059	3271.275146
2020-01-17	3329.620117	3290.069824
2020-01-21	3320.790039	3304.266357
...	...	...
2020-11-24	3635.409912	3513.168457
2020-11-25	3629.649902	3562.987793
2020-11-27	3638.350098	3558.089844
2020-11-30	3621.629883	3577.118896
2020-12-01	3662.449951	3567.544434

Рис.3.11 Порівняння реальних даних з прогнозом.

GRU(рис.3.12) – механізм вентилів для рекурентних нейронних мереж, представлений в 2014 році. Було встановлено, що його ефективність при вирішенні задач моделювання музичних і мовних сигналів порівнянна з використанням довгої короткострокової пам'яті (LSTM). У порівнянні з LSTM у даного механізму менше параметрів, тому що відсутній вихідний вентиль.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gru (GRU)                    (None, 30, 40)            5160
_________________________________________________________________
gru_1 (GRU)                  (None, 40)                9840
_________________________________________________________________
dense (Dense)                (None, 25)                1025
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 26
=================================================================
Total params: 16,051
Trainable params: 16,051
Non-trainable params: 0
_________________________________________________________________
```

Рис.3.12 Структура мережі GRU.

GRU алгоритм виявився не підходящим для вирішення задачі прогнозування. Можливі причини:

- Велика кількість параметрів;
- Невдалий підбір параметрів;
- Не типове застосування.

Значення RMSE похибки 63.95740400041853, що в 3 рази більше, ніж проста комірка RNN.

Підбір параметрів мережі.

Спробуємо поєднати нейрони GRU послідовно з LSTM(рис.3.13).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gru (GRU)                    (None, 30, 10)            390
_________________________________________________________________
lstm (LSTM)                  (None, 40)                8160
_________________________________________________________________
dense (Dense)                (None, 25)                1025
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 26
=================================================================
Total params: 9,601
Trainable params: 9,601
Non-trainable params: 0
_________________________________________________________________
```

Рис.3.13 Архітектура комбінацій алгоритмів.

Слід зауважити, що швидкість навчання мережі збільшилась в 2-3 рази без втрати якості навчання, навіть зі збільшенням.

RMSE похибка становить найменшу з попередніх варіантів, а саме – 7.52862548828125.

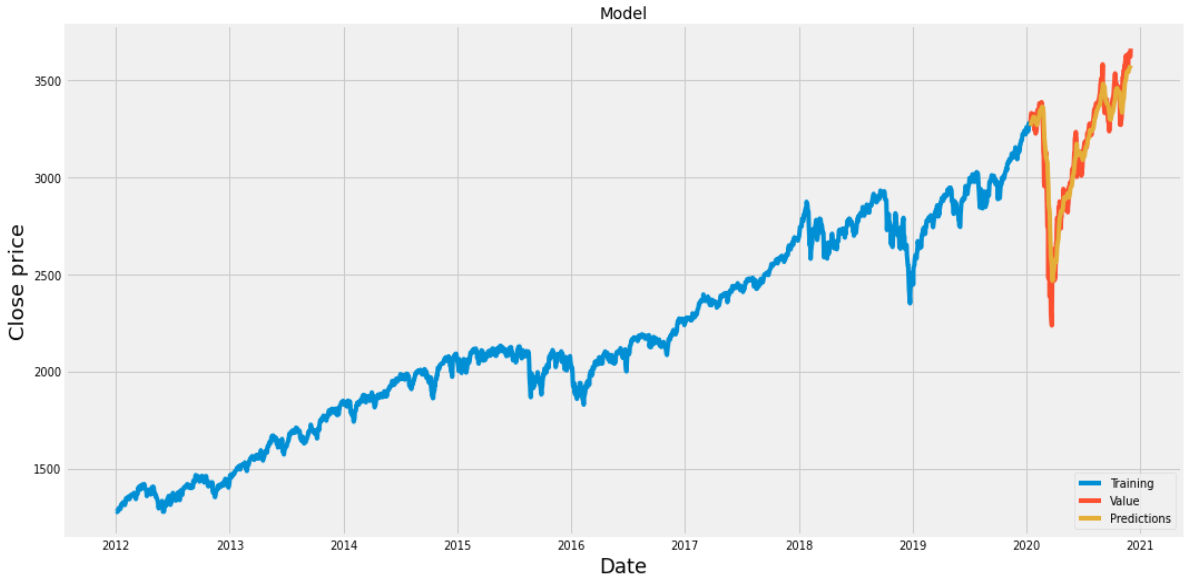

Рис.3.14 Прогноз комбінованої архітектури.

Date	Close	Predictions
2020-01-14	3283.149902	3263.550293
2020-01-15	3289.290039	3269.107178
2020-01-16	3316.810059	3274.377441
2020-01-17	3329.620117	3283.142334
2020-01-21	3320.790039	3293.192383
...	...	...
2020-11-24	3635.409912	3542.593750
2020-11-25	3629.649902	3552.910156
2020-11-27	3638.350098	3562.395020
2020-11-30	3621.629883	3571.583008
2020-12-01	3662.449951	3576.244385

Рис.3.15 Порівняння прогнозу комбінованої архітектури з реальними даними.

ВИСНОВКИ ДО РОЗДІЛУ 3

Отже, на основі спроектованої архітектури мережі та обраної бази даних для навчання та перевірки було виконано такі кроки:

- Приведення даних до масивів;
- Задання параметрів архітектури, а саме: оптимізатор, кількість нейронів у шарі, функція оцінки навчання;
- Навчено нейронну мережу на перших 90% даних фондового індексу.

Результат навчання було розраховано за допомого функції втрат, а саме RMSE, завдяки якій можливо спостерігати точність прогнозування мережі на часовому ряді.

Нейронна мережа змогла дати непогане прогнозування, але опиратися на ці дані не є хорошою ідеєю, адже з навчальних матеріалів було використано лице історичні дані, які не є максимальними показниками для майбутніх прогнозів.

ВИСНОВОК

В ході дослідження було застосовано технологію глибокого навчання штучних нейронних мереж для задачі прогнозування часових рядів. В даній роботі було вирішено такі задачі:

- Проведено аналіз предметної області штучних нейронних мереж, а саме – глибокого навчання;
- Проведено порівняння існуючих програмних реалізацій прогнозування цін;
- Спроектовано архітектуру нейронної мережі з LSTM нейронами;
- Проведено навчання нейронної мережі на основі історичних цін фондового індексу S&P 500;
- Проаналізовано оцінку роботи нейронної мережі на майбутніх даних.

Отже, виходячи з результатів дослідження, можна сказати, що реалізована нейронна мережа з короткостроковою довгою пам’яттю добре підходить для задача прогнозування, коли минулі дані мають значення і варто їх враховувати.

Середня квадратична похибка на етапі навчання становить 9.4839e-04, що можна вважати задовільним результатом.

Похибка даних на яких мережа не навчалась та прогнозованих даних за допомогою RMSE становить - 12.659767150878906, при RMSE = 0 – результат прогнозовано на 100% вірно, а сама модель досягає максимальної точності прогнозування перших 10-ти днів, після чого спостерігається спад точності і збільшується різниця між реальними даними та прогнозом.

Отже, виходячи з вище сказаного, можна сказати, що ціль даної роботи буда досягнута.

ДЖЕРЕЛА

1. Chollet F. Deep Learning with Python / François Chollet., 2017. – 384 с. – (ISBN 9781617294433).

2. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow / Aurélien Géron., 2019. – (ISBN: 9781492032649).

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

4. Ярушкина Н. Интеллектуальный анализ временных рядов: Учебное пособие / Н.Г. Ярушкина, Т.В. Афанасьева, И.Г. Перфильева – М.: Инфра-М, 2015 – 160 с.

5. Matplotlib documentation [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://matplotlib.org/3.3.3/contents.html.

6. Прогнозирование с помощью нейронных сетей / М. В. Ульянова [Электронный ресурс]. / URL: http://www.scienceforum.ru/2018/pdf/5783.pdf.

7. Архитектура нейронной сети / Сайт «ИНТУИТ» [Электронный ресурс]. / URL: https://www.intuit.ru/studies/courses/6/6/lecture/178.

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

9. Анализ временных рядов [Электронный ресурс]: / Электрон. дан. - Идентификация модели временных рядов - http://www.statsoft.ru/home/textbook/modules/sttimser.html.

10. Машинное обучение / Сайт «Habr» [Электронный ресурс]. / URL: https://habr.com/company/wunderfund/blog/331310.

11. Raschka S. Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 3rd Edition / S. Raschka, V. Mirjalili., 2019. – 770 с. – (1789955750).

12.

13. Нейронные сети. / Я. Еремин / [Электронный ресурс]. / URL: http://elanina.narod.ru/lanina/ind/neiro/index.html.

14. Искусственная нейронная сеть с нуля на Python с библиотекой NumPy [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://neurohive.io/ru/tutorial/nejronnaja-set-na-numpy/.

15. Shukla N. Machine Learning with TensorFlow / N. Shukla, K. Fricklas., 2018. – 272 с. – (ISBN 9781617293870).

16. Basic regression: Predict fuel efficiency [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/tutorials/keras/regression.

17. The Sequential class [Електронний ресурс] – Режим доступу до ресурсу: https://keras.io/api/models/sequential/.

18. Keras API reference [Електронний ресурс] – Режим доступу до ресурсу: https://keras.io/api/.

19. Layer activation functions [Електронний ресурс] – Режим доступу до ресурсу: https://keras.io/api/layers/activations/.

20. NumPy v1.18 Manual [Електронний ресурс] – Режим доступу до ресурсу: https://numpy.org/doc/1.18/user/basics.html.

21. NumPy Reference¶ [Електронний ресурс] – Режим доступу до ресурсу: https://numpy.org/doc/1.18/reference/index.html.

22. API reference [Електронний ресурс] – Режим доступу до ресурсу: https://pandas.pydata.org/docs/reference/index.html#api.

23. Pilgrim M. Dive Into Python 3 / Mark Pilgrim., 2009. – 360 с. – (1430224150).

24. Lutz M. Learning Python, 5th Edition / Mark Lutz., 2013. – 1648 с. – (1449355730).

25. Vanderplas J. Python Data Science Handbook: Essential Tools for Working with Data / Jake Vanderplas.. – 548 с. – (1491912057; т. 1).