Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
Кафедра Комп'ютерних інтелектуальних технологій та систем

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Інтелектуальна система підтримки прийняття рішень
для трейдера на фінансовому ринку

Виконав:

студент 2 курсу, групи КІТм-21-1
Антонов Д. О.
Спеціальність 123 Комп'ютерна інженерія
Тип програми освітньо-професійна
Освітня програма Комп'ютерні інтелектуальні технології
Керівник проф. Корабльов М.М

Допускається до захисту

(підпис)

Зав. кафедри проф. Руденко О.Г.

(підпис)

2022 р.

Харківський національний університет радіоелектроніки

Факультет	Комп'ютерної інженерії та управління
Кафедра	Комп'ютерних інтелектуальних технологій та систем
Рівень вищої освіти	другий (магістерський)
Спеціальність	123 Комп'ютерна інженерія
Тип програми	освітньо-професійна
Освітня програма	Комп'ютерні інтелектуальні технології

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Антонову Данилу Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інтелектуальна система підтримки прийняття рішень для трейдера на фінансовому ринку

затверджена наказом по університету від "____" _____ 2022 р. № _____

2. Термін подання студентом роботи до екзаменаційної комісії 10 грудня 2022р.

3. Вхідні дані до роботи

1) виявлення основних сигналів за графіками ринку акцій трейдера;
2) побудова тестових моделей нейронних мереж для підтримки рішень трейдера;
3) середовище моделювання – Pycharm;
4) мова програмування – Python.

4. Перелік питань, що потрібно опрацювати в роботі

1) огляд предметної області;
2) аналіз предмету дослідження;
3) дослідження нейронних мереж;
4) дослідження інтелектуальних систем підтримки прийняття рішень;
5) дослідження сигналів графіків трейдера;
6) розробка інтелектуальної системи підтримки прийняття рішень;
7) експериментальні дослідження;
8) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів): Презентація - 15 слайдів

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Видача та узгодження теми проекту	07.09.2022	виконано
2	Огляд стану проблеми та постановка задачі	09.09-14.09	виконано
3	Аналіз літератури за напрямком магістерської	14.09-21.09	виконано
	роботи		
4	Аналіз сигналів на графіках трейдерів	21.09-28.09	виконано
5	Аналіз інтелектуальних систем підтримки	21.09-28.09	виконано
	прийняття рішень		
6	Розробка тестових нейронних мереж	28.09-12.10	виконано
7	Експериментальні дослідження	12.10-02.11	виконано
8	Підготовка графічного матеріалу	23.11-07.12	виконано
9	Перевірка виконаного проекту керівником	10.12.2022	виконано
10	Захист проекту	20.12.2022	

Дата видачі завдання 01 вересня 2022 р.

Студент _____ (підпис)

Керівник роботи _____ (підпис) _____ (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка магістерської кваліфікаційної роботи: 81 с., 40 рис., 12 джерел.

МАШИННЕ НАВЧАННЯ, ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ, НЕЙРОННІ МЕРЕЖІ, ШТУЧНИЙ ІНТЕЛЕКТ, ТРЕЙДИНГ, АЛГОРИТМИ, ГРОШІ

Ціль роботи – побудувати ІСППР для трейдера, яка б дозволила йому отримати вигоду на ринках акцій, використовуючи алгоритми машинного навчання.

Об’єкт дослідження – фінансовий ринок акцій.

Предмет дослідження – нейромережеві моделі та алгоритми, які можуть бути використані для передбачення майбутньої ціни на ринку акцій.

Методи дослідження – нейронні мережі та алгоритми машинного навчання.

В ході цієї роботи була розроблена інтелектуальна системи підтримки прийняття рішень трейдера на фінансовому ринку. В роботі була проведена дослідницька робота для відповіді на питання "як машинне навчання може допомагати у продажах та покупках акцій так, щоб приносити прибуток?".

На основі побудованої нейронної мережі було створено інтелектуальну систему підтримки прийняття рішень. Дана система представлена як клієнтський додаток. Дана система показала, що вона може виконувати поставлені завдання.

ABSTRACT

Explanatory note of the master’s qualification work: 81 pages, 40 figures, 12 sources.

MACHINE LEARNING, INTELLIGENT DECISION SUPPORT SYSTEMS, NEURAL NETWORKS, ARTIFICIAL INTELLIGENCE, TRADING, ALGORITHMS, MONEY

The goal of the work is to build an DSS for a trader that would allow him to profit on the stock markets using machine learning algorithms.

The object of research is the financial market of shares.

The subject of research is neural network models and algorithms that can be used to predict the future price on the stock market.

Research methods – neural networks and machine learning algorithms.

In the course of this work, an intelligent decision support system was developed for the trader in the financial market. In the paper, a lot of research work was done to answer the question "how can machine learning help in selling and buying stocks in a way that brings profit?".

Based on the constructed neural network, an intelligent decision support system was created. This system is presented as a console application. This system has shown that it can perform the assigned tasks.

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп’ютерної інженерії та управління
Кафедра Комп’ютерних інтелектуальних технологій та систем

АНОТАЦІЯ
КВАЛІФІКАЦІЙНОЇ РОБОТИ

рівень вищої освіти другий (магістерський)

Інтелектуальна система підтримки прийняття рішень
для трейдера на фінансовому ринку

Виконав:
студент 2 курсу, групи КІТм-21-1
Антонов Д. О.
Спеціальність 123 Комп’ютерна інженерія
Тип програми освітньо-професійна
Освітня програма Комп’ютерні інтелектуальні технології

Керівник проф. Корабльов М.М.

2022 р.

АНОТАЦІЯ

Антонов Д.О. Інтелектуальна система підтримки прийняття рішень для трейдера на фінансовому ринку – Магістерська кваліфікаційна робота.

Актуальність теми дослідження. Розпізнавання шаблонів є ключем до успішної торгівлі. Історично трейдери спостерігали закономірності в ринкових даних і використовували їх, щоб робити прогнози та максимізувати віддачу від своєї торгової діяльності. Ці стратегії можна представити як набір правил, які запускають купівлю та продаж, коли виконуються певні умови. Трейдери часто шукають моделі руху (математичні розрахунки на основі інформації про ціни, волатильність тощо) у технічних торгових індикаторах. Наприклад, перетин ковзного середнього – це проста торгова стратегія, яка базується на розумінні того, що поведінка індикаторів ковзного середнього відносно інших ковзних середніх допомагає визначити тенденції. Спостерігати за ринком і торгувати такими стратегіями можливо, але люди повільні та непослідовні. Машини швидші та точніші, і часто корисно кодувати стратегії в алгоритмах «якщо це станеться, зроби це» високочастотних торгових платформ, які можуть обробляти тисячі транзакцій за секунду. Це покращення порівняно з ручною торгівлею та більшість торгівлі, яку ми робимо сьогодні, є алгоритмічною. Однак він покладається на те, що люди визначають хороші шаблони та кодують алгоритми, щоб скористатися ними. Крім того, прибуток від алгоритмічної торгівлі в останні роки знизився через гостру конкуренцію. Якщо всі так зроблять, переваги буде менше. Навпаки, машинне навчання пропонує кілька переваг перед традиційною алгоритмічною торгівлею. Алгоритми машинного навчання можуть виявляти шаблони у великих обсягах даних. Вони використовуються для визначення релевантності історичних даних, які можна застосувати до алгоритмічних торгових стратегій. Машинне навчання дозволяє трейдерам прискорити й автоматизувати один із найскладніших,

трудомістких і складних аспектів алгоритмічної торгівлі, забезпечуючи конкурентну перевагу над торгівлею на основі правил.

У магістерській роботі досліджено науково-прикладну проблему створення інтелектуальної систем підтримки прийняття рішень для трейдера на фінансовому ринку, яка може частково допомгти трейдерові у повсякденному житті виконуючи частину його роботи.

Об'єктом дослідження є фінансовий ринок акцій.

Предметом дослідження є нейромережеві моделі та алгоритми, які можуть бути використані для передбачення майбутньої ціни на ринку акцій.

Дослідження базується на системному аналізі результатів сучасних теоретичних і прикладних розробок вітчизняних і зарубіжних учених в IT галузі. Для вирішення поставлених завдань використано: методи системного аналізу, методи побудови нейронних мереж, методи побудови інтелектуальних систем підтримки прийняття рішень, методи об'єктно-орієнтованого програмування, методи побудови програмних застосунків з командним інтерфейсом.

Метою даної роботи є розробка проекту інтелектуальної системи підтримки прийняття рішень здатної надати трейдеру інформацію яку він може використати та зробити більш точне рішення щодо певних акцій. Вимогами до системи є:

- збирати дані акції в реальному часі у момент запуску застосунку;
- приймати рішення: купити, продати або нічого не робити;
- навчатися у процессі роботи;
- видавати трейдерові найбільш підходящий алгоритм дії.

У першому розділі розглянуто аналіз предметної області і були поставлені задачі дослідження. Технічний аналіз фінансового ринку – це комплекс інструментів для прогнозування вартості активів. Допомагає правильно оцінювати графіки котирувань і приймати коректні торгові рішення. В основі технічного аналізу лежить комбінація знань про математику і психології людини. Застосування алгоритмів і індикаторів до

попередніх цін акцій допоможе спрогнозувати, як котирування будуть змінюватися в майбутньому. Технічний аналіз акцій хороший наочністю і простотою. Інструменти вже вбудовані в графіки котирувань на спеціалізованих інтернет-майданчиках, а ключові фінансові показники компаній можна вивчати в щоквартальних звітах. Було розглянуто поняття рівнів підтримки і опору. Підтримка – це мінімальна ціна, за якою трейдерам вигідно продавати акції. Опір – максимальна вартість, за яку їх готові купити. Було розглянуто різні варіанти графіків. Лінійний графік – найпростіший і зрозумілий, з його допомогою можна легко оцінити напрямок котирувань. Лінія формується з цін закриття торгової сесії за певний проміжок часу. Свічковий графік – це зручний інструмент, який відображає 4 параметри котирувань по кожному періоду. Бари – улюблений графік західних трейдерів. Будується за аналогією зі свічками, різниця – у візуалізації. Були розглянуті види трендів ринку. Тренд – це рух котирувань в заданому напрямку, чинний протягом певного відрізку часу. Його легко побачити на графіку. Аналіз трендів допомагає спрогнозувати зміну цін в майбутньому, розробляти торгові стратегії і ефективніше застосовувати біржові індикатори. Було розглянуто багато індикаторів. Трейдери використовують технічні індикатори, щоб отримати додаткову інформацію про ціновий рух активу. Ці індикатори спрощують виявлення патернів і сигналів спотової покупки або продажу у нинішньому ринковому середовищі. Є багато різних типів індикаторів, і вони широко використовуються денними трейдерами, свінг-трейдерами, а іноді навіть довгостроковими інвесторами. Далі була розглянута саме теоретична інформація щодо ІСППР. В рамках інформації про ІСППР було розглянуто такі поняття як знання, дані, властивості та моделі знань, проблеми винятків. Також, було розглянуто етапи прийняття рішень. Процес прийняття рішення здійснюється у кілька основних етапів: постановки задачі, формування рішень, вибору рішення. Етап постановки задачі складається з фаз аналізу та діагностики проблеми і визначення цілей рішення. Етап формування рішень

складається з фаз формулювання обмежень і критеріїв прийняття рішень та визначення альтернатив рішення. Етап вибору рішення складається з фаз оцінки альтернатив та остаточного вибору рішення. Була розглянута послідовність прийняття рішень. Також описано помилки прийняття рішень. Часто в процесі прийняття рішень ОПР припускають помилки. До найбільш поширених типових належать такі помилки: прийняття так званих однобічних рішень; відсутнім є системний підхід у прийнятті рішення; під час вибору варіантів перевагу надають «звичній» альтернативі; розглядають лише позитивні варіанти, а можливий ризик не враховують. В контексті трейдингу було розглянуто емоції в прийнятті рішень трейдерами. Звільнити себе від впливу емоцій - напевно, одна з найбільш важких завдань в торгівлі. Почасти, саме цим пояснюється те, що результати торгівлі на папері не завжди відповідають тим результатам, які трейдер міг би отримати, торгуючи на реальні гроші. Було розглянуто основні напрями ІСППР. Серед сучасних напрямів розробки людино-машинних систем, системи автоматичного керування, експертні системи та СППР. Найбільш придатними для розв’язання багатьох задач, зокрема задачі розподілу ресурсів, виявляються СППР. Саме за допомогою СППР ОПР має можливість безпосередньо за допомогою обчислювальних засобів проектувати, порівнювати та обирати альтернативні варіанти рішень різноманітними способами. Єдиної класифікації СППР зараз не існує.

У другому розділі було розроблено архітектуру ІССПР, описано її модулі: модуль подання даних до системи, модуль машинного навчання та модуль який відповідає за вивід інформації користувачеві. Розглянуто декілька моделей ІССПР. А саме авторегресійної інтегрованої ковзної середньої, глибинні нейронні мережі, відповідність шаблону, Q-навчання та глибинне Q-навчання. За останні кілька років було досягнуто неймовірних успіхів у застосуванні RNN для різноманітних проблем. Проте якщо розглядати LSTM для вирішення нашої задачі, нашої довжини послідовності їй може бути не достатньо. Найбільш за все нам підходить підхід Глибинного

Q-навчання. Він задовільняє усі наші потреби і бере найбільш привабливі риси нейромережевих підходів та Q-навчання. Так как підхід Глибинного Q-навчання побудований на базі звичайного Q-навчання. На вхід мережі подаються поточні числові дані графіків, а виходом є відповідне значення Q для кожної можливої дії. Також було обрано алгоритм навчання нейронної моделі.

У третьому розділі йдеться про експериментальну реалізацію усіх підходів із другого розділу. Спочатку було зібрано тестові дані. Так як підходящих під задачу тестових наборів данних знайдено не було, було прийнято рішення сформувати власні. Тестові дані було зібрано руками за допомогою ресерсу Yahoo Finance. Тестові дані заключають у собі різні показники, такі як Смуги Болліджера. Ці дані були використані для заповнення таблиці для подальшого прогнозування ринку цінних паперів. Однакові данні використовувалися під час усіх тестів, із різними алгоритмами та нейронними мережами. Слід зазначити, що це не цифри з голови, а реальні дані які будуть використовуватися як тестові. Усього було зібрано подібні тестові дані для акцій таких компаній: Apple, Amazon, EA, Ebay, Facebook, Google, IBM, Microsoft, Netflix, Nvidia, Oracle, Tesla, Twitter та ін. Далі був написаний python код для тестування кожного підходу і отримано та описано всі результати і доведено що глибинне Q-навчання може показувати дуже гарні результати.

На основі побудованої нейронної мережі було створено інтелектуальну систему підтримки прийняття рішень. Дана система представлена як консольний додаток. Дана система показала, що вона може виконувати поставлені завдання.

МАШИННЕ НАВЧАННЯ, ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ, НЕЙРОННІ МЕРЕЖІ, ШТУЧНИЙ ІНТЕЛЕКТ, ТРЕЙДИНГ, АЛГОРИТМИ, ГРОШІ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БЗ — База знань

БД — База даних

ЕС — Експертна система

ІСППР — Інтелектуальна система підтримки прийняття рішень

НМ — Нейронна мережа

ОПР — Особа, яка приймає рішення

СППР — Система підтримки прийняття рішень

ШІ — Штучний інтелект

ВСТУП

Трейдинг – це купівля та продаж фінансових інструментів для отримання прибутку, ці інструменти варіюються від різноманітних активів яким призначено фінансова вартість, яка змінюється зростаючи чи знижуючись. Ви можете торгувати точно в напрямку, у якому вони рухаються.

Трейдер – це особа, яка займається купівлею та продажем фінансового активу на будь-якому фінансовому ринку. На цьому фінансовому ринку трейдер може купувати або продавати як для себе, так і від імені іншої особи чи установи. Основною відмінністю між інвестором і трейдером є тривалість, протягом якої він або вона утримує актив.

Основною проблемою трейдингу є момент коли трейдеру требя зробити рішення що зараз робити – продати, купити, чи просто чекати. Також, важливо прийняти правильне рішення про те, які певні активи будуь найбільш прибутковими за певним проміжком часу. Так как як на кону стоять реальні гроші – нехтувати важливістю цих рішень просто неможливо.

Сьогодні існує певний клас актуальних завдань, вирішення яких без використання інтелектуальних систем підтримки прийняття рішень (ІСППР) неможливо або складно піддається реалізації. Для вирішення таких проблем людський інтелект малоефективний, традиційні обчислення трудомісткі або фізично неадекватні, оскільки не відображають або погано відображають реальні фізичні процеси і об'єкти, відповідно стає необхідним використовувати штучні нейронні мережі для вирішення класифікаційних задач.

Завдання класифікації – вказати належність вхідній вибірці, представленого вектором числових атрибутів, до одного або декількох визначеним класам. Практичні застосування такого завдання включають розпізнавання тексту, розпізнавання мови, класифікацію сигналів

електрокардіограми, класифікацію клітин крові, тощо.

Використання нейромережевих технологій відкриває якісно новий рівень вивчення процесів в такій невизначеній системі, як організм людини. Відмінною особливістю нейронних мереж є те, що вони не запрограмовані - вони не використовують ніяких правил виведення для постановки діагнозу, але вони вчаться робити це на прикладах. Діагностика - це окремий випадок класифікатора подій та основна категорія подій, які не включені до набору навчальних даних. Це відображає перевагу нейромережевих технологій: вони можуть виконувати таку класифікацію, узагальнювати попередній досвід і застосовувати його в нових випадках .

Нейронні мережі здатні вирішувати традиційні математичні задачі з недоступною обробкою, порівняння, класифікації образів; здатністю до самонавчання та самоорганізації. Вони забезпечують діагностику, ідентифікацію та класифікацію складних об’єктів з чіткими, складними нелінійними технічними умовами.

Метою цієї роботи є розробка інтелектуальної системи підтримки прийняття рішень, яка, аналізуючи певні показнкики певних фінансових активів допоже трейдеру відповідати на найголовніші питання – «чи мені продати зараз?», «Чи мені купити зараз?», та пошуку найвигідніших варіантів на ринку за певний проміжок часу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Поняття «технічний аналіз фінансового ринку»

Технічний аналіз фінансових ринків - це набір інструментів для прогнозування вартості активів, який допомагає вам правильно оцінювати ринкові графіки та приймати правильні торгові рішення. Технічний аналіз заснований на поєднанні математичних знань і людської психології. Застосування алгоритмів і індикаторів до попередніх цін на акції допомагає передбачити, як зміниться ринок у майбутньому.

Технічний аналіз фінансових ринків базується на трьох аксіомах:

- ціна акцій визначається безліччю факторів. Поточна вартість – це дзеркало того, що відбувається: співвідношення попиту і пропозиції, стану економіки, пов'язаних з конкретною компанією новин, коментарів регуляторів і аналітиків;
- ціна завжди в тренді. Графік весь час рухається в конкретному напрямку – той, хто вгадує подальшу тенденцію, отримує прибуток;
- історія циклічна. Графіком рухає людська психологія, рано чи пізно після піку тренд розвертається і йде вниз, а після досягнення мінімуму незмінно починає підніматися – і так по нескінченному колу.

Технічний аналіз акцій наочний і простий. Інструменти вже вбудовані в графіки котирувань на спеціалізованих інтернет-майданчиках, а ключові фінансові показники компаній можна вивчати в щоквартальних звітах.

1.1.1 Рівні підтримки і опору

Підтримка - це найнижча ціна, за якою трейдер може продати акції та отримати прибуток. Опір – це максимальна ціна, яку вони готові купити.

Загальна ситуація на фінансових ринках залежить від напрямку ринку. Верхня лінія - це лінія опору, а нижня - лінія підтримки. Під час одного тренду ціна знаходиться між ними. Коли графік перетинає будь-яку з них, психологія ринку змінюється. Такий стан називається пробою.

Ціни на графіку котирувань можуть відображатися по-різному, основних способів три. Це свічки, лінії і бари. Розглянемо докладніше кожен з них.

Лінійний графік – найпростіший і зрозумілий, з його допомогою можна легко оцінити напрямок котирувань. Лінія формується з цін закриття торгової сесії за певний проміжок часу.

Свічковий графік – це зручний інструмент, який відображає 4 параметри котирувань по кожному періоду. Свічка будується наступним чином: прямокутник формується на підставі цін відкриття і закриття торгової сесії, а лінії-хвости показують максимальну і мінімальну вартість, зафіксовані за певний інтервал часу. Зелене тіло означає зростання котирувань, червоне – зниження. При необхідності можна налаштувати інші, більш зручні кольори свічок в індивідуальному порядку.

Бари – улюблений графік західних трейдерів. Будується за аналогією зі свічками, різниця – у візуалізації. Кожен бар складається з лінії, що з'єднує мінімальне і максимальне значення ціни за торгову сесію. На відрізку є горизонтальні лінії-«вуха»: ліве позначає вартість закриття, праве – відкриття. Якщо ціна впала, бар буде червоним, якщо зросла – зеленим. Який вибрати графік для технічного аналізу ринку – свічковий або барний – залежить від індивідуальних переваг трейдера.

1.1.2 Види трендів

Тренд – це рух котирувань в заданому напрямку, чинний протягом певного відрізку часу. Його легко побачити на графіку. Аналіз трендів допомагає спрогнозувати зміну цін в майбутньому, розробляти торгові

стратегії і ефективніше застосовувати біржові індикатори.

Напрямок графіка поділяють на три види:

- висхідний тренд. Сприятливий час для «биків» – трейдерів, які заробляють на підвищенні вартості акцій. Лінія котирувань тягнеться вгору, при цьому можливі короткострокові падіння і коригування, але загальна тенденція прагнення до максимуму очевидна;

- нисхідний тренд. Час, коли на ринок виходять «ведмеді» – інвестори, що працюють на коротких дистанціях і отримують прибуток від падіння котирувань. Нисхідний тренд складається з низки падаючих «максимумів» і «мінімумів», кожен з яких нижчий за попередні;

- бічний (флет). Період руху котирувань паралельно осі, «піки» і «мінімуми» знаходяться недалеко один від одного. При цьому графік обмежений вищезгаданими лініями підтримки і опору. При відсутності форс-мажорних обставин фінансовий ринок знаходиться в бічному тренді 60% часу. Якщо ціна виходить за лінію опору, трейдери починають активно продавати акції через невпевненість у подальшому зростанні котирувань. Тренд розгортається, і ціна падає до тих пір, поки не стає привабливою для покупки, виходячи за лінію підтримки.

Існує ще одна класифікація трендів – за тривалістю. Тут також виділяють три види. Довгостроковий тренд триває 1-2 роки. В цей час на ринок виходять великі трейдери і акціонери. Середньостроковий триває 1-6 місяців і є корекційним, що йде врозріз з основним. Короткостроковий тренд – це невеликі коливання курсу протягом 1-4 тижнів, які практично неможливо оцінити і передбачити за допомогою технічного аналізу. А ось довго- і середньострокові – можна, що і слід робити трейдеру, оскільки більшість тенденцій фінансового ринку циклічні і відбуваються за одним і тим же сценарієм.

1.1.3 Індикатори

Трейдери використовують технічні індикатори, щоб отримати додаткову інформацію про рух цін на активи. Ці індикатори дозволяють легко виявити моделі купівлі або продажу та сигнали в поточному ринковому середовищі. Існує багато різних типів індикаторів, і вони широко використовуються денними трейдерами, свінг-трейдерами та іноді довгостроковими інвесторами. Давайте розглянемо приклад.

Одним із моїх улюблених місць для отримання інформації про ринки та публічні компанії є finance.yahoo.com. Ви можете переглянути історичні дані з технічними показниками, прочитати фінансову звітність компанії, новини.

Були вибрані технологічні компанії з ринку NASDAQ, до яких маю особистий інтерес. Для проекту створено кілька технічних індикаторів подібним чином, який ви знайдете в yahoo finance. Давайте поглянемо на історичні дані Tesla, щоб зрозуміти, з чим мається справа.

Смуги Боліндже́ра показують смуги або діапазони волатильності, в якому певна ціна цінного паперу рухається вгору або вниз. Волатильність відображається на основі стандартного відхилення для конкретного цінного папера, яке позначається верхньою та нижньою лінією/смугою, оскільки стандартне відхилення є мірою волатильності. Далі відображено приклад вигляду індикатору «Смуги Боліндже́ра» (рисунок 1.1).

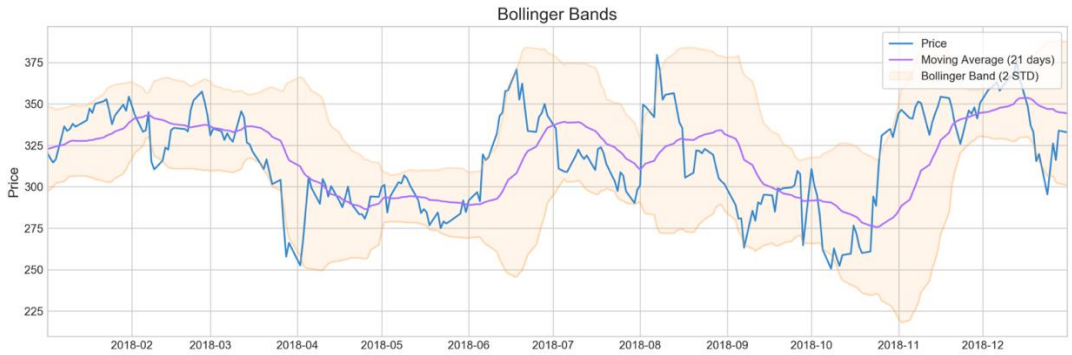

Рисунок 1.1 – Індикатор «Смуги Боліндже́ра»

Розбіжність конвергенції ковзного середнього (MACD) – це індикатор моментуму, який показує співвідношення між двома ковзними середніми ціни цінного паперу. Зазвичай, коли MACD (фіолетова лінія) перевершує сигнал (помаранчева лінія), це означає, що акція зростає і деякий час буде продовжувати рости. Нижче відображено приклад вигляду індикатора «MACD» (рисунок 1.2).

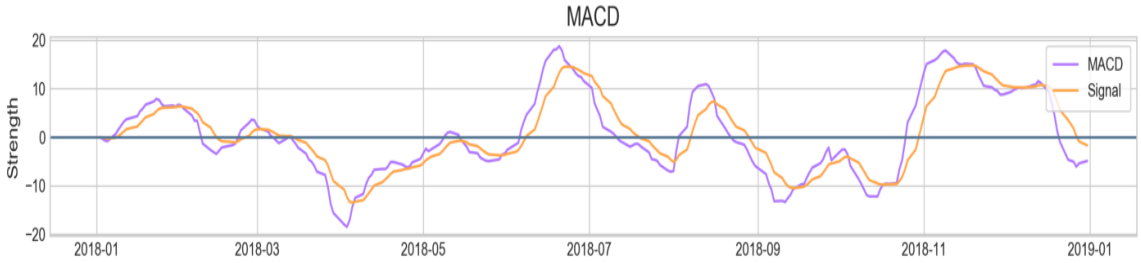

Рисунок 1.2 – Індикатор «MACD»

Індекс відносної сили (RSI) – ще один індикатор імпульсу, який може визначити, перекуплені чи перепродані акції. Він коливається від 0 до 100, але зазвичай звертається увага, коли індекс наближається до 20, і це буде сигналом для покупки. Якщо наближається до 80 – краще швидше продати. Далі відображено приклад індикатора «RSI» (рисунок 1.3).

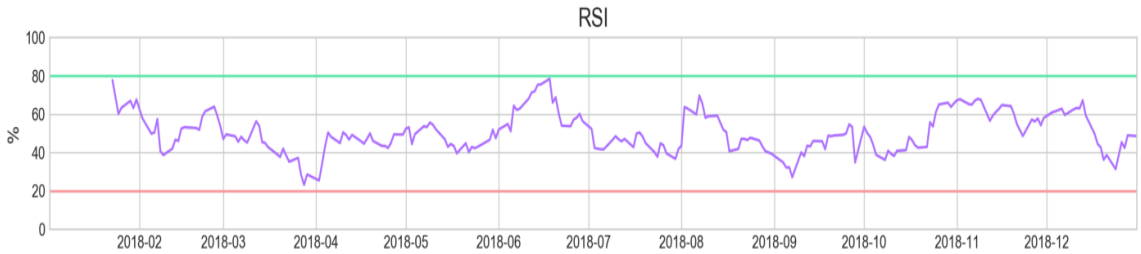

Рисунок 1.3 – Індикатор «RSI»

Це наглядно демонструює, що трейдер у своїй роботі дуже часто використовує певні індикатори, та, відшовхуючись від них будує свою стратегію та робить певні дії щоб фіксувати прибуток і мінімізувати втрати. Це дуже корисно для нашої майбутньої ІСППР, яка також буде покладатися

на ці індикатори задля того щоб зробити певні висновки і побудувати власну стратегію підсказок для трейдера. Таким чином, ІСППР та трейдер будут на одній хвилі, використовуючи спільні показники аналізу ринку.

1.2 Подання знань в інтелектуальних системах

1.2.1 Підходи до подання знань в інтелектуальних системах

Філософія визначає знання як «відображення об'єктивних властивостей і зв'язків світу». Знання є інформаційною основою інтелектуальних систем, тому що інтелектуальні системи завжди порівнюють зовнішні ситуації зі своїми власними знаннями і керуються ними при прийнятті рішень. Знання – це систематизована інформація, яку можна певним чином поповнювати, і не менш важливо, що на її основі можна отримати нову інформацію, тобто нові знання.

Існує багато підходів до визначення поняття «знання». На сучасному етапі парадигми (конкретні набори ключових принципів), характерні для символічних підходів, можна вважати домінуючими парадигмами, що лежать в основі найвідоміших моделей представлення знань у системах штучного інтелекту. Цю парадигму можна охарактеризувати як вербальну дедуктивність або вербальну логіку, залежно від певних факторів. Інформаційні одиниці подаються в усній формі, тобто в наближеній до словесній формі, у вигляді чітко сформульованого твердження або набору фактів. Основним механізмом отримання нової інформації на основі наявної інформації є дедукція, тобто висновки від загального до часткового. Такий дедуктивний підхід ідеальний з логічної точки зору. В його основі лежать транзитивність імплікації (якщо з a випливає b, з b випливає c, то з a випливає c і дія квантора загальності (якщо деяка властивість P виконується для будь-якого елемента множини M, а $x \in M$, то P виконується для x. Але вербально дедуктивне задання знань не є повним, оскільки: дедуктивний

висновок не виступає єдино можливим. Мислення людини багато в чому є рефлекторним, інтуїтивним. Воно, як правило, спирається на підсвідомі процеси.

Люди часто роблять висновки за аналогією та асоціаціями. Ці висновки не завжди правильні, але вони значною мірою доповнюють процес дедуктивного мислення. Без підсвідомості це було б неможливо (як правило, відкриття - це підсвідоме породження гіпотез, перевірених апріорі або експериментально). Не всі знання є усними. Так, чітко збережених у пам'яті людини тверджень немає. Відомо, що в основі діяльності мозку людини лежить передача сигналу між нейронами. Часто людина не може сформулювати свої знання. Наприклад, ми всі знаємо, що таке «стіл». Але просити людей дати визначення цьому поняттю може призвести до проблем.

Таким чином, поняттями можна оперувати і не знаючи чіткого їх визначення. Типовими є слова: "Я не можу пояснити чому, але мені здається». Тому необхідно розвивати інші моделі знань, окрім вербально-дедуктивних. Наприклад, у рамках конекціоністського підходу можна розглядати моделі на основі однорідного поля знань. Однорідним полем знань називається сукупність простих однорідних елементів, які обмінюються між собою інформацією: нові знання народжуються на основі певних процедур, визначених над полем знань.

1.2.2 Дані та знання

Сучасний етап розвитку інформатики, як відомо, характеризується еволюцією моделей даних у напрямку переходу від традиційних (реляційних, ієрархічних, мережевих) моделей до моделей знань. Знання, як і всю інформацію, можна розглядати як дані. Проте знання – це високоорганізовані дані, що характеризуються специфічною внутрішньою структурою та добре розвиненими зв’язками між різними інформаційними одиницями. Інший фундаментальний аспект полягає в тому, що інформаційні

системи, засновані на знаннях, повинні мати можливість отримувати нові знання на основі існуючих знань. Це вже доведено прикладами дедуктивного виведення нових знань із ясних фактів.

Розширена частина БЗ складається з фактів, що чітко запам’ятовуються, а навмисна – це частина правил, яка дозволяє засвоєння нових фактів. Наявність розгорнутої та навмисної частини є однією з головних ознак, яка відрізняє знання від звичайних даних. Інклюзивні зв’язки, які описують базу даних, часто називають правилом або набором правил, яким дотримується кожен запис у базі даних.

Приклад. Розглянемо базу даних (БД) деканату, що містить дані про те, які курси прослухав кожний студент. Відомо, що жоден студент не має права слухати курс "Бази 18 даних", якщо він не прослухав курсу "Основи програмування". Нехай у базі даних зберігається інформація про Іванова, Петрова та Сидорова, які прослухали обидва курси. Екстенсіональна частина: Прізвище Дисципліна Іванов Бази даних Петров Бази даних Сидоров Бази даних Інтенсіональна БД (або її вже можна назвати БЗ) може мати такий вигляд: Правило: Якщо Прослухав (X, Бази даних), то Прослухав (X, Основи програмування). Наявність такого правила дозволяє скоротити базу знань: твердження, істинність яких можна встановити за допомогою правил, не обов’язково запам’ятовувати в явному вигляді.

Пошук в інтенсіональній БЗ складається з двох етапів: пошук потрібного факту в екстенсіональній частині; дедуктивне виведення факту на основі правил інтенсіональної частини. Реалізовувати інтенсіональні правила можна навіть у рамках реляційної моделі даних шляхом програмування відповідних запитів. Знання можуть бути неповними. Це означає, що для доведення або спростування певного твердження може не вистачати інформації. У багатьох системах логічного виведення прийнято постулат замкненості світу: на запит про істинність деякого твердження система відповідає "так" тоді і тільки тоді, коли його можна довести; якщо ж довести неможливо, система відповідає "ні".

Водночас "неможливо довести через нестачу інформації" і "доведено, що ні" – це зовсім не те саме. З огляду на це бажано, щоб експертна система запитувала у користувача про факти, яких не вистачає. Знання можуть бути недостовірними. Наприклад, на виготовлення продукції можуть впливати випадкові чинники (об’єктивна невизначеність) або експерт може бути не зовсім упевненим у деякому факті чи правилі (суб’єктивна невизначеність).

У процесі логічної побудови необхідно враховувати недостовірність знань і недостовірність наявних фактів. Звичайно, можна просто відкинути факти та правила висновків, які викликають запитання, але цінну інформацію потрібно відкинути. Крім того, експертним системам часто доводиться мати справу з неточно визначеними поняттями, такими як «великий» і «маленький».

1.2.3 Властивості та моделі знань

Сформульовані такі особливості знань, які відрізняють їх від звичайних даних: внутрішня інтерпретованість – кожна інформаційна одиниця повинна мати унікальне ім’я, за яким інформаційна система її знаходить, а також відповідає на запити, в яких це ім’я згадується; структурованість – знання повинні мати гнучку структуру; одні інформаційні одиниці можуть включатися до складу інших (відношення типу "частина – ціле", "елемент – клас"); зв’язність – в інформаційній системі повинна бути передбачена можливість встановлення різних типів зв’язків між різними інформаційними одиницями (причинно-наслідкові, просторові та ін.); семантична метрика – на множині інформаційних одиниць корисно задавати відношення, які характеризують ситуаційну близькість цих одиниць; активність – виконання програм в інтелектуальній системі повинно ініціюватися поточним станом бази знань.

Інші властивості знань часто відфільтровуються, наприклад масштабування. Це означає, що формально різні поняття фактично

з'являються на одній шкалі понять, різні точки якої відповідають силі одного і того ж чинника. Тому температура може бути вище або нижче. Це породжує такі поняття, як «холодний», «теплий» і «гарячий», щоб формалізувати завдання знання в системах штучного інтелекту.

Модель знань – це фіксована система формалізму (понять і правил), згідно з якою інтелектуальна система зберігає знання в пам’яті та виконує над ними операції. Нам потрібна модель розподілу знань. Створити спеціальну мову для опису знань і маніпулювання ними. Формалізувати процедуру порівняння нових знань з наявними. Для формалізації механізму логічного виведення.

Як уже зазначалося, найбільш розробленими на сучасному етапі є моделі присвоєння знань, засновані на парадигмі лінгвістичної дедукції. Найвідомішими з них є чотири класи моделей: семантичні мережі, фреймові, логічні,продукційні. Лінгвістичні дедуктивні моделі розподілу знань мають багато спільних рис, тому можна припустити, що всі вони мають єдину концептуальну основу.

1.2.4 Проблема винятків

З успадкуванням пов’язана дуже серйозна проблема – проблема винятків, яка полягає в тому, що деякі підкласи можуть не успадковувати ті чи інші властивості надкласів. Якщо казати другими словами, характерні риси класу успадковуються всіма його підкласами, Але деякі є винятками. Нехай відомо, що літають всі птахи, крім пінгвінів. Існують деякі інші види птахів, які не літають, але для наших цілей це не має суттєвого значення. Якби це твердження відразу потрапило до БЗ саме в такому вигляді, особливих проблем не виникало б. Хоча і в цьому випадку слід було б передбачити належну обробку винятків.

Експерт не завжди може сформулювати свої знання в явному вигляді. Зокрема, він може не знати або не пам’ятати всіх винятків, отже він може

спочатку включити до БЗ твердження про те, що всі птахи літають, а потім пригадати, що пінгвіни не літають, і додати це до БЗ, що в результаті ми б могли отримати БЗ, подібну до такої: Усі птахи літають. Ластівка є птахом. Юкко є ластівкою. Пінгвін є птахом. Пінгвіни не літають. Бакс є пінгвіном. Якби три останні твердження не були включені до бази знань, системі просто дійшла б хибного висновку, що Бакс літає. Але включення даних відомостей до бази знань ще більше ускладнює ситуацію. Система знань стає суперечливою: з одного боку, система повинна дійти висновку, що Бакс літає, а з іншого – що Бакс не літає. У даному разі кажуть про втрату монотонності дедуктивної системи.

1.3 Системи підтримки прийняття рішень

Процес прийняття рішення здійснюється у кілька основних етапів: постановки задачі, формування рішень, вибору рішення.

1.3.1 Основні етапи прийняття рішень

Етап визначення проблеми складається з етапів аналізу та діагностики проблеми та визначення цілей вирішення. На цьому етапі ми визначаємо та описуємо проблемну ситуацію та збираємо відповідну інформацію та дані. Визначається мета рішення, яке необхідно прийняти. Це дозволяє спрямувати ваш пошук рішень і усунути ті, які не відповідають вашим цілям.

Етап прийняття рішення складається з етапів, на яких формулюються обмеження та критерії прийняття рішення та визначаються варіанти рішення. На цьому етапі визначаються обмеження для розрізнення прийнятних і неприйнятних варіантів і критерії, які сприяють вибору найкращого варіанту рішення.

Етап вибору рішення складається з етапу оцінки альтернатив і етапу вибору остаточного рішення. На цій завершальній стадії варіанти

оцінюються з набору прийнятних альтернатив відповідно до обраних критеріїв, після чого остаточно вибирається рішення. Альтернативні значення зазвичай не збігаються, але вони однакові за умови, що є одна непряма перевага.

1.3.2 Послідовність прийняття рішень

Процес прийняття рішення складається з таких кроків:

- визначення цілей, критеріїв оптимальності, критеріїв добору „кандидатів” на отримання ресурсів;
- формування множини допустимих альтернатив;
- вибір методів розв’язання задачі;
- порівняння та упорядкування множини альтернатив за обраними критеріями;
- добір кращих варіантів за критерієм оптимальності та вибір рішення.

1.3.3 Помилки прийняття рішень

У процесі прийняття рішень щодо ОПР часто трапляються помилки. Деякі з найпоширеніших типових помилок включають: Прийняття так званих односторонніх рішень. Немає системного підходу до прийняття рішень. При виборі варіанту пріоритет має «звичайний» вибір. Розглядаються тільки позитивні варіанти, можливі ризики не враховуються.

Емоції керують вами у прийнятті рішень. Рішення приймаються імпульсивно. Рішення приймаються поспішно. Рішення керуються припущеннями, прихованими бажаннями та хибними припущеннями, а не надійною та об’єктивною інформацією. Наявні факти неправильно тлумачаться в рішеннях. Прийняття нерелевантних рішень (неправильні рішення, передчасне або передчасне виконання, на жаль, це особливість сучасної української економіки тощо).

1.3.4 Емоції в прийнятті трейдерських рішень

Звільнення від емоційного впливу, мабуть, одне з найскладніших завдань у трейдингу. Це частково пояснює той факт, що результати торгівлі на папері не завжди збігаються з результатами, які трейдери отримують під час торгівлі реальними грошима. Але це нормально. Ви не можете усунути емоції на 100%, але ви можете усвідомлювати їх присутність і розуміти, коли вони починають ставати на шляху вашого успіху. Намагайтеся бути максимально «роботом» під час торгівлі. Зрештою, не дивно, що багато компаній використовують алгоритми, а не живих трейдерів.

Так як же перемогти емоції в торгівлі?

Крок 1: Визначте емоцію і її ефект. Наприклад, припустимо, що ви виробили продуманий план і відкрили позицію в лонг по XYZ в очікуванні пробою. Хоча ваш план був правильним, ви занадто рано вийшли з позиції, тому що акція перед пробоєм сходила вниз. В даному випадку, вашій емоцією був страх, а її ефект полягав в передчасному виході з позиції.

Крок 2: Проаналізуйте проблему. Ви знаєте, що страх змусив вас вийти з позиції, незважаючи на наявність ідеального плану. Чим же був викликаний страх? Ви купили занадто багато акцій? Проявили нетерпіння? Засумнівалися в собі? Після виявлення причини емоції, можна сконцентруватися на усуненні подібної пастки в майбутньому.

Крок 3: Усунення проблеми. Припустимо, ви зрозуміли, що страх змусив вас передчасно закрити операцію, тому що ви купили занадто багато акцій. Тепер від проблеми легко позбутися. Наступного разу просто купуйте менше акцій, коли входите в угоду.

Слід враховувати, що процес прийняття рішень людиною має певні обмеження стосовно можливості аналізу, оброблення даних, одержання рішень прогнозованої якості та швидкості прийняття обґрунтованих рішень. Робота ОПР обмежена як відносинами між окремими особами, так і внутрішніми психологічними і фізіологічними причинами. Людина має

можливість одночасно оперувати лише обмеженим числом операндів і понять, щонайбільше 7 ± 2. Крім того, при аналізі і розв'язанні багатокритеріальних задач особи ОПР досить часто проявляють мінливість, невпевненість, нелогічність, намагання суттєво спростити задачу, тож у таких випадках можливості обчислювальних машин значно перевищують можливості людини, тому така ситуація призвела до створення напряму розробки систем та методологій, які мають можливість об'єднати переваги людини і комп'ютера та компенсувати їх недоліки, тобто людино-машинних систем.

1.3.5 Основні напрями створення систем підтримки прийняття рішень

Серед сучасних напрямів розробки людино-машинних систем – системи автоматичного керування, експертні системи та СППР. Найбільш придатними для розв'язання багатьох задач, зокрема задачі розподілу ресурсів, виявляються СППР. Саме за допомогою СППР ОПР має можливість безпосередньо за допомогою обчислювальних засобів проектувати, порівнювати та обирати альтернативні варіанти рішень у різноманітні способи.

У сучасних умовах застосування нових та альтернативних підходів до формування та прийняття якісних рішень неможливо уявити без використання електронних інформаційних систем та інформаційних технологій. Причиною широкого впровадження інформаційних технологій є їхня зростаюча роль майже в кожному секторі суспільства та зростаючий потенціал цих технологій. Застосування сучасних підходів, заснованих на інформаційних технологіях, дає можливість використовувати обчислювальні потужності комп'ютерів для обчислення, обробки, аналізу та прогнозування даних у режимі реального часу для підтримки прийняття рішень. За цих обставин автоматизована не стільки ручна праця, скільки інтелектуальна. Коротше кажучи, людино-машинні системи стають все більш

інтелектуальними системами підтримки прийняття рішень. Такі системи для багатьох практичних завдань виявилися більш ефективними порівняно з іншими, тому останній час вони стають все більш популярними і все частіше приходять на заміну людям.

1.3.6 Системи підтримки прийняття рішень із передбаченням

У багатьох випадках автоматизації прийняття рішень доцільно використовувати СППР, які дозволяють приймати важливі рішення, керуючись подіями, які ще не здійснилися, тож такі СППР надають можливість розробляти декілька можливих сценаріїв типу «що було б, якби», визначати оптимальні дії тощо. СППР інтегрують в собі такі якості, які роблять їх не тільки дуже корисними для системних задач управління і прийняття рішень. По суті вони стають незамінними інструментами аналізу даних в сучасних умовах економічного розвитку.

Системи СППР створюються для підтримки прийняття рішень ОПР в складних та слабоструктурованих ситуаціях, оскільки історично виник ряд різних напрямків діяльності стосовно прийняття рішень. Тож, розрізняють кілька видів СППР, що відображають основні аспекти процесу прийняття рішень: аналіз рішень (Decision Analysis); обчислення (визначення) рішень (Decision Calculus); дослідження рішень (Decision Research) та процес впровадження (Implementation Process). Кожний із зазначених напрямів постає самостійною перспективою розвитку СППР, проте в „чистому вигляді” вони практично не зустрічаються.

1.3.7 Вимоги до сучасних систем підтримки прийняття рішень

За сучасним станом розвитку система СППР має відповідати таким вимогам: оперує зі слабоструктурованими рішеннями; використовує

слабоструктуровані та нечіткі дані; підтримує як взаємозалежні, так і послідовні рішення; може застосовувати знання; підтримує моделювання та прогнозування; може бути легко побудована, якщо може бути сформульована логіка конструкції СППР; повинна підтримувати три фази процесу прийняття рішень: інтелектуальну частину, проектування та вибір; система призначена для ОПР різного рівня; повинна бути простою у застосуванні та модифікації; система може бути адаптована до індивідуального та групового застосування; СППР підтримує різні стилі та методи рішень, що можуть бути корисними при застосуванні групою ОПР; система виявляє гнучкість і адаптується до змін в організації та в її оточенні; СППР дозволяє людині керувати процесом прийняття рішення за допомогою комп’ютера, а не навпаки; система підтримує еволюційне застосування та легко адаптується до мінливих вимог; застосування СППР підвищує ефективність процесу прийняття рішень.

1.3.8 Класифікація СППР

Єдиної класифікації СППР зараз не існує. Розглянемо деякі основні підходи до поділу СППР на види за різними характеристиками.

На рівні користувача виділяють такі види СППР: кооперативна – дозволяє ОПР змінювати, поповнювати або поліпшувати рішення, пропоновані системою, надсилаючи потім ці зміни в систему для перевірки; пасивна – допомагає у процесі ухвалення рішення, але не може внести пропозицію, яке рішення прийняти; активна – може зробити пропозицію, яке рішення варто вибрати.

На технічному рівні розрізняють такі СППР: персональна настільна СППР – мала система, що обслуговує лише один комп'ютер користувача; СППР рівня підприємства – підключена до великих сховищ даних і обслуговує багатьох менеджерів підприємства.

На концептуальному рівні відрізняють такі типи СППР: керована

повідомленнями (Communication-Driven DSS) – підтримує групу користувачів, що працюють над виконанням загальної задачі; керована даними (Data-Driven DSS, Data-oriented DSS) – в основному орієнтується на доступ і маніпуляції з даними; керована документами (Document-Driven DSS) – здійснює пошук і маніпулювання неструктурованою інформацією, заданої в різних форматах; – керована знаннями (Knowledge-Driven DSS) – забезпечує рішення задач у виді фактів, правил, процедур; керована моделями (Model-Driven DSS) – забезпечує доступ і маніпуляції з математичними моделями (статистичними, фінансовими, оптимізаційними, імітаційними).

Відзначимо, що деякі OLAP-системи, які дозволяють здійснювати складний аналіз даних, можуть бути віднесені до так званих гібридних СППР, що забезпечують моделювання, пошук і оброблення даних та відповідають властивостям декількох видів СППР. Залежно від оброблюваних даних СППР можна поділити на оперативні та стратегічні.

1.3.9 Оперативні і стратегічні СППР

Оперативні СППР призначені для негайного реагування на зміни поточної ситуації у керуванні фінансово-господарськими процесами компанії, об’єднання, галузі чи держави, тому такі СППР називають Виконавчі Інформаційні Системи (Executive Information Systems) і зазвичай системи надають кінцеві множини звітів, побудовані за даними із транзакційної інформаційної системи оперативного обліку підприємства. Вони забезпечують адекватне відображення в режимі реального часу. Це стосується основних аспектів виробничої і фінансової діяльності підприємства.

Для оперативних СППР характерними є наступні риси. Звіти грунтуються на стандартних, для організації, запитах, кількість яких відносно невелика, СППР подає звіти у максимально зручному вигляді, що включає надання, окрім таблиць, ділової графіки, мультимедійної інформації, СППР

зазвичай орієнтовані на конкретну сферу, наприклад, фінанси, маркетинг, керування ресурсами.

Стратегічні СППР орієнтовані на аналіз значних обсягів різнорідної інформації. Ця інформація збирається з різних джерел. Вважається, що найважливішою метою цих СППР є пошук найбільш раціональних варіантів розвитку бізнесу компанії із урахуванням впливу різних факторів, зокрема кон'юнктури цільових, для компанії, ринків, зміни фінансових ринків і ринків капіталів, зміни у законодавстві тощо. Стратегічні СППР будують на принципах багатовимірного подання та аналізу даних у системах OLAP (On-line Analytical Processing).

1.3.10 Підходи до проектування архітектури СППР

Існують різні підходи до проектування СППР. Системний аналіз цих підходів дозволяє узгодити існуючі вимоги до СППР з вибраною архітектурою СППР, а проектування СППР передбачає визначення типу архітектури і подальшу деталізацію основних функцій системи оброблення даних та генерування результатів (СОДГР) СППР, тому розробку системи СОДГР починають з вибору та опису алгоритмів відповідно до методів оброблення даних, форм і засобів подання даних і знань, які можуть використовуватись в СППР.

На завершальній стадії проектування важливими є функції системи подання результатів, форми подання даних і результатів обробки даних тощо. Розгляд цих питань і складає основу проектування архітектури СППР. За категорією класифікації – концептуальна модель (схема) виділяють: інформаційний підхід до проектування, підхід, що грунтується на знаннях та інструментальний підхід.

З точки зору інформаційного підходу СППР належить до класу інформаційних систем, основною метою яких є вдосконалення характеру діяльності управлінського персоналу корпорації (удосконалення самого

характеру, який шляхом використання інформаційних технологій Ми надамо вам необхідну інформацію під час. У межах цього підходу запропоновано дві моделі СППР: „Спрага” та еволюціонуюча модель.

Бази даних СППР включають як кількісну, так і якісну інформацію, що надходить із різних джерел. Засоби створення і ведення БД повинні надавати такі можливості: об’єднувати різні джерела інформації, використовуючи процедуру їх “добування” даних; представляти логічну структуру у термінах користувача; надавати повний набір функцій управління даними.

База моделей повинна забезпечувати гнучкість моделювання. Загалом, за рахунок використання готових блоків моделей і підпрограм. Управління моделями дає, наприклад, такі можливості, як каталогізування та обслуговування широкого спектру моделей, які підтримують всі рівні управління; легко і видко створювати нові моделі; пов’язувати моделі з відповідними БД. Подальшим розвитком СППР є еволюціонуюча СППР.

Крім інтерфейсу користувача, бази даних, бази моделей ця система включає базу текстів і базу правил. Завдяки цьому розширюється їх функціональні можливості, а інформаційна база СППР дає змогу використовувати як менш структуровані види інформації (наприклад, тексти звичайною мовою), так і більш структуровану інформацію (наприклад, правила подання знань, евристичні процедури).

Одним із перспективних напрямів розвитку систем підтримки прийняття рішень є об’єднання технологій підтримки рішень і технології.

Елементи штучного інтелекту, такі як використання звичайної мови для спілкування з системою, методологія експертних систем, інженерія знань і комп’ютерних мов штучного інтелекту знайшла застосування у трьох базових компонентах СППР: БД і СУБД, база моделей і система управління базою моделей, інтерфейсі користувача. Але є концепції створення СППР, в яких система знань в СППР виступає як один з визначальних чинників. Відмінною особливістю СППР, що ґрунтуються на знаннях, є явне виділення нового аспекту підтримки рішень – спроможність “розуміти” проблему,

тобто здатність прийняти запит користувача, зібрати відповідну інформацію і підготувати звіт.

Мовна система забезпечує зв’язок між користувачем і всіма компонентами комп’ютерної системи, бо за її допомогою користувач формулює проблему і керує процесом її рішення. Використовується синтаксичні та семантичні засоби, запропоновані мовною системою. Система знань вміщує інформацію стосовно предметної області, а типи цих систем відрізняються за характером подання в них даних і використаними моделями формалізації знань (ієрархічні структури, графи, семантичні мережі, фрейми, обчислення предикатів тощо).

Система обробки завдань – це механізм, який з'єднує мовну систему і систему знань. Цей процесор задач забезпечує збір інформації, формулювання моделі та її аналіз. Він бере опис проблеми, виконаний відповідно до синтаксису мовної системи, і використовує знання відповідно до правил, прийнятих системою знань, щоб створити інформацію, необхідну для підтримки прийняття рішень.

Структурну схему СППР, яка грунтується на знаннях, зображено нижче (рисунок 1.5). Проблемний процесор є динамічною компонентою СППР, що відображає (моделює) поведінку особи, яка вирішує проблему. Він повинен мати можливість інтегрувати інформацію від користувача через мовну систему і систему знань і може перетворювати формулювання проблеми у докладні процедури, виконання яких дає відповідь (розв’язок задачі). У складніших випадках проблемний процесор повинен вміти формулювати моделі, необхідні для вирішення поставленої проблеми.

Торгівля штучним інтелектом зараз процвітає, оскільки його функції ідеально підходять для світу фінансів. AI-рішення здатні швидко підраховувати цифри та приймати оптимальні рішення на основі великих масивів даних, що дуже застосовно до реалій фондового ринку. Машинне навчання для трейдингу дозволяє фінансовим компаніям отримати повне уявлення про ситуацію на фондовому ринку за допомогою поглибленого

безперервного аналізу коливань цін на акції та обробки неструктурованих даних. Він також виявляється корисним для ідентифікації складних торгових моделей, інформуючи про правильні рішення про продаж/купівлю в режимі реального часу.

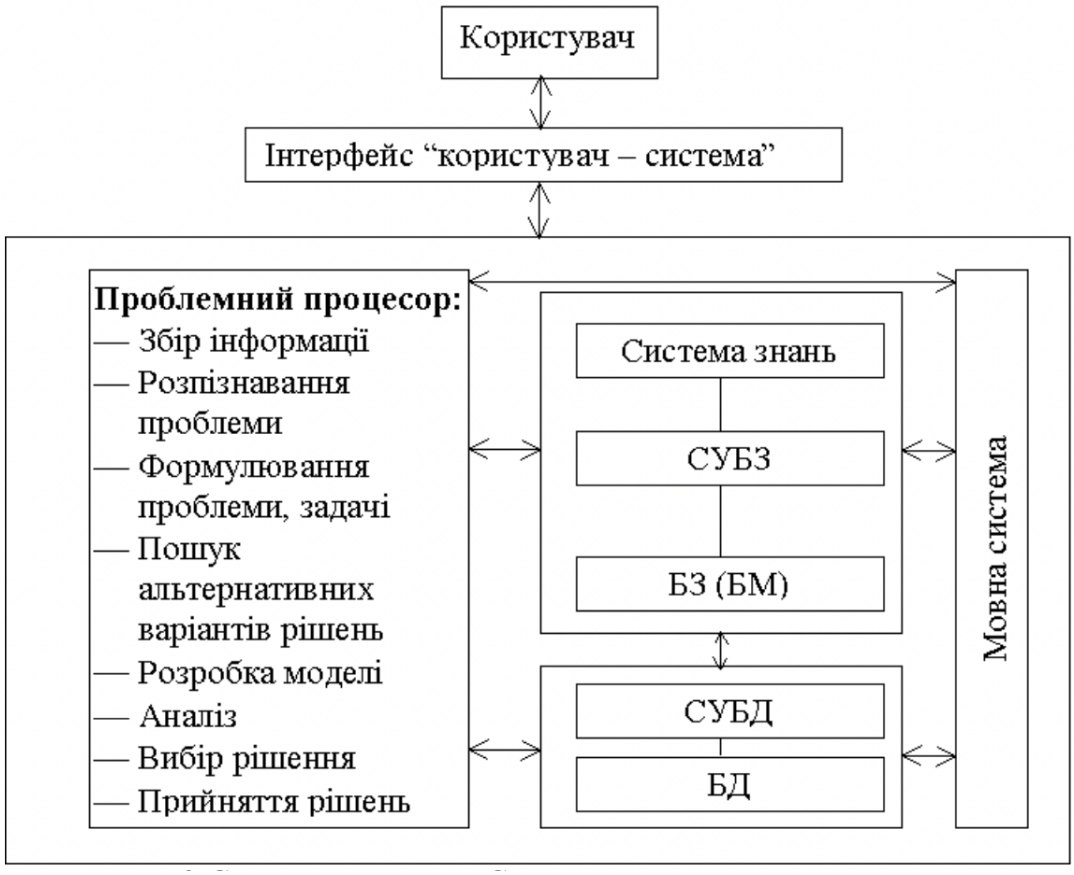

Рисунок 1.5 – Структурна схема СППР, яка грунтується на знаннях

1.4 Аналіз існуючих рішень

Торгові стратегії штучного інтелекту стають все більш складними, оскільки системи навчаються на власному досвіді. Таким чином, сьогодні вони пропонують незаперечні переваги користувачам.

Програмне забезпечення штучного інтелекту аналізує історичні дані та бачить повторювані закономірності в динаміці цін на акції, щоб визначити правильну стратегію для інвестора.

Програми штучного інтелекту можуть включати дані з новин і

соціальних мереж у свої аналізи, приймаючи рішення на основі набору даних, набагато більшого, ніж це дозволив би звичайний технічний аналіз.

Враховуючи кожну мілісекунду під час торгівлі акціями, програми AI для торгівлі можуть заощадити ваш час і гроші, сприяючи миттєвому прийняттю рішень і дій.

Ці переваги говорять на користь використання AI у вашій інвестиційній діяльності. Ось кілька способів застосування штучного інтелекту в повсякденній торгівлі. Найкращим результатом використання ШІ в торгівлі на фондовому ринку є торговий сигнал. Такі сигнали є результатом аналізу великих даних систем штучного інтелекту щодо конкретних активів, які дають точні рекомендації для прийняття успішних торгових рішень, таких як найкраща ціна входу, стоп-лосс і норма прибутку. Таким чином, ці сигнали дозволяють набагато краще керувати ризиками активів, не даючи трейдерам зайти занадто далеко нижче маржі збитків у надії на відновлення ціни.

Торгові сигнали генеруються системами штучного інтелекту на основі розширеного аналізу численних показників, таких як динаміка цін, оцінка валют і навіть аналіз даних про конкретні активи з новин і соціальних мереж. Технічний аналіз динаміки цін на акції також включено в набір даних.

Де взяти торгові сигнали AI? Як правило, компанії, які інвестували в налаштування власного програмного забезпечення для аналізу фондового ринку в реальному часі, не розголошують свої секрети та продають сигнали на основі передплати. Ось деякі плюси та мінуси покладатися на торгові сигнали.

Серед плюсів можна виділити багато речей.

Ви витратите набагато більше часу на аналіз трендів і динаміку цін на активи. Використання сигналу штучного інтелекту для торгівлі акціями для певного активу економить час для інших важливих дій.

Вам не потрібно витрачати місяці чи роки на вивчення науки біржової торгівлі, щоб заробити на фондовому ринку. Системи штучного інтелекту

вже мають необхідний досвід.

Незважаючи на те, що машини ніколи не перевершать людей, вони все одно добре справляються з аналізом великих даних. Вони обробляють дані з різних джерел, щоб надати надійні підказки.

Але існують і мінуси.

У більшості випадків користувачі все ще повинні здійснювати дії з продажу/купівлі, тоді як сигнали є лише рекомендацією інформаційного характеру.

Сигнали є загальними, і не всі вони можуть відповідати вашому рівню ризику та стилю торгівлі.

Сьогодні, під час криз і крахів, фондові ринки дуже непередбачувані, і жодна машина не може вловити ці потрясіння та адаптувати свої поради до цих коливань досить швидко, щоб захистити ваші активи. Важливо пам’ятати про ширший ринковий статус кожного разу, коли ви слідкуєте за торговими сигналами, оскільки програмне забезпечення для торгівлі штучним інтелектом має обмежені знання про ринок.

Як бачите, торгові сигнали пропонують інвесторам певні переваги, але містять певні ризики, про які вам слід знати, перш ніж довіряти свої гроші машинам.

Але це все стосується кожного AI-додатку – усі вони дуже ризиковані. Проте, а які все ж таки є існуючі рішення?

Що ж, серед існуючіх рішень можна виділити таку компанії як Datrics – вона має в собі все що треба знати про існуючі рішення – Datrics це професійна компанія яка витратила на розробку AI додатків мільйони доларів, а тому тягатися з нею простому студенту дуже важко. Слід пам’ятати, що ринок акцій – це дуже серйозна сфера із дуже серйозними гравцями на ній. Мінуси у всіх таких професійних додатків однакові – вони коштують багато грошей та не мають відкритого вихідного коду. Обидва ці мінуси можна зрозуміти, ніхто не захоче показувати на яких саме AI алгоритмах працює їх додаток який приносить їм велику перевагу на ринку.

Тож це виклик нам – спробувати розробити безкоштовний додаток із відкритим вихідним кодом який зможе хоч трохи бути схожим на ті додатки, які використовують великі компанії і не хочуть ділитися кодами.

1.5 Постановка задачі

У своїй роботі трейдер весь час стикається з важливим питанням – коли йому купувати, а коли йому продавати так, щоб мало того не втратити свої гроші, але і збільшити їхню кількість. У відповідях на ці питання і визначається досвід трейдера. Існує безліч практик, які трейдери використовують для відповіді на ці питання, з якими теоретично може допомогти машинне навчання.

У процесі написання кваліфікаційної роботи треба провести дослідницьку роботу в цій темі, а саме дати відповідь на питання "чи має машинне навчання у продажах та покупках акці, щоб приносити прибуток". Також, треба використати та протестувати різні алгоритми та нейронні моделі щоб визначити найбільш оптимальний та робочий підхід у використанні машинного навчання у повсякденному житті трейдера. Використовуючи вибрану модель, у ході цкваліфікаційної роботи треба розробити безкоштовний додаток із відкритим вихідним кодом, який аналізує поведінку певних акцій за час та використовує алгоритми машинного навчання для пошуку сигналів, про які говорить поведінка певних акцій. При виявленні цих сигналів, сервіс повідомляє про це трейдеру, і радить те, що вважає найбільш підходящим для акції: що буде вигідніше зараз – купити, продати або нічого не робити.

2 СТВОРЕННЯ ІСППР ДЛЯ ТРЕЙДЕРА ФІНАНСОВОГО РИНКУ

2.1 Створення архітектури системи

Система складається з декількох основних частин або модулів, кожен з яких виконує свою унікальну функцію:

- модуль подання даних;
- модуль машинного навчання;
- модуль інформування.

Модуль подання знань до системи підключається до онлайн систем для моніторингу графіків, бере необхідну інформацію про акції які нас ціклявать у числовому вигляді і передає її до наступного модулю. Також, цей модуль відповідає за загрузку вже підготовлених тестових даних у вигляді csv таблиць до системи задля тестування різних підходів на однаковій тестовій інформації.

Модуль машинного навчання відповідає за використання поданих до системи даних. Саме у цьому модулі використовується навчання нейромережі (якщо воно необхідне), та її подальше використання. Після цього результат роботи нейромережі подається до наступного модуля.

Модуль інформування відповідає за вивід результату роботи нейромережі до кінцевого юзера. Для тестування роботи нейромережі цілком підходить звичайний консольний вивід, тому буде використаний саме він.

2.2 Порівняння моделей побудови ІССПР для трейдера

Розглянемо різні способи вирішення завдання. А саме - модель авторегресійної інтегрованої ковзної середньої, глибинні нейронні мережі, відповідність шаблону, Q-навчання та глибинне Q-навчання.

2.2.1 Модель авторегресійної інтегрованої ковзної середньої

Модель авторегресійної інтегрованої ковзної середньої (ARIMA) використовується для прогнозування даних часових рядів на основі припущення, що точки даних корельовані одна з одною. Якщо дані не корельовані – модель не зможе робити прогнози.

Стохастичний процес називають процесом ковзної середньої порядку q, якщо до загальної моделі входять лише q складових. Позначимо коефіцієнти обмеженого ряду $MA(q)$ літерою b, тоді модель ковзної середньої порядку q має вигляд:

$$y = \varepsilon_t + b_2\varepsilon_{t-1} + b_2\varepsilon_{t-2} + \cdots + b_q\varepsilon_{t-q} = \\ = \left(1 + b_1L + b_2L^2 + \cdots + b_qL^q\right)\varepsilon_t = b(L)\varepsilon', \quad (2.1)$$

де ε_t – випадкова величина, білий шум;

$b(L) = \left(1 + b_1L + b_2L^2 + \cdots + b_qL^q\right)$ – лінійний оператор;

$(q+1)$ – кількість невідомих параметрів, що треба оцінити на підставі вибіркових спостережень.

Процес – стаціонарний, оскільки є окремим випадком загальної лінійної моделі, а саме, φ_1 включно до $j = q$ дорівнюють b_j, решта φ_1 дорівнюють нулю.

Операторний багаточлен $b(L)$ можна розкласти на множники, використовуючи корені рівняння $b(L) = 0$. Отже, лінійний оператор $b(L)$ можна записати у вигляді:

$$b(L) = 1 + b_1L + b_2L^2 + \cdots + b_qL^q = b\prod_{k=t}^{q}(L - Z_j), \quad (2.2)$$

де Z_j - корені рівняння.

$MA(q)$-процес, відповідно, має вигляд:

$$y_t = \prod_{j=t}^{q}(L - Z_j)\varepsilon_t \ . \tag{2.3}$$

За умови оберненості кожен скінченний $MA(q)$-процес може бути представлений у вигляді нескінченного авторегресійного процесу:

$$y_t = a_1 y_{t-1} + \ a_2 y_{t-2} + \cdots + \varepsilon_t \tag{2.4}$$

Автоковаріація та дисперсія $MA(q)$-процесу відповідно дорівнюють:

$$Y_k = \left(-b_k + \sum_{j=q}^{q-k} b_j b_{j+k}\right)\sigma_\varepsilon^2, \tag{2.5}$$

$$k = 1, \dots q, k > q$$

$$Y_0 = D(y_t) = \sigma_E^2\left(1 + b_1^2 + \cdots + b_q^2\right). \tag{2.6}$$

Автокореляційна функція процесу має вигляд:

$$\rho_k = \frac{y_k}{y_0} = \ \frac{-b_k + \sum_{j=1}^{q-k} b_j b_{j+k}}{1 + b_1^2 + \cdots + b_q^2}, \tag{2.7}$$

для $k > q$.

Автокореляційну функцію використовують для визначення порядку $MA(q)$-процесу. Функція автокореляції (ACF) покаже, чи точки даних мають такий зв'язок. Як працює ACF? Автокореляційна функція (ACF) із затримкою k, позначеною pk, стаціонарного стохастичного процесу, визначається як $\rho k \ = \frac{y_k}{y_0}$, де $y_k = \ cov(y_i, y_i + k)$ для будь-якого i.

Причому, y_0 є дисперсією випадкового процесу.

Середнє значення часового ряду $y_1, \dots, y_n$ є:

$$y = \frac{1}{n}\sum_{i=1}^{n} y_i. \quad (2.8)$$

Ми побачимо результати цього тестування у 3 розділі, а поки перейдемо до глибинних нейронних мереж.

2.2.2 Глибинні нейронні мережі

Коли трейдери використовують історичні дані разом із технічними індикаторами для прогнозування руху акцій, вони шукають знайомі моделі. Деякі типи нейронних мереж чудово знаходять закономірності та мають різноманітні застосування для розпізнавання зображень або обробки тексту.

Для нас можуть підійти згорткові та рекурентні мережі.

Згорткові нейронні мережі (CNN) - це клас штучних нейронних мереж, які є більш вдосконаленими моделями в порівнянні з багатошаровим персептроном. CNN об'єднує згорткові шари і так звані шари пулінгу, котрі використовуються для зменшення розмірності зображення, з повнозв'язними шарами, використовуваними в багатошаровому персептроні. Шари згортки і пулінгу необхідні для отримання ознак вхідних даних, а повнозв'язні шари використовуються для перетворення витягнутих і оброблених ознак у вихідні сигнали мережі.

Згортковий шар виконує математичну операцію, яку називають згорткою. В даному наборі даних MNIST всі цифрові зображення зберігаються в двовимірному форматі, де двовимірна матриця меншого розміру, звана в термінах згорткових нейромереж ядром, або фільтром, використовується для виконання операції згортки з кожним нейроном згорткового шару. По суті, згортка є не що інше, як скалярний добуток між ядром і вхідним тензором. Доповнюючи нулі в масивах зображень іноді необхідні для здійснення згортки, щоб зберегти пропорції в розмірах фільтра і тензора, а шари пулінгу дозволяють зменшити розмірність зображення і в той же час використовувати меншу кількість параметрів.

Так само, як і в архітектурі багатошарового персептрону, в CNN використовується алгоритм зворотного поширення помилки для вивчення просторової ієрархії ознак і коригування ваг нейронів. Як правило, для навчання моделі CNN потрібне використання графічних процесорів через велику кількість параметрів, що налаштовуються. Тепер глибинні згорткові нейронні мережі вважаються одним з найбільших досягнень в задачах класифікації зображень. Нижче наведено простий приклад архітектури згорткових нейронних мереж, призначеної для завдання класифікації цифр на зображеннях (рисунок 2.1). Для цього набору даних є десять класів, таким чином, загальне число вихідних сигналів мережі дорівнює десяти

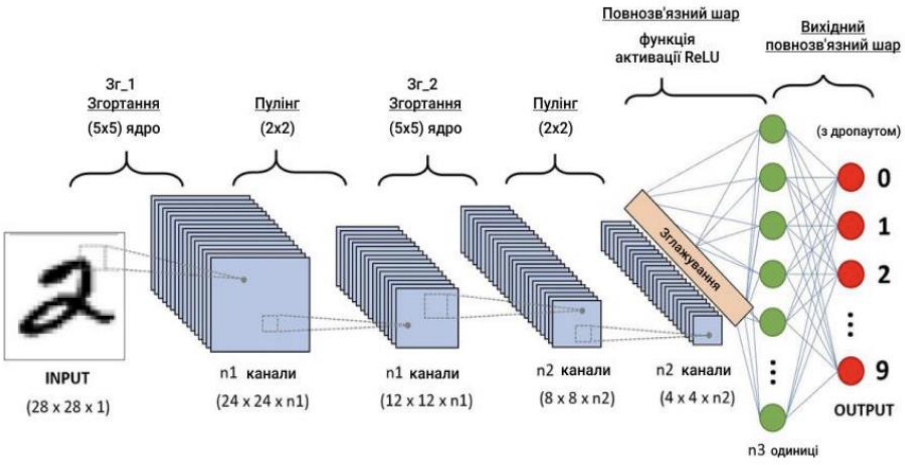

Рисунок 2.1 – Архітектура згорткової нейронної мережі

Рекурентні мережі (LSTM) також добре навчаються на основі послідовних даних, тобто часових рядів.

Традиційні нейронні мережі не можуть цього зробити, і це є серйозним недоліком. Наприклад, уявіть, що ви хочете класифікувати, яка подія відбувається в кожному моменті фільму. Незрозуміло, як традиційна нейронна мережа могла використовувати свої міркування про попередні події у фільмі, щоб інформувати про наступні.

Рекурентні нейронні мережі вирішують цю проблему. Це мережі мають

петлі в них, що дозволяє інформації зберігатися (рисунок 2.2).

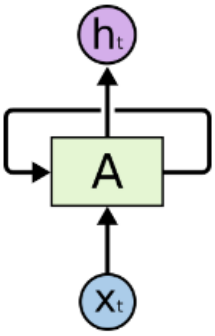

Рисунок 2.2 – Петля рекурентної нейронної мережі

На наведеній вище діаграмі частина нейронної мережі A дивиться на деякий вхідний сигнал x_t і виводить значення h_t. Цикл дозволяє передавати інформацію від одного кроку мережі до іншого.

Завдяки цим циклам рекурентні нейронні мережі здаються чимось загадковими. Однак якщо ви подумаєте трохи більше, виявиться, що вони не дуже відрізняються від звичайної нейронної мережі. Повторювану нейронну мережу можна розглядати як кілька копій однієї мережі, кожна з яких передає повідомлення наступнику. Нижче показано, що станеться, якщо розгорнемо цикл (рисунок 2.3).

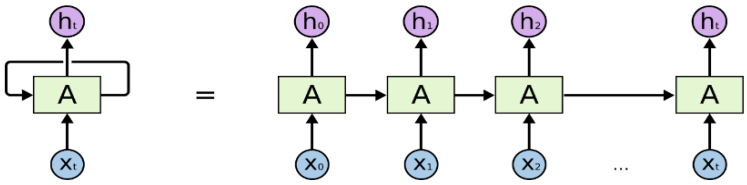

Рисунок 2.3 – Розгорнутий цикл рекурентної нейронної мережі

Така ланцюгова природа показує, що рекурентні нейронні мережі тісно пов’язані з послідовностями та списками. Це природна архітектура нейронної мережі, яка використовується для таких даних.

За останні кілька років було досягнуто неймовірних успіхів у

застосуванні RNN для різноманітних проблем: розпізнавання мовлення, моделювання мови, переклад, підписи до зображень та інше.

Розглянемо попередника LSTM – RNN. Однією з привабливостей RNN є ідея, що вони можуть зв'язати попередню інформацію з поточним завданням, наприклад, використання попередніх відеокадрів може допомогти зрозуміти поточний кадр. Якби RNN могли це зробити, вони були б надзвичайно корисними. Але чи можуть вони?

Іноді нам потрібно лише переглянути останню інформацію, щоб виконати поточне завдання. Наприклад, розглянемо на нешому прикладі модель, яка намагається передбачити наступну ціну на основі попередніх. Якщо намагатися передбачити останню цифру в послідовності «100 200 300», не потрібен додатковий контекст – цілком очевидно, що наступною цифрою буде 400. У таких випадках, коли розрив між релевантною інформацією та місцем, де вона потрібна, невеликий, RNN можуть навчитися використовувати минулу інформацію (рисунок 2.4).

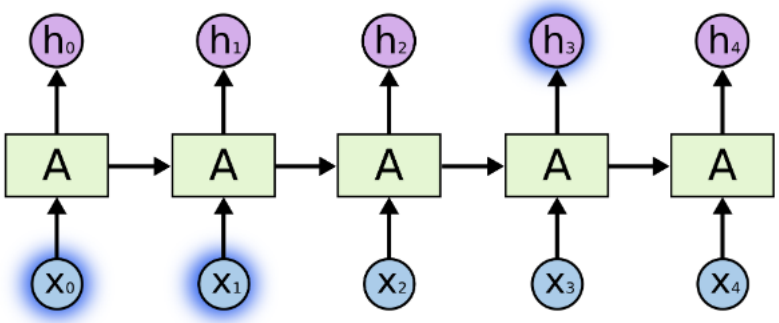

Рисунок 2.4 – Схема рекуретної нейронної мережі яка використовує попередню інформацію

Але є також випадки, коли нам потрібно більше контексту. Спробуйте передбачити останню цифру в послідовності «10 4031 1 2332 100 12 3030 900». Цілком можливо, що розрив між релевантною інформацією та точкою, де вона потрібна, стане дуже великим.

На жаль, у міру того, як цей розрив зростає, RNN стають нездатними навчитися з’єднувати інформацію (рисунок 2.5).

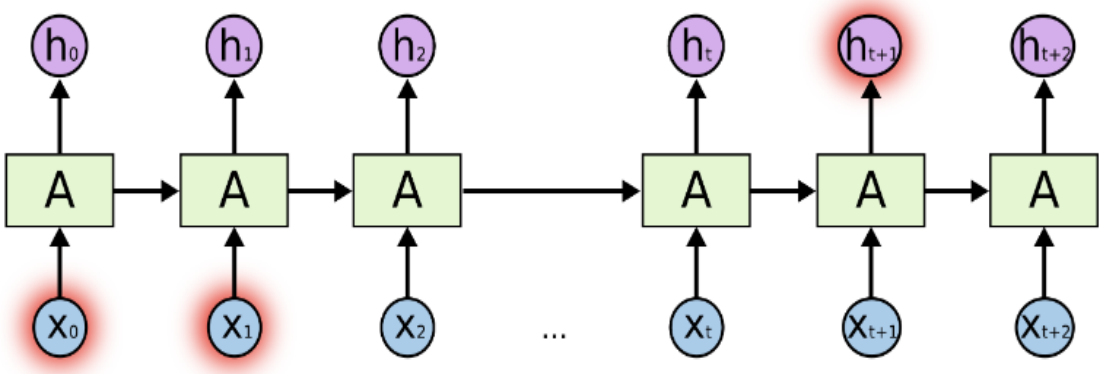

Рисунок 2.5 – Схема рекуретної нейронної мережі із великим розривом в інформації

І тут з’являється LSTM. LSTM спеціально розроблені, щоб уникнути проблеми довгострокової залежності. Запам’ятовування інформації протягом тривалого періоду часу – це практично їх поведінка за замовчуванням, а не те, що їм важко навчитися!

Усі рекурентні нейронні мережі мають вигляд ланцюжка повторюваних модулів нейронної мережі. У стандартних RNN цей повторюваний модуль матиме дуже просту структуру, таку як один шар tanh (рисунок 2.6).

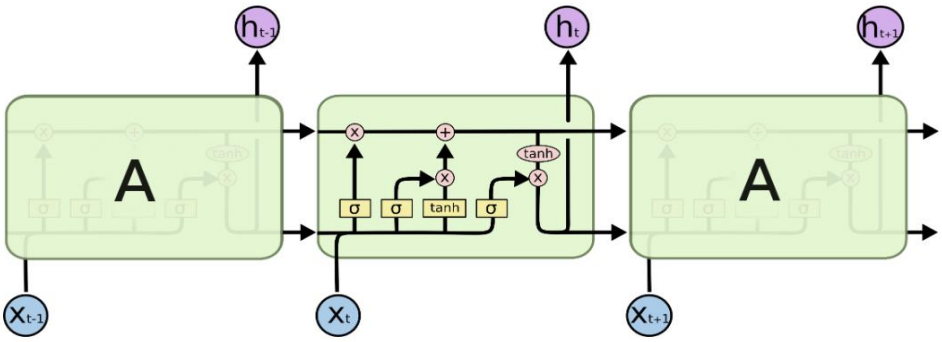

Рисунок 2.6 – Модуль повторюванності у LSTM

Однак у мережі LSTM є і мінуси. Це все ще рекурентна мережа, тому,

якщо вхідна послідовність має 1000 символів, комірка LSTM викликається 1000 разів, довгий градієнтний шлях. Хоча додавання каналу довгострокової пам’яті допомагає, існує обмеження щодо того, скільки він може вмістити.

Проте якщо розглядати LSTM для вирішення нашої задачі, нашої довжини послудовності їй може бути не достатньо.

2.2.3 Відповідність шаблону

Ідея цієї методики полягає в тому, щоб взяти послідовність з 9 днів у тестовому наборі, знайти подібні послідовності в наборі і порівняти їх віддачу за 10 днів. Якщо алгоритм знаходить більше однієї послідовності, він просто усереднює результат.Зіставлення шаблонів – це підхід комп’ютерного зору високого рівня, який виявляє частини зображення, які відповідають заздалегідь визначеному шаблону. Розширені алгоритми зіставлення шаблонів виявляють шаблони незалежно від орієнтації чи локальної яскравості.

Підходи зіставлення шаблонів є універсальними та простими у застосуванні, що робить їх одним із найбільш використовуваних способів локалізації об’єктів. Їх корисність здебільшого обмежена потужністю комп’ютера, оскільки визначення великих і складних шаблонів може зайняти багато часу.

2.2.4 Q-навчання

Q-навчання – це безмодельний алгоритм навчання з підкріпленням для вивчення значення дії в певному стані. В алгоритмі Q-навчання метою є ітераційне вивчення оптимальної функції Q-значення за допомогою рівняння оптимальності Беллмана. Для цього зберігаються всі Q-значення в таблиці, яка оновлюється на кожному кроці за допомогою ітерації Q-навчання.

Що ж, давайте зафіксуємо деякі визначення та рівняння, які нам

знадобляться для реалізації алгоритму Q-Learning.

У RL є середовище, у якому треба навчитися. Для цього створюється агента, який взаємодіятиме з середовищем за допомогою процесу проб і помилок. На кожному кроці часу t агент перебуває в певному стані s_t і вибирає дію a_t для виконання. Середовище виконує вибрану дію та повертає винагороду агенту. Чим вища винагорода, тим краща дія. Середовище також повідомляє агенту, закінчив він чи ні. Отже, епізод можна представити як послідовність стан-дія-винагорода.

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots \tag{2.9}$$

Мета агента полягає в тому, щоб максимізувати загальну винагороду, яку він отримає від середовища. Функція для максимізації називається функцією очікуваної дисконтованої прибутковості, яку позначаємо як G.

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0} y^k R_{t+k+1}\,. \tag{2.10}$$

Для цього агенту потрібно знайти оптимальну політику π, яка є розподілом ймовірностей даного стану за діями.

За оптимальної політики виконується рівняння оптимальності Беллмана:

$$q_*(s,a) = E[R_{t+1} + \gamma \max q_*(s',a')], \tag{2.11}$$

де q – функція «Дія-значення» або функція «Q-значення».

В алгоритмі Q-навчання метою є ітераційне вивчення оптимальної функції Q-значення за допомогою рівняння оптимальності Беллмана. Для цього зберігаємо всі Q-значення в таблиці, яку будемо оновлювати на кожному кроці за допомогою ітерації Q-Learning:

$$q^{new}(s,a) = (1 - \alpha)\, q(s,a) + \alpha\, (R_{t+1} + \gamma\, max\, q(s',a')), \qquad (2.12)$$

де γ – швидкість навчання, важливий гіперпараметр, який нам потрібно налаштувати, оскільки він контролює конвергенцію.

Тепер впроваджуватимемо алгоритм Q-Learning. Але повинні говорити про компроміс між розвідкою та розробкою. Але чому? На початку агент не має жодного уявлення про оточення. Він більше схильний досліджувати нові речі, ніж використовувати свої знання, тому що… у нього немає знань. За допомогою часових кроків агент буде отримувати все більше і більше інформації про те, як працює середовище, і тоді він, швидше за все, буде використовувати свої знання, ніж досліджувати нові речі. Якщо пропустимо цей важливий крок, функція Q-value зійдеться до локального мінімуму, який у більшості випадків далекий від оптимальної функції Q-value. Щоб впоратися з цим, матимемо поріг, який розкладатиме кожен епізод за допомогою формули експоненціального розпаду. Роблячи це, на кожному кроці часу t будемо вибірку змінної рівномірно по $[0,1]$. Якщо змінна менша за порогове значення, агент досліджуватиме середовище. Інакше він буде використовувати свої знання.

$$N(t) = N_0 e^{-\lambda t}, \qquad (2.13)$$

де N – початкове значення,

λ – константа, яка називається константою розпаду.

Приклад експоненційного розпаду зображено нижче (рисунок 2.7).

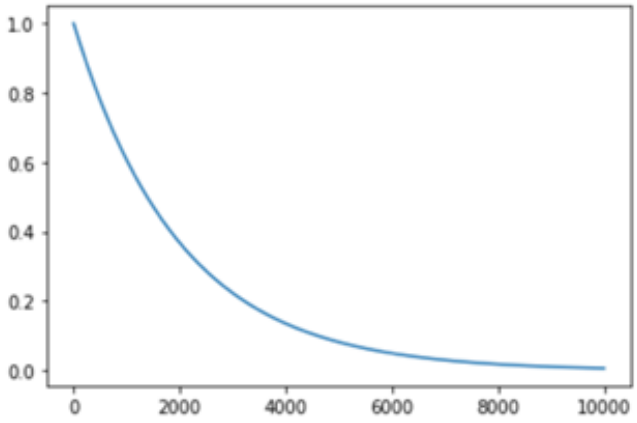

Рисунок 2.7 – Приклад експоненційного розпаду при $N_0 = 1$ та $\lambda = 0.0005$

Проте, предікт у процессі реального часу за допомогою лише Q-learning може не спрацювати.

2.2.4 Глибинне Q-навчання

Хоча Q-learning – дуже потужний алгоритм, його головна слабкість – відсутність спільності. Якщо ви розглядаєте Q-learning, як оновлення чисел у двовимірному масиві (простір дій * простір станів (action space * state space)), воно фактично нагадує динамічне програмування. Це вказує на те, що для станів, які агент Q-Learning не бачив раніше, він не знає, яку дію зробити. Інакше кажучи, агент Q-Learning немає можливості оцінювати значення для невидимих станів. Щоб впоратися з цією проблемою, DQN позбавляється двомірного масиву, ввівши нейронну мережу.

У глибокому Q-навчанні (DQN) використовуємо нейронну мережу для апроксимації функції Q-значення. Стан надається як вхід, а Q-значення дозволених дій є прогнозованим виходом.

DQN мінімізує середню квадратичну помилку між цільовими значеннями Q і використовує нейронну мережу для виконання апроксимації функції. Основним обмеженням підходу лише критики є те, що він працює лише з дискретними та кінцевими просторами станів і дій, що непрактично для великого портфеля акцій, оскільки ціни, звичайно, безперервні. Проте,

поки що не ставимо за ціль великі портфелі акцій, тому, це потенційно саме те що нам потрібно.

Основна ідея Q-навчання полягає в тому, що якби мали функцію $Q^*: State * Action = Result$, який міг би сказати нам, якою буде наша віддача, якби виконали дію в певному стані, тоді могли б легко створити політику, яка максимізує наші винагороди. Однак знаємо не все про світ, тому не маємо доступу до Q^*. Але оскільки нейронні мережі є універсальними апроксиматорами функцій, можемо просто створити їх і навчити виводити Q^*. Для нашого правила оновлення навчання використаємо той факт, що кожна функція Q для певної політики підкоряється рівнянню Беллмана, що було розглянуто в Q-learning розділі.

У DQN політика – це просто нейронна мережа, якій передаються спостереження. Наша оптимальна Q-функція дорівнює політиці, яка дає найкращу можливу суму винагород за певну пару стан-дія. Іншими словами, враховуючи те, що щойно сталося, якого правила можемо дотримуватися, щоб отримати максимальну можливу винагороду відтепер? Політика – це правило, яке вказує нам, які дії потрібно виконати з огляду на спостереження, і в DQN цю політику вивчає нейронна мережа. Таким чином, ця теоретично оптимальна політика є те, до чого намагаємося зблизитися.

Але як дізнаємося про винагороди в майбутньому? Завдяки буферу відтворення досвіду можемо взяти зразки, які вже відбулися, і побачити, що сталося за один крок у майбутньому, що надасть перевагу. Потім апроксимуємо нашу помилку передбачення за допомогою спеціально написанної функції втрат:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')} \sim U(D)[(r + \gamma max Q(s', a';\ \theta_i^-) - Q(s, a; \theta_i))^2], \quad (2.14)$$

де $\mathbb{E}_{(s,a,r,s')} \sim U(D)$ – приклади із буферу відтворення досвіду;

$\gamma max Q(s', a';\ \theta_i^-)$ – оцінка винагороди цільової мережі;

$Q(s, a; \theta_i)$ – оцінка винагороди Q-мережі.

2.2.4 Висновки щодо порівняльного аналізу підходів

Найбільш за все нам підходить підхід глибинного Q-навчання. Він задовільняє усі наші потреби і бере найбільш привабливі риси нейромережевих підходів та Q-навчання. Так как підхід глибинного Q-навчання побудований на базі звичайного Q-навчання. Використаємо просту нейромережу із прямим зв'язком.

На вхід мережі подаються поточні числові дані графіків, а виходом є відповідне значення Q для кожної можливої дії.

Навчати нашу модель глибинного Q-навчання будемо наступим чином:

Крок 1. Вибираємо дані з пам'яті відтворення досвіду, спочатку встановлюючи пакет у порожній список.

Крок 2. Перебираємо пам'ять за допомогою циклу for.

Крок 3. Оскільки маємо справу з даними часових рядів, нам потрібно брати вибірку з кінця пам'яті замість випадкової вибірки з неї.

Крок 4. Тепер, коли у нас є пакет даних, нам потрібно повторити кожен пакет – стан, нагорода, наступний_стан і готово – і навчити модель.

Крок 5. Якщо агент не в термінальному стані, розраховуємо загальну винагороду зі знижкою як поточну винагороду.

Крок 6. Далі визначаємо цільову змінну, яка також передбачена моделлю.

Крок 7. Далі підганяємо модель.

Крок 8. Наприкінці навчання хочемо зменшити параметр E, щоб повільно припинити виконання випадкових дій.

Тож, закінчуючи порівняльний аналіз маємо список підходів до проектування ІСППР які повинні бути протестовані експерементально та теоритичного фаворита – глибинне Q-навчання, яке побудовано на базі звичайного Q-навчання.

3 ЕКСПЕРЕМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Вибір тестових даних

Тестові набори данних під задачу було сформовано емпірічно. Тестові дані було зібрано руками за допомогою ресурсу Yahoo Finance. Дані заключають у собі різні показники, такі як дату, ціну при відкритті, закритті та інші (рисунок 3.1), та волотильність і інші, більш серйозні показники (рисунок 3.2). Ці дані будуть використані для заповнення таблиці для подальшого прогнозування ринку цінних паперів.

	A	B	C	D	E	F	G	H
1	Date	Open	High	Low	Close	Adj Close	Volume	Return
2	########	0.546875	0.546875	0.544643	0.544643	0.433865	5762400	-0.004
3	########	0.546875	0.549107	0.546875	0.546875	0.435643	3572800	0
4	########	0.558036	0.5625	0.558036	0.558036	0.444534	3516800	0
5	########	0.555804	0.555804	0.553571	0.553571	0.440977	3348800	-0.004
6	########	0.587054	0.589286	0.587054	0.587054	0.46765	10393600	0
7	########	0.571429	0.571429	0.569196	0.569196	0.453425	7520800	-0.004
8	########	0.580357	0.584821	0.580357	0.580357	0.462315	3976000	0
9	########	0.587054	0.591518	0.587054	0.587054	0.46765	8887200	0
10	########	0.587054	0.589286	0.584821	0.584821	0.465871	2805600	-0.004
11	########	0.578125	0.578125	0.575893	0.575893	0.458759	6160000	-0.004
12	########	0.575893	0.575893	0.571429	0.571429	0.455203	5924800	-0.008
13	########	0.555804	0.555804	0.553571	0.553571	0.440977	7039200	-0.004
14	########	0.535714	0.535714	0.533482	0.533482	0.424975	10976000	-0.004
15	########	0.508929	0.508929	0.504464	0.504464	0.401859	11547200	-0.009
16	########	0.477679	0.477679	0.475446	0.475446	0.378743	5941600	-0.005
17	########	0.493304	0.495536	0.493304	0.493304	0.392968	4788000	0
18	########	0.511161	0.513393	0.511161	0.511161	0.407193	6966400	0
19	########	0.511161	0.515625	0.511161	0.511161	0.407193	1982400	0
20	########	0.513393	0.515625	0.513393	0.513393	0.408971	3466400	0
21	########	0.491071	0.491071	0.486607	0.486607	0.387633	4188800	-0.009
22	########	0.486607	0.488839	0.486607	0.486607	0.387633	4586400	0
23	########	0.473214	0.473214	0.470982	0.470982	0.375187	3460800	-0.005
24	########	0.46875	0.46875	0.466518	0.466518	0.37163	3640000	-0.005
25	########	0.459821	0.459821	0.455357	0.455357	0.36274	2788800	-0.01
26	########	0.466518	0.46875	0.466518	0.466518	0.37163	3068800	0
27	########	0.486607	0.491071	0.486607	0.486607	0.387633	4810400	0
28	########	0.459821	0.459821	0.457589	0.457589	0.364518	5577600	-0.005
29	########	0.435268	0.435268	0.433036	0.433036	0.344958	6092800	-0.005
30	########	0.439732	0.441964	0.439732	0.439732	0.350293	3528000	0
31	########	0.428571	0.428571	0.424107	0.424107	0.337846	4244800	-0.01
32	########	0.450893	0.453125	0.450893	0.450893	0.359183	4872000	0
33	########	0.457589	0.459821	0.457589	0.457589	0.364518	2710400	0
34	########	0.473214	0.477679	0.473214	0.473214	0.376965	3690400	0
35	########	0.475446	0.477679	0.475446	0.475446	0.378743	2940000	0
36	########	0.470982	0.470982	0.46875	0.46875	0.373408	4043200	-0.005
37	########	0.466518	0.466518	0.464286	0.464286	0.369852	3427200	-0.005
38	########	0.464286	0.464286	0.462054	0.462054	0.368074	1344000	-0.005
39	########	0.462054	0.462054	0.457589	0.457589	0.364518	2900800	-0.01

Рисунок 3.1 – Перша частина таблиці тестових даних на прикладі акцій компанії Apple

Change	Volatility	MA7	MA21	Momentun	RSI	MACD	Signal	Upper_bar	Lower_band
-0.02009	0.051939	0.564094	0.55389	-2.45982	52.11011	0.011549	0.017082	0.668147	0.439633
0.002232	0.050151	0.556122	0.555485	-2.4308	52.50058	0.009186	0.015503	0.668293	0.442676
0.011161	0.048342	0.553571	0.558886	-2.43527	54.49843	0.00812	0.014026	0.66719	0.450583
-0.00446	0.046669	0.55389	0.563775	-2.45536	53.52845	0.006837	0.012588	0.660224	0.467327
0.033483	0.045603	0.560587	0.569728	-2.45313	59.36866	0.008424	0.011756	0.654537	0.484919
-0.01786	0.044114	0.560587	0.574192	-2.44196	55.37177	0.008147	0.011034	0.647206	0.501178
0.011161	0.042909	0.562819	0.577806	-2.44643	57.30633	0.008728	0.010573	0.643467	0.512145
0.006697	0.041944	0.568878	0.58057	-2.41295	58.46966	0.009617	0.010382	0.642379	0.518761
-0.00223	0.040916	0.574298	0.582164	-2.4308	57.90311	0.010027	0.010311	0.642517	0.521812
-0.00893	0.039716	0.576849	0.581952	-2.41964	55.5841	0.009521	0.010153	0.642362	0.521541
-0.00446	0.038517	0.579401	0.578975	-2.41295	54.41072	0.00866	0.009854	0.6346	0.523351
-0.01786	0.037485	0.574617	0.574724	-2.41518	49.87481	0.006462	0.009176	0.623004	0.526443
-0.02009	0.037179	0.569515	0.570259	-2.42411	45.29986	0.003064	0.007953	0.615382	0.525136
-0.02902	0.03874	0.558673	0.565264	-2.42857	39.64311	-0.00195	0.005973	0.615174	0.515353
-0.02902	0.042615	0.542729	0.558567	-2.44643	34.94389	-0.00817	0.003145	0.616876	0.500259
0.017858	0.043898	0.529656	0.553359	-2.46652	39.6825	-0.01152	0.000211	0.614585	0.492133
0.017857	0.043714	0.520408	0.550276	-2.49554	44.06955	-0.0126	-0.00235	0.613231	0.487322
0	0.04343	0.511798	0.548363	-2.52455	44.06955	-0.01329	-0.00454	0.613583	0.483143
0.002232	0.042965	0.506059	0.547088	-2.5067	44.6531	-0.01351	-0.00633	0.614006	0.480169
-0.02679	0.04395	0.499362	0.543155	-2.48884	39.34739	-0.01566	-0.0082	0.614196	0.472113
0	0.044639	0.496811	0.539435	-2.48884	39.34739	-0.01717	-0.00999	0.613833	0.465036
-0.01563	0.04617	0.496174	0.535927	-2.48661	36.4198	-0.0194	-0.01187	0.616022	0.455832
-0.00446	0.047625	0.492347	0.5321	-2.51339	35.60473	-0.02129	-0.01376	0.617501	0.4467
-0.01116	0.049525	0.484375	0.527211	-2.51339	33.58119	-0.02341	-0.01569	0.617965	0.436457
0.011161	0.050194	0.477997	0.523065	-2.52902	37.41192	-0.02392	-0.01733	0.61667	0.429461
0.020089	0.04964	0.474171	0.518282	-2.53348	43.70549	-0.02244	-0.01835	0.608353	0.428212
-0.02902	0.05053	0.470025	0.512968	-2.54464	37.7937	-0.02334	-0.01935	0.60359	0.422346
-0.02455	0.052898	0.462372	0.505952	-2.53348	33.6466	-0.02574	-0.02063	0.597468	0.414436
0.006696	0.054221	0.457908	0.498937	-2.51339	35.71821	-0.02679	-0.02186	0.586858	0.411016
-0.01563	0.05629	0.451849	0.491284	-2.54241	33.11972	-0.02856	-0.0232	0.575716	0.406852
0.026786	0.056254	0.451212	0.485332	-2.56696	41.03874	-0.02748	-0.02405	0.561977	0.408686
0.006696	0.055794	0.449936	0.479911	-2.56027	42.86014	-0.02578	-0.0244	0.546412	0.413409
0.015625	0.054755	0.448023	0.476084	-2.57589	46.97635	-0.02292	-0.0241	0.533397	0.418772
0.002232	0.053665	0.450574	0.473321	-2.54911	47.55751	-0.02023	-0.02333	0.52425	0.422391
-0.0067	0.052752	0.455676	0.47162	-2.54241	45.93105	-0.01843	-0.02235	0.520527	0.422713
-0.00446	0.05196	0.459184	0.471088	-2.52679	44.8303	-0.01717	-0.02131	0.520063	0.422114
-0.00223	0.051218	0.464605	0.4696	-2.52455	44.25917	-0.01616	-0.02028	0.51763	0.421571
-0.00446	0.050598	0.465561	0.467049	-2.53125	43.07687	-0.01554	-0.01933	0.511354	0.422745

Рисунок 3.2 – Друга частина таблиці тестових даних на прикладі акцій компанії Apple

Вони містять різні показники, такі як волотильність певної акції у той день та показники різних сигналів технічного аналізу розглянутих у першому розділі цієї дипломної роботи. Слід зазначити, що це не цифри з голови, а реальні дані які будуть використовуватися як тестові. Усього було зібрано подібні тестові дані для акцій таких компаній: Apple, Amazon, EA, Ebay, Facebook, Google, IBM, Microsoft, Netflix, Nvidia, Oracle, Tesla, Twitter та ін.

3.1 Тестування системи на базі моделі авторегресійної інтегрованої ковзної середньої

Як було зазначено у другому розгляді під час опису цієї моделі, функція автокореляції (ACF) покаже, чи точки даних мають такий зв’язок. Напишемо python код щоб перевірити це припущення. Результат зображено нижче (рисунок 3.3).

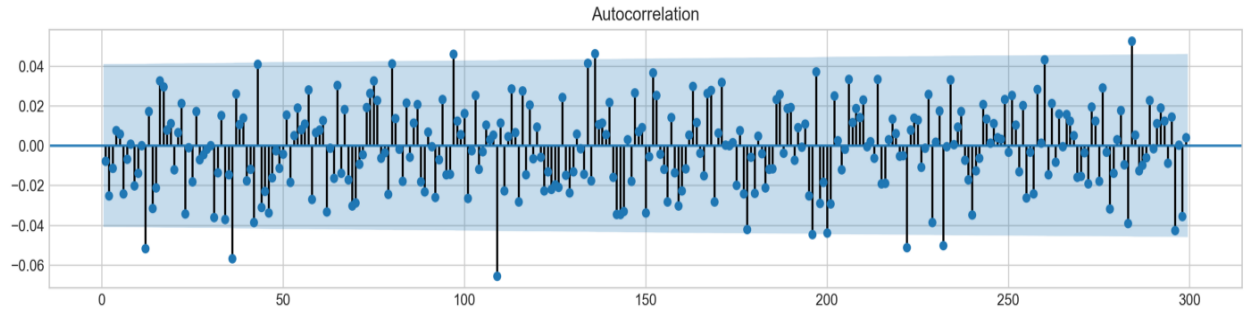

Рисунок 3.3 – Функція автокореляції

Точки даних дійсно не корельовані, тому використання ARIMA для прогнозування майбутніх значень є недоцільним.

3.2 Тестування системи на базі глибинних нейронних мереж

Спочатку було протестовано розпізнавання шаблонів історичних даних за допомогою згорткової мережі. Вона може приймати будь-яку кількість функцій і навчатися з них одночасно. У цьому прикладі мережа повинна була вчитися на основі послідовностей 21 дня та передбачити повернення акцій на наступний день. Під поверненням мається на увазі різниця в ціні на початку і в кінці дня. Якщо ціна пішла вгору – прибутковість позитивна, вниз – негативна. Код наведено нижче (рисунок 3.4).

```
keras.backend.clear_session()

n_steps = X_train.shape[1]
n_features = X_train.shape[2]

model = Sequential()

model.add(Conv1D(filters=20, kernel_size=2, activation='relu', input_shape=(n_steps, n_features)))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(5, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse', metrics=['mse'])
```

Рисунок 3.4 – Код тестування системи на базі глибинної нейронної мережі

Мережа була схильна до переобладнання, тобто вона дуже добре вивчила шаблони в даних, але не змогла зробити жодних значущих прогнозів на основі тестових даних. Точність була настільки ж високою, як і випадкове припущення. Результат роботи зображено нижче (рисунок 3.5). Фіолетовий колір – предсказання, синій – реальний графік за той самий день.

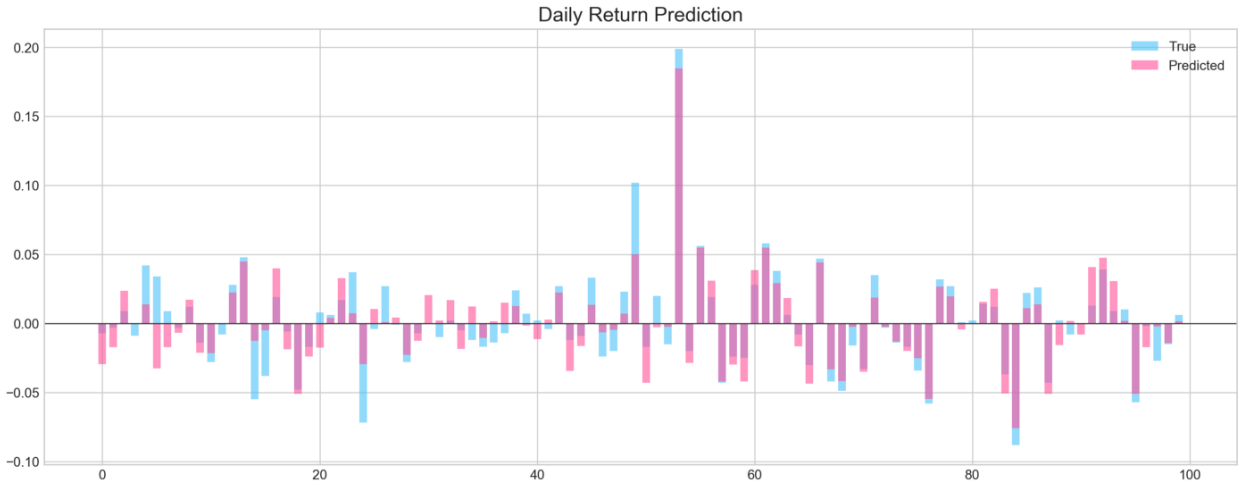

Рисунок 3.5 – Результат роботи глибинної нейронної мережі

Рекурентні мережі (LSTM) також добре навчаються на основі послідовних даних, тобто часових рядів. Створимо рекурентну нейронну мережу за допомогою коду (рисунок 3.6).

```
keras.backend.clear_session()

n_steps = X_train.shape[1]
n_features = X_train.shape[2]

model = Sequential()
model.add(LSTM(100, activation='relu', return_sequences=True, input_shape=(n_steps, n_features)))
model.add(LSTM(50, activation='relu', return_sequences=False))
model.add(Dense(10))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

```
model.summary()
```

Рисунок 3.6 – Код створення рекурентної нейронної мережі

Далі за допомогою тестових даних проходить навчання нейромережі (рисунок 3.7).

```
model.fit(X_train, y_train, epochs=100, verbose=0, validation_data=[X_test, y_test], use_multiprocessing=True)

plt.figure(figsize=(16,4))
plt.plot(model.history.history['loss'], label='Loss')
plt.plot(model.history.history['val_loss'], label='Val Loss')
plt.legend(loc=1)
plt.title('LSTM - Training Process')
plt.show()
```

Рисунок 3.7 – Код навчання рекурентної нейронної мережі

Далі йде опрацювання навченої нейромережі (рисунок 3.8).

```
pred, y_true, y_pred = functions.evaluation(
                    X_test, y_test, model, random=False, n_preds=50,
                    show_graph=True)
```

Рисунок 3.8 – Код запуску навченої нейромережі в роботу

Давайте подивимося на прогнози (рисунок 3.9). Мережа пішла легким шляхом і вирішила, що щоденна віддача буде негативною. Це було правильно приблизно в 50% випадків.

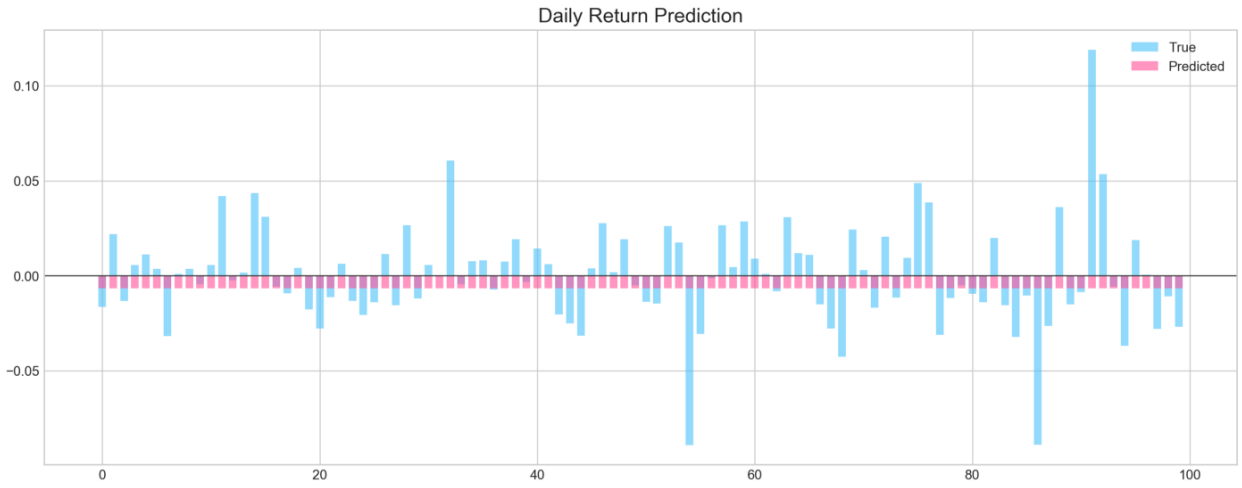

Рисунок 3.9 – Результат роботи рекурентної нейронної мережі

Було спробувано сотні комбінацій мережевої архітектури, кількості функцій, довжини послідовностей, налаштування гіперпараметрів, передбачення наступного дня/тижня/місяця. Результати були такими ж гарними, як і випадкове припущення.

Зрозуміло, що нейронні мережі тут також не працюватимуть.

3.3 Тестування системи на базі відповідності шаблону

Нижче зображено код (рисунок 3.10) та результат його роботи, де рожева лінія – це послідовність 9 днів за якими відбувалося навчання (рисунок 3.11)

```
for i in range(10):
    if i + check_step >= len(binary_test):
        break
    found_patterns = []
    sample = binary_test[i:i+step]
    sample_check = binary_test[i+check_step]

    # Looking for patterns in training set
    # that matches with pattern from testing set
    for j in range(len(binary_train)-check_step):
        if accuracy_score( binary_train[j:j+step], sample ) == 1.0:
            found_patterns.append(i)

    p = []
    prediction = None
    if len(found_patterns) != 0:
        for k in found_patterns:
            p.append( binary_train[k+check_step] )

        prediction = sum(p)/len(p)
        if sample_check == prediction:
            total_predictions.append(1)
        else:
            total_predictions.append(0)

accuracy = sum(total_predictions)/len(total_predictions)
print('Accuracy:', accuracy)
```

Рисунок 3.10 – Код тестування підходу «відповідність шаблону»

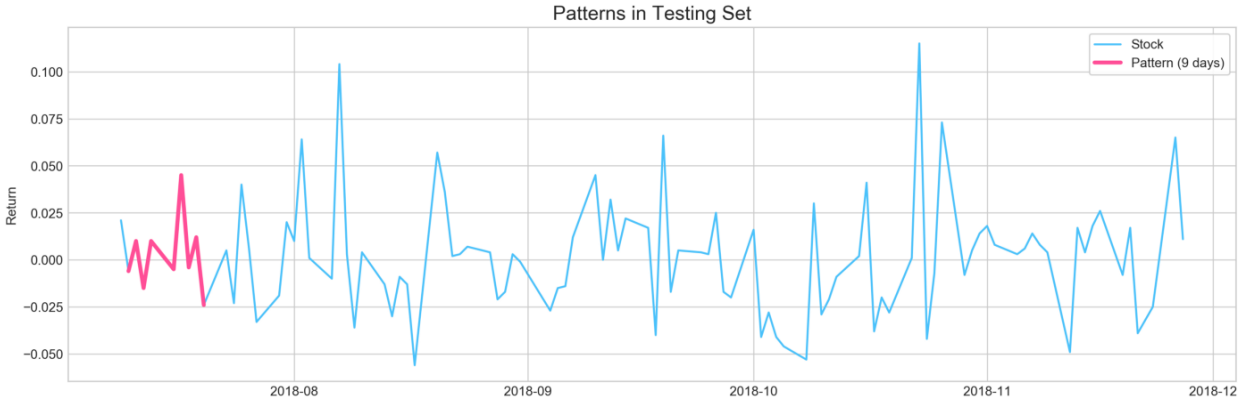

Рисунок 3.11 – Результат роботи алгоритму «відповідність шаблону» за 4 дні

Алгоритм знайшов 5 збігів, три з них мають позитивну віддачу на 10-й день, два – негативну. Усереднюючи це, маємо позитивний прибуток як прогноз. Те саме, що фактичне повернення з тестового набору. Це працює.

Але перегляд усього тестового набору шаблонів дав приблизно 50%

потребуємого рівня точності. Тут також не сталося магії (рисунок 3.12).

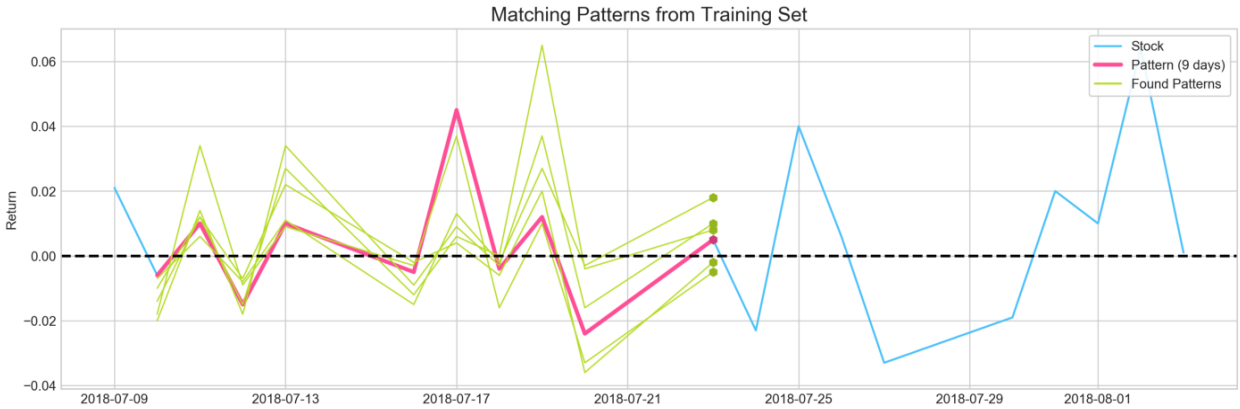

Рисунок 3.12 – Результат роботи алгоритму «відповідність шаблону» за 1 місяць

3.3 Тестування системи на базі алгоритму Q-навчання

Жодна з методик не спрацювала, як і гадалося у попередньому розділі, але все ж таки потрібно заробляти гроші на фондовому ринку. Є альтернатива денній торгівлі. Класичний підхід із використанням технічних індикаторів може запропонувати хороший прибуток від короткострокових інвестицій – коливається від кількох днів до приблизно місяця. Ось як це працює: моделюємо торгову стратегію, використовуючи смуги Болліджера використовуючи алгоритм із назвою Q-навчання.

Запускаємо симуляцію, ніби купуються акції, коли ціна наближається до нижньої смуги, і навпаки. На початку 2018 року алгоритм мав 15 000 доларів для інвестування в Tesla. До кінця року він зміг отримати понад 100% прибутку лише за 19 угод. Результат роботи зображено нижче (рисунок 3.13).

Смуги Болліджера чудово працювали на Tesla, але не так добре на інших акціях. Amazon, наприклад, мав негативний прибуток. Отже, він не ідеальний, але він працював на більшості акцій (загалом 19).

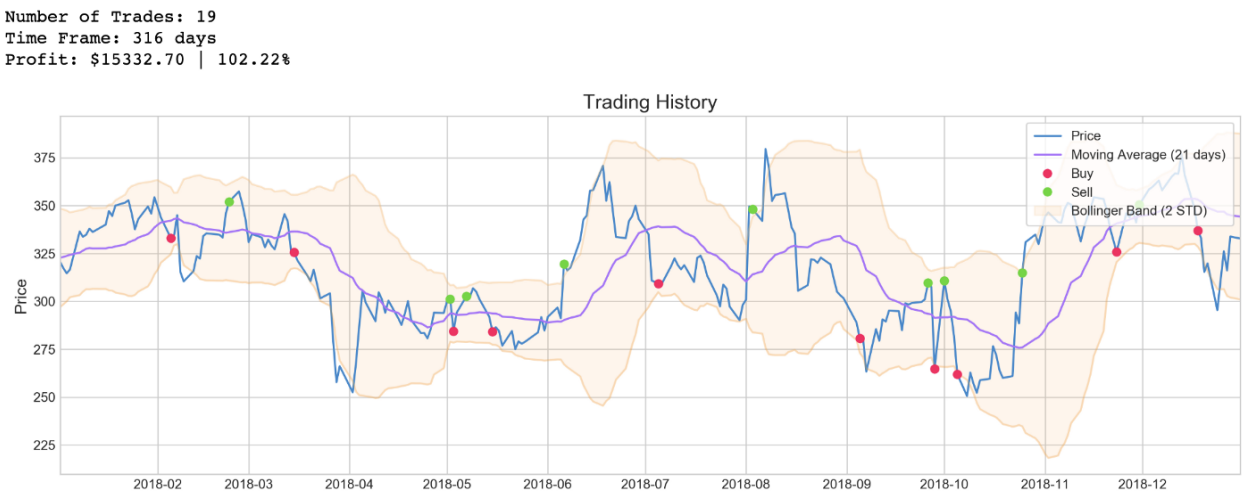

Рисунок 3.13 – Результат роботи алгоритму купівлі/продажу за допомогою Q-learning та смуг Боліндже́ра

3.4 Тестування системи на базі глибинного Q-навчання

Оскільки Глабинне Q-навчання це фаворит, запунимося на ньому більш детально. Використовуються здебільшого бібліотеки `tensorflow` та `pandas` (рисунок 3.14).

```
import math
import random
import numpy as np
import tensorflow as tf
import pandas_datareader as data_reader

from tqdm import tqdm
from collections import deque
```

Рисунок 3.14 – Імпорт необхідних бібліотек

Тепер визначаємо сам алгоритм за допомогою класу `AITrader`, нижче наведено кілька важливих моментів.

У торгівлі мається 3 простори дій: купити, продати та сидіти.

Встановлюємо для пам’яті відтворення досвіду масив `deque` з 2000

елементами всередині. Створюємо порожній список із інвентарем, який містить акції, які вже куплено.

Потрібно встановити гамма-параметр на 0,95, що допоможе максимізувати поточну винагороду в довгостроковій перспективі;

Параметр epsilon використовується для визначення того, чи слід використовувати випадкову дію чи використовувати модель для дії. Починаємо з встановлення значення 1.0, щоб спочатку виконувати випадкові дії, коли модель не навчена;

З часом треба зменшити кількість випадкових дій і натомість здебільшого використовувати навчену модель, тому встановлено epsilon_final на 0,01;

Потім встановлено швидкість зменшення E в параметрі *epsilon_decay*;

Починається побудова класу *AITrader* (рисунок 3.15)

```
class AITrader:
    def __init__(self, state_size, action_space=3, model_name="AITrader"):
        self.state_size = state_size
        self.action_space = action_space
        self.memory = deque(range(2000))
        self.inventory = []
        self.model_name = model_name

        self.gamma = 0.95
        self.epsilon = 1.0
        self.epsilon_final = 0.01
        self.epsilon_decay = 0.995

        self.model = self.model_builder()
```

Рисунок 3.15 – Головний класс AITrader

Далі потрібно почати визначати нашу нейронну мережу.

Першим кроком для визначення нейронної мережі є визначення функції під назвою *model_builder*, яка не приймає жодних аргументів,

лише ключове слово *self*.

Потім визначається модель за допомогою *tf.keras.models.Sequential()*. Визначити за допомогою станів моделі, які є попередніми n днями та цінами акцій за дні. Стан – це вектор чисел, який використовує повністю зв'язану мережу або щільну мережу.

Далі додається перший щільний шар за допомогою *tf.keras.layers.Dense()* і вказується кількість нейронів у шарі (32) і встановлюється активація *relu*. Також потрібно визначити форму введення на першому шарі за допомогою *input_dim=self.state_size*.

Збираємося використовувати 3 прихованих шари в цій мережі, тому додається ще 2 і змінюємо архітектуру до 64 нейронів у другому та 128 для останнього шару.

Потім потрібно визначити вихідний рівень і скомпілювати мережу.

Щоб визначити вихідний рівень, потрібно встановити кількість нейронів на кількість дій, які можна виконати, у цьому випадку 3. Також збираємося змінити функцію активації на relu, оскільки використовується середньоквадратична помилка для втрати (рисунок 3.16).

```
def model_builder(self):
    model = tf.keras.models.Sequential()

    model.add(tf.keras.layers.Dense(units=32, activation='relu', input_dim=self.state_size))
    model.add(tf.keras.layers.Dense(units=64, activation='relu'))
    model.add(tf.keras.layers.Dense(units=128, activation='relu'))
    model.add(tf.keras.layers.Dense(units=self.action_space, activation='linear'))

    model.compile(loss='mse', optimizer=tf.keras.optimizers.Adam(lr=0.001))
    return model
```

Рисунок 3.16 – Функція побудови моделі

Нарешті, скомпільовується модель. Оскільки це завдання регресії, не можна використовувати точність як втрату, тому використовується mse. Потім використовується оптимізатор Adam і встановлюється швидкість

навчання на 0,001 і повертається модель (рисунок 3.17)

```
model.compile(loss='mse', optimizer=tf.keras.optimizers.Adam(lr=0.001))
return model
```

Рисунок 3.17 – Компіляція і повернення моделі

Щоб повернути модель, потрібно додати self.model = self.model_builer до функції __init__. Ця функція створить мережу, ініціалізує її та збереже в аргументі self.model.

Тепер, коли нейронну мережу визначена, створюється функцію для торгівлі, яка приймає стан як вхідні дані та повертає дію для виконання в цьому стані. Для цього створюється функцію під назвою trade, яка приймає один аргумент: стан.

Для кожного стану нам потрібно визначити, чи використовувати випадково згенеровану дію чи нейронну мережу. Для цього використовується бібліотеку random, і якщо вона менша за E, повертається випадкову дію за допомогою random.randrange() і передається self.action_space. Якщо число більше E, використовується модель для вибору дії. Для цього визначаются дії, що дорівнюють self.model.predict, і передається стан як аргумент. одне число з np.argmax повертається, щоб повернути лише дію з найвищою ймовірністю.

Функція приймає як вхідні дані форму та генерує випадкове число (рисунок 3.18). Якщо число менше або дорівнює E, це призведе до випадкової дії (це завжди буде так на початку). Якщо він більший за E, модель використовуватиме модель для виконання прогнозу щодо вхідного стану та повернення дії, яка має найвищу ймовірність.

```
def trade(self, state):
    if random.random() <= self.epsilon:
        return random.randrange(self.action_space)

    actions = self.model.predict(state[0])
    return actions
```

Рисунок 3.18 – Функція трейду

Тепер, коли реалізувано функцію торгівлі, створюємо спеціальну функцію навчання (рисунок 3.19). Ця функція візьме групу збережених даних і навчить модель на цьому. Використаємо для цього покроковий алгоритм який було описано у попередньому розділі під час розгляду процессу навчання моделі Глибинного Q-навчання.

```
def batch_train(self, batch_size):
    batch = []
    for i in range(len(self.memory) - batch_size + 1, len(self.memory)):
        batch.append(self.memory[i])

    for state, action, reward, next_state, done in batch:
        reward = reward
        if not done:
            reward = reward + self.gamma * np.amax(self.model.predict(next_state)[0])

            target = self.model.predict(state)
            target[0][action] = reward

            self.model.fit(state, target, epochs=1, verbose=0)

    if self.epsilon > self.epsilon_final:
        self.epsilon *= self.epsilon_decay
```

Рисунок 3.19 – Функція тренування моделі

Тепер, коли створено клас `AITrader`, потрібно створити кілька допоміжних функцій, які використовуватимуться в процесі навчання. Зокрема, потрібно визначити наступні 3 функції: `sigmoid`, `stocks_price_format` та `dataset_loader`.

`sigmoid` – це функція активації, яка зазвичай використовується наприкінці мережі для двійкової класифікації, оскільки вона масштабує число в діапазоні від 0 до 1 (рисунок 3.20). Це використовуватиметься для нормалізації даних про вартість акцій.

```
def sigmoid(x):
    return 1 - (1 + math.exp(-x))
```

Рисунок 3.20 – Функція `sigmoid`

`stocks_price_format` – це функція форматування для друку цін на акції, які вже куплено або продано (рисунок 3.21).

```
def stocks_price_format(n):
    if n < 0:
        return "- # {0:2f}".format(abs(n))
    else:
        return "$ {0:2f}".format(abs(n))
```

Рисунок 3.21 – Функція stocks_price_format

`dataset_loader` – ця функція з'єднується з джерелом даних і отримує з нього дані про акції, у цьому випадку дані завантажуються з Yahoo Finance (рисунок 3.22).

```
def dataset_loader(stock_name):
    dataset = data_reader.DataReader(stock_name, data_source="yahoo")
    start_date = str(dataset.index[0]).split()[0]
    end_date = str(dataset.index[1]).split()[0]
    return dataset['Close']
```

Рисунок 3.22 – Функція `dataset_loader`

Нижче можна побачити набір даних AAPL. За допомогою цієї інформації можливо створити стани для цієї мережі (рисунок 3.23).

Date	High	Low	Open	Close	Volume	Adj Close
2010-01-04	30.642857	30.340000	30.490000	30.572857	123432400.0	26.782711
2010-01-05	30.798571	30.464285	30.657143	30.625713	150476200.0	26.829010
2010-01-06	30.747143	30.107143	30.625713	30.138571	138040000.0	26.402260
2010-01-07	30.285715	29.864286	30.250000	30.082857	119282800.0	26.353460
2010-01-08	30.285715	29.865715	30.042856	30.282858	111902700.0	26.528664

Рисунок 3.23 – Дані які беруться в онлайн режимі із Yahoo Finance

Тепер, коли є функція dataset_loader, потрібно створити функцію, яка приймає ці дані та генерує з них стани. Спочатку розглянуто можливість перевести проблему торгівлі на фондовому ринку в середовище навчання з підкріпленням. Кожна точка на біржовому графіку – це просто плаваюче число, яке представляє ціну акцій у певний момент часу.

Завдання полягає в тому, щоб передбачити, що станеться в наступний період, і, як згадувалося, є 3 можливі дії: купити, продати або сидіти. Це проблема регресії. Скажімо, є `window_size` = 5, тому використовуються 5 станів для прогнозування цілі, яка є безперервним числом. Замість того, щоб прогнозувати реальні числа, натомість передбачаємо одну з 3 можливих дій.

Далі змінено стани вхідних даних на різниці в курсах акцій, які відображатимуть зміни цін з часом.

Щоб реалізувати це в Python, створено функцію state_creator, яка приймає 3 аргументи: `data, timestep і window_size`.

Спочатку потрібно обчислити початковий ідентифікатор. Коли початковий ідентифікатор додатний, створюється стан, а якщо він від’ємний, додається інформація, аж до поки ідентифікатор не дійде до `window_size`.

Далі визначається порожній список під назвою state і проходиться список `window_data`.

Щоб виконати функцію, повертається масив NumPy стану (рисунок 3.24).

```
def state_creator(data, timestep, window_size):
    starting_id = timestep - window_size + 1

    if starting_id >= 0:
        windowed_data = data[starting_id:timestep + 1]
    else:
        windowed_data = starting_id * [data[0]] + list(data[0:timestep + 1])

    state = []
    for i in range(window_size - 1):
        state.append(sigmoid(windowed_data[i + 1] - windowed_data[i]))

    return np.array([state])
```

Рисунок 3.24 – Функція створення стану

Тепер, коли створено функцію state_creator, завантажуємо набір тестових даних. Спочатку визначається нова змінна під назвою stock_name, і для цього прикладу використається знову AAPL. Потім визначається змінна під назвою data за допомогою функції dataset_loader (рисунок 3.25).

Date	
2010-01-04	30.572857
2010-01-05	30.625713
2010-01-06	30.138571
2010-01-07	30.082857
2010-01-08	30.282858
2010-01-11	30.015715
2010-01-12	29.674286
2010-01-13	30.092857
2010-01-14	29.918571

Рисунок 3.25 – Результат dataset_loader на акціях Apple

Перш ніж перейти до навчання нашої моделі, треба визначити кілька гіперпараметрів (рисунок 3.26).

```
window_size = 10
episodes = 1000

batch_size = 32
data_samples = len(data) - 1
```

Рисунок 3.26 – Гіперпараметри системи

Тепер настав час визначення торгового агента,і переглядання короткого викладу нейронної моделі (рисунок 3.27).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 32)                352
_________________________________________________________________
dense_1 (Dense)              (None, 64)                2112
_________________________________________________________________
dense_2 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 387
=================================================================
Total params: 11,171
Trainable params: 11,171
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 3.27 – Виклад моделі

Тепер потрібно навчити модель, це буде зробелно за допомогою циклу for, який буде проходити через усі епізоди. Потім потрібно визначити початковий стан за допомогою state_creator. Потім визначається 2 змінні, щоб можна було відстежувати `total_profit`, і встановлюється інвентар на 0 на початку епізоду з trader.inventory.

Далі визначається часовий крок (1 часовий крок дорівнює 1 дню) за допомогою циклу `for`, який показує, скільки зразків мається. Для цього потрібно визначити дію, `next_state` і винагороду. Тоді оновлюється інвентар на основі заданої дії. На основі дій можна розрахувати винагороду та оновити `total_profit`. Потім потрібно перевірити, чи це останній зразок у наборі даних. Далі потрібно додати всі дані до буфера відтворення досвіду

трейдера за допомогою `trader.memory.append()`. Потім змінюємо стан на `next_state`, щоб можна було повторювати весь епізод. Нарешті, роздруковуємо `total_profit if done = True` і додаємо заяви про друк до того, коли купується чи продається, і як який прибуток.

Перед початком тренувального процесу потрібно зробити ще дві речі - перевірити, чи є в пам'яті більше інформації, ніж розмір `batch_size`. Якщо є, викликається `trader.batch_train` і передається аргумент `batch_size`. Потім перевіряється, чи кількість епізодів ділиться на 10, і якщо це так, зберігається модель за допомогою `trader.model.save()` у файлі H5.

Результат роботи нейромережі зображено нижче (рисунок 3.28). Як можна побачити, модель працює та приносить прибуток.

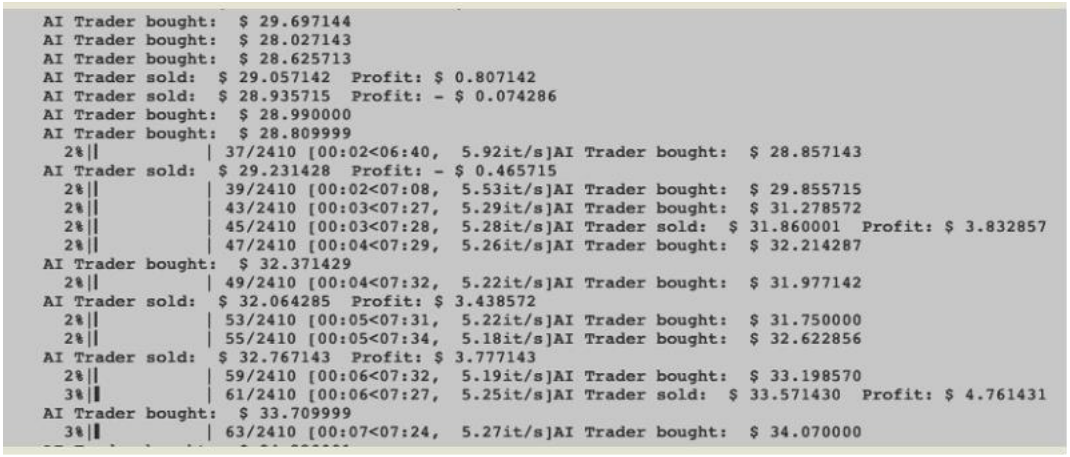

Рисунок 3.28 – Результат роботи моделі

3.4 Висновки за результатами тестування систем

Було протестовано різні підходи до вирішення задачі та експерементально доведено, що із переліченних підходів, найбільш за все підходить саме глибинне Q-навчання. Цей підхід вчиться усьому під час своєї роботи, якщо йому не вистачає знань зробити вибір. Чим більше він працює, тим менше він навчається.

ВИСНОВКИ

У ході цієї роботи була розроблена інтелектуальна системи підтримки прийняття рішень трейдера на фінансовому ринку. В роботі була проведена дослідницька робота для отримання відповіді на питання "як машинне навчання може допомагати у продажах та покупках акцій так, щоб приносити прибуток?".

Було розглянуто термінологію і базові поняття та алгоритми із повсякденного життя трейдера, які можна покращити за допомогою систем підтримки прийняття рішень. Описано теорію систем підтримки прийнття рішень, іх класифікацю, арзітектуру та проведено аналіз існуючих рішень.

Проаналізувавши усе це набагато краще зрозуміло, як саме система може допомагати трейдерові у роботі. Для цього було протестовано багато різних підходів, таких як модель авторегресійної ковзної середньої, систему на базі глибинних нейронних мереж, відповідність шаблону, алгоритм навчання із підкріплення під назвою Q-навчання і врешті обрано фаворита який працює на базу Q-навчання та нейромережі і має горду назву глибинне Q-навчання.

Але ж не може бути все так просто? Так, звісно студент не може написати систему підтримки прийняття рішень для трейдера та стати мільонером, або допомогти стати мільонерами своїм пикладачам. Світ трейдингу набагато складніше за графіки та цифри із них. На ціну ценних паперів впливають економіка, величезна купа новин, інсайди, та ін., тому простий аналіз графіків хоч і працює, але він не застрахований від непередбачуванних подій із реального світу. Наприклад, Ілон Маск каже що збирається купити компанію Twitter та ціна на акції Twitter взлітає у той же час. Могла б наша система таке передбачити? Авжеж ні.

Але так як розроблена система це тільки помічник трейдера а не сам трейдер, точність роботи алгоритмів та реальний життєвий досвід трейдера

можуть працювати у симбіозі та підкорювати верховини світу трейдерства. Принаймні, мені хочется у це вірити.

Ніяка система у нашому житті не бездоганна, тому, завжди є що поліпшити. Можна розділити майбутні можливі оновлення на дві категорії – вертикальні та горизонтальні.

Вертикальні майбутні імовірні поліпшення – це вдосконалення самої системи підримки прийняття рішень для трейдера на фінансовому ринку, яка була написана в ходу цього дипломного проекту. До вертикальних майбутніх імовірних поліпшень можна додати:

- додавання тенденції наступних показників до нашого повідомлення;
- використання LSTM нейромережі замість щільних шарів;
- побудова більш гарного інтерфейсу користувача для масового юзера;

Горизонтальні майбутні імовірні поліпшення – це інші системи із нейромережевеми підходами, які повинні працювати поряд із створенною у цій дипломній роботі системою, та комунікувати одна із одною і якості мікросервесів щоб збільшити точність прогнозів та кількість вхідних даних. До горизонтальних майбутнії імовірних поліпшень можливо додати такі системи:

- система агрегації і аналізу новин щодо компаній, якції яких юзер моніторить;
- система агрегації і аналізу фінансових даних компанії, що впливають на ціну;
- система агрегації і анаілзу економіки країни, до якої належить компанія.

У ході виконання кваліфікаційної роботи експерементально доведено, чому усі підходи окрім глибинного Q-навчання не працюють у цьому випадку і розроблено інтелектуальну систему підтримки прийняттия рішень трейдера на фінансовому рику на базі глибинного Q-навчання, яка дійсно працює і яку можна вдосконалити вертикально та горизонтально щоб ще більше поліпшити результати її роботи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Нильсен М.А. Нейронні мережі та глибоке навчання. - Determination Press. -2015.

2. H. Nguyen and H. La, “Review of Deep Reinforcement Learning for Robot Manipulation,” in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, Naples, Italy. – 2019

3. CS231n: Convolutional Neural Networks for Visual Recognition [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу : http://cs231n.github.io/neural-networks-1/

4. A Begginers Guide to Q-learning [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу : https://towardsdatascience.com/a-beginners-guide-to-q-learning-c3e2a30a653c.- 20.05.2022 – Загл. з екрану.

5. Q-learning algorithm: from eplanation to implementation [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу : https://towardsdatascience.com/q-learning-algorithm-from-explanation-to-implementation-cdbeda2ea187.- 22.05.2022 – Загл. з екрану.

6. 10 trading indicators every trader should know [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу : https://www.ig.com/en/trading-strategies/10-trading-indicators-every-trader-should-know-190604.- 26.05.2022 – Загл. з екрану.

7. SR Trend Lines - Bollinger Bands [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу : https://www.fxacademy.com/learn/advanced-sr-trend-lines/sr-trend-lines-bollinger-bands.- 26.05.2022 – Загл. з екрану.

8. Technical Analysis [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу : https://www.investopedia.com/terms/t/technicalanalysis.asp.- 26.05.2022 – Загл. з екрану.

9. Романенко В.Д., Милявский Ю.Л., Канцедал Г.О. Стабилизация неустойчивого курса криптовалюты на основе модального управления импульсным процессом когнитивной карты. XXVI Міжнародна конференція з автоматичного керування. Автоматика 2020: матеріали 26 наукової конференції, м. Київ, 13-15 жовтня 2020р.

10. Fawcett T. An introduction to ROC analysis: Pattern Recognition Letters / USA, 2006, pages 861-874.

11. Xiao-Yuan Jing, Fei Wu, Dong Xiwei, Baowen Xu. An Improved SDA Based Defect Prediction Framework for Both Within-Project and Cross-Project ClassImbalance Problems. IEEE Transactions on Software Engineering. 2016.

12. Субботін С. О. Нейронні мережі: теорія та практика: навч. посіб. – Житомир: Вид. О. О. Євенок, 2020. – 160 с

13. Корабльов М.М. Інтелуктуальна система підтримки прийняття клінічних рішень на основі мультиагентного підходу та міркувань по прецедентам // Сучасні інформаційні технології і системи: монографія / за заг. ред. В.С. Пономаренка. – Х.: ХНЕУ ім. С. Кузнеця, 2022. – С. 139-164.

14. Mykola Korablyov, Natalia Axak, Oleksandr Fomichov and Andrii Chuprina. Hybrid Neuro-Fuzzy Model with Immune Training for Recognition of Objects in an Image / Proceedings of the 9th International Conference "Information Control Systems & Technologies", Odessa, Ukraine, September 24–26, 2020. – pp. 267-281

15. Кораблев Н.М., Фомичев А.А., Соловьев Д.Н., Чуприна А.А. Гибридные модели принятия решений с использованием иммунного похода // Информационные управляющие системы и технологии. Проблемы и решения: монография. Под науч. ред. проф. Вычужанина Владимира. – Одесса: Экология, 2019. – С. 100-116.

16. Korablyov, M., Axak, N., Soloviov, D. Hybrid evolutionary decision-making model based on neural network and immune approaches (2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2018 – Proceedings 1,8526594, с. 378-381.