Київський національний торговельно-економічний університет

Кафедра інформаційних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Засоби машинного навчання в фінансовому менеджменті»

(за матеріалами ТОВ «Адвансед Нетворк Консалтінг» м. Київ)

Студента 2м курсу, 7 групи,
факультету обліку, аудиту
та інформаційних систем,
денної форми навчання
спеціальності
122 «Комп'ютерні науки» Хурчак
Валерій
Юрійович

_____
(підпис студента)

Науковий керівник
канд фіз.-матем. н.
доцент Нагірна Алла
Миколаївна

_____
(підпис наукового керівника)

Гарант освітньої програми
доктор т.н.
професор Краскевич
Валерій
Євгенович

_____
(підпис гаранта освітньої програми)

Київ 2018

ЗМІСТ

ВСТУП……..4

РОЗДІЛ 1. Огляд та сучасний стан машинного навчання у фінансовому менеджменті

1.1 Огляд існуючих досліджень……..5

1.2 Приклади застосування машинного навчання у фінансовому менеджменті…...11

Висновки до розділу 1……14

РОЗДІЛ 2. Комплексний аналіз методів і інструментів машинного навчання у фінансовому менеджменті

2.1 Аналіз типів та методів машинного навчання……..…15

2.2 Особливості використання програмних продуктів в задачах машинного навчання……...……...20

Висновки до розділу 2……25

РОЗДІЛ 3. Застосування моделей машинного навчання для менеджменту торгів індексу S&P 500

3.1 Постановка задачі……..……..26

3.2 Розробка моделі алгоритмічних торгів індексу S&P 500……...……..27

3.2.1 Підготовка масиву даних для тренування нейронних мереж

3.2.2 Навчання моделей з використанням алгоритмів регресії та класифікації

3.3 Аналіз ефективності застосування створеної моделі……54

Висновки до розділу 3……60

ВИСНОВКИ……63

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……....65

АНОТАЦІЯ

Хурчак В.Ю. Засоби машинного навчання в фінансовому менеджменті.

Дослідження присвячене проблемі застосування методів машинного навчання в задачах фінансового менеджменту та прогнозування ринку цінних паперів. В роботі теоретично обґрунтована та експериментально перевірена методика використання моделей машинного навчання в якості засобу прогнозування фінансових інструментів та керування ризиками під час інвестування. Запропоновано алгоритми та методологію для побудови нейронних мереж. Реалізовано модель прогнозування зваженого біржового індексу на їх основі. Розроблена система може бути використана в якості помічника фінансових менеджерів для прийняття рішень щодо купівлі та продажу акцій.

Khurchak V.Y. Means of machine learning in financial management.

Graduate research is devoted to the problem of application machine learning methods for tasks of financial management and securities market prediction. The paper theoretically grounded and experimentally tested method of using the machine learning models for forecasting financial instruments and managing risks while investing. There have been developed the algorithms and methodology for building neural networks. Implemented the model for prediction of weighted stock index based on them. The system can be used as a financial managers' assistant in order to take decisions on buying and selling stocks.

ВСТУП

Застосування машинного навчання у фінансовому менеджменті включає аналіз фінансових показників за допомогою автоматизованих систем, що здатні знаходити взаємозв'язки між наявними даними та надавати рекомендації щодо ефективних стратегій поведінки на фінансовому ринку.

Об'єктом дослідження є задачі менеджменту фінансів під час проведення торгів на біржі.

Предметом дослідження є методика використання алгоритмів машинного навчання в задачах фінансового менеджменту.

Метою дослідження є розробка методу та експериментальної системи прогнозування фондового ринку акцій за допомогою методів машинного навчання.

Методи дослідження

- вивчення і теоретичний аналіз наукових досліджень та виклад основних теоретичних положень дослідження;
- аналіз існуючих програмних інструментів для реалізації методів машинного навчання;
- цілеспрямоване експериментальне дослідження методів машинного навчання при побудові моделей для прогнозування цінних паперів.

Практична значущість дослідження полягає в тому, що запропоновано та експериментально обґрунтовано підхід для прогнозування фондового ринку з використанням моделей машинного навчання для підвищення прибутковості торгів.

РОЗДІЛ 1. Огляд та сучасний стан машинного навчання у фінансовому менеджменті

1.1 Огляд існуючих досліджень

Моделі, що використовуються вченими в спробі вирішити проблему прогнозування фінансових часових рядів в більшості спираються на факт, що в ціновому русі існує постійність (позитивна автокореляція). Наприклад, коли ціни підвищуються, вони, ймовірно, продовжуватимуть збільшуватися і навпаки. Популярна модель, що описує «трендовий» характер прибутків називається *Модель авторегресії – ковзного середнього* (форм. 1.1). Вона була досліджена Джорджом Боксом та Гвилимом Дженкінсом у 1970 році [9].

$$\vartheta_t = \sum_{i=1}^{p} \alpha_t \vartheta_t + \sum_{i=1}^{q} \beta_i \epsilon_{t-i} + \epsilon_t + \delta \quad (1.1)$$

де ϵ_t - незалежна однаково розподілена випадкова величина звичайного розподілу

Перші дослідження з виявлення постійності фінансового ринку були проведені в сфері акцій. Юджен Фама виявив, що більшість акцій Dow Jones демонструють позитивну щоденну послідовну кореляцію. Було помічено, що автокореляції портфелів акцій формують хвиле образну модель, що містить дуги зростаючого прибутку з позитивними автокореляціями для портфелів з короткостроковим «інвестиційним горизонтом» та досягаючими мінімуму значеннями для стратегій терміном 3-5 років [13].

Ло та МакКінлі знайшли значну позитивну автокореляцію індексу дохідності для тижневих та місячних «періодів проведення» та негативну кореляцію для окремих цінних паперів.

Технічний аналіз

У випадках, коли горизонт прогнозу орієнтований на середньо та довгострокові терміни, наприклад від кількох днів до місяців, методи, що базуються на фундаментальних макроекономічних трендах є найбільш придатними для прогнозування зміни цін. Такі техніки часто приймають форму технічного аналізу з метою зробити цілеспрямовані прогнози, або щоб з'ясувати поворотні моменти в трендах і разом з цим найліпший час для входу або виходу з торгів.

Браун та Дженнінгс показали, що технічний аналіз корисний коли ціни не є повністю інформативними та гравці на ринку мають раціональні вірування щодо співвідношення між цінами та сигналами ринку.

Нефтсі показав, що частка правил використовуваних технічними аналітиками створюють успішні техніки прогнозування, але навіть ці чітко визначенні правила показали себе неефективними в передбачені економічних часових рядів Гауса.

Брок у 1992 році дослідив 26 технічних правил торгів використовуючи дані з цінами акцій за останні 90 років та з'ясував, що вони перевершили результати ринку. Також він використав генетичне програмування для розвитку технічних правил торгів на іноземних ринках валют. Пізніше він привів докази, що використання технологічних правил торгів у періоди втручання США в роботу іноземних валютних ринків може бути прибутковим [5].

Мікроструктура ринку

На відміну від економетрики та технічного аналізу, що більше концентруються на вартості фінансових активів, мікроструктура ринку приділяє

більшу увагу самому процесу торгів, аналізуючи як конкретні механізми впливають на такі феноменя як торгові витрати, ціни, обсяги цінних паперів та поведінку ринку в цілому. Ця сфера допомогає пояснити втрат, що перешкоджають прибутковості активів.

Як результат стрімких змін в структурі та технології в світових фінансових ринках, інтерес до мікроструктури ринку істотно виріс за останні два десятиліття. Структурні зміни з'явилися через глобалізацію та зростання конкуренції між ринками. Регуляторні зміни також відіграли важливу роль. Наприклад управління Комісії з цінних паперів (SEC) в США призвело до створення Електронних комунікаційних мереж (ECN), котрі запекло конкурували з фондовими біржами за долю ринку [19].

Еволюційні техніки

Було зроблено багато спроб здійснити прогнозування фінансового ринку більш емпіричним шляхом, при якому робиться мінімальна кількість здогадок про те, як відбуваються часові ряди спостережень. Один з популярних шляхів досягнення цього – використання еволюціоних технік, де параметри моделі або самі моделі розвиваються, використовуючи прийоми схожі на генетичні процеси в світі біології, що охоплює такі функції як перехрещування та мутація.

Нілі, Веллер та Діттмар (1997) використали прийом генетичного програмування для вдосконалення технічних правил торгів. Вони помітили докази значного позанормового прибутку для кожного проаналізованного курсу валют.

Парк та Ірвін підсумували ефективність еволюційних технік у 2004 році та дійшли висновку, що генетичне програмування показало кращі результати для валютного ринку на відміну від акцій та фьючерсів [8].

Породжувальне машинне навчання

Лінійно-квадратичне оцінювання (фільтр Калмана)

Фільтр Калмана (форм. 1.2) найчастіше представляють у вигляді двох стадій: передбачення та уточнення. Стадія передбачення використовує оцінку стану з попереднього моменту часу для отримання оцінки стану в поточний момент часу (форм. 1.3). Передбачена оцінка стану також відома як апріорна оцінка стану, оскільки, хоча це й оцінка стану в поточний момент часу, вона не включає інформацію про спостереження з поточного моменту часу. У фазі уточнення поточне апріорне передбачення просуває стан до наступного запланованого спостереження, а уточнення включає і це спостереження (форм. 1.4). Проте, це не є обов'язковим якщо спостереження з якоїсь причини не доступне, уточнення може бути пропущене, і виконано декілька кроків передбачення. Аналогічно, якщо в один і той же момент часу доступно декілька незалежних спостережень, може бути виконано декілька кроків уточнення.

$$h_t = Ah_{t-1} + \mu_t^h \quad \mu_t^h \sim N(0, \sum H)$$
$$\vartheta_t = Bh_t + \mu_t^\vartheta \quad \mu_t^\vartheta \sim N(0, \sum V) \tag{1.2}$$

де h_t - прихований стан у часі t;

A - описує як він змінюється при переході з одного спостереження до наступного;

μ_t^h- коваріація шумів, що є незалежною однаково розподіленою випадковою величиною.

Передбачення:

$$h_t = Ah_{t-1}$$
$$P_t = AP_{t-1}A^T + \sum H \tag{1.3}$$

Уточнення:

$$K_t = P_t B^T (B P_t B^T + \sum V)^{-1}$$
$$h_t = h_t + K_t(\vartheta_t - B h_t)$$
$$P_t = (I - K_t B) P_t$$

(1.4)

Фільтр Калмана широко використовується на фінансових ринках як в своїй базовій формі, так і в варіаціях з різноманітними припущеннями щодо лінійності та транзитивності станів (Розширений фільтр Калмана, Неврівноважений фільтр Калмана) [15]. Найліпші результати були досягнуті при використанні на високочастотному валютному ринку.

Прихована модель Маркова

Процес Маркова порядку L є таким, де умовна незалежність $p(h_t|h_{1:t}) = p(h_t|h_{t-L:t-1})$ стала, наприклад коли L = 1, $p(h_t)$ залежить лише попереднього відрізку часу $p(h_{t-1})$. Прихована модель Маркова є такою, де приховані стани h_t дотримуються марковського процесу та видимі спостереження v_t створюються стохастично за допомогою процесу емісії, що визначає як створюються спостереження при визанченному прихованому стані, наприклад $p(\vartheta_t|h_t)$.

Спільний розподіл прихованих станів та спостережень за T часових кроків може бути представлений формулою (1.5).

$$p(h_{1:T}, \vartheta_{1:T}) = p(\vartheta_1|h_1)p(h_1)\prod_{t=2}^{T} p(\vartheta_t|h_t)p(h_t|h_{t-1})$$

(1.5)

Спільна ймовірність стану у час t та спостережень до та включаючи цей час може бути рекурсивно описаним процесом відомим як *пряма рекурсія* (форм. 1.6).

$$p(h_t, \vartheta_{1:t}) = p(\vartheta_t|h_t) \sum_{h_{t-1}} p(h_t|h_{t-1}) p(h_{t-1}, \vartheta_{1:t-1}) = \alpha(h_t)$$

(1.6)

Ймовірність усіх спостережень з часу t до T за наявності попереднього стану також може бути порахована рекурсивно використовуючи *зворотню рекурсію* (форм. 1.7).

$$p(\vartheta_{t:T}|h_{t-1}) = \sum_{h_t} p(\vartheta_t|h_t) p(h_t|h_{t-1}) p(\vartheta_{t+1:T}|h_t) = \beta(h_{t-1})$$

(1.7)

Пряма та зворотня рекурсія можуть бути поєднані, щоб отримати спільну ймовірність кожного прихованого стану використовуючи повний набір спостережень T. Це називається *алгоритмом прямого-зворотного ходу* (форм. 1.8).

$$p(h_t|\vartheta_{1;T}) = \frac{\alpha(h_t)\beta(h_t)}{\sum_{h_t} \alpha(h_t)\beta(h_t)}$$

(1.8)

Варто зазначити, що прихована модель Маркова та лінійна динамічна система по факту еквівалентні, з відмінністю в тому, що перша має дискретне представлення станів, а друга – послідовне. Прихована модель Маркова – перевірений спосіб моделювання фінансових часових рядів, тому що ринок часто представлений у різних режимах (прихованих станах), що спричиняють рух цін [3].

Розрізнювальне машинне навчання

Проблеми прогнозування фінансового ринку можуть бути виражені у намаганні знайти відношення між вихідним параметром *y* та набором D вхідних параметрів *x* де $x = \{x_1, x_2 \ldots x_D\}$. Якщо *y* представляє майбутній прибуток активу або ціну у деякий момент у майбутньому, функція *f* може бути натренована з

використанням набору наявних даних таким чином, що передбачення зможуть формуватися використовуючи нові дані, представлені моделі.

Новим та популярним методом прогнозування фінансових ринків є **штучні нейронні мережі**, що іноді включають в себе елементи технічного аналізу. Штучні нейронні мережі втілюють набір граничних функцій, що зв'язані мж собою адаптивними зваженими, що тренуються використовуючи історичні дані для того щоб мати змогу робити прогнози у майбутньому [1].

1.2 Приклади застосування машинного навчання у фінансовому менеджменті

Штучний інтелект та машинне навчання вже широко використовуються в підрозділах фінансових організацій. Великі обсяги клієнтських даних оброблюються новими алгоритмами для того щоб оцінювати якість кредитування та розмір ставок на позики. Також, такі дані можуть допомогти оцінити ризики при продажу страхових полісів.

Інструменти обчислювання кредитного рейтингу, що використовують машинне навчання розроблені для того, щоб прискорити прийняття рішень щодо видачі позик. Дані про транзакції та історію платежів в багатьох фінансових організаціях є основою для більшості моделей кредитного рейтингу. Ці моделі використовують регресію, дерева прийняття рішень та статистичний аналіз щоб генерувати кредитний рейтинг використовуючи лімітовану кількість структурованої інформації.

Фінансові підрозділи страхових компаній використовують машинне навчання для аналізу складних даних щоб зменшити витрати та підвисити прибутковість. Через те, що аналіз даних для управління цінами є ядром бізнесу страхування, технології часто спираються на аналіз «великих даних». Деякі страхові компанії активно використовують машинне навчання щоб вдосконалити ціноутворення та маркетинг страхових продуктів використовуючи дані у реальному часі, такі як поведінка в інтернет магазинах та телеметрія (сенсори в електронних

приладах). Фірми зазвичай мають доступ до таких даних через партнерства, поглинання та інші підрозділи свого бізнесу [21].

Чатботи та віртуальні помічники, що допомагають клієнтам у відповідях на питання та вирішенні проблем. Ці автоматизовані програми використовують обробку природної мови для взаємодії з клієнтами використовуючи текст або голос і використовують машинне навчання для щоб поступово самовдосконалюватися.

Чатботи вводяться в експлуатацію широким колом фірм, що надають фінансові сервіси, часто в їх соціальні мережі та мобільні додатки. Окрім безпосереднього надання допомоги клієнтам, чатботи надають фінансовим організаціям інформацію про їх клієнтів, що базується на взаємодії з ними.

Оптимізація капіталу – це традиційна функція у банківському управлінні, що активно використовує математичні підходи. Інструменти машинного навчання, побудовані на засадах великих даних та математичних концептів підвищують ефективність, точність та швидкість оптимізації капіталу. Машинне навчання находить найкращу комбінацію початкового зниження маржинальності торгів у деякий проміжок часу базуючись на показниках за попередні періоди [20].

Штучний інтелект може доповнювати традиційні моделі впливу на ринок. Фірми використовують машинне навчання для створення торгових роботів, що потім навчають себе як реагувати на зміни ринку. Аналіз впливу на ринок включає оцінку власних торгів підприємства на ринкові ціни. Зазвичай вплив торгів фірми на ринок дуже складно моделювати, особливо для менш ліквідних цінних паперів, де є недолік схожих даних за попередні періоди. Прийоми машинного навчання можуть допомогти в поліпшені моделей, що вже використовуються та введені нових підходів для мінімізації впливу на ринкові ціни і ліквідність торгів для великих «вхідних» та «вихідних» позицій.

Машинне навчання часто використовується для того, щоб ідентифікувати групи позик, що мають схожу поведінку. Таким чином, компанії отримують більше зрізків даних, на які можна спиратися та робити кращі розрахунки руху ціни.

Трейдингові фірми використовують штучний інтелект для того, щоб вдосконалити свої здібності продажів клієнтам. Наприклад, аналіз попередньої поведінки проведення торгів може допомогти передбачити наступне замовлення клієнта. Торгівля на фондових біржах генерує дуже велику кількість даних, що сприяють ефективному функціонуванню інструментів машинного навчання. Якщо сучасний тренд щодо використання сервісів типу «голос на текст» продовжиться, це призведе до отримання додаткових даних про торги проведені по телефону, що можуть бути інтегровані з потоками інформації електронних платформ.

При менеджменті інвестиційними портфелями, інструменти машинного навчання використовуються щоб ідентифікувати нові сигнали зміни цін і зробити більш ефективними велику кількість доступної інформації та ринкових досліджень порівняно з теперішніми моделями. Серед менеджерів портфелями, існують спеціальні підрозділи які за допомогою машинного навчання розробляють структуру портфелів [23].

З точки зору мікрофінансового аналізу, машинне навчання має потенціал значно вдосконалити ефективність обробки інформації, таким чином зменшивши асиметричність даних штучний інтелект може посилити інформаційну функцію фінансової системи. Механізми таких вдосконалень можуть включати наступні наслідки для фінансових ринків:

- Можливість певних учасників ринку збирати та аналізувати інформацію в більших обсягах. Зокрема, ці інструменти можуть допомогти учасникам ринку розуміти взаємозв'язки між формуванням ринкових цін та різноманітними факторами, такими як аналіз сентиментів. Це може

зменшити асиметрію даних і таким чином посприяти ефективності та стабільності ринків.

- Машинне навчання може знизити витрати учасників ринку. Більш того, штучний інтелект може дозволити їм регулювати свої торгові та інвестиційні стратегії відповідно до змін середовища, таким чином поліпшивши виявлення цін та зменшивши загальні транзакційні витрати в системі.

Зі сторони клієнтів та інвесторів машинне навчання також може надати наступні переваги:

- Клієнти та інвестори зможуть користуватися менш низькими комісіями та відсотками за кредитами.
- Мати більш широкий доступ до фінансових сервісів. Наприклад, використання роботів помічників може полегшити людям використання ринків активів для того, щоб робити інвестиції. Більш того, через поліпшену побудову кредитних рейтингів малий та середній бізнес матиме ширший вибір фінансових установ для кредитування.
- Машинне навчання зможе забезпечить більш зручні та персоналізовані фінансові сервіси шляхом аналітики великих даних. Наприклад, з'явиться можливість аналізу персоніфікованих потреб кожного окремого клієнта.

Висновки до розділу 1

Протягом останніх десятиліть наукове суспільство знаходиться у постійному пошуку засобів для більш ефективного прогнозування фінансового ринку, приділяючи все більше уваги дослідженню методів машинного навчання.

Вищенаведені приклади дають можливість усвідомити масштаби застосування нейронних мереж. Потенціал використання таких алгоритмів є очевидним для фінансових установ, що готові інвестувати у розробку новітніх систем з використанням моделей машинного навчання.

Розділ 2. Комплексний аналіз методів і інструментів машинного навчання у фінансовому менеджменті

2.1 Аналіз типів та методів машинного навчання

Дослідники в обласні комп'ютерних наук та статистики розробили просунуті техніки для отримання прихованих показників з великих розрізнених наборів даних. Дані можуть бути різних типів, з різних джерел та різної якості (структуровані та неструктуровані). Ці прийоми можуть використати можливості комп’ютерів вчитися, використовуючи попередній досвід для вирішення різноманітних задач. Нещодавнє зростання обчислювальної потужності разом із різким збільшенням кількості та доступності інформації призвели до відродження інтересу щодо потенційного застосування штучного інтелекту.

Багато інструментів машинного навчання базуються на статистичних методах. Залежності, що виявляються моделями машинного навчання не обмежені лінійними залежностями, тому домінують над традиційними економічним та фінансовим аналізом.

Існує декілька категорій алгоритмів машинного навчання. Ці категорії варіюються відповідно до рівня людського втручання для підготовки тренувальних даних:

- Глибоке навчання – форма машинного навчання, що використовує алгоритми котрі діють пластами і використовують структуру побудови людського мозку як прообраз

- В керованому машинному навчанні, алгоритм використовує тренувальні дані, що мають позначки для частки спостережень. Наприклад, набір даних транзакцій може містить позначки для деяких параметрів, що визначають чи є вони шахрайськими. Алгоритм вивчить загальне правило класифікації, та буде використовувати його для позначення інших спостережень в наборі даних.
- Некероване навчання відносить до ситуацій коли наявні дані, що використовуються алгоритмом не містять позначок. Задачею алгоритму є визначити закономірності шляхом виявлення кластерів спостережень, що залежать від схожих похідних характеристик. Наприклад, алгоритм некерованого машинного навчання може бути налаштованим для пошуку цінних паперів, що мають характеристики схожі на неліквідні цінні папери.
- Машинне навчання з підкріпленням – алгоритм використовує дані без позначок, обирає дію для кожного спостереження та отримує зовнішній сигнал щодо правильності обраної дії.

Методи керованого машинного навчання намагаються встановити зв’язок між вхідними параметрами (що називаються незалежними змінними) та цільовим атрибутом (залежною змінною). Виявлена взаємодія між змінними має вигляд структури, що називається моделлю. Зазвичай моделі описують та пояснюють феномени, що приховані всередині набору даних і можуть бути використані для прогнозування значення цільового атрибуту, маючи відомі значення вхідних атрибутів.

Існує дві основних групи моделей керованого машинного навчання: моделі класифікації та моделі регресії. Регресійні моделі трансформують область вхідних значень у діапазон дійсних чисел. Наприклад, алгоритм регресії може прогнозувати попит на конкретний продукт маючи його характеристики. З іншого боку,

класифікатори зіставляють область вхідних значень і заздалегідь відомі класи. Наприклад, алгоритм класифікації може поділити осіб, що взяли позики на порядних (повністю виплатили суму у встановлений термін) та непорядних (прострочили платіж). Існує багато альтернативних форм моделей класифікації, наприклад метод опорних векторів, дерева прийняття рішень, алгебраїчні функції.

В типовому сценарії керованого машинного навчання, тренувальний набір даних має ціль скласти опис явища для того щоб прогнозувати нові спостереження. Набір даних має форму колекції рядів, що можуть мати дублікати. Кожен ряд представлений вектором атрибутів. Заголовки набору даних надають опис атрибутів та їх призначення.

Атрибути (поля, змінні, властивості) зазвичай належать до одного з двох типів: номінальні (значення та члени несортованого списку) або чисельні (значення є дійсними числами).

Зазвичай допускається, що ряди тренувального набору даних генеруються випадковим і незалежним чином відповідно до невідомого та фіксованого розподілу ймовірностей.

Для вирішення задачі класифікації ведеться пошук функції, що зіставляє набір всіх можливих прикладів у визначений набір позначок класу.

Алгоритм індукції, або індуктор – це сутність, що бере тренувальний набір даних и формує модель яка узагальнює взаємозв’язок між вхідними атрибутами та цільовим атрибутом. Наприклад, індуктор може приймати в якості вхідних параметрів окремі тренувальні ряди з відповідними позначками класів і створювати класифікатор.

Класифікатор згенерований індуктором може бути використаний для того, щоб класифікувати невідомий ряд даних шляхом явного присвоєння до конкретного класу або створюючи вектор ймовірностей, що відображає умовну

ймовірність того, що даний об’єкт належить до кожного окремо взятого класу (ймовірнісний класифікатор).

Оцінка продуктивності індуктора – фундаментальний аспект машинного навчання. Класифікатор та індуктор можуть бути оцінені використовуючи певні критерії. Оцінка важлива для розуміння якості моделі (індуктора), для вдосконалення параметрів в ітераційному процесі навчання та для вибору найбільш приємної моделі серед набору представлених варіантів [12].

Існує декілька критеріїв для оцінки моделей та індукторів. Зазвичай найбільш продуктивними моделями вважаються ті, що мають найвищу точність. Однак, існують інші критерії котрі також є важливими. Серед них обчислювальна складність та зрозумілість створеного класифікатора.

Помилка узагальнення – ймовірність неправильно класифікувати об’єкт, обраний відповідно до розподілу діапазону позначок об’єкту. Хоча помилка узагальнення – натуральний критерій, її реальне значення відоме у рідких випадках. Причиною цього є те, що часто розподіл позначок є невизначеним. Якщо не враховувати помилку, зазвичай оцінка буде оптимістично зміщеною, особливо якщо алгоритм машинного навчання «перетренований».

Іншим корисним критерієм для порівняння індукторів і класифікаторів є їх обчислювальна складність. Обчислювальна складність – кількість процесорного часу, що використовується кожним індуктором. Виділяють три метрики обчислювальної складності:

- Обчислювальна складність для створення нового класифікатора – найбільш важлива метрика, особливо коли потрібно масштабувати алгоритм машинного навчання для великого набору даних. Більшість алгоритмів мають обчислювальну складність більше ніж лінійну, що викликає значне сповільнення алгоритму при роботі з великими обсягами інформації.

- Обчислювальна складність для оновлення класифікатора – при надходженні нових даних обчислювальна потужність необхідна для оновлення всіх існуючих класифікаторів.
- Обчислювальна складність для класифікації нового об’єкта – зазвичай вплив цього типу є невеликим. Однак для окремих методів або додатків, що працюють у режимі реального часу цей параметр може бути критичним.

Класичні алгоритми індукції застосовуються з практичним успіхом в багатьох ситуаціях, з обмеженою кількістю даних. Однак, для дослідження ширших тенденцій вимагає взаємодії з великими сховищами даних, що викликає проблеми швидкодії. Управління та аналіз великих сховищ даних вимагає спеціальне і дуже дороге програмне забезпечення та комп’ютерні комплектуючі. Великі бази даних створюють наступні складності:

- Обчислювальна складність – час обробки алгоритмів при великих обсягах даних є значно довшим.
- Погана точність класифікації через складності у пошуку правильного класифікатора. Через великий обсяг даних ймовірність обрати перенавчений класифікатор зростає.
- Проблеми зберігання даних. В більшості алгоритмів машинного навчання, весь набір даних повинен бути зчитаний з жорсткого диску у оперативну пам’ять перед початком процесу навчання. Це створює складнощі, адже обсяг оперативної пам’яті значно менший ніж обсяг жорсткого диску.

Складнощі в реалізації алгоритмів класифікації при роботі з великими даними виникає через підвищення кількості записів та об’єктів в базі даних [6]. Існують наступні підходи для вирішення подібних проблем:

- Методи зразків – обираються певні записи з набору даних використовуючи техніки зразків.

- Агрегація – зменшує кількість записів поєднуючи декілька рядів в один, або ігноруючи деякі, «неважливі» зразки даних.
- Паралельна обробка - використовуються технології паралельної обробки для того, щоб одночасно аналізувати декілька аспектів даних.
- Ефективні методи зберігання даних, що дозволяють алгоритму оперувати великою кількістю рядів.
- Зменшення діапазону пошуку алгоритму.

2.2 Особливості використання програмних продуктів в задачах машинного навчання

Використання сучасних технологій стає все більшою перевагою у світі фінансів. Фінансові установи все більше трансформуються у технологічні компанії та значно розширюють сферу своєї діяльності замість того щоб зберігати свій фокус лише на фінансових ринках. Швидкість і частота фінансових транзакцій разом із великими обсягами даних сприяють впровадженню новітніх технологій для аналізу великих даних та отриманню додаткової інформації, що слугує інструментом для прийняття рішень.

У даному підрозділі розглянуті особливості застосування популярних програмних продуктів для вирішення задач машинного навчання та побудови моделей, що застосовуються установами різних типів.

Хоча машинне навчання найбільшим чаном концентрується на моделюванні даних, процес також включає декілька інших етапів, кожен з котрих використовує спеціалізовані програмні бібліотеки (Рис. 2.1).

Фаза моделювання даних не може розпочатися доки дані не проаналізовані. Але перед цим етапом проводиться підготовка даних, що створює фундамент для застосування алгоритмів.

Мова програмування Python має нескінченну кількість інструментів, що можуть бути використані на різних етапах роботи з даними.

Екосистема мови програмування Python може бути поділена на три основні групи пакетів:

- Обробка даних, завантажених в оперативну пам’ять комп’ютера
- Оптимізація коду та обчислювальної потужності
- Робота з «великими даними»

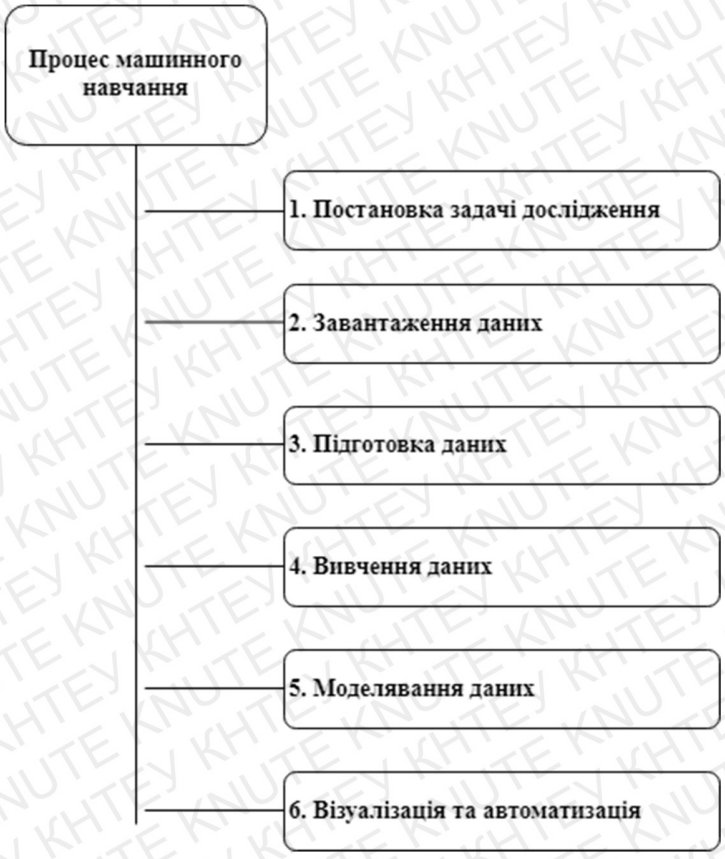

Рис. 2.1. Етапи процесу машинного навчання

Перша група пакетів зазвичай використовується для простих задач коли дані можуть бути оброблені однією машиною. Друга група використовується для оптимізації коду після того, як була закінчена стадія прототипування та виникають проблеми пов’язані з швидкодією при роботі в реальних умовах. Третій тип є

специфічним для використання мови програмування Python у поєднанні з технологіями великих даних [13].

Під час прототипування наступні бібліотеки стають у нагоді для маніпуляції даними:

- SciPy – надає високорівневі команди та класи для маніпуляції даними і візуалізації даних, що заначно підвищує продуктивність та інтерактивність процесу моделювання з використання мови програмування Python. В додачу до цього бібліотека містить математичні алгоритми, що прискорює процес розробки спеціалізованих додатків.
- NumPy – пакет мови програмування Python, що містить потужні структури даних, а саме багатовимірні масиви та матриці. Наступний фрагмент коду демонструє завантаження набору даних з файлу в оперативну пам’ять та створення масиву (Рис. 2.2).

```
import numpy as np
import urllib

# download the file
raw_data = urllib.urlopen(url)
# load the CSV file as a numpy matrix
dataset = np.loadtxt(raw_data, delimiter=",")
# separate the data from the target attributes
X = dataset[:,0:7]
y = dataset[:,8]
```

Рис. 2.2. Створення масиву NumPy

- Matplotlib – найбільш популярна бібліотека мови програмування Python для візуалізації даних, що надає широкий спектр функціоналу для побудови 2D та 3D графіків.
- Pandas – пакет для маніпуляції наборами даних, що створює спеціальний об'єкт під назвою dataframe який містить ряди та стовбці.
- SymPy – пакет, що використовується для комп’ютерної алгебри.

- StatsModels – пакет, що містить статистичні методи та алгоритми.
- Scikit-learn – бібліотека, що містить різноманітні алгоритми машинного навчання, включаючи регресію та класифікацію.
- RPy2 – дозволяє використовувати функції, що написані за допомогою мови програмування R в інтерактивному середовищі мови програмування Python.
- NLTK – набір інструментів для роботи з текстом та обробкою природньої мови.

Після закінчення стадії прототипування, алгоритм потребує доопрацювання та застосування технік оптимізації коду та швидкодії. Існує набір інструментів, що допомагають створити необхідну інфраструктуру.

- Numbda і NumbdaPro – бібліотеки, що забезпечують «компіляцію за вимогою» для додатків написаних мовою програмування Python. Також дозволяють використовувати графічні процесори для поліпшення продуктивності програм.
- PyCUDA – дозволяє писати код, що виконується на графічному процесорі замість звичайного, що є ідеальною опцією для додатків з важкими математичними розрахунками. Найкращий результат досягається при паралельній обробці зразків даних на різних ядрах процесору.
- Cython – комбінує мови програмування C та Python. Через те, що C – мова програмування низького рівня, швидкість обчислень збільшується.
- Blaze – містить структури даних, розмір яких може бути більшим за обсяг оперативної пам’яті комп’ютера, що дозволяє працювати з більшими обсягами даних.
- PySpark – підключає середовище мови програмування Python до фреймворку великих даних Spark, щоб забезпечити можливість розподілених обчислень.

Неперевершена швидкість Apache Spark досягається через використання оперативної пам’яті (RAM) замість традиційного жорсткого диска (HDD\SSD) [22]. Spark побудований навколо концепцій «гнучкого розподіленого набору даних» (RDD) та «прямого ациклічного графа» (DAG), що представляють трансформації та залежності наборів даних. Обробка даних виконується за допомогою серії трансформацій доки не буде досягнутий фінальний результат. Такі трансформації контролюються «планувальником», що контролює обробку на різних «вузлах» кластеру (Рис. 2.3).

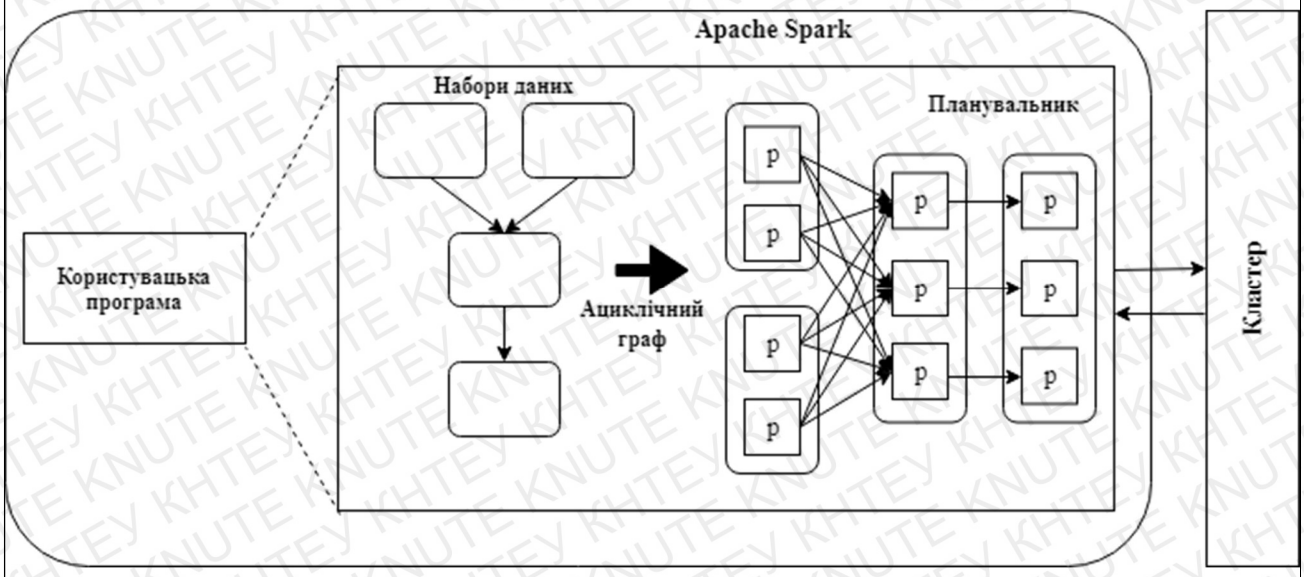

Рис. 2.3. Архітектура фреймворку Apache Spark

Окремим класом програмних продуктів для машинного навчання є хмарні сервіси. Нині існує три головних технології, що дозволяють швидке тренування моделей та не вимагають наявності широкої експертизи в області машинного навчання:

- Amazon Machine Learning services
- Azure Machine Learning
- Google Cloud AI

Однак, активне використання такого роду технологій значно підвищує витрати на процес розробки.

Висновки до розділу 2

В цьому розділі були розглянуті основні методи та етапи процесу машинного навчання, що будується навколо обробки наборів даних та подальшої побудови моделей з їх використанням.

Через зростаючу популярність машинного навчання з'явилась безліч інструментів, призначених для вирішення тих чи інших задач.

Етапи прототипування та реального застосування моделей мають свої особливості, адже для впровадження алгоритму у реальні бізнес системи потрібно враховувати такі аспекти як швидкодія та загальні затрати на підтримку системи.

РОЗДІЛ 3. Застосування моделей машинного навчання для менеджменту торгів індексу S&P 500

3.1 Постановка задачі

Основною задачею даного дипломного проекту є демонстрація та аналіз ефективності застосування розповсюджених алгоритмів машинного навчання для фінансового менеджменту ринку цінних паперів та підвищення прибутковості інвестиційного портфелю порівняно з традиційними методами прогнозування.

В якості об'єкту прогнозування був обраний індекс S&P 500 створений компанією Standard and Poor's у 1957 році. У фондовий індекс S&P 500 включено 500 акціонерних компаній США, що мають найбільшу капіталізацію. Акції всіх компаній зі списку торгуються на найбільших американських фондових біржах, таких як Нью-Йоркська фондова біржа та NASDAQ. Індекс є середньозваженим акцій всіх цих компаній, що робить його еталоном для відслідковування тенденцій ринку та економічних циклів.

Через велику популярність індексу багато компаній та фізичних осіб намагаються будувати свій інвестиційний портфель відповідно до пропорцій капіталу індексу. Для торгів потребується купівля активів, що є прямою складовою індексу.

Традиційним способом торгів індексу є CFD («контракт на зміну»). CFD – це договір між клієнтом та брокером, предметом якого є так звані «фьючерси». Він дозволяє не купувати самі активи, а вести розрахунки лише на основі різниці цін договору.

Задачею моделі машинного навчання, що розглядається є зменшення ризику під час торгівлі CFD і разом з тим максимізація отримуваного прибутку. Нейронна

мережа буде аналізувати історичні данні торгів та на основі цього прогнозувати різницю стартової та кінцевої цін наступного торгового дня. Це дозволить фінансовим менеджерам більш зважено заключати нові CFD.

Для розробки вибрана проста стратегія – за умови, що торговий день згідно прогнозу буде «висхідним» відкрити позицію коли біржа Волл Стріт починає торги о пів на десяту ранку та продати її під час закриття в чотири години вечора. Якщо ціна закриття позиції вище стартової ціни – приріст позитивний.

Згідно теорії ймовірності, вірогідність передбачення «висхідного»/«низхідного» торгового дня становить 50%. Отже, модель машинного навчання можна вважати ефективною, якщо точність прогнозування буде вищою за 50%.

В розділі показано, як тренувати методи машинного навчання, описано основні етапи процесу і наведено реалізацію необхідних математичних та програмних функцій.

3.2 Розробка моделі алгоритмічних торгів індексу S&P 500

3.2.1 Підготовка масиву даних для тренування нейронних мереж

Для побудови фінансової системи автоматичних торгів було використано мову програмування Python та прийоми машинного навчання. В якості джерела для отримання історичних даних, необхідних для тренування моделі та формування прогнозів був обраний сервіс Yahoo Finance. Отже, для початку роботи необхідно створити локальне середовище розробки з наступними інструментами:

- Мова програмування Python та зокрема IPython notebook. IPython notebook також відомий як Jupyter Notebook – інтерактивне обчислювальне середовище, в якому можна комбінувати виконання коду, змістовне документування, математична нотація для функцій, графіки та інші медіа файли (HTML, LaTeX, PNG, SVG).

- Програмний пакет Yahoo Finance Python, що встановлюється наступною командою терміналу:

pip install yahoo-finance

- Пакет машинного навчання GraphLab

Для початку роботи перш за все потрібно створити новий Jupyter «щоденник». Найлегше це зробити наступною інструкцією командної строки:

jupyter notebook

Команда створить новий файл «щоденника» в поточній папці та запустить його в браузері с портом **8888 (**Рис. 3.1).

```
$ jupyter notebook
[I 20:55:34.440 NotebookApp] Serving notebooks from local di
rectory: C:\Projects\conda
[I 20:55:34.440 NotebookApp] 0 active kernels
[I 20:55:34.440 NotebookApp] The Jupyter Notebook is running
 at: http://localhost:8888/
[I 20:55:34.440 NotebookApp] Use Control-C to stop this serv
er and shut down all kernels (twice to skip confirmation).
```

Рис. 3.1. Запуск щоденника Jupyter

Першим кроком напишемо в «щоденнику» код для завантаження історичних цін індексу S&P 500 (Рис. 3.2).

```
import graphlab as gl
from __future__ import division
from datetime import datetime
from yahoo_finance import Share
import pandas as pd

# Скачуємо історичні ціни
today = datetime.strftime(datetime.today(), "%Y-%m-%d")

# ^GSPC це символ Yahoo Finance, що ссилається на S&P 500
frame = pd.read_csv('Data/^GSPC.csv')

# Збираємо історичні показники в початку 2001 року до сьогодні
df = frame[(frame['Date'] > '2001-01-01') & (frame['Date'] <= today)]

# Рядки мають наступний вигляд
print df.head(1)
```

```
         Date         Open         High          Low        Close  \
0  2018-10-05  2902.540039  2909.639893  2869.290039  2885.570068

     Adj Close      Volume
0  2885.570068  3328980000
```

Рис. 3.2. Завантаження набору даних із зовнішнього джерела

В цьому прикладі створюються двовимірний «фрейм» даних за допомогою бібліотеки Pandas. Змінна **df –** список словників, кожен об’єкт словника – це день торгів з полями **Open**, **High**, **Low**, **Close**, **Adj_Close**, **Volume**, **Symbol** та **Date**.

Протягом кожного торгового дня ціна зазвичай змінюється починаючи зі стартової **Open** до кінцевої **Close**, при цьому досягаючи максимального та мінімального значення **High** і **Low**. Потрібно повністю прочитати словник та створити списки кожного значущого показника. Також дані повинні бути відсортовані починаючи з найбільш нових спостережень, тому необхідно «перевернути» масив (Рис. 3.3). Окрім зазначених стовбців індекс може включати наступні допоміжні показники:

- Просте рухоме середнє – арифметичне рухоме середнє, що розраховується додаванням цін «закриття» цінних паперів за визначений період та діленням на ту саму кількість сумарних показників.
- Індекс товарного каналу – зміна відхилення ціни від її середнього значення.

```
l_date = []
l_open = []
l_high = []
l_low = []
l_close = []
l_volume = []

# Перевертаэмо список
df[::-1]
for quotes in df:
    l_date.append(df['Date'])
    l_open.append(pd.to_numeric(df['Open']))
    l_high.append(pd.to_numeric(df['High']))
    l_low.append(pd.to_numeric(df['Low']))
    l_close.append(pd.to_numeric(df['Close']))
    l_volume.append(pd.to_numeric(df['Volume']))
```

Рис. 3.3. Трансформація вхідного набору даних

Тепер можна сформувати всі завантаженні показники в об'єкт **SFrame**, котрий дуже добре масштабується на основі стовбців даних та підтримує компресію. Ще однією перевагою є те, що цей об'єкт може бути більшим за кількість оперативної пам'яті комп'ютера тому як зберігається на жорсткому диску. Отже, наступим кроком конвертуємо словник даних (Рис. 3.4).

```
qq = gl.SFrame({'datetime' : l_date,
        'open' : l_open,
        'high' : l_high,
        'low' : l_low,
        'close' : l_close,
        'volume' : l_volume})

# перевіряємо що дані відсортовані у "висхідному" порядку
qq.head(3)
```

Рис. 3.4. Створення об'єкту SFrame

Після цього зберігаємо данні на жорсткий диск (Рис. 3.5).

```
qq.save("SP500_daily.bin")

# після того як дані збережені, можна використати наступну інструкцію щоб їх загрузити
qq = gl.SFrame("SP500_daily.bin/")
```

Рис. 3.5. Збереження підготовлених даних на жорсткий диск

Для візуалізації «розподілу» даних показників використовується бібліотека **pyplot** (Рис. 3.6). В цьому прикладі проілюстрована динаміка **Close** з 2001 по 2018 рік (Рис. 3.7).

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(qq['Close'])
```

Рис. 3.6. Застосування бібліотеки pyplot

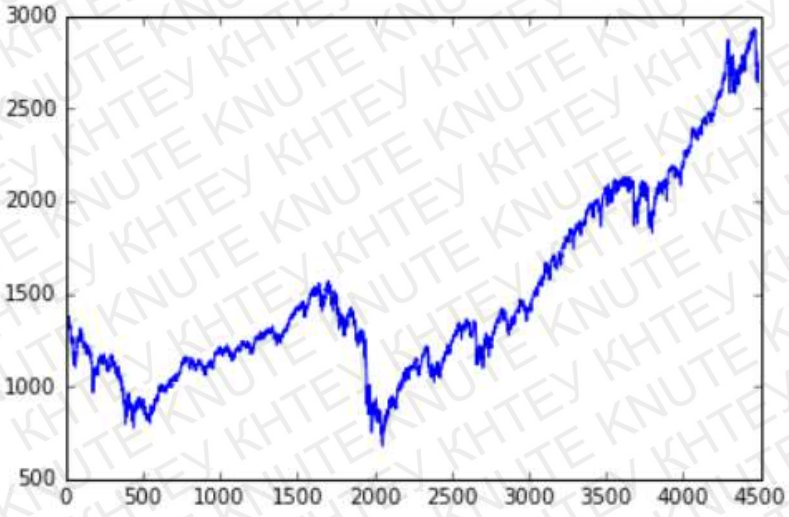

Рис. 3.7. Графік зміни ціни закриття

Ціллю моделі є прогнозування того, чи кінцева ціна буде вище за початкову ціну. В цьому випадку можливо отримати прибуток при купівлі даного активу. Отже, необхідно додати стовбець **Outcome** (результат), що буде містити цільову або прогнозовану змінну (Рис. 3.8). Кожен рядок цього стовбця бути мати значення:

- +1 для «висхідного» дня з **кінцевою** ціною вище **початкової** ціни
- -1 для «низхідного» дня з **кінцевою** ціною нижче **початкової** ціни

```
# Додаємо змінну 'результату', 1 якщо торгова сессія була прибутковою (Close>Open), інакше -1
qq['Outcome'] = qq.apply(lambda x: 1 if x['Close'] > x['Open'] else -1)
# Також потрібно додати 3 нові колонки 'Ho' 'Lo' та 'Gain'
# Вони знадобляться для подальшого тестування моделі
qq['Ho'] = qq['High'] - qq['Open'] # Різниця між Highest та Opening ціною
qq['Lo'] = qq['Low'] - qq['Open'] # Різниця між Lowest та Opening ціною
qq['Gain'] = qq['Close'] - qq['Open']
```

Рис. 3.8. Додавання стовбців до набору даних

Через те що останні торгові дні потребують оцінки, потрібно змістити данні спостережень на один або декілька днів. Для операцій подібного роду використовується спеціальний об'єкт програмного пакету **GraphLab** під назвою **TimeSeries**. **TimeSeries** має метод **shift,** що зміщує дані на задану кількість рядків (Рис. 3.9).

```
ts = gl.TimeSeries(qq, index='Date')

ts['Outcome'] = ts.apply(lambda x: 1 if x['Close'] > x['Open'] else -1)


ts_1 = ts.shift(1)
ts_2 = ts.shift(2)
```

Рис. 3.9. Застосування методу shift

Змінні **ts_1** та **ts_2** містять масиви даних зі зміщенням в 1 та 2 дні відповідно. Часто рішення щодо кількості рядків для зміщення є довільним та може по різному впливати на кінцеву результативність прогнозування, тому значення обирається експериментальним шляхом.

Наступним кроком потрібно додати «прогнозувальні змінні». Такі змінні мають певні властивості та мають бути обрані для тренування моделі і прогнозування результату. Таким чином, вибір фактора прогнозування є критичним, якщо не найважливішим компонентом для моделей складання прогнозів [16]. В наведеному прикладі фактором вартим уваги може бути ситуація,

коли кінцева ціна позиції на сьогодні вища за аналогічний показник вчора, ця тенденція може бути розширена аналізом значень двох попередніх днів, і так далі. Такий вибір може бути виражений наступним фрагментом коду, що додає два нових стовбця з властивостями до існуючого набору даних (Рис. 3.10).

```
ts['Feat1'] = ts['Close'] > ts_1['Close']
ts['Feat2'] = ts['Close'] > ts_2['Close']
```

Рис. 3.10. Додавання нових властивостей

Перед безпосереднім застосуванням тренувальних алгоритмів необхідно розділити наявний масив даних на дві частини: тренувальну та перевірочну (Рис. 3.11). Класичною пропорцією для такого поділу є 80/20 відсотків.

```
ratio = 0.8
training = ts.to_sframe()[0:round(len(ts)*ratio)]
testing = ts.to_sframe()[round(len(ts)*ratio):]
```

Рис. 3.11. Поділ масиву на тренувальний та перевірочний

Це пізніше дозволить оцінити точність прогнозування за допомогою моделі, використовуючи окремий набір даних. Це дозволяє уникнути явища, відомого під назвою «перенавчання моделі». Перенавчання – це стан, за якого статистична модель починає описувати випадкові помилки в даних, замість виявлення залежностей між змінними. В регресійному аналізі перенавчання може створювати оманливі коефіцієнти детермінації та значення вірогідності. «Перенавчена» модель має забагато умов відносно до кількості спостережень. Коли це відбувається, регресійні коефіцієнти представляють погрішності замість істинних зв'язків сукупності об'єктів.

На рисунку зображено «перенавчену» модель (Рис. 3.12). Випадкова погрішність, що притаманна даним призводить до розташування точок у довільному порядку біля прямої. Хвиляста лінія зображує перенавчену модель.

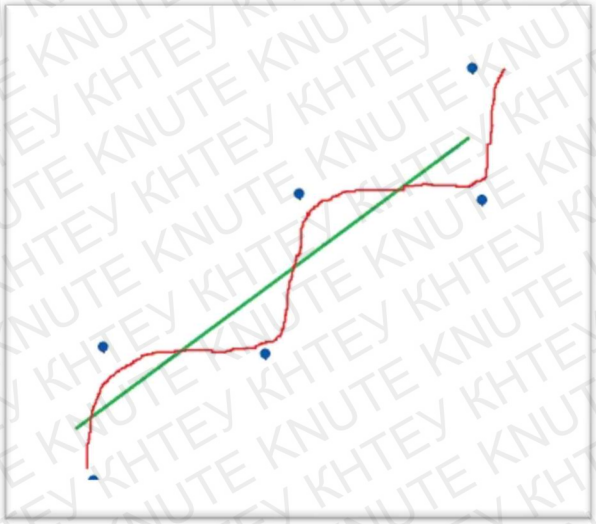

Рис. 3.12. Графік властивостей перенавченої моделі

Процес вибору алгоритму для тренування нейронної мережі є експериментальним та значною мірою залежить від розміру вхідного набору даних, кількості властивостей та ступеню кореляції їх значень. Для тренування даної моделі розглянемо декілька прийомів, що дозволить оцінити точність прогнозування за допомогою кожного з них і вибрати оптимальний.

3.2.2 Навчання моделей з використанням алгоритмів регресії та класифікації

Дерево прийняття рішень

Загальний алгоритм дерева прийняття рішень може бути описаний наступним чином:

- Вибір найліпшого атрибута/властивості. Найліпший атрибут вважається таким, що максимально ефективно розбиває дані.
- Постановка «доречного» питання.
- Дотримання ланцюга відповідей
- Повернення до кроку 1 доки не буде знайдено відповідь

Дерева прийняття рішень можуть зображувати будь-яку Булеву функцію вхідних атрибутів. Розглянемо дерева рішень для виконання функції трьох різних Булевих вентиля: І, АБО, виключне АБО.

Логічний вентиль І реалізує логічну кон'юнкцію, двомісну логічну операцію, що має значення «істина», якщо всі операнди мають значення «істина». Функція може бути розширена до будь-якої кількості вхідних параметрів (Табл. 3.1).

Таблиця 3.1

Булева функція І

A	B	A і B
Ні	Ні	Ні
Ні	Так	Ні
Так	Ні	Ні
Так	Так	Так

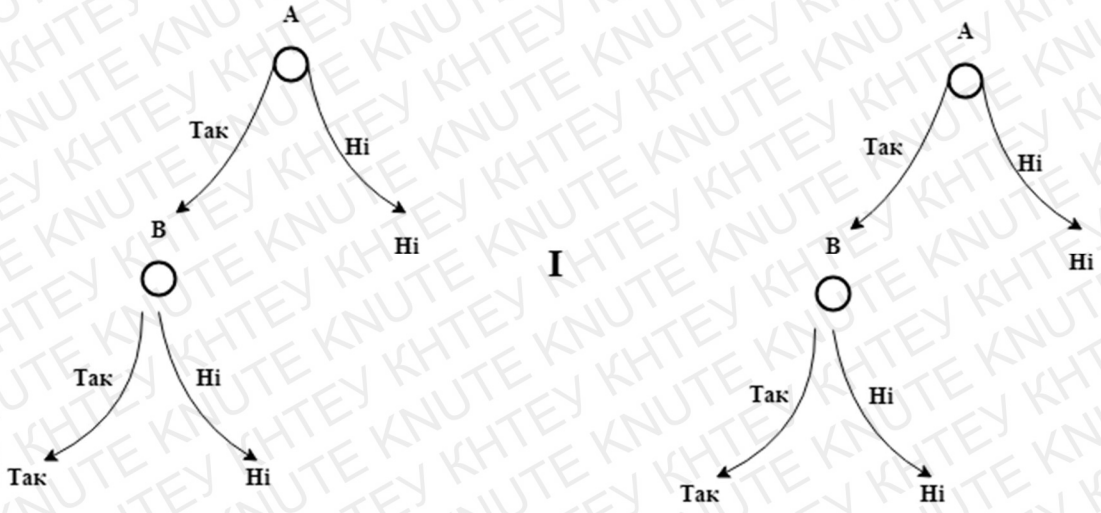

Рис. 3.13. Принцип роботи логічного вентилю І

Логічний вентиль АБО реалізує логічну диз'юнкцію, двомісну логічну операцію, що має значення «істина», якщо хоча б один з операндів має значення «істина». Значення «хиба» вихідного параметру досягається лише у випадку, коли

обидва вхідні параметри хибні. Таким чином, функція АБО ефективно знаходить **максимум** з двох бінарних чисел (Табл. 3.2).

Таблиця 3.2

Булева функція АБО

A	**B**	**A або B**
Ні	Ні	Ні
Ні	Так	Так
Так	Ні	Так
Так	Так	Так

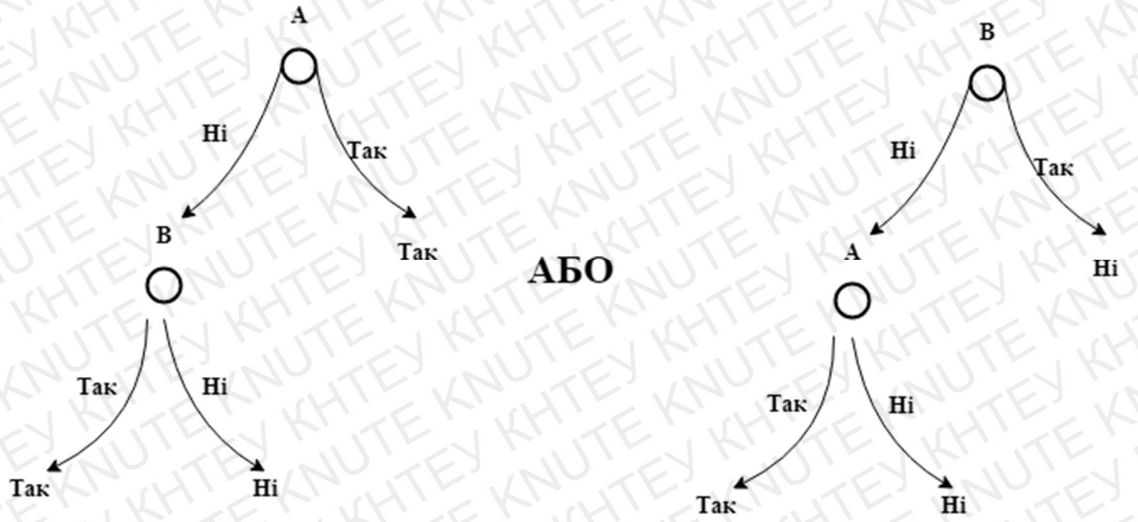

Рис. 3.14. Принцип роботи логічного вентилю АБО

Логічний вентиль **виключне АБО** реалізує операцію виключної диз'юнкції, що є бітовою операцією та приймає значення «істина» тоді і лишн тоді коли значення «істина» має рівно один з її операндів. Виключна диз'юнкція є запереченням логічної еквівалентності. У випадку двох змінних результат виконання операції є істинним тоді і тільки тоді, коли лише один з аргументів є істинним. Для функції трьох і більше змінних результат виконання операції буде

істинним тільки тоді, коли аргументів рівних 1 на заданому наборі буде непарна кількість (Табл. 3.3).

Таблиця 3.3

Булева функція виключне АБО

A	B	A XOR B
Ні	Ні	Ні
Ні	Так	Так
Так	Ні	Так
Так	Так	Ні

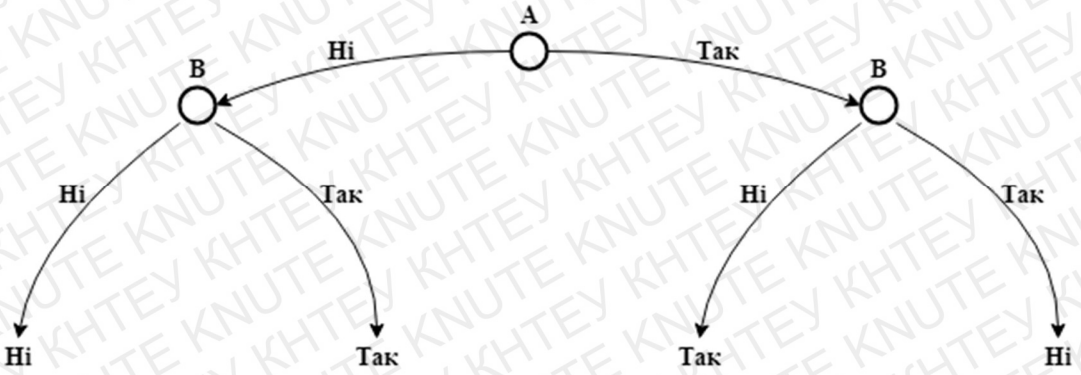

Рис. 3.15. Принцип роботи логічного вентилю виключне АБО

Базовий алгоритм, що використовується в деревах прийняття рішень має назву ID3. Алгоритм ID3 будує дерева рішень, використовуючи підхід «з верху до низу» та «жадібні» алгоритми.

Для створення моделі дерева прийняття рішень бібліотека **GraphLab** містить метод **create**, що використовується для навчання моделі тренувальним набором даних. Метод приймає наступні параметри:

- **training** – тренувальний набір даних зі стовпцями властивостей та результату
- **target** – ім’я колонки, що містить цільову змінну

- **validation_set** – масив даних для моніторингу узагальненої точності моделі. В даному випадку не використовується.
- **features** – список імен колонок з властивостями, що будуть використані в якості параметрів для тренування
- **verbose** - визначає чи виводити детальну інформацію про процес тренування у консоль
- **max_depth** – максимальна глибина дерева

Наступний фрагмент коду побудує дерево рішень (Рис. 3.16).

```
l_features = ['Ho', 'Lo']

max_tree_depth = 6
decision_tree = gl.decision_tree_classifier.create(
    training, validation_set=None,
    target='Outcome', features=l_features,
    max_depth=max_tree_depth, verbose=False)
```

Рис. 3.16. Побудова моделі дерева рішень

Під час тренування моделі були обрані властивості **Ho** та **Lo**, що представляють дельти між стартовою та граничними цінами протягом дня.

Для вимірювання продуктивності моделі використовуються наступні показники:

- Точність – важлива метрика для оцінки доброякісності моделі прогнозування. Вираховується шляхом ділення правильних передбачень та загальну кількість спостережень. Так як модель навчається з використанням тренувальних даних, точність тренувального набору оцінюється вище ніж для перевірочного.
- Влучність – частка позитивних передбачень. Еталонне значення цього показника наближається до одиниці. Побудоване дерево рішень є класифікатором програмного пакету **GraphLab**, що має метод **evaluate** для отримання важливих метрик навченої моделі.

Повнота – кількісно визначає здатність класифікатора передбачати позитивні випадки. Повнота може бути інтерпретована як ймовірність того, що випадково-обраний позитивний випадок був коректно ідентифікований класифікатором. Еталонні моделі мають повноту, що наближається до одиниці.

Нижченаведений фрагмент коду показує **точність** навченої моделі для обох наборів даних: тренувального та перевірочного (Рис. 3.17). Для перевірочного набору даних вона складає 82 %, що є гарним показником порівняно зі «сліпим» вибором.

```
decision_tree.evaluate(training)['accuracy'], decision_tree.evaluate(testing)['accuracy']
```

```
(0.8758006126427179, 0.8207126948775055)
```

Рис. 3.17. Оцінка точності дерева рішень

Для прогнозування даних **GraphLab** має інтерфейс, що підтримує різні типи навчених моделей. Використаємо метод **predict**, що потребує перевірочний набір щоб спрогнозувати цільову змінну, а саме **Outcome**.

```
predictions = decision_tree.predict(testing)

testing['Predictions'] = predictions

testing[['Date', 'Outcome', 'Predictions']].head(10)
```

Рис. 3.18. Додання стовбця передбачень до масиву

Додаємо колонку **Predictions** до перевірочного масиву даних. Для оцінки точності моделі можна порівняти передбачення з реальним результатом для 10 рядків (Рис. 3.18).

Рядки можна поділити на дві групи. Помилково-позитивні випадки коли модель прогнозує позитивний результат коли як реальний результат з перевірочного набору є негативним. І навпаки, помилково-негативні випадки коли модель прогнозує негативний результат, що насправді виявляється позитивним.

Стратегія торгів, що розглядається чекає на позитивно-прогнозований результат щоб купити індекс S&P 500 за стартовою ціною і продати за кінцевою ціною. Через це цілю моделі є щонайменший відсоток помилково-позитивних результатів для уникнення втрат. Іншими словами, модель повинна наближатися до найвищого відсотку *влучності*.

Таблиця 3.4

Перші десять спостережень

Date	Outcome	Predictions
2015-04-15 00:00:00+00:00	1	1
2015-04-16 00:00:00+00:00	-1	-1
2015-04-17 00:00:00+00:00	-1	-1
2015-04-20 00:00:00+00:00	1	1
2015-04-21 00:00:00+00:00	-1	1
2015-04-22 00:00:00+00:00	1	1
2015-04-23 00:00:00+00:00	1	1
2015-04-24 00:00:00+00:00	1	1
2015-04-27 00:00:00+00:00	-1	-1
2015-04-28 00:00:00+00:00	1	-1

Таблиця з 10 перших передбачень перевірочного набору містить одне помилково-негативне значення за 2015-04-28 та одне помилково-позитивне за 2015-04-21 (Табл. 3.4). Маючи ці дані можливо розрахувати основні показники продуктивності моделі:

- точність = 8/10 = 0.8 або 80%
- влучність = 4/5 = 0.8 або 80%
- повнота = 4/5 = 0.8 або 80%

В даному випадку вони є однаковими.

Для валідації моделі створюємо симуляцію того, як будуть проводитися торги використовуючи прогнозовані значення. Якщо передбачений результат дорівнює +1, очікується «висхідний» день. Для висхідного дня індекс купується на початку сесії та продається під її кінець того ж самого дня. І навпаки, якщо передбачений результат дорівнює -1, очікується «низхідний» день, протягом якого торгувати невигідно.

Логістична регресія

Логістична регресія – це широко використовувана статистична модель, що в своїй базовій формі використовує логістичну функцію для моделювання залежної бінарної змінної. Математично, бінарна логістична модель має залежну змінну з двома можливими значеннями – «виграш» та «програш», вони виражені фіктивною змінною, де два значення помічаються як 0 та 1.

Тренувальний набір даних обробляється **методом максимальної правдоподібності**. Цей метод є загальним алгоритмом навчання, що використовується багатьма алгоритмами машинного навчання. Найкращі коефіцієнти призведуть до того, що модель буде прогнозувати значення максимально наближене до 1 (наприклад «успіх») для класу за замовчанням і значення максимально наближене до 0 («програш») для іншого класу. Роль методу максимальної правдоподібності в тому, що процедура пошуку намагатиметься знайти значення для коефіцієнтів (бета значення), що мінімізують похибку в ймовірностях прогнозованих моделлю.

Для того, щоб зіставити прогнозуємі значення до ймовірностей використовується синусоїдна функція. Функція трансформує реальне значення в іншу змінну між 0 та 1 (Рис. 3.19).

$$S(z) = \frac{1}{1+e^{-z}}, \quad (3.1)$$

де S(x) – вихідний параметр зі значення від 0 до 1 (оцінка ймовірності);

z – вхідний параметр функції (передбачення);

e – основа натурального логарифму.

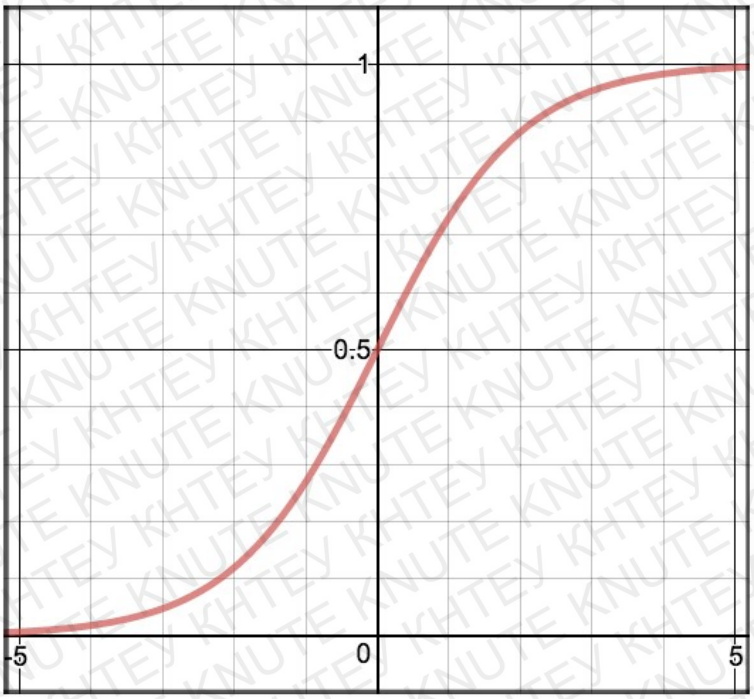

Рис. 3.19. Графік синусоїдної функції

Нинішня функція прогнозування провертає оцінку ймовірності між 0 та 1. Для того щоб зіставити це з дискретним класом («виграш»/»програш»), потрібно обрати граничне значення для точки переходу, значення понад якої класифікуються як 1, а нижче як 0.

$$\begin{cases} p \geq 0.5, class = 1 \\ p < 0.5, class = 0 \end{cases}, \quad (3.2)$$

де p – граничне значення.

Наприклад, якщо граничне значення дорівнює 0.5, а функція прогнозування повертає значення 0.7 – таке спостереження буде класифіковане як позитивне. Для логістичної регресії з декількома класами зазвичай обирається клас з найвищою

прогнозованою ймовірністю. Таке граничне значення називається «границею прийняття рішення» (Рис. 3.20).

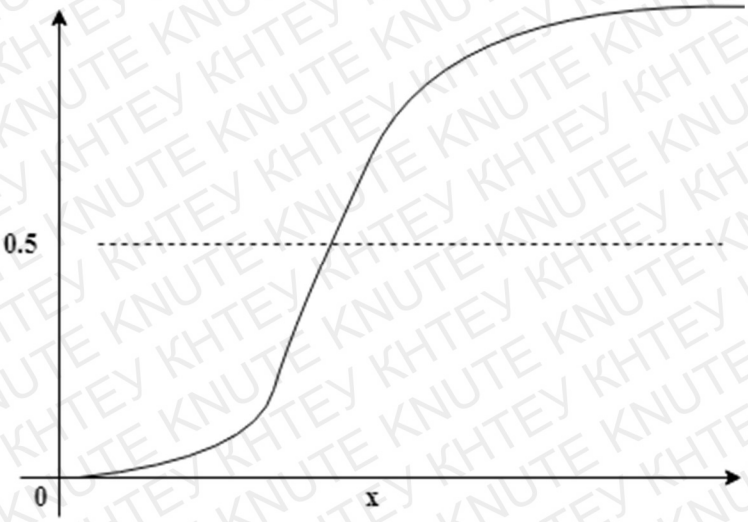

Рис. 3.20. Границя прийняття рішень

Використовуючи відомості про логістичну функцію та границю прийняття рішень, тепер можна написати функцію прогнозування. Функція прогнозування в логістичній регресії повертає значення ймовірності спостереження, що є позитивним. Таке спостереження класифікується як позитивне та набуває значення 1, може бути виражене формулою P(class = 1). Тому як ймовірність наближується до 1, модель визначає, що спостереження належить до класу 1.

Програмний пакет **GraphLab** містить модель під назвою Logistic Classifier. Ця модель має програмний інтерфейс, схожий на дерево прийняття рішень та використовує метод **create** для побудови моделі (Рис. 3.21). Для прогнозування потрібно використати вектор ймовірностей замість прогнозованого вектору класів (складається з +1 для позитивних результатів, та -1 для негативних). Для отримання більшої точності прогнозування обираємо граничне значення більше за 0.5.

```
model = gl.logistic_classifier.create(training, target='Outcome', features=l_features,
                                      validation_set=None, verbose=False)
predictions_prob = model.predict(testing, 'probability')
```

Рис. 3.21. Побудова моделі логістичної регресії

Змінна **predictions_prob** містить прогнозовані значення для перевірочного набору даних, що виражають ймовірність прибуткового результату торгового дня та знаходяться у діапазоні від 0 до 1. Для того щоб трансформувати наявні ймовірності у вектор класів «виграш»/«програш» використаємо наступний фрагмент коду з граничним значенням рівним 0.6 (Рис. 3.22).

```
testing['prob'] = predictions_prob
testing['predictions_logistic'] = testing.apply(lambda x: 1 if x['prob'] > 0.6 else -1)
```

Рис. 3.22. Трансформація масиву передбачень

Для оцінки точності моделі використовується **функція втрат**. Оптимальною функцією втрат для моделей логічної регресії вважається **перехресна ентропія**. Вона вимірює продуктивність моделі класифікації, в якій вихідний параметр виражає ймовірність в діапазоні від 0 до 1 (Рис. 3.23). Перехресна ентропія втрат зростає за умови, коли прогнозована ймовірність відхиляється від дійсної позначки. Тому наприклад прогнозування ймовірності 0.012 коли дійсне спостереження має позначку 1 призведе до поганого результату прогнозування та втрат. Еталонна модель має логарифм втрат рівний 0.

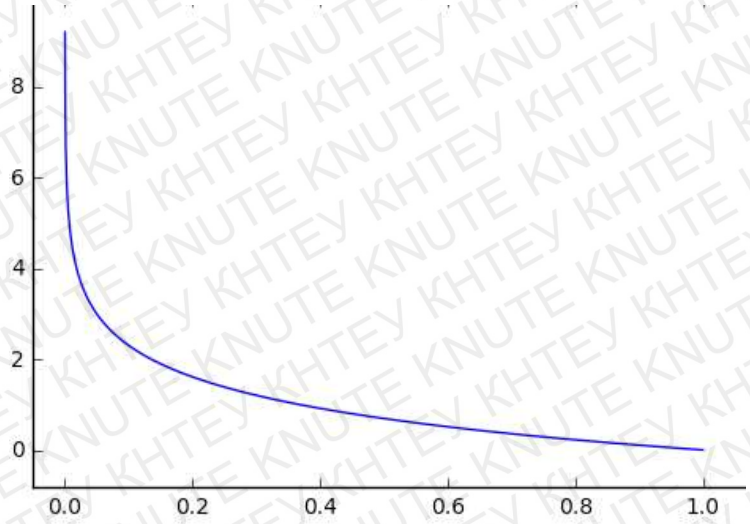

Рис. 3.23. Графік функції втрат

Перехресна ентропія втрат може бути поділена на дві окремі функції втрат. Одна для y = 1, а інша для y = 1 (Рис. 3.24).

Переваги використання логарифмів стають явними якщо графічно зобразити функції для y = 1 та y = 0. Це однорідні монотонні функції, що завжди зростають або спадають та полегшують обчислення градієнту і мінімізацію втрат.

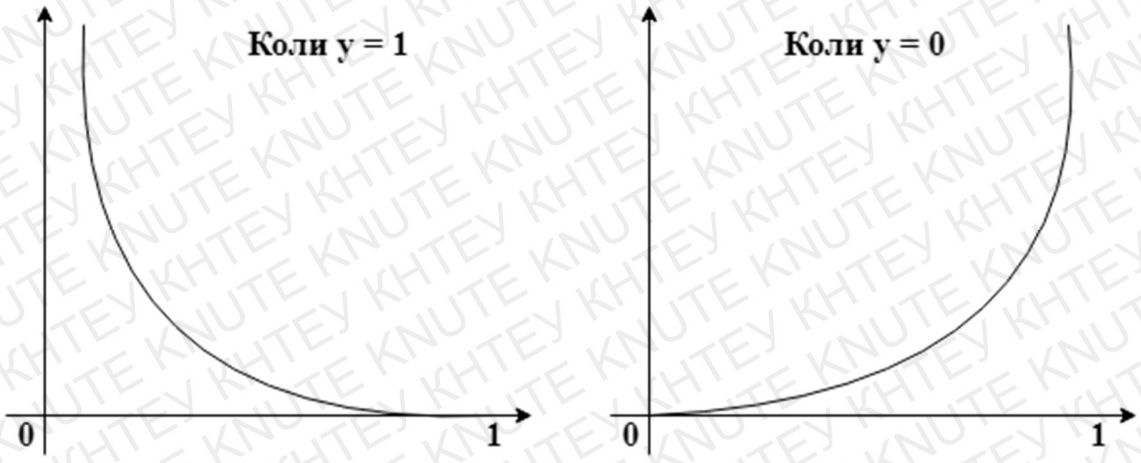

Рис. 3.24. Графік функції втрат перехресної ентропії

Варто зазначити, що функція втрат «штрафує» помилково-негативні передбачення більше, ніж «відзначає» коректні передбачення. В наслідок чого, підвищує точність прогнозу (ближче до 0 та 1) та зменшує втрати через логістичну природу функції.

В результаті перевірки моделі отримуємо точність близько 83% (Рис. 3.25).

```
model.evaluate(testing)['accuracy']
```

```
0.8318485523385301
```

Рис. 3.25. Оцінка точності моделі логістичної регресії

Лінійна регресія

Лінійна регресія – алгоритм машинного навчання з «вчителем», де прогнозуємі вихідні параметри є послідовними і що має постійний нахил. Ві використовується для прогнозування значень, що знаходяться в послідовному діапазоні (напр. продажі, ціна) замість того, щоб класифікувати їх як окремі категорії.

Проста лінійна регресія виражається формулою (3.3):

$$y = mx + b\ , \tag{3.3}$$

де **m** та **b** – змінні алгоритму, що намагається навчитися генерувати найбільш точні передбачення. **x** відображає вхідні дані, а **y** – передбачення.

Головна відмінність цієї моделі в тому, що вона оперує послідовними значеннями замість бінарних класів. Модель не потрібно тренувати використовуючи цільову змінну, що дорівнює +1 для «висхідних» днів та -1 для «низхідних», цільова змінна повинна бути послідовною.

Враховуючи те, що ми прагнемо передбачити позитивний прибуток, тобто ситуацію коли ціна «закриття» більше ніж ціна «відкриття», цільовою змінною повинене бути стовбець **Gain** з тренувального набору даних. Також, список властивостей повинен складатися з послідовних значень, таких як **Open** та **Close**.

Ця модель, на відміну від розглянутих раніше моделей класифікації, буде використовувати інший набір властивостей для навчання. Список параметрів для методу **create** бібліотеки **GraphLab** виглядає наступним чином (Рис. 3.26):

- **training –** тренувальний набір даних, що складається з колонок властивостей та цільової колонки.
- **target –** назва колонки, що містить цільову змінну.
- **validation_set –** набір даних для моніторингу узагальненої продуктивності моделі. В даному випадку він відсутній.
- **features –** список назв колонок властивостей, що використовуються для навчання моделі.
- **verbose -** визначає чи виводити детальну інформацію про процес тренування у консоль.
- **max_iterations –** максимальна кількість дозволених «проходів» по даним. Більша кількість може призвести до більш точно тренованої моделі.

```
l_lr_features = ['Open', 'Close']

model = gl.linear_regression.create(training, target='Gain',
                                    features = l_lr_features,
                                    validation_set=None, verbose=False,
                                    max_iterations=100)
predictions = model.predict(testing)
```

Рис. 3.26. Побудова моделі лінійної регресії

В якості стовбців властивостей були обрані ціна «відкриття» **Open** та ціна «закриття» **Close**.

Через те, що модель лінійної регресії прогнозує послідовні значення, потрібно зробити оцінку успішності ймовірностей та нормалізувати всі значення, щоб отримати вектор ймовірностей.

```
predictions_max, predictions_min = max(predictions), min(predictions)
predictions_prob = (predictions - predictions_min)/(predictions_max - predictions_min)
```

Рис. 3.27. Нормалізація передбачень моделі

Досі передбачення були у вигляді прогнозованих прибутків в масиві типу **SArray**, коли як нормалізовані передбачення містяться у змінній **predictions_prob** (Рис. 3.27). Для більшої точності прогнозування треба відфільтрувати передбачення з ймовірністю менше **0.6**, тобто для днів де прогнозоване значення буде менше – торгову сесію відкривати невигідно.

Підсилені дерева

Підсилення градієнту – це техніка машинного навчання для проблем регресії та класифікації, що створює модель прогнозування у формі збірки «слабких» моделей прогнозування, наприклад дерев прийняття рішень.

Для мінімізації функції втрат використовується *градієнтний спуск* і оновлення передбачень згідно з відсотком навчання. Градієнтний спуск – це ітераційний алгоритм оптимізації першого порядку, в якому для знаходження

локального мінімуму функції здійснюються кроки, пропорційні протилежному значенню градієнту функції в поточній точці.

Спуск починається з локального максимуму функції у напрямі негативного градієнту. З кожним новим кроком негативний градієнт перераховується використовуючи координати поточного місцезнаходження і відбувається наступний крок у заданому напрямку. Цей процес виконується ітеративно доки не буде досягнутий локальний мінімум градієнту (Рис. 3.28).

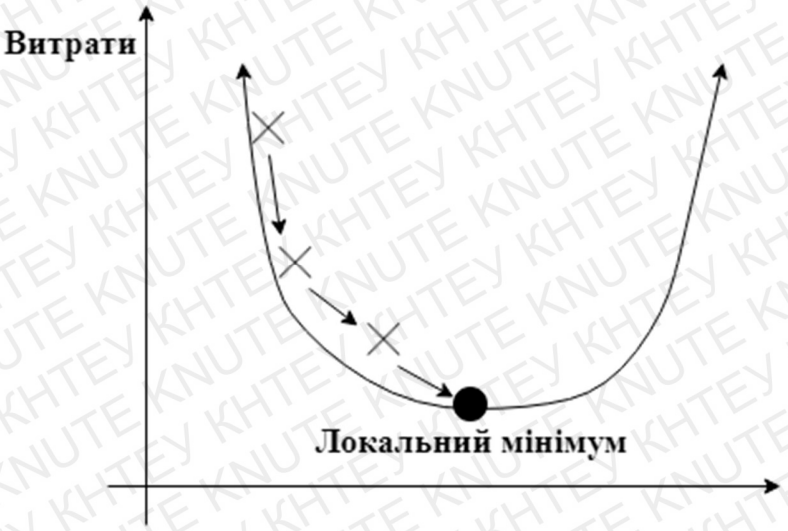

Рис. 3.28. Графік градієнтного спуску

Розмір таких кроків називається «відсоток навчання». З високим відсотком навчання, кожен крок має більшу пройдену відстань. Проте є ризик пропустити точку локального мінімуму тому як нахил спуску змінюється протягом всього часу. Через це спуск краще здійснювати малими кроками, що надає більшу точність. Недоліком цього є значно більший час навчання та потребуємі обчислювальні потужності. Градієнт може бути виражений формулою (3.4).

$$f(m,b) = \begin{bmatrix} \frac{df}{dm} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum -2x_i(y_i - (mx_i + b)) \\ \frac{1}{N}\sum -2(y_i - (mx_i + b)) \end{bmatrix}, \quad (3.4)$$

Для обчислення градієнту виконується ітерація точок даних використовуючи нові значення *m* та *b* і обчислені часткові похідні. Новий градієнт показує функцію витрат в поточній позиції (значення параметрів) і цільовий напрям руху для отримання нових значень параметрів. Нижче приведений фрагмент коду на програмній мові Python реалізує алгоритм (Рис. 3.29).

```
def update_weights(m, b, X, Y, learning_rate):
    m_deriv = 0
    b_deriv = 0
    N = len(X)
    for i in range(N):
        m_deriv += -2*X[i] * (Y[i] - (m*X[i] + b))

        b_deriv += -2*(Y[i] - (m*X[i] + b))

    m -= (m_deriv / float(N)) * learning_rate
    b -= (b_deriv / float(N)) * learning_rate

    return m, b
```

Рис. 3.29. Алгоритм обчислення градієнтного спуску

Збірка «слабких» моделей має на меті максимальне усунення різниці між реальними та прогнозованими значеннями, що виражається в **шумах**, **дисперсії** та **зміщенні**. Вона допомагає зменшити ці фактори.

Збірка має форму колекції «прогнозувальників», що збираються разом (середнє значення) для того, щоб створити кінцевий прогноз. Основна мотивація збірки – підвищення точності прогнозування коли багато різних «прогнозувальників» намагаються передбачити одну й ту саму цільову змінну.

Техніки збірки поділяються на наступні типи:

- **Пакування** – прийом за якого незалежні «прогнозувальники» будуються, а потім усереднюють передбачення за допомогою схрещення технік (середнє зважене, більшість голосів і т.д.). Зазвичай для кожного прогнозувальника беруться випадкові частки набору даних, для того, щоб вхідний набір був

різним для кожного випадку (Рис. 3.30). Результатом застосування техніки є те, що береться множина некорельованих прогнозувальників для створення фінальної моделі, що в свою чергу знижує похибку через зниження дисперсії.

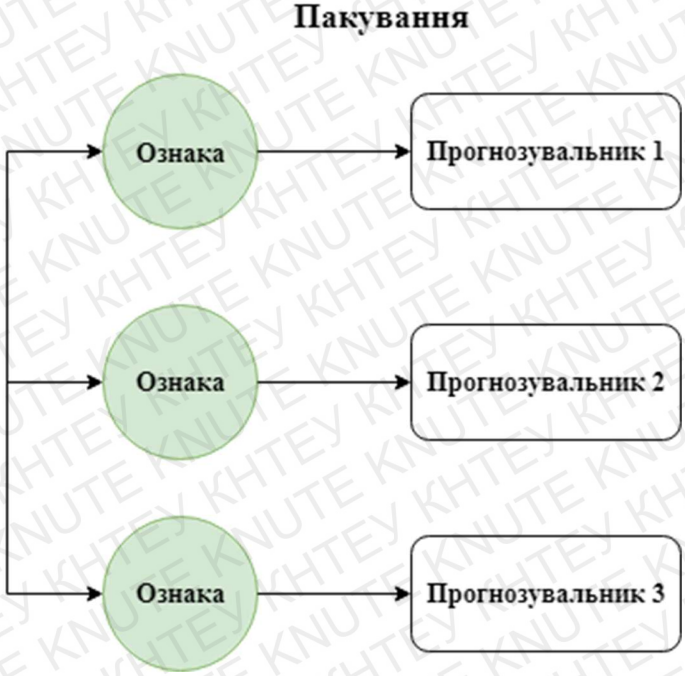

Рис. 3.30. Принцип роботи пакування моделей

- **Підсилення** – прийом, за якого прогнози робляться незалежно, але послідовно. Кожне наступне прогнозування враховує помилки попереднього (Рис. 3.31). Через це спостереження розподіляються між моделями нерівномірно.

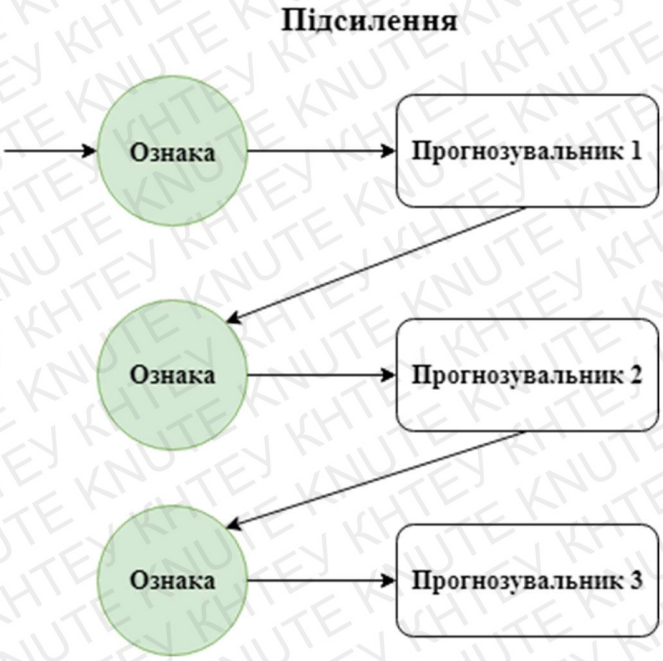

Рис. 3.31. Принцип роботи підсилення моделей

Якщо раніше для тренування застосовувалися звичайні дерева прийняття рішень, то в даному випадку використовується алгоритм підсиленого класифікатора дерева рішень, що реалізується у програмній бібліотеці **GraphLab**. На вхід приймаються такі ж самі параметри, що й для інших моделей класифікації. Окрім цього, задається параметр **max_iterations**, що визначає максимальну кількість ітерацій для підсилення градієнту алгоритму. Нижченаведений фрагмент коду демонструє процес створення моделі та прогнозування результатів, використовуючи перевірочний набір даних (Рис. 3.32).

```
boosted_model = gl.boosted_trees_classifier.create(training, target='Outcome', features=l_features,
                                                   validation_set=None, max_iterations=12, verbose=False)
predictions_prob = boosted_model.predict(testing, 'probability')
```

Рис. 3.32. Створення моделі підсиленого дерева

Точність даної моделі становить приблизно 83%, що на 1% відсоток більше ніж звичайне дерево прийняття рішень (Рис. 3.33).

```
boosted_model.evaluate(testing)['accuracy']
```

```
0.8296213808463252
```

Рис. 3.33. Оцінка точності моделі підсиленого дерева

Випадковий ліс

Випадковий ліс – ансамблевий метод машинного навчання для класифікації, регресії та інших завдань, які оперують за допомогою побудови численних дерев прийняття рішень під час тренування моделі і продуктують моду для класів або усереднений прогноз (регресія) побудованих дерев. Алгоритм відноситься до сімейства алгоритмів машинного навчання з вчителем. Ліс являє собою збірку декількох дерев прийняття рішень та в більшості випадків використовує метод *пакування*.

Алгоритм забезпечує додаткову випадковість моделі, додаючи нові дерева. Замість пошуку найбільш вагомої властивості набору даних розбиваючи вузол, він шукає найкращу властивість серед випадкової підмножини властивостей. Це призводить до більшої різноманітності і кращих результатів моделі загалом (Рис. 3.34).

Процес побудови випадкового лісу відбувається згідно наступних кроків:

1. Випадково обрати **K** властивостей серед загальних **m** властивостей де **K < m**.
2. Серед властивостей K обчислити вузол d, використовуючи найкращу точку поділу.
3. Розділити вузол на дочірні вузли за допомого найкращого поділу.
4. Повторити кроки 1-3 доки не буде досягнута *i* кількість вузлів.
5. Побудувати ліс повторюючи перші чотири кроки *n* разів.

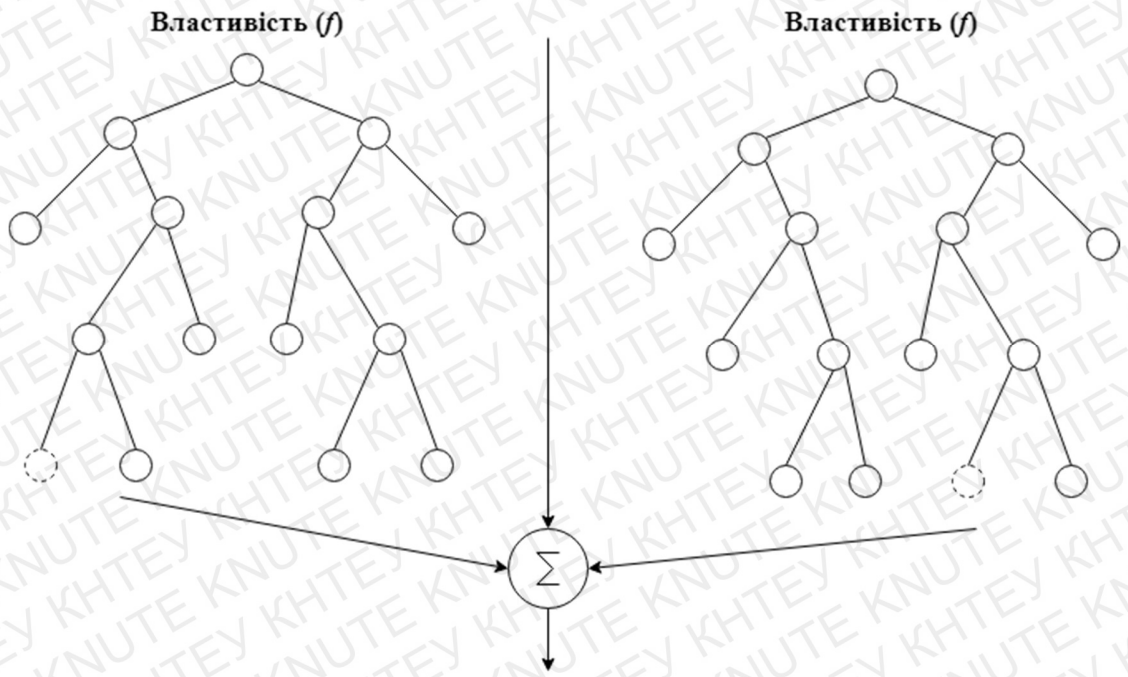

Рис. 3.34. Принцип роботи моделі випадкового лісу

Бібліотека **GraphLab** має вбудовану реалізацію класифікатору випадкового лісу, що збирає декілька дерев. Максимальна кількість дерев, що використовується в моделі визначається параметром **num_trees**. Це дозволяю уникнути зайвої складності та феномену «перенавчання» нейронної мережі.

Нижченаведений фрагмент коду виконує побудову та перевірку моделі випадкового лісу (Рис. 3.35).

```
random_f_model = gl.random_forest_classifier.create(training, target='Outcome', features=l_features,
                                    validation_set=None, verbose=False, max_iterations = 10)

predictions_prob = random_f_model.predict(testing, 'probability')
```

Рис. 3.35. Побудова моделі випадкового лісу.

Точність моделі становить більше 84%, що є найліпшим показником серед проаналізованих алгоритмів.

3.3 Аналіз ефективності застосування створеної моделі

Після створення та тренування нейронних моделей необхідно визначитися зі стратегією їх використання та проаналізувати ефективність і можливі ризики.

Для оцінки прибутковості найбільш часто використовуються наступні показники:

- Доходи та витрати (**pnl**)
- Коефіцієнт Шарпа

Доходи та витрати дозволяють оцінити прибуток кожного окремо взятого торгового дня. Цей показник обчислюються шляхом віднімання стартової ціни на цінні папери від кінцевої.

Коефіцієнт Шарпа був винайдений лауреатом Нобелевської премії Вільямом Шарпом і використовується для того, щоб допомогти інвесторам розуміти повернення інвестицій відносно до кількості ризику. Коефіцієнт – це середній дохід зароблений в додачу до безризикового відсотку відносно до загального ризику. Він вираховується наступною формулою (3.5):

$$Sharpe\ ratio = \sqrt{252} * \frac{mean(pnl)}{sd(pnl)}, \quad (3.5)$$

де mean(pnl) – середнє значення списку доходів та витрат за торговий період;
sd(pnl) – середнє квадратичне відхилення.

Згідно до вищезазначеної стратегії торгів, задачею нейронної мережі є купівля та подальший продаж CFD з ціллю мінімізації ризику.

Наприклад покупка одного CFD індексу S&P 500 зі стартовою ціною, що дорівнює 2000 та його продаж з кінцевою ціною 2020. В даному випадку прибуток складає 20 балів. Припустимо, що кожний бал має значення 25 доларів США:

Валовий прибуток = 20 * 25 = 500 доларів.

Зазвичай біржовий брокер утримує комісію рівну 0.6 балів. Тому чистий прибуток рахується наступним чином:

Чистий прибуток = (20 – 0.6) * 25 = 485 доларів.

Іншим важливим аспектом використання моделі є додаткові заходи для мінімізації втрат під час торгів. Такі випадки можуть трапитися коли прогнозований результат дорівнює +1, а реальне значення -1, тобто помилково позитивні передбачення. В цьому випадку кінцева торгова сесія є «низхідним» днем де ціна закриття нижче ніж ціна відкриття і модель приносить негативний результат з грошовими втратами.

Запобіжник втрат – сигнал моделі, що запобігає максимальним втратам які можуть відбутися під час торгової сесії. Цей захід спрацьовує у випадках, коли ціна активу стає нижче фіксованого значення, що було встановлене заздалегідь. Тренувальний набір даних має стовбець **Low**, що дорівнюються найнижчий ціні активу протягом торгового дня. Таким чином якщо запобіжник втрат буде мати значення -3, а різниця між стартовою та найнижчою ціною позиції становитиме -5, то позиція буде автоматично закрита і грошові втрати мінімізовані. Наступний фрагмент коду демонструє допоміжну функцію, щоб імітувати торгову сесію з запобіжником. Функція **trade_with_stop** приймає параметри **bar**, **slippage** та **stop**. **bar** містить дані торгів за один торговий день. **slippage** – процентна комісія біржового брокера. **stop –** значення запобіжника втрат. Функція повертає фінальний показник прибутку (Рис. 3.36).

```
def trade_with_stop(bar, slippage = 0, stop=None):

    bar['Gain'] = bar['Gain'] - slippage
    if stop<>None:
        real_stop = stop - slippage
        if bar['Lo']<=stop:
            return real_stop

    return bar['Gain']
```

Рис. 3.36. Функція запобіжника втрат

Транзакційні витрати накладаються при купівлі та продажу цінних паперів. Вони включають брокерські комісії та спреди (різниця між ціною, що дилер заплатив за акцію і ціною, що заплатив покупець). Такі витрати повинні враховуватися в процесі оцінки ефективності моделі. В цьому прикладі розглянуті два види таки витрат: ковзна комісія брокера та фіксована комісія купівлі/продажу акції.

Для візуалізації ефективності моделі було використано допоміжну функцію, що будує графік кумулятивного прибутку згідно з передбаченнями моделі, використовуючи перевірочний набір даних (Рис. 3.37).

```
import numpy as np

def plot_equity_chart(pnl):
    equities = []
    temp = 0
    for x in pnl:
        equities.append(x if len(equities) == 0 else x + equities[-1])

    plt.plot(equities)
    print 'Mean of PnL:' + "{:10.4f}".format(pnl.mean())
    print 'Round turns: %5d' % len(pnl)
    print 'Sharpe:' + "{:10.4f}".format(np.sqrt(252) *
                                        (pnl.mean() / np.std(np.array(pnl), axis=0)))
```

Рис. 3.37. Функція візуалізації ефективності моделі

Окрім безпосередньої побудови графіку функція обчислює наступні показники ефективності:

- Доходи і витрати
- Коефіцієнт Шарпа
- Кількість циклів торгів

Використовуючи вищенаведену функцію та «запобіжник втрат» можна оцінити ефективність навчених моделей, прогнозований прибуток та ризиковість їх використання.

1. Дерева прийняття рішень (Рис. 3.38).

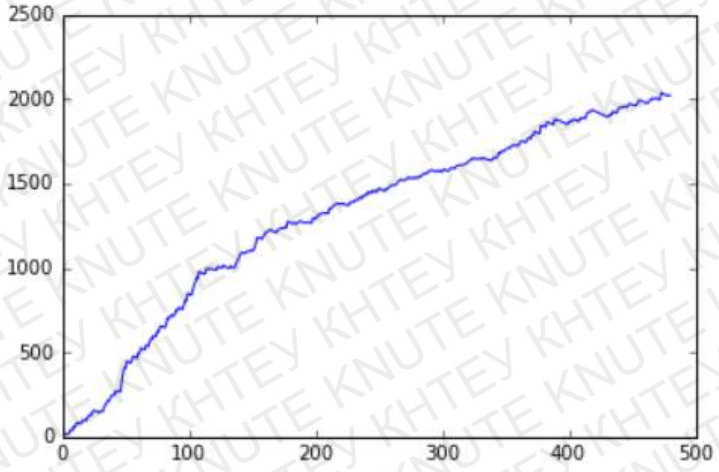

Рис. 3.38. Графік чистого прибутку дерева прийняття рішень

Загальна кількість передбачень: 898

Середній показник доходу за один цикл торгів: 8.9514

Кількість циклів торгів: 481

Коефіцієнт Шарпа: 11.5922

Чистий прибуток з урахуванням комісій: 46476 доларів

2. Логістична регресія (Рис. 3.39).

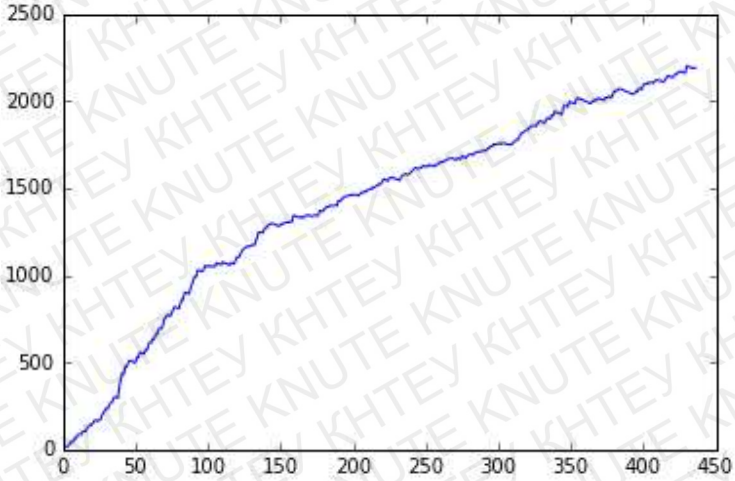

Рис. 3.39. Графік чистого прибутку логістичної регресії

Загальна кількість передбачень: 898

Середній показник доходу за один цикл торгів: 10.3136

Кількість циклів торгів: 437

Коефіцієнт Шарпа: 13.9456

Чистий прибуток з урахуванням комісій: 50337 доларів

3. Лінійна регресія (Рис. 3.40).

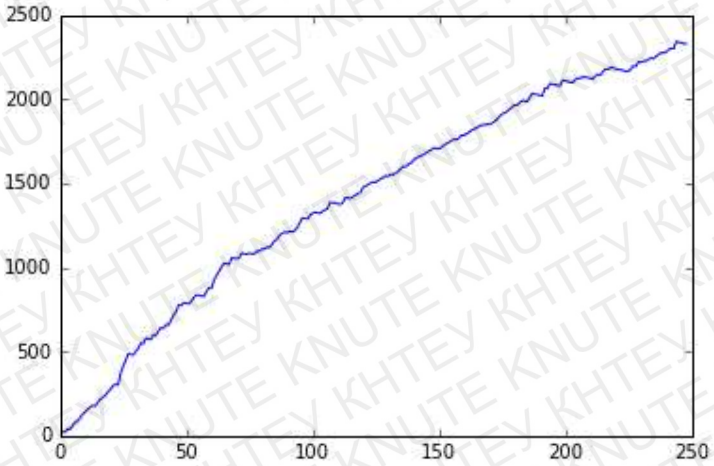

Рис. 3.40. Графік чистого прибутку лінійної регресії

Загальна кількість передбачень: 898

Середній показник доходу за один цикл торгів: 17.3989

Кількість циклів торгів: 249

Коефіцієнт Шарпа: 25.7398

Чистий прибуток з урахуванням комісій: 53521 доларів

Через те, що модель використовує передбачення з ймовірністю більше 60% вона має значно більший показник прибутку на один цикл торгів. Це дозволяє досягти кращих результатів при меншій кількості торгових днів порівняно з попередніми моделями.

4. Підсилене дерево (Рис. 3.41).

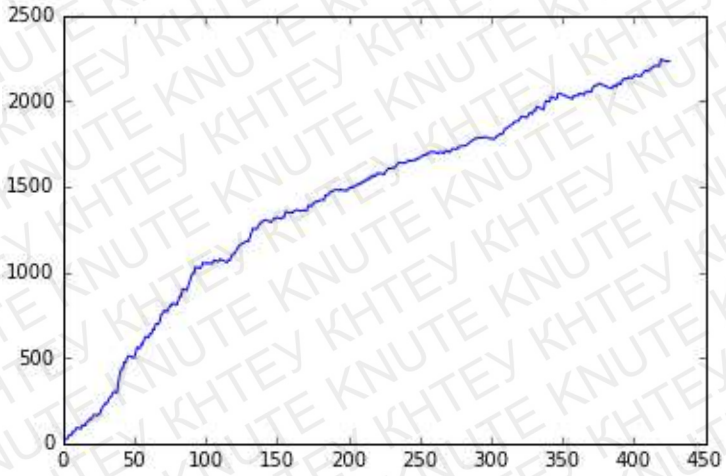

Рис. 3.41. Графік чистого прибутку підсиленого дерева

Загальна кількість передбачень: 898

Середній показник доходу за один цикл торгів: 10.3676

Кількість циклів торгів: 426

Коефіцієнт Шарпа: 13.9563

Чистий прибуток з урахуванням комісій: 51296 доларів

5. Випадковий ліс (Рис. 3.42).

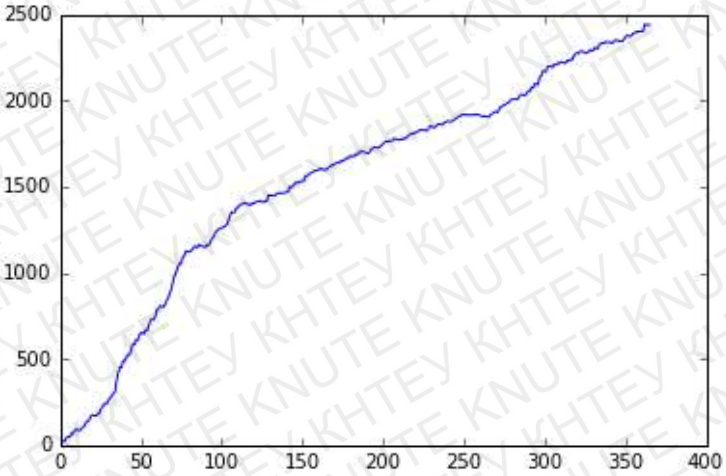

Рис. 3.42. Графік чистого прибутку випадкового лісу

Загальна кількість передбачень: 898

Середній показник доходу за один цикл торгів: 10.7622

Кількість циклів торгів: 366

Коефіцієнт Шарпа: 16.0067

Чистий прибуток з урахуванням комісій: 56128 доларів

Разом із найвищою точністю серед розглянутих моделей, випадковий ліс забезпечує ще й найвищу прибутковість.

Висновки до розділу 3

В даному розділі алгоритми машинного навчання були застосовані на практиці для вирішення задач фінансового менеджменту, а саме прогнозування фондового індексу S&P 500.

Одним з найважливішим критеріїв успішного застосування прийомів машинного навчання є ретельна підготовка наборів для тренування моделей і перевірки їх точності.

В даному розділі алгоритми машинного навчання були застосовані на практиці для вирішення задач фінансового менеджменту, а саме прогнозування фондового індексу S&P 500.

Одним з найважливішим критеріїв успішного застосування прийомів машинного навчання є ретельна підготовка наборів для тренування моделей і перевірки їх точності.

Властивості спостережень набору даних зазвичай обираються експериментальним шляхом та здебільшого залежать від розміру набору, ступеня кореляції значень та типу алгоритму машинного навчання.

Було розглянуто 5 різних алгоритмів: дерева прийняття рішень, логістична регресія, лінійна регресія, підсилені дерева та випадковий ліс (табл. 3.5).

Таблиця 3.5

Звідна таблиця ефективності моделей

Модель	Кількість передбачень	Дохід/цикл	Кількість циклів	Коеф Шарпа	Чистий прибуток
Дерева рішень	898	8.9514	481	11.5922	46476 $
Логістична регресія	898	10.3136	437	13.9456	50337 $
Лінійна регресія	898	17.3989	249	25.7398	53521 $
Підсилені дерева	898	10.3676	426	13.9563	51296 $
Випадковий ліс	898	10.7622	366	16.0067	56128 $

Властивості спостережень набору даних зазвичай обираються експериментальним шляхом та здебільшого залежать від розміру набору, ступеня кореляції значень та типу алгоритму машинного навчання.

Було розглянуто 5 різних алгоритмів: дерева прийняття рішень, логістична регресія, лінійна регресія, підсилені дерева та випадковий ліс (табл. 3.5).

Найліпші показники продемонстрували алгоритми лінійної регресії та випадкового лісу. Причиною цього є високий ступінь кореляції між використаними властивостями та прогнозованим показником, а також хронологічний характер вхідних даних, що сприяє побудові рядів послідовних значень властивостей.

Всі розглянуті моделі мають точність вище 80 %, що є значно кращим показником порівняно з методом «сліпого» вибору та традиційними статичними методами. Отже, це доводить ефективність та доцільність використання нейронних мереж для подібного роду задач.

ВИСНОВКИ

Пошук шляхів застосування новітніх алгоритмів машинного навчання призводять до появи все більшої кількості наукових досліджень, що мають на меті підвисити ефективність діяльності на фінансових ринках та надати конкурентні переваги підприємствам, що впроваджують новітні підходи з використанням нейронних мереж.

Застосування алгоритмів машинного навчання надає переваги, як для самих компаній, що отримують доступ до нової інформації і здатність її ефективно аналізувати, так і для клієнтів, що отримують кращий, більш персоніфікований сервіс.

Процес побудови моделей машинного навчання має здебільшого експериментальний характер і потребує певної підготовки та аналізу тренувального набору даних перед безпосереднім застосуванням алгоритму машинного навчання.

Існує декілька класів програмних продуктів, що мають на меті полегшення розробки і автоматизацію рутинних задач машинного навчання. При застосуванні моделей у реальному середовищі часто виникає низка складностей пов’язаних з продуктивністю програми та її швидкодією. Використання фреймворків для розподіленої обробки інформації та технологій «великих даних» здебільшого вирішують ці проблеми.

В проведеному дослідженні запропоновано методику торгів зваженим індексом фондової біржі S&P 500 та розроблено декілька моделей, що виконують поставлену задачу з точністю більше 80%.

Розглянуті основні алгоритми регресії та класифікації, що використовують методи статистичного аналізу для побудови моделей, а також техніки «збірки» та градієнтного спуску, що підвищують точність прогнозування.

Застосування моделей машинного навчання у галузі фінансового менеджменту набуває все більшого значення та стає надійним інструментом для керування ризиками та прийняття рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Aggarwal C.C., Charu C. Data Classification Algorithms and Applications. 2015: Chapman & Hall /CRC.
2. Aggarwal C.C., Charu C. Outlier Analysis. Springer, 2013.
3. Aggarwal C.C., Hinneburg A., Keim A. On the Surprising Behavior of Distance Metrics in High Dimensional Space // ICDT Conference. 2001.
4. Aggarwal C.C. On the Eects of Dimensionality Reduction on High Dimensional Similarity Search // ACM PODS Conference. 2001.
5. Agyemang M., Barker K., Alhajj R. A Comprehensive Survey of Numeric and Symbolic Outlier Mining Techniques // Intelligent Data Analysis, No. 10(6), 2006. pp. 521–538.
6. Barber D. Bayesian Reasoning and Machine Learning: Kindle Edition, 2012.
7. Barnett V., Lewis T. Outliers in Statistical Data. Wiley, 1994.
8. Beckman R., Cook R. Outliers // Technometrics, No. 25(2), 1983. pp. 119– 149.
9. Box G. and Jenkins G. Time series analysis: Forecasting and control. // SanFrancisco, USA: Holden-Day, Inc., 1970.
10. Chandola V., Banerjee A., Kumar V. Anomaly Detection for Discrete Sequences: A Survey // IEEE Transactions on Knowledge and Data Engineering, No. 24(5), 2012. pp. 823–839.
11. Chandola V., Banerjee A., Kumar V. Anomaly Detection: A Survey // ACM Computing Surveys, No. 41(3), 2009.
12. Downey A. Probability and Statistics for Programmers: Kindle Edition, 2011.
13. Dunnung T., Friedman E. Practical Machine Learning: A New Look at Anomaly Detection. O'Reilly Media, 2004
14. Elkan C., Noto K. Learning Classifiers from only Positive and Unlabeled Data // ACM KDD Conference. 2008.

15. Engle R. F. Autoregressive conditional heteroscedasticity with estimates of variance. Econometrica, 1978.
16. Geron A. Hands-On Machine Learning with Scikit-Learn and TensorFlow // O'Reilly Media, 2017.
17. Haider S., Abbas A., Zaidi A.K. A multi-technique approach for user identification through keystroke dynamics. // IEEE International Conference on Systems, Man and Cybenetics. 2000.
18. Hawkins D. Identification of Outliers. Chapman and Hall, 1980.
19. Koolen W., Rooij S. Switching investments. Theoretical ComputerScience, 2013.
20. Koolen W., Vovk V. Buy low, sell high. In Algorithmic Learning Theory, 2012.
21. Nobuchika M. "Will FinTech create shared values?" // Columbia Business School conference, 2017.
22. Ryza S., Laserson U., Wills J. Advanced Analytics with Spark: Kindle Edition, 2015.
23. Schoutens W. Processes in Finance: Pricing Financial Derivatives. // Wiley, 2003.
24. Shafer G., Vovk V. Probability and Finance: It's Only a Game! // John Wiley& Sons, 2001.