ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Методика оптимізації процесу аналізу даних продажів за допомогою машинного навчання»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

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

_____ _____
(підпис) Ім'я, ПРІЗВИЩЕ здобувача

Виконав: Владислав КАРПЕНКО
здобувач вищої освіти
група ІСДМ-64

Керівник: Ольга ПОЛОНЕВИЧ
науковий ступінь, вчене звання к.т.н., доцент

Рецензент:
науковий ступінь, вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти Магістр
Спеціальність Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ
Каміла СТОРЧАК
«____» _____20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ СТУДЕНТУ
Карпенку Владиславу Олеговичу
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: «Методика оптимізації процесу аналізу даних продажів за допомогою машинного навчання»

керівник кваліфікаційної роботи Ольга ПОЛОНЕВИЧ, к.т.н., доцент
(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024 року № 320.

2. Строк подання кваліфікаційної роботи: 27 грудня 2024 року.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, технічна документація бібліотек Python для машинного навчання, стандарти оформлення коду Python, технічна документація модуля візуалізації Matplotlib.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Дослідження ефективності популярних алгоритмів аналізу даних продажів та вибір методології для подальшої розробки
2. Розробка оптимальної методики аналізу даних продажів
3. Генерування віртуальної бази даних продажів та тестування створеної програми для визначення ефективності методики

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання: 15 жовтня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	15.10 – 23.10.24	
2	Порівняння підходів у аналітиці продажів за допомогою різних алгоритмів машинного навчання	24.10 – 01.11.24	
3	Вивчення функціоналу бібліотек, необхідних для розробки програмного забезпечення	02.11 – 07.11.24	
4	Розробка програми аналізу даних продажів	08.11 – 16.11.24	
5	Генерація тестового датасету для тестування	17.11 – 21.11.24	
6	Тестування та визначення ефективності розробленої методики за конкретними метриками	22.11 – 11.12.24	
7	Оформлення роботи: вступ, висновки, реферат	12.12 – 20.12.24	
8	Розробка демонстраційних матеріалів	21.12 – 25.12.24	

Здобувач вищої освіти _____ Владислав КАРПЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник роботи
кваліфікаційної роботи _____ Ольга ПОЛОНЕВИЧ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 86 стор., 76 рис., 2 табл., 31 джерело.

Об'єкт дослідження - процес аналізу даних продажів із використанням алгоритмів машинного навчання.

Предмет дослідження - методи та алгоритми машинного навчання застосовані для аналізу даних продажів.

Мета роботи - розробка методики аналізу даних продажів із використанням машинного навчання для підвищення точності прогнозів та оптимізації процесів.

Методи дослідження: порівняння алгоритмів машинного навчання, розробка програмного забезпечення з використанням Python-бібліотек, тестування створеної програми на синтетичному датасеті, верифікація моделі.

Короткий зміст роботи: проведено огляд літератури та аналіз сучасних методів машинного навчання, таких як Random Forest, K-means та LSTM, розроблено методику підготовки даних, що включає очищення, нормалізацію та усунення викидів. Для сегментації клієнтів і кластеризації продажів застосовано алгоритм K-means, а також розроблено та протестовано програму аналізу даних продажів на синтетичному наборі даних. Ефективність запропонованої методики верифіковано шляхом порівняння отриманих результатів із традиційними підходами.

Результати роботи демонструють можливість застосування розробленої методики для прогнозування попиту, аналізу поведінки клієнтів і визначення ефективності маркетингових кампаній.

Ключові слова: МАШИННЕ НАВЧАННЯ, АНАЛІЗ ДАНИХ, ПРОДАЖІ, RANDOM FOREST, K-MEANS, LSTM, ОПТИМІЗАЦІЯ, ПРОГНОЗУВАННЯ, КЛАСТЕРИЗАЦІЯ, PYTHON.

ABSTRACT

The text part of the master's qualification work consists of 86 pages, 76 figures, 2 tables, and 31 references.

The study focuses on optimizing the sales data analysis process using machine learning.

Object of research - the process of sales data analysis using machine learning algorithms.

Subject of research - methods and algorithms of machine learning applied to sales data analysis.

Purpose of the work - to develop a methodology for sales data analysis using machine learning to improve forecast accuracy and optimize business processes.

Research methods: comparison of machine learning algorithms, development of software using Python libraries, testing the developed program on a synthetic dataset, and model verification.

Summary of the work: A review of the literature and an analysis of modern machine learning methods, such as Random Forest, K-means, and LSTM, were conducted, and a data preparation methodology was developed, including data cleaning, normalization, and outlier removal. The K-means algorithm was applied for customer segmentation and sales clustering, and a sales data analysis program was developed and tested on a synthetic dataset. The effectiveness of the proposed methodology was verified by comparing the obtained results with traditional approaches.

The results demonstrate the applicability of the developed methodology for demand forecasting, customer behavior analysis, and evaluating the effectiveness of marketing campaigns.

Keywords: MACHINE LEARNING, DATA ANALYSIS, SALES, RANDOM FOREST, K-MEANS, LSTM, OPTIMIZATION, FORECASTING, CLUSTERING, PYTHON.

ЗМІСТ

ВСТУП

Актуальність теми дослідження пов’язана з бурхливим розвитком технологій машинного навчання та їх широким впровадженням у різні сфери економіки. В умовах сучасної цифрової економіки обробка великих обсягів даних стала однією з головних задач для підприємств. Зростання обсягів інформації, яка генерується під час комерційної діяльності, спричинило необхідність у застосуванні новітніх методів аналізу даних, серед яких важливу роль відіграють алгоритми машинного навчання. Вони здатні автоматизувати багато бізнес-процесів, підвищити точність прогнозів, оптимізувати роботу підприємств та сприяти кращому розумінню поведінки клієнтів. Механізми машинного навчання дозволяють забезпечити ефективне прогнозування попиту, сегментацію клієнтів, аналіз впливу сезонності на продажі, а також надання персоналізованих рекомендацій для підвищення прибутковості.

Зокрема, в умовах високої конкуренції на ринку важливою є швидкість та точність прийняття рішень. Для компаній, які працюють на основі великих обсягів даних, здатність аналізувати та обробляти ці дані за допомогою машинного навчання може стати вирішальним фактором для підвищення ефективності та конкурентоспроможності. Це особливо актуально для сфери продажів, де прогнози попиту, адаптивність ціноутворення, виявлення змін у споживчих уподобаннях і персоналізовані пропозиції стають ключовими аспектами для досягнення успіху.

У цьому контексті розробка методик оптимізації процесу аналізу даних продажів з використанням машинного навчання стає важливою задачею, що вимагає комплексного підходу до вибору алгоритмів, обробки даних та оцінки ефективності моделей. Технології машинного навчання можуть значно спростити обробку інформації, скоротити час, що витрачається на аналіз, а також підвищити точність та достовірність отриманих результатів, що важливо для прийняття бізнес-рішень.

Метою дослідження є розробка методики аналізу даних продажів з використанням машинного навчання для підвищення ефективності прогнозування

та прийняття рішень в умовах невизначеності та високої конкуренції. Для досягнення цієї мети необхідно вирішити ряд завдань: провести огляд сучасних методів машинного навчання, які застосовуються для аналізу даних продажів, визначити найбільш підходящі алгоритми для цієї задачі, розробити методику підготовки даних та оцінки точності прогнозних моделей, а також застосувати розроблену методику на реальних даних для перевірки її ефективності.

Об'єктом дослідження є процес аналізу даних продажів за допомогою машинного навчання, а предметом — методи та алгоритми машинного навчання, які застосовуються для оптимізації цього процесу. У рамках роботи буде вивчено вплив різних факторів на ефективність моделей, таких як якість даних, вибір алгоритмів, етапи підготовки даних і методи оцінки точності.

Методи дослідження включають аналіз існуючих наукових праць і розробок в області машинного навчання, порівняння різних алгоритмів, а також експериментальну перевірку результатів на реальних даних. Основними методами для розв'язання задач будуть: аналіз літератури, порівняння алгоритмів, статистичний та експериментальний аналіз даних.

Наукова новизна роботи полягає в адаптації існуючих алгоритмів машинного навчання для оптимізації процесів аналізу продажів, а також у розробці методики, яка дозволяє підвищити точність прогнозування, знизити витрати на обробку даних і забезпечити більш ефективне управління бізнес-процесами. Пропонована методика дозволяє застосувати машинне навчання для різних аспектів аналізу продажів, включаючи прогнозування попиту, визначення ефективності маркетингових кампаній, сегментацію клієнтів і виявлення шахрайства.

1 ТЕОРЕТИЧНІ ОСНОВИ ТА ОГЛЯД МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ АНАЛІЗУ ДАНИХ ПРОДАЖІВ

1.1 Загальні поняття про машинне навчання та його роль в аналізі даних

Штучний інтелект є одним із ключових аспектів розвитку сучасних технологій, відіграючи центральну роль у трансформації економіки, науки та суспільства. Його застосування охоплює широкий спектр сфер — від автоматизації бізнес-процесів і медицини до аналізу великих даних і прогнозування поведінки споживачів. Завдяки здатності обробляти та аналізувати величезні обсяги інформації з високою точністю, технології штучного інтелекту сприяють підвищенню ефективності прийняття рішень, знижують витрати та відкривають нові можливості для інновацій. У контексті глобальної цифровізації штучний інтелект стає необхідним інструментом для конкурентоспроможності компаній і прогресу людства.

Прогноз динаміки розвитку ринку машинного навчання у період з 2023 по 2034 рік, опублікований Precedence Research, демонструє стрімке зростання, що підтверджує зростаючу роль цієї технології у сучасній економіці (рис. 1.1). Згідно з поданими даними, у 2023 році обсяг ринку оцінювався в 51,48 мільярда доларів США, але до 2034 року ця сума має зрости до вражаючих 1,407,65 мільярда доларів. Такий показник відображає середньорічний приріст, що значно перевищує традиційні темпи зростання у багатьох інших галузях.

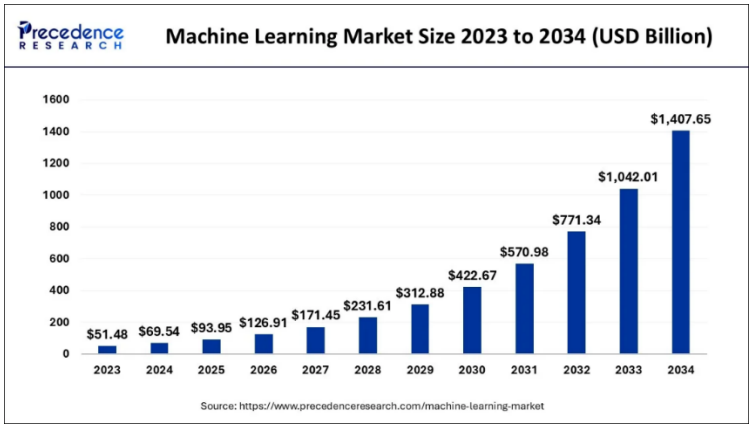


Рис.1.1. Розмір ринку машинного навчання та прогноз на 2024–2034 роки [2]

Основними рушійними силами такого зростання є широке застосування машинного навчання у різних сферах. Серед ключових секторів, які активно інтегрують ML, виділяються фінанси, охорона здоров'я, виробництво, транспорт і електронна комерція. Використання машинного навчання забезпечує підвищення ефективності роботи через автоматизацію процесів, покращення аналізу даних та формування точних прогнозів. Наприклад, у сфері фінансів ML використовується для виявлення шахрайства та алгоритмічної торгівлі, тоді як у медицині — для діагностики захворювань і персоналізованого лікування.

Зростання обсягу ринку також свідчить про збільшення доступності технологій машинного навчання, включаючи хмарні платформи та інструменти для розробки моделей. Крім того, розвиток ML стимулюється попитом на рішення, що здатні ефективно працювати з великими обсягами даних (Big Data) і забезпечувати конкурентні переваги на ринку.

Ці дані вказують на те, що машинне навчання стає важливим елементом стратегії розвитку для багатьох компаній, які прагнуть залишатися конкурентоспроможними у глобальній економіці. У майбутньому очікується подальше розширення сфер його застосування, що сприятиме ще більшому впливу ML на науково-технічний прогрес і суспільний розвиток. [2]

Машинне навчання (machine learning) є ключовим напрямом розвитку сучасного штучного інтелекту, який спрямований на надання комп'ютерам здатності самостійно навчатися на основі даних. Цей підхід дозволяє вирішувати широкий спектр задач, зокрема автоматизацію складних процесів, аналіз великих обсягів інформації та формування прогнозів. Основною характеристикою машинного навчання є його здатність адаптивно вдосконалювати моделі, підвищуючи точність і ефективність аналізу в міру зростання обсягів доступних даних. На відміну від традиційних статистичних методів, які базуються на фіксованих формулах, машинне навчання використовує алгоритми, що вчаться на основі закономірностей у даних. Це забезпечує гнучкість підходу та його адаптацію до специфіки різних галузей і завдань.

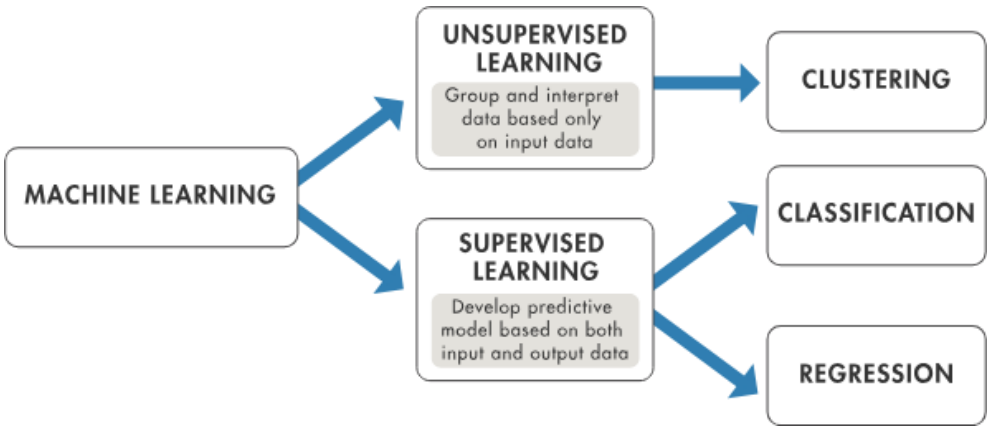


Рис.1.2. Методи машинного навчання [1]

Існують дві основні категорії машинного навчання: контрольоване (supervised learning) та неконтрольоване (unsupervised learning), вони схематично зображені на рис. 1.2. Контрольоване навчання базується на роботі з даними, де результати вже відомі. Наприклад, цей підхід використовується для класифікації (визначення типу об’єкта, такого як "спам" чи "не спам") та регресії (прогнозування безперервних значень, наприклад, цін або температур). Досягнення контрольованого навчання в таких галузях, як медична діагностика, кредитний скоринг або прогнозування ризиків, забезпечують його актуальність для задач із високою потребою у точності. Неконтрольоване навчання, навпаки, працює з невідомими вихідними значеннями і застосовується для пошуку внутрішніх закономірностей у даних. Найпоширенішим методом неконтрольованого навчання є кластеризація, яка використовується для групування даних у певні категорії або виявлення прихованих структур (рис. 1.3). Наприклад, у маркетингових дослідженнях цей метод дозволяє сегментувати споживачів для формування цільових пропозицій.

Рис.1.3. Пошук прихованих шаблонів у даних шляхом кластеризації [1]

Схема на рис. 1.4 ілюструє основні підходи та методи машинного навчання.

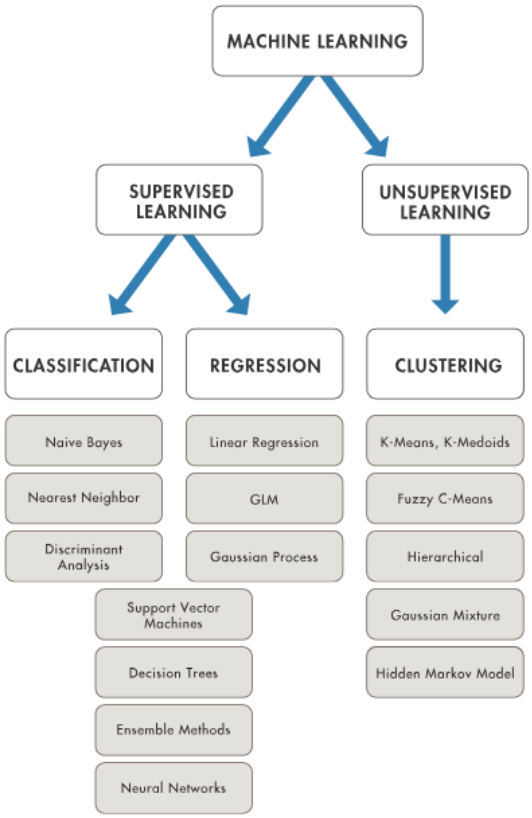


Рис.1.4. Техніки машинного навчання

Контрольоване навчання спрямоване на створення моделей, які прогнозують результати на основі відомих вхідних і вихідних даних. Воно охоплює два основні напрями: класифікацію та регресію. Класифікація використовується для прогнозування дискретних значень, тобто розподілу даних на категорії. Серед основних методів класифікації виділяють: методи Naive Bayes, алгоритм Nearest Neighbor, дискримінантний аналіз (Discriminant Analysis), метод опорних векторів (Support Vector Machines), дерева рішень (Decision Trees), ансамблеві методи (Ensemble Methods) та нейронні мережі (Neural Networks). Регресія, натомість,

спрямована на передбачення безперервних значень і використовується для моделювання залежностей у числових даних. До її методів належать: лінійна регресія (Linear Regression), узагальнені лінійні моделі (GLM), процеси Ґаусса (Gaussian Process).

Неконтрольоване навчання, на відміну від контрольованого, не вимагає міток даних для навчання моделі. Воно зосереджене на пошуку прихованих закономірностей у даних і включає техніки кластеризації. Найпоширенішими методами кластеризації є алгоритми K-means та K-Medoids, нечітке групування (Fuzzy C-Means), ієрархічна кластеризація (Hierarchical), ґаусівські суміші (Gaussian Mixture) та моделі Hidden Markov.

Ця структура дозволяє вибирати методи залежно від характеру даних і завдань. Контрольоване навчання підходить для задач, де результати можна визначити наперед, тоді як неконтрольоване ефективне для первинного аналізу та виявлення прихованих структур у даних. Такий підхід забезпечує гнучкість і широку функціональність машинного навчання у вирішенні реальних задач.

Машинне навчання є основним інструментом у сучасній аналітиці даних, дозволяючи організаціям ефективно працювати з великими обсягами інформації, прогнозувати майбутні події та оптимізувати процеси. Цей підхід значно підвищує ефективність рішень, пропонуючи автоматизацію, інтерпретацію та швидку реакцію на зміни.

Основні ролі машинного навчання:

1. Автоматизація обробки даних: ML автоматизує очищення, трансформацію та інженерію ознак, знижуючи витрати часу і зусиль. Завдяки автоматизації рутинних завдань аналітики зосереджуються на стратегічних аспектах.

2. Виявлення прихованих закономірностей: кластеризація (наприклад, алгоритм K-means) використовується для сегментації клієнтів, зменшення розмірності (PCA) дозволяє спрощувати аналіз великих наборів даних.

3. Прогнозування майбутніх трендів: регресія та класифікація забезпечують точні прогнози, наприклад, для продажів чи ризиків шахрайства.

4. Реальна часова аналітика: алгоритми обробляють дані в реальному часі, забезпечуючи миттєве прийняття рішень у таких сферах, як кібербезпека.

5. Візуалізація та інтерпретація: використання Explainable AI (XAI) та інтерактивних візуалізацій робить складні результати більш зрозумілими.

6. Прескриптивна аналітика: рекомендаційні системи пропонують індивідуальні рішення для оптимізації процесів і підвищення доходів.

Машинне навчання підвищує ефективність бізнесу, автоматизуючи рутинні завдання та оптимізуючи аналіз даних. Алгоритми, як K-means та PCA, забезпечують сегментацію клієнтів і спрощення великих наборів даних, що дозволяє зосередитися на стратегічних цілях.

Прогнозування й аналітика в реальному часі роблять машинне навчання незамінним для прийняття рішень. Регресія та класифікація прогнозують продажі чи ризики, а Explainable AI забезпечує прозорість результатів, що сприяє довірі до систем.

Машинне навчання є невід'ємною частиною сучасної аналітики даних. Його застосування сприяє ефективній автоматизації, глибокому аналізу та прийняттю рішень на основі даних (табл. 1.1). Завдяки впровадженню ML компанії отримують конкурентні переваги, підвищують продуктивність і покращують досвід клієнтів. [2]

Таблиця 1.1

Порівняння ключових ролей ML

Роль ML	**Приклад використання**	**Перевага**
Автоматизація обробки даних	Очищення даних у банківських транзакціях	Зниження витрат часу
Прогнозування	Передбачення поведінки клієнтів	Підвищення точності бізнес-планування

Реальна часова аналітика	Виявлення шахрайства у фінансовій сфері	Швидке реагування на загрози
Візуалізація	Інтерактивні панелі для представлення звітів	Покращення комунікації результатів
Прескриптивна аналітика	Оптимізація логістики в ритейлі	Підвищення ефективності операцій

Машинне навчання є надзвичайно гнучким інструментом, який знаходить застосування у багатьох галузях. У медицині алгоритми ML використовуються для аналізу медичних зображень, передбачення захворювань або визначення ефективності лікування. У фінансовій сфері вони допомагають прогнозувати ринкові ризики, оцінювати кредитоспроможність клієнтів і виявляти шахрайство в операціях. У виробництві та логістиці ML сприяє оптимізації процесів, забезпечуючи ефективність управління ресурсами, прогнозування потреб або планування маршрутів. Особливо перспективним напрямом є його використання у сфері енергетики, де алгоритми машинного навчання допомагають передбачати навантаження на електромережі, оптимізувати споживання ресурсів і знижувати витрати.

Крім того, важливим є те, що машинне навчання відкриває нові горизонти для розвитку цифрової економіки та інновацій. У сучасному світі, де обсяг інформації збільшується в геометричній прогресії, традиційні підходи до обробки даних втрачають свою ефективність. Машинне навчання дозволяє не лише аналізувати ці дані, але й виявляти приховані закономірності, формувати рекомендації та приймати обґрунтовані рішення. Наприклад, у сфері електронної комерції алгоритми ML забезпечують персоналізований підхід до клієнтів через аналіз їхньої поведінки, тоді як у медіа-індустрії — пропонують рекомендації фільмів чи музики на основі вподобань користувача. У транспортній інфраструктурі машинне навчання допомагає оптимізувати маршрути, знижуючи витрати на перевезення та підвищуючи якість обслуговування клієнтів.

Окремо слід зазначити розвиток спеціалізованого підходу до машинного навчання — глибокого навчання (deep learning), яке дозволяє обробляти дані у більш складних формах. Глибоке навчання забезпечує автоматичне виділення релевантних характеристик даних, що є важливим для роботи з великими наборами інформації, такими як зображення, відео або природна мова. Цей підхід вимагає великих обсягів даних та обчислювальних ресурсів, однак він є незамінним у таких завданнях, як розпізнавання облич або автоматичний переклад.

Зважаючи на стрімкий розвиток цифрових технологій, машинне навчання стає критично важливим інструментом для ефективного аналізу даних та забезпечення конкурентоспроможності бізнесу. Його широкі можливості щодо адаптації до різних галузей роблять машинне навчання універсальним рішенням для сучасного суспільства, відкриваючи нові перспективи для прогресу та інновацій. [1]

1.2 Огляд алгоритмів машинного навчання, що застосовуються для аналізу продажів

Машинне навчання є важливим інструментом для аналізу продажів, що дозволяє компаніям отримувати цінні інсайти та підвищувати ефективність бізнес-рішень. Алгоритми ML забезпечують автоматизацію складних процесів, таких як прогнозування попиту, сегментація клієнтів та розробка рекомендаційних систем. У сфері аналізу продажів використовуються як алгоритми контрольованого навчання, що передбачають результати на основі навчальних даних, так і неконтрольованого навчання, спрямовані на виявлення прихованих закономірностей у даних.

У сучасних умовах конкуренції машинне навчання (ML) стало одним із ключових інструментів для оптимізації продажів, аналізу клієнтської поведінки та прогнозування попиту (рис. 1.5). Згідно зі звітом McKinsey & Company, організації, які впроваджують ML у продажах, демонструють зростання доходу на 5–10% у порівнянні з конкурентами, які покладаються на традиційні методи. У 2023 році

глобальний ринок ML у продажах оцінювався у 15,6 мільярда доларів і, за прогнозами, до 2028 року досягне 36,2 мільярда доларів із середньорічним темпом зростання 18,5%.

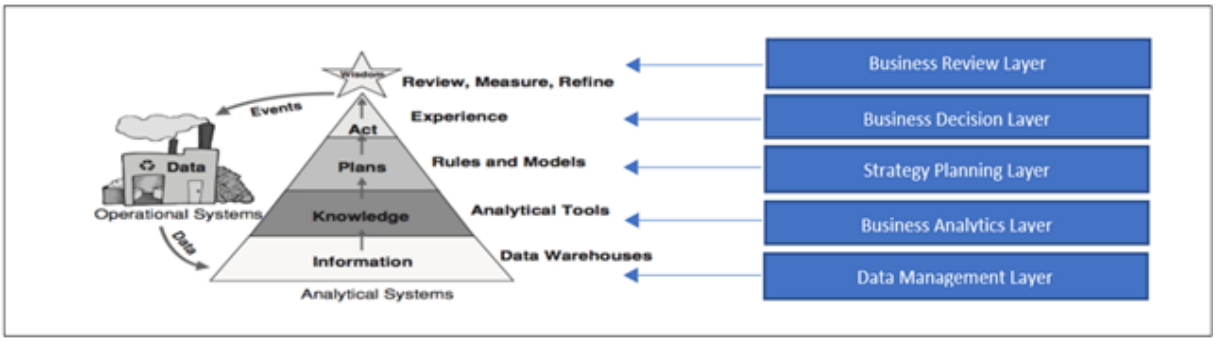


Рис.1.5. Рівні включення аналітичної системи до процесу продажів [5]

Напрями впровадження ML у продажах:

1. Прогнозування попиту
2. Динамічне ціноутворення
3. Сегментація клієнтів
4. Рекомендаційні системи
5. Виявлення шахрайства

Прогнозування обсягів продажів є важливою складовою стратегічного планування. ML дозволяє враховувати історичні дані, сезонність і зовнішні фактори (наприклад, економічну ситуацію). Алгоритми регресії, такі як лінійна регресія та дерева рішень, є найчастіше застосовуваними для цієї задачі. Наприклад, використання ML у прогнозуванні дозволило Walmart знизити витрати на 10% завдяки точнішому плануванню запасів.

Завдяки ML компанії можуть адаптувати ціни в реальному часі залежно від попиту, конкурентних цін та інших змінних. Алгоритми класифікації, такі як Random Forest, та рекурентні нейронні мережі (RNN) дозволяють прогнозувати поведінку клієнтів і встановлювати оптимальні ціни. Згідно з даними Boston Consulting Group, компанії, які впровадили динамічне ціноутворення, збільшили свій дохід на 3–5%.

ML дозволяє створювати сегменти клієнтів на основі їхньої поведінки, демографії та купівельних звичок. Алгоритми кластеризації, такі як K-means, широко використовуються для сегментації. Наприклад, Amazon використовує ML для персоналізації пропозицій, що підвищує рівень конверсії на 29%.

Рекомендаційні системи базуються на алгоритмах асоціативного аналізу (Apriori, FP-Growth) та нейронних мережах. Вони визначають товари, які клієнти найімовірніше купуватимуть разом. Наприклад, Netflix та eBay звітують про зростання продажів на 20–30% завдяки ML-рекомендаціям.

Шахрайські транзакції є серйозною проблемою для компаній. ML-алгоритми, такі як SVM і Gradient Boosting, дозволяють ідентифікувати аномальні патерни в даних і мінімізувати ризики. За оцінками PwC, впровадження систем на основі ML знижує втрати від шахрайства на 35%.

Алгоритми ML у продажах:

- Регресія: для прогнозування обсягів продажів, наприклад, використання лінійної регресії дозволяє досягати точності до 90% у прогнозах.
- Кластеризація (K-means): дозволяє сегментувати клієнтів за поведінковими параметрами. У ритейлі такі моделі підвищують ефективність маркетингових кампаній на 25%.
- Дерева рішень та Random Forest: ефективні для аналізу факторів, які впливають на продажі.
- LSTM (Long Short-Term Memory): для аналізу сезонних змін і довгострокових прогнозів. Наприклад, у сфері FMCG такі алгоритми підвищують точність прогнозів на 15–20%.

Прогнозування з використанням ML базується на двох основних етапах. Першим етапом є кластеризація продуктів та клієнтів. Для сегментації продуктів та клієнтських сегментів використовується алгоритм K-means та аналіз головних компонент (PCA). Ці методи дозволяють розділити клієнтів на групи залежно від їхнього внеску в обсяги продажів. Наприклад, дані сегменти дозволяють класифікувати клієнтів за їхньою поведінкою (регулярні покупці, сезонні клієнти тощо).

Другим етапом є прогнозування майбутніх трендів. Алгоритми прогнозування, такі як ARIMA (AutoRegressive Integrated Moving Average), використовуються для визначення трендів у продажах. Поєднання ARIMA з Python-скриптами та автоматизованими інструментами Power BI створює ефективну систему для побудови прогнозів та прийняття обґрунтованих рішень.

Схема на рис. 1.6 ілюструє процес описової аналітики, що застосовується для аналізу минулої ефективності бізнесу. Дана система включає вибір таблиць і ключових характеристик (факторів), які мають значення для аналізу, аналіз атрибутів даних, таких як клієнти, продукти, локації, час та обсяги продажів, групування й фільтрація атрибутів для визначення основних показників, таких як дохід, кількість проданих одиниць, ціна за одиницю, скасовані замовлення тощо. Цей підхід дозволяє побудувати загальну картину бізнесу в минулому та оцінити загальний стан компанії.

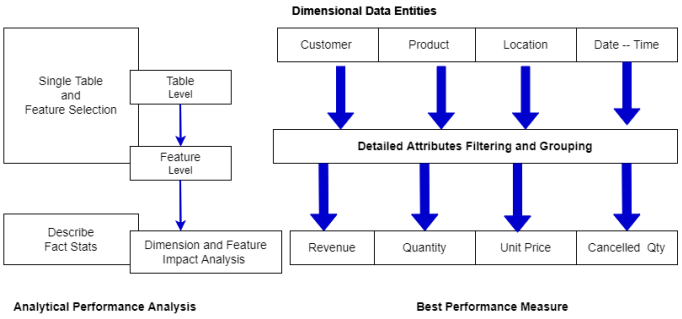


Рис.1.6. Описова аналітика ефективності бізнесу [5]

Схема, показана на рис. 1.7, демонструє більш детальний підхід до аналізу минулої ефективності з використанням діагностичної аналітики. Основні етапи: вибір і аналіз даних на рівні таблиць та окремих характеристик, використання динамічних фільтрів для порівняння результатів за різними групами даних та застосування карт і фільтрів типу "What If" для оцінки впливу різних факторів на кумулятивні результати. Ця модель допомагає зрозуміти, чому певні результати було досягнуто, і визначити основні чинники, що вплинули на минулу ефективність.

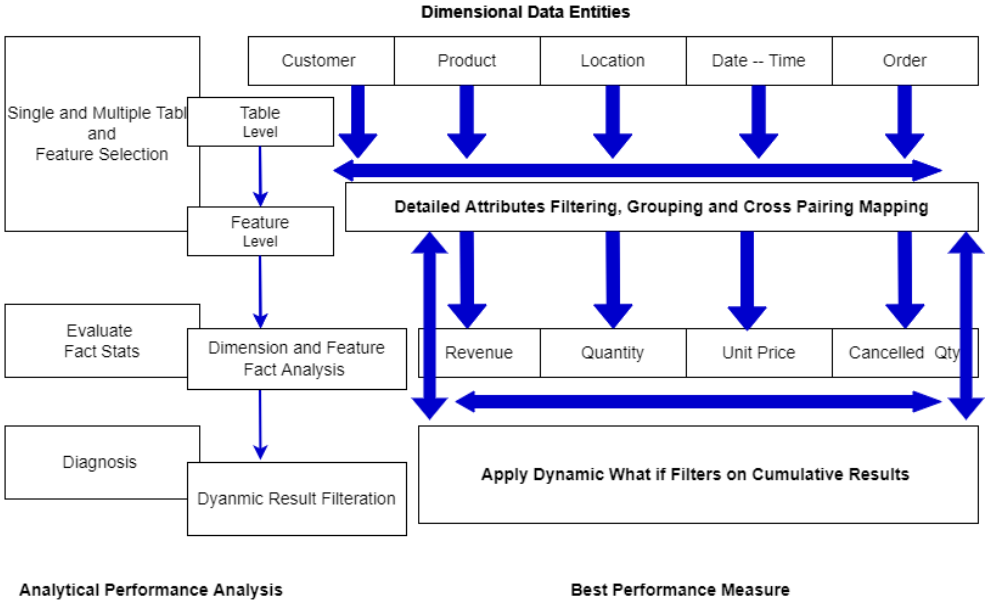


Рис.1.7 – Аналіз ефективності продажів за допомогою діагностичної аналітики [5]

Рис. 1.8 відображає підхід до прогнозування майбутньої ефективності бізнесу з використанням машинного навчання. Основні етапи включають використання кластеризації клієнтів (наприклад, RFM-аналізу) для ідентифікації купівельних моделей і впливу клієнтів, прогнозування поведінки клієнтів і сегментація на основі категорій купівель, прогнозування продажів з урахуванням майбутніх попиту, трендів та інших факторів, а також аналіз важливості характеристик, таких як продажі, продукти, попит і майбутні перспективи. Ця модель дозволяє бізнесу приймати стратегічні рішення на основі прогнозів, оптимізуючи взаємодію з клієнтами та управління ресурсами. [5]

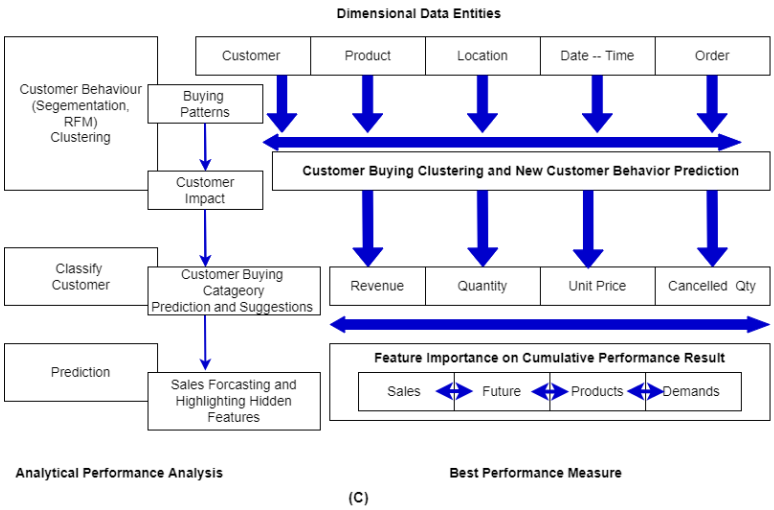


Рис.1.8. Алгоритм прогнозування ефективності продажів [5]

Машинне навчання є ключовим інструментом для трансформації процесів у продажах. Його використання дозволяє автоматизувати прогнозування, оптимізувати ціни, розробляти персоналізовані рекомендації та підвищувати безпеку транзакцій. Завдяки впровадженню ML компанії отримують значні конкурентні переваги, оптимізуючи витрати та підвищуючи рівень задоволеності клієнтів.

1.3 Дослідження впливу якісної підготовки даних на точність алгоритмів

Одна з основних проблем, яка залишається актуальною, – це якість вихідних даних, від яких безпосередньо залежить продуктивність і точність моделей машинного навчання. Ефективність моделей ML напряму залежить від якості даних. Джерело, метод збору, частота та тривалість захоплення даних визначають точність прогнозів і продуктивність моделей.

Ключові аспекти, що визначають якість, включають точку захоплення, метод збору, рівень шуму, частоту та тривалість збору даних. Точка захоплення є критично важливою: наприклад, розміщення датчика температури безпосередньо на машині забезпечує точність на 20–30% вищу порівняно із віддаленими джерелами. Подібно, у промислових додатках коректне позиціонування датчика дозволяє зменшити вплив зовнішніх факторів, таких як температура навколишнього середовища, до мінімуму.

Метод збору даних також суттєво впливає на їх релевантність. Інфрачервоні зображення дозволяють візуалізувати розподіл тепла з точністю до 90%, тоді як фізичні датчики краще фіксують механічний знос. Наприклад, у виробничій лінії дефекти виявляються на 40% швидше завдяки візуальному аналізу, ніж за допомогою традиційних методів.

Шум у даних є ще одним важливим фактором. Реальні дані майже ніколи не є чистими, і хоча помірний шум підвищує стійкість моделі на 10–15%, надмірний або некоректний шум може знижувати точність прогнозів на 25–30%. Це може статися, наприклад, через збої датчиків, які додають хибні змінні або викиди. Згідно з дослідженнями, 18% втрат продуктивності моделей пов’язано саме з некоректним обробленням шуму в даних.

Частота збору даних також має вирішальне значення. Для вібраційних даних двигуна необхідна висока частота зразків — до 1 000 вимірювань на секунду, тоді як для температурних змін у приміщенні достатньо частоти в межах 1–2 вимірювань за хвилину. Невідповідна частота може призвести або до втрати критичних змін, або до збереження зайвих і дубльованих даних, що збільшує обсяги обробки на 15–20%.

Останнім, але не менш важливим фактором, є тривалість збору даних. Наприклад, аналіз витоків струму потребує кількох хвилин безперервного збору зразків, щоб зафіксувати подію повністю. Водночас тривалі періоди моніторингу дозволяють уникнути збоїв, підвищуючи достовірність результатів на 25%.

Таким чином, якість даних є визначальним фактором для ефективності моделей машинного навчання. Правильний підхід до збору, обробки та забезпечення якості даних може підвищити продуктивність моделей на 30–50%, забезпечуючи точність і надійність прогнозів у реальних умовах.

Однією з головних проблем машинного навчання є робота з високовимірними даними, де кількість вимірів (атрибутів або ознак) може бути значною. Інтуїтивно здається, що збирання якомога більшої кількості атрибутів дозволить точніше відобразити стан об’єкта. Проте це має зворотний ефект: зі збільшенням кількості атрибутів (вимірності) кількість можливих унікальних комбінацій зростає експоненціально, зменшуючи охоплення вибірки. Це явище відоме як "прокляття розмірності" (Curse of Dimensionality).

Висока вимірність не лише збільшує складність обчислень, але й вимагає більших ресурсів, таких як обчислювальна потужність, пам’ять та пропускна здатність мережі. Ефективність моделі машинного навчання значно знижується зі

збільшенням вимірності, що показано на графіку на рис. 1.9. Тому грамотний вибір релевантних ознак є ключовим для підвищення продуктивності та точності моделі.

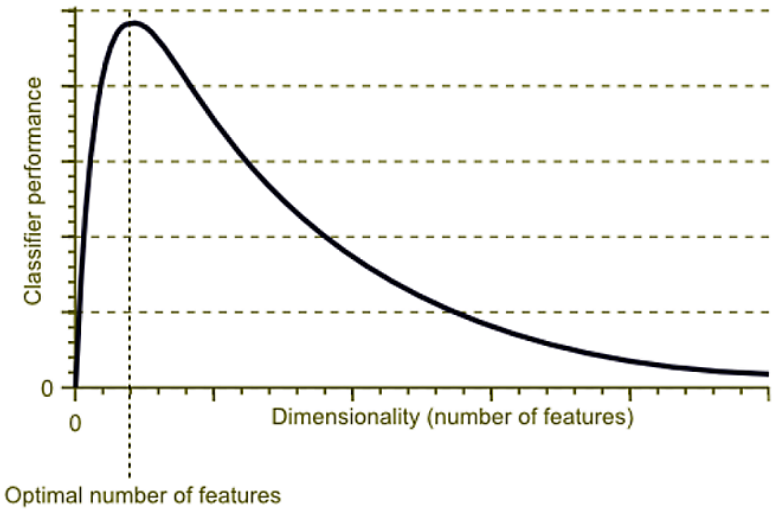


Рис.1.9. Залежність ефективності моделі машинного навчання від вимірності даних [6]

Вибір ознак, або атрибутів, полягає у визначенні характеристик, які найкраще підходять для створення стабільної прогнозної моделі та підвищують її ефективність. Завдання полягає у виборі підмножини атрибутів із вхідного набору даних, які найкраще прогнозують вихідний набір. Основна ідея вибору ознак схожа на фільтрування шуму в галасливому середовищі: усунення нерелевантних атрибутів дозволяє створити скорочений набір ознак із максимальною прогностичною здатністю.

У деяких випадках релевантні атрибути можна визначити інтуїтивно. Наприклад, у наборі даних, що містить параметри роботи електродвигуна, атрибути «дата» чи «назва заводу» навряд чи впливатимуть на прогноз несправності. В інших випадках, коли значення атрибутів неочевидні, можна використовувати методи аналізу, такі як локалізація, сегментація потоку даних та анотація (для аналізу зображень і відео), або техніки видалення стоп-слів, інформаційного приросту (IG), квадрату (X^2) і «мішок слів» (BOW) для обробки нативної мови. Для дискретних і часових даних підходять такі методи, як лінійний

дискримінантний аналіз (LDA), аналіз головних компонент (PCA) та дисперсійний аналіз (ANOVA).

Використання нерелевантних ознак негативно впливає на модель, спотворюючи її результати та змушуючи враховувати незначущі атрибути. Наприклад, у моделі класифікації виживання пасажирів «Титаніка» атрибут «ім'я пасажира» не дає корисної інформації. Техніки, такі як F-тест, покроковий відбір ознак (Forward Selection) та регресія LASSO, дозволяють оцінювати значущість ознак, аналізуючи їхній внесок у продуктивність моделі.

Ще однією серйозною проблемою є дисбаланс даних, коли представленість класів у наборі навчальних даних є нерівномірною. У таких випадках більшість класів домінує над меншістю, що робить модель упередженою до більшості. Дисбаланс може виникати через особливості збору даних і є звичним явищем у природі, наприклад, позитивні зразки (якісна продукція) зазвичай значно переважають негативні (дефектна продукція). Як показано на рис. 10, надійність моделі значно залежить від збалансованості класів у навчальному наборі. Збалансований розподіл даних між класами забезпечує точніші прогнози та зменшує ймовірність упередженості.

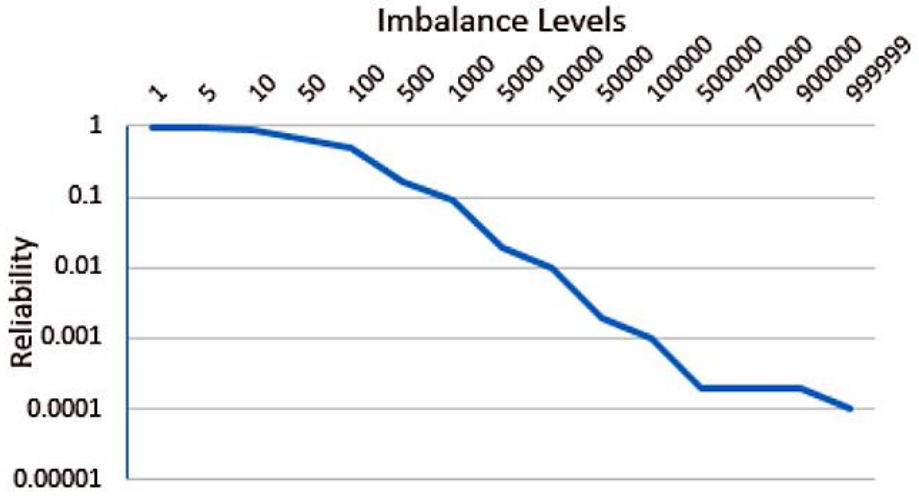


Рис.1.10. Вплив дисбалансу класів на надійність роботи алгоритму [6]

Внутрішньокласова варіативність стосується розбіжностей між зразками, які належать до одного класу. Наприклад, у навчальному наборі даних для класифікації різних типів транспортних засобів зразок автомобіля може бути червоним чи синім, хетчбеком чи седаном, квадратним чи обтічним, із жорстким дахом чи кабріолетом тощо. Це є розширенням концепції представлення класів, яка була розглянута раніше. Набір даних із достатньою варіативністю всередині класів покращує ефективність моделі та знижує ризик перенавчання в межах одного класу. Варіативність також забезпечує узагальнення моделі для неконтрольованих реальних застосувань. На практиці для створення синтетичних даних, які охоплюють різноманітні варіації, використовуються інструменти для аугментації зображень та симуляції пристроїв.

Ці два аспекти є ключовими при оцінюванні придатності набору даних для моделі. Високий дисбаланс класів (упередженість) або низька міжкласова варіативність можуть призвести до перенавчання моделі, що зменшить її здатність до узагальнення. У протилежному випадку це може викликати недонавчання моделі та низьку швидкість її збіжності (рис. 1.11). Для оптимізації продуктивності моделі необхідно досягти правильного балансу між цими аспектами.

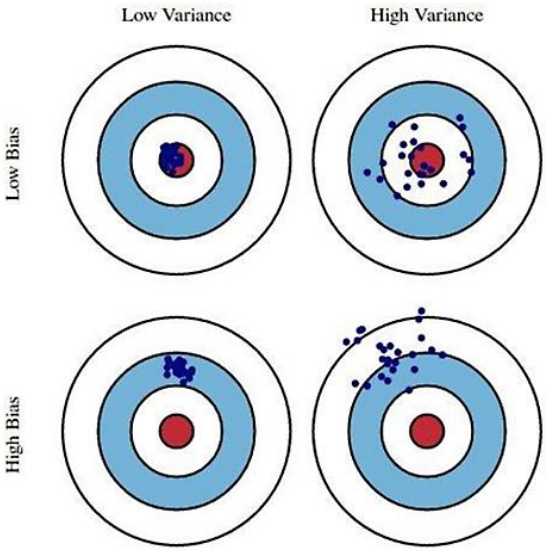


Рис.1.11. Баланс між упередженістю та варіативністю [6]

Один із способів виміряти упередженість та варіативність даних для моделі — порівняти точність передбачень (precision) із повнотою охоплення класів (recall) під час інференції та налаштувати набір даних на основі отриманих результатів. Точність (precision) відповідає на запитання, який відсоток передбачень, зроблених на валідаційному наборі даних, є дійсно правильними. Повнота (recall) вказує, який відсоток екземплярів класу був фактично ідентифікований. Для того щоб модель була ефективною, вона повинна мати оптимальні показники як точності, так і повноти.

Розрахунок точності виконується наступним методом:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Тоді як відкликання можна розрахувати таким чином:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Де TP = загальна кількість позитивних результатів, що вказує на правильні прогнози; FP = помилкові позитивні результати та FN = помилкові негативні результати, що вказує на неправильні прогнози.

Зібраний для аналізу набір даних може містити багато домішок, які впливають на якість моделей машинного навчання. Наприклад, у потоці даних із датчиків можуть з’являтися раптові викиди, спричинені тимчасовими збоями датчиків, помилками передачі чи збереження даних. Для потоків візуальних даних проблеми можуть включати фоновий шум, затемнення або перебої в подачі сигналу. Ці аномалії створюють нові ознаки, які насправді є нерелевантними до задачі, що вирішується моделлю. Основна мета очищення даних — видалення домішок, пропущених значень, дублікатів чи некоректних елементів даних. Очищення зазвичай виконується як на етапі навчання, так і під час інференції для підготовки потоку даних до подальшої обробки.

Шум у даних визначається як елемент, що не відповідає решті потоку даних. Важливо зазначити, що шум — це не завжди викид. Викид може бути елементом, що відрізняється за масштабом чи віддаленістю від решти даних, але все ж є валідним. Наприклад, у потоці даних про зріст осіб викидом може бути значення 200 см або 121 см, що вказує на дуже високого чи низького члена групи, але значення -50 см чи 1500 см є явним шумом. Шумові елементи не додають цінності моделі, тоді як викиди можуть зробити модель більш стійкою та уникнути перенавчання.

У візуальній аналітиці шум може виникати через нерелевантні об'єкти, затемнення, розмиті чи спотворені зображення. Для усунення небажаних особливостей із вхідних даних використовуються техніки попередньої обробки зображень, такі як морфологічні операції, локалізація та сегментація. У задачах обробки природної мови шум можна зменшити шляхом видалення зайвих пробілів, розділових знаків, стоп-слів, виправлення регістру, перевірки орфографії тощо. У потоках даних шум фільтрується за допомогою методів вимірювання відстаней, фільтрів (наприклад, низькочастотних, фільтрів Калмана, перетворення Фур'є) та інтерполяції. Помилкове маркування ознак (label noise) може викликати шум міток, що негативно впливає на точність моделі.

Пропущені значення у потоках даних виникають через помилки збору або обробки. Причинами можуть бути як людські фактори (ігнорування, пропуски), так і технічні збої (помилки, втрата з'єднання). Пропущені дані класифікуються на три основні категорії:

- Пропущені не випадково (MNAR): зумовлені відомими факторами, такими як квантовані обмеження.
- Пропущені випадково (MAR): несподівані втрати через зовнішні чинники.
- Пропущені повністю випадково (MCAR): непередбачувані втрати без очевидних причин.

Обробка пропущених даних залежить від властивостей потоку, обсягу даних і ступеня їхньої розрідженості. Один із простих методів — видалення рядків із пропущеними значеннями. Це дозволяє уникнути додавання змодельованих чи

наближених значень, але водночас зменшує обсяг даних, що може створити упередженість, якщо пропущені значення належать до конкретного класу (рис. 1.12).

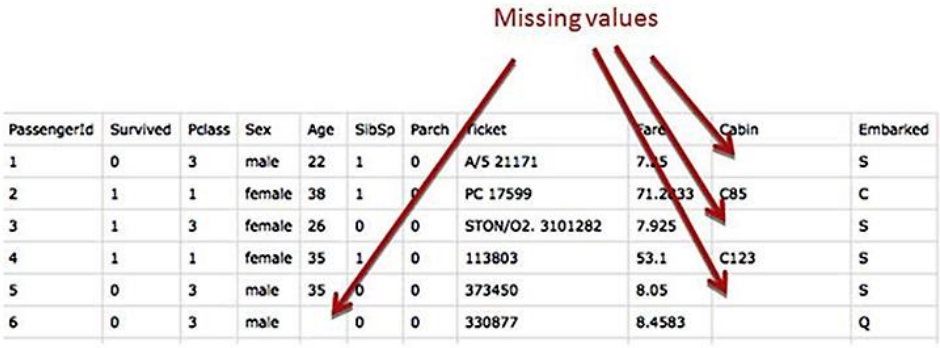


Рис.1.12. Відсутні значення в наборі даних [6]

Пропущені значення можна заповнювати середнім, медіаною чи модою, що підходить для даних із низькою варіативністю, наприклад, кімнатної температури. Для змінних із високою варіативністю (наприклад, вібраційні датчики) потрібні точніші методи, такі як регресія або K-ближчих сусідів, але вони складніші в реалізації. При значній втраті даних (понад 20%) слід переглянути методи збору, щоб уникнути упередженості.

Масштабування ознак вирівнює вплив величин, забезпечуючи рівномірний розподіл ваг між атрибутами. Це особливо важливо для алгоритмів KNN, K-means та SVM. Стандартизація (Z-оцінка) перетворює дані на нормальний розподіл із середнім 0 і стандартним відхиленням 1, що пришвидшує збіжність моделей і підвищує їх ефективність.

Іншим способом масштабування є метод Min-Max, який є досить простим. Цей метод масштабує дані в діапазон від 0 до 1. Недоліком цього методу є те, що масштабування до обмеженого діапазону може призвести до зменшення стандартного відхилення (рис. 1.13).

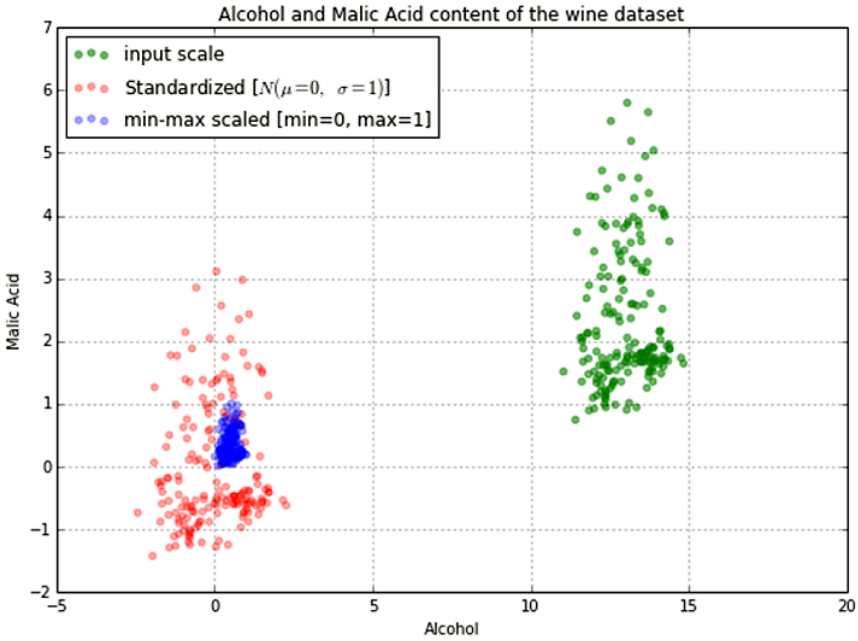


Рис.1.13. Ефект стандартизації [6]

Терміни "масштабування ознак" і "нормалізація ознак" часто використовуються як взаємозамінні, хоча вони мають різні підходи. Масштабування трансформує діапазон даних (наприклад, метод Min-Max), тоді як нормалізація змінює їхній розподіл, перетворюючи його на нормальний. Для специфічних завдань також використовуються інші методи, такі як нормалізація середнього, масштабування одиничного вектора чи Box-Cox трансформація.

У глибоких мережах навіть незначні зміни у вхідних параметрах можуть посилюватися з глибиною. Нормалізація ознак у таких випадках допомагає зменшити внутрішній зсув і прискорити збіжність моделі.

Інженерія ознак — це процес перетворення сирих даних на вектор ознак, придатний для алгоритмів машинного навчання. Реальні дані представлені в різних формах (текст, числа, зображення, часові ряди), і їх потрібно ефективно перетворювати. Цей процес також включає усунення шуму та непотрібних варіацій, потребуючи знання предметної області для створення інформативних ознак. [6]

Підготовка даних є критично важливим етапом у процесі побудови моделей машинного навчання, адже від якості даних залежить точність і надійність

прогнозів. Ретельно підготовлені дані дозволяють моделям ефективно аналізувати інформацію та уникати упередженості. Усунення шуму та обробка пропущених значень сприяють зменшенню впливу помилок і покращенню загальної продуктивності моделей. Балансування класів у наборі даних гарантує справедливий розподіл ваги між усіма категоріями, що дозволяє уникнути перенавчання та забезпечити коректне узагальнення. Масштабування ознак вирівнює вплив різних атрибутів, сприяючи швидшій збіжності моделей і зменшенню ресурсних витрат. Крім того, правильна нормалізація даних допомагає усунути внутрішній зсув у глибоких мережах, підвищуючи їхню стабільність. Інженерія ознак дозволяє перетворити сирі дані у формат, придатний для навчання, забезпечуючи максимальну інформативність. Використання відповідних методів очищення, балансування та трансформації даних забезпечує ефективність моделей у реальних сценаріях. Підготовка даних є не лише технічним, а й стратегічним процесом, що вимагає глибоких знань предметної області. Загалом, якісно підготовлені дані є основою успішного впровадження машинного навчання в різних сферах.

2 РОЗРОБКА МЕТОДИКИ АНАЛІЗУ ДАНИХ ПРОДАЖІВ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

2.1 Вибір алгоритмів та інструментів для побудови моделі

Машинне навчання є важливим інструментом у сучасній системі продажів, здатним значно підвищити ефективність і точність бізнес-процесів. Його інтеграція у сферу продажів дозволяє аналізувати великі обсяги даних, виявляти приховані закономірності та адаптуватися до швидко змінюваних ринкових умов. Однією з ключових переваг машинного навчання є його здатність персоналізувати взаємодію з клієнтами, що сприяє підвищенню рівня задоволеності та лояльності. Крім того, алгоритми машинного навчання дозволяють точніше прогнозувати попит, оптимізувати стратегії ціноутворення та визначати потенційно вигідні сегменти клієнтів. За допомогою інструментів машинного навчання команди продажів можуть скоротити час на виконання рутинних завдань, зосереджуючись на стратегічних аспектах своєї роботи. Прогностична аналітика, основана на ML, забезпечує реалістичне планування та розподіл ресурсів, зменшуючи невизначеність у прийнятті рішень. Системи машинного навчання також дозволяють ефективно виявляти ризики, пов’язані з відтоком клієнтів, і розробляти заходи для їх утримання. Такий підхід сприяє довгостроковому зростанню бізнесу, дозволяючи компаніям залишатися конкурентоспроможними на динамічному ринку. У поєднанні з сучасними технологіями обробки даних, машинне навчання стає ключовим компонентом цифрової трансформації у продажах. Застосування його інструментів відкриває нові можливості для автоматизації, персоналізації та прогнозування, що значно підвищує загальну продуктивність і прибутковість компаній. [7]

Для аналізу продажів доцільно використовувати алгоритми машинного навчання, які забезпечують високу точність прогнозів, адаптивність до змінних умов та здатність працювати з великими наборами даних. Серед найбільш релевантних варіантів можна виділити такі алгоритми: K-means, Random Forest та нейронні мережі.

Алгоритм K-means є одним із найпопулярніших методів неконтрольованого навчання, який застосовується для кластеризації даних. Його доцільно використовувати для сегментації клієнтів або продуктів у продажах. Наприклад, K-means дозволяє розділити клієнтів на групи за подібністю поведінкових характеристик, таких як частота покупок, середній чек або категорія товарів, що купуються. Це сприяє персоналізації маркетингових кампаній та підвищенню ефективності взаємодії з клієнтами. Основною перевагою K-means є його швидкість та простота реалізації, що дозволяє ефективно обробляти навіть великі обсяги даних (рис. 2.1). Проте цей алгоритм найкраще працює, коли кількість кластерів є відомою наперед, що вимагає попереднього аналізу даних.

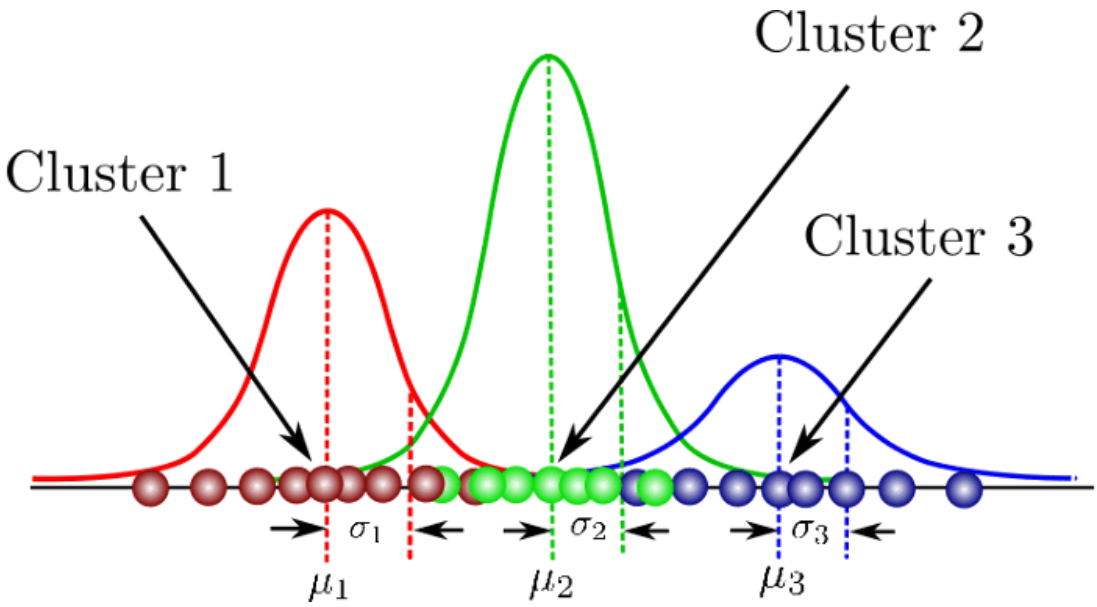


Рис.2.1. Візуалізація принципу кластеризації K-means

Random Forest є алгоритмом контрольованого навчання, який поєднує у собі кілька дерев рішень для досягнення більш точних і стабільних прогнозів. Його використання доцільне для задач прогнозування продажів, таких як оцінка майбутнього попиту чи визначення факторів, які найбільше впливають на успіх продукту. Завдяки механізму побудови ансамблю дерев, Random Forest стійкий до шуму в даних і може працювати з набором даних, що містить велику кількість ознак. Крім того, алгоритм дозволяє оцінити важливість кожної ознаки, що робить

його цінним інструментом для визначення ключових факторів, які впливають на продажі. Однак Random Forest може бути обчислювально затратним для дуже великих наборів даних.

Нейронні мережі, особливо рекурентні нейронні мережі (RNN) або моделі типу Long Short-Term Memory (LSTM), є найбільш доцільними для аналізу часових рядів, таких як історичні дані про продажі. Ці моделі здатні враховувати взаємозв'язки між даними у часовій послідовності, що дозволяє їм ефективно прогнозувати сезонні тренди, пікові періоди продажів та довгострокову динаміку попиту. Наприклад, у роздрібній торгівлі нейронні мережі можуть прогнозувати, як зовнішні фактори, такі як погода або святкові періоди, вплинуть на обсяги продажів. Висока точність прогнозів, яку забезпечують нейронні мережі, є їхньою головною перевагою, але вони вимагають значних обчислювальних ресурсів та ретельного налаштування для ефективного функціонування.

Вибір алгоритму машинного навчання залежить від конкретної задачі та характеристик набору даних. K-means є ідеальним для кластеризації, Random Forest — для прогнозування та аналізу важливості ознак, а нейронні мережі чудово підходять для роботи з часовими рядами. Кожен з цих алгоритмів робить вагомий внесок у покращення процесу аналізу продажів, підвищуючи ефективність та точність управлінських рішень.

Метод кластеризації K-means є ефективним інструментом для групування точок даних, що мають схожі характеристики. Цей метод належить до неконтрольованого машинного навчання, що означає його здатність групувати немарковані дані шляхом виявлення подібностей між ними. Зокрема, ця техніка є ідеальною для аналізу обраного набору даних. У межах дослідження передбачається демонстрація того, як клієнтів можна розподілити на групи на основі їхніх характеристик, а також зробити висновки на основі такого аналізу. Основною метою є визначення продуктів, які приносять прибуток чи збитки, а також класифікація клієнтів на групи.

Перед проведенням аналізу необхідно здійснити перевірку даних. Лише після перевірки цілісності даних та обробки спотворень можна бути впевненими у

точності отриманих результатів. Під час перевірки на наявність пропущених значень та дублікатів була створена описова характеристика даних (рис. 2.2), яка дозволяє забезпечити їхню придатність для подальшого аналізу.

```
In [7]: df.describe().T
Out[7]:
```

	count	mean	std	min	25%	50%	75%	max
Order_ID	730.0	29721.597260	17350.560439	35.00	14479.000000	29350.500	44978.500	59909.00
Order_Quantity	730.0	24.857534	14.182970	1.00	13.000000	25.000	37.000	50.00
Sales	730.0	1502.777653	2893.936648	8.60	171.098375	438.700	1480.855	27663.92
Profit	730.0	207.479233	878.631028	-4437.91	-77.132500	0.035	180.000	8417.57

Рис.2.2. Описова характеристика даних

Дані представлені у вигляді загальної характеристики, яка включає 730 рядків для кожного стовпця. Проведений аналіз дозволив визначити середні значення, стандартне відхилення, мінімальні значення та інші ключові статистичні параметри.

Після цього було застосовано функцію value_counts() для аналізу частоти кожного значення в даних (рис. 2.3). Ця функція дозволила визначити розподіл значень у кожному стовпці, а також частоту їхньої появи. Такий підхід сприяє кращому розумінню структури даних і є важливим етапом для подальшого аналізу.

```
In [8]: df['Region'].value_counts()
Out[8]: West                   232
        Atlantic               170
        Northwest Territories  150
        Prarie                  79
        Ontario                 67
        Nunavut                 32
        Name: Region, dtype: int64
```

Рис.2.3. Аналіз частоти кожного значення

Наприклад, у випадку зі стовпцем "Регіон" було виявлено, що найбільша кількість продажів здійснювалась клієнтами з регіону "Захід". На другому місці за кількістю покупок знаходиться регіон "Атлантичний" із 170 випадками, тоді як найменше покупок було зафіксовано у регіоні "Нунавут".

Також було проведено перевірку на унікальність даних у розділі "Підкатегорія продуктів", оскільки в деяких наборах даних одна і та ж категорія може бути записана в різний спосіб (рис. 2.4). У даному випадку дані виявилися однорідними, що значно спрощує подальший аналіз.

```
In [11]: df['Product_Sub-Category'].value_counts()

Out[11]: Binders and Binder Accessories    137
         Paper                             120
         Telephones and Communication       94
         Office Furnishings                 83
         Storage & Organization             63
         Appliances                         54
         Computer Peripherals               54
         Labels                             38
         Envelopes                          33
         Office Machines                    24
         Copiers and Fax                    13
         Chairs & Chairmats                 10
         Tables                              5
         Scissors, Rulers and Trimmers       1
         Pens & Art Supplies                 1
         Name: Product_Sub-Category, dtype: int64
```

Рис.2.4. Перевірка на унікальність підкатегорій

Під час перевірки кількості значень у стовпці "Order_ID" було виявлено повторювані покупки з однаковими ідентифікаторами. Це свідчить про необхідність обробки таких випадків, оскільки наявність дублікатів і неунікальних значень може призвести до неточності в аналізі даних. Для забезпечення достовірності результатів ці дублікатні записи потребують усунення або корекції (рис. 2.5).

```
In [10]: df['Order_ID'].value_counts()

Out[10]: 24132    6
         8995     4
         47846    4
         32611    3
         8992     3
                 ..
         35908    1
         35584    1
         33255    1
         28420    1
         20003    1
         Name: Order_ID, Length: 571, dtype: int64
```

Рис.2.5. Виявлення дублікатів даних

З цією метою було використано функцію nunique(), за допомогою якої дані у стовпці "Order_ID" були перетворені на унікальні значення.

На наступному етапі було створено стовпчасту діаграму для аналізу взаємозв’язку між регіонами та прибутком (рис. 2.6). Як видно з графіка, регіон "Прерії" зазнав значних збитків, які становлять майже 2000 доларів. У той же час регіон "Захід" є найбільш прибутковим із показником понад 15 000 доларів. Це можна пов’язати з тим, що саме "Захід" мав найбільшу кількість продажів. Така візуалізація сприяє глибшому розумінню регіональних показників та їх впливу на загальний прибуток.

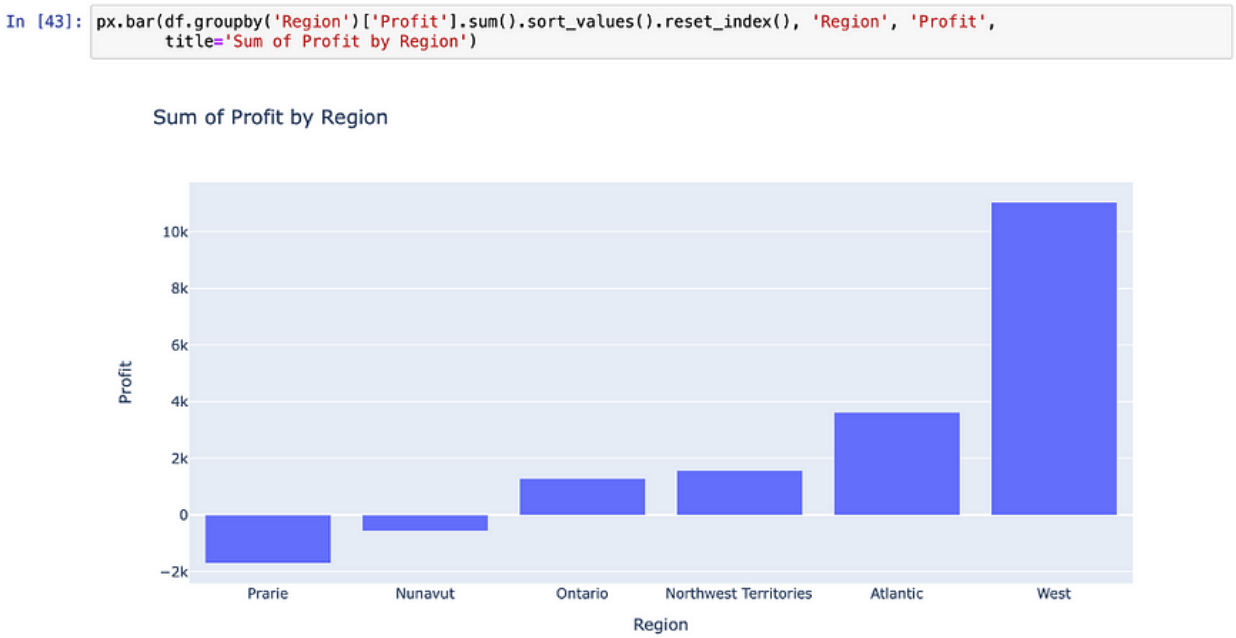


Рис.2.6. Аналіз взаємозв’язку між регіонами та прибутком

Дані стовпця "Order_ID" були згруповані разом із показниками "Продажі", "Прибуток" та "Кількість замовлень". Ці три змінні були обрані через їхній числовий характер, що дозволяє дослідити взаємозв’язки між ними та отримати більш глибоке розуміння структури продажів і прибутковості.

```
In [46]: rfm = df.groupby(by='Order_ID')[['Sales','Profit','Order_Quantity']].sum()
         rfm
```

Out[46]:

Order_ID	Sales	Profit	Order_Quantity
35	1892.8480	48.99	14
293	244.5700	46.71	27
322	2915.8555	-33.83	66
450	543.7200	-211.13	35
515	394.2700	30.94	19
...	...	...	...
59585	712.0400	-110.93	45
59651	283.2100	-196.06	36
59750	223.5900	-66.05	34
59878	362.0400	-138.40	29
59909	411.5600	-188.02	46

521 rows × 3 columns

Рис.2.7. Поєднання взаємозв’язків структури продажів

Перед початком кластеризації було прийнято рішення обробити вибіркові значення (викиди), оскільки вони можуть стати джерелом викривлення результатів аналізу. Усунення викидів сприяє підвищенню точності та надійності отриманих даних. На рис. 2.8 та 2.9 представлено розподіл даних до видалення вибіркових значень, що дозволяє оцінити вплив таких записів на структуру даних.

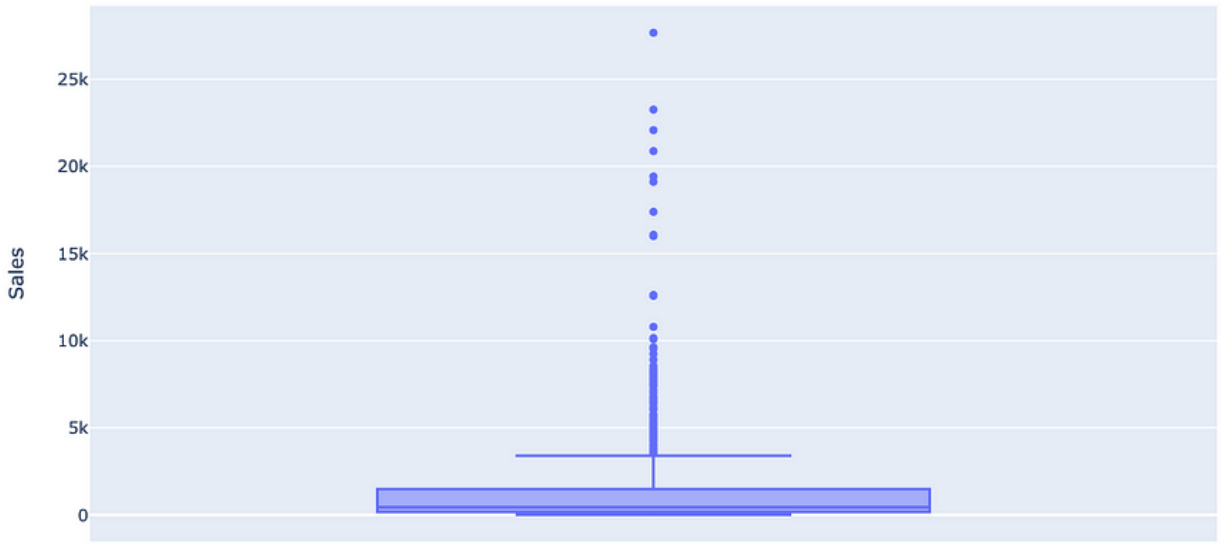


Рис.2.8. Розподіл даних до видалення викидів

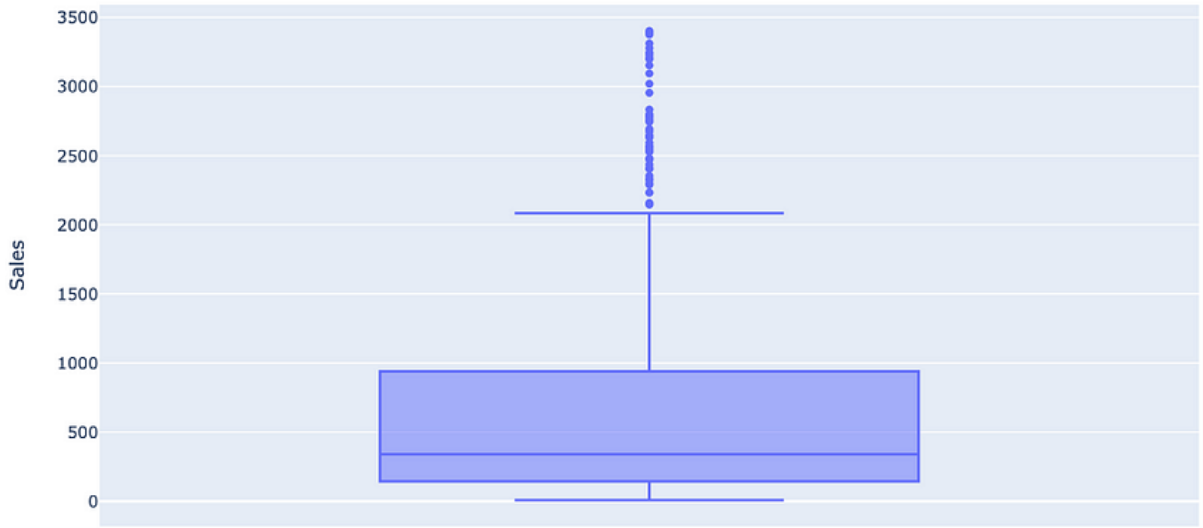


Рис.2.9. Розподіл даних після видалення викидів

Після завершення попередньої обробки даних було застосовано метод Elbow для визначення оптимальної кількості кластерів. Використання показника суми квадратів відстаней у межах кластерів (Within Cluster Sum of Squares, WCSS) дозволило оцінити якість групування. На основі отриманих результатів було встановлено, що оптимальним є поділ даних на три кластери. Цей підхід забезпечує баланс між деталізацією кластеризації та зменшенням втрат інформації (рис. 2.10).

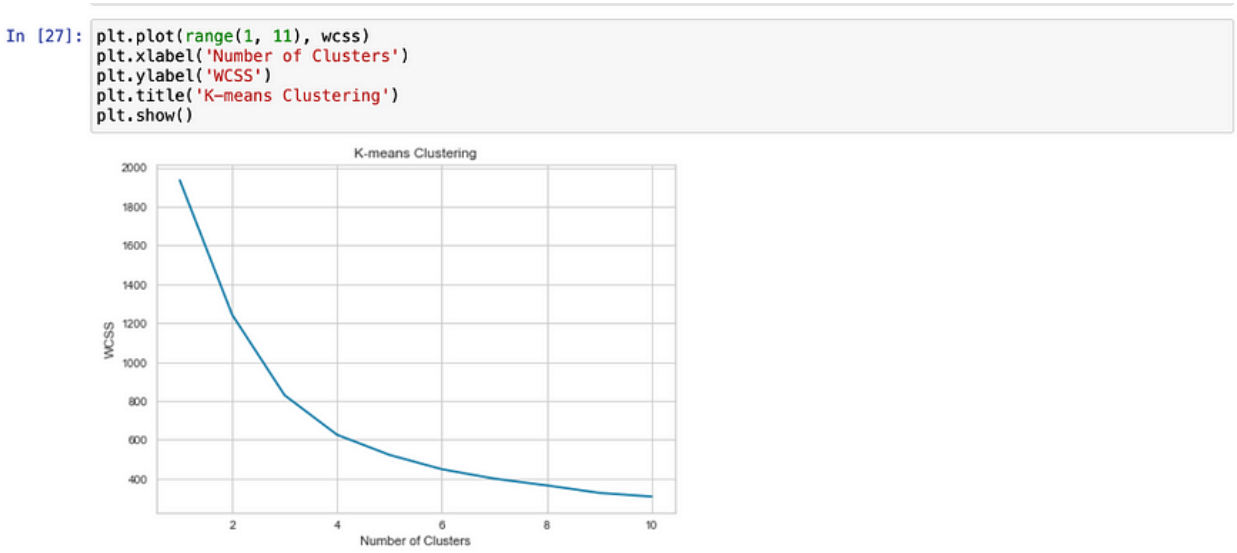


Рис.2.10. Визначення оптимального поділу на кластери

На наступному етапі було проведено оцінювання кластеризації за допомогою коефіцієнта силуету (Silhouette Scoring), який використовується для аналізу якості групування даних. Цей метод дозволяє оцінити, наскільки близькими є точки одного кластера одна до одної порівняно з точками інших кластерів. На основі побудованих графіків було визначено, що товщина кластерів відображає їхній розмір, а показник силуету допомагає виявити невдалу кластеризацію (рис. 2.11). Зокрема, класи, що мають середнє значення коефіцієнта силуету нижче встановленого рівня, вважаються небажаними для поділу даних. Це дозволяє забезпечити точнішу та ефективнішу кластеризацію.

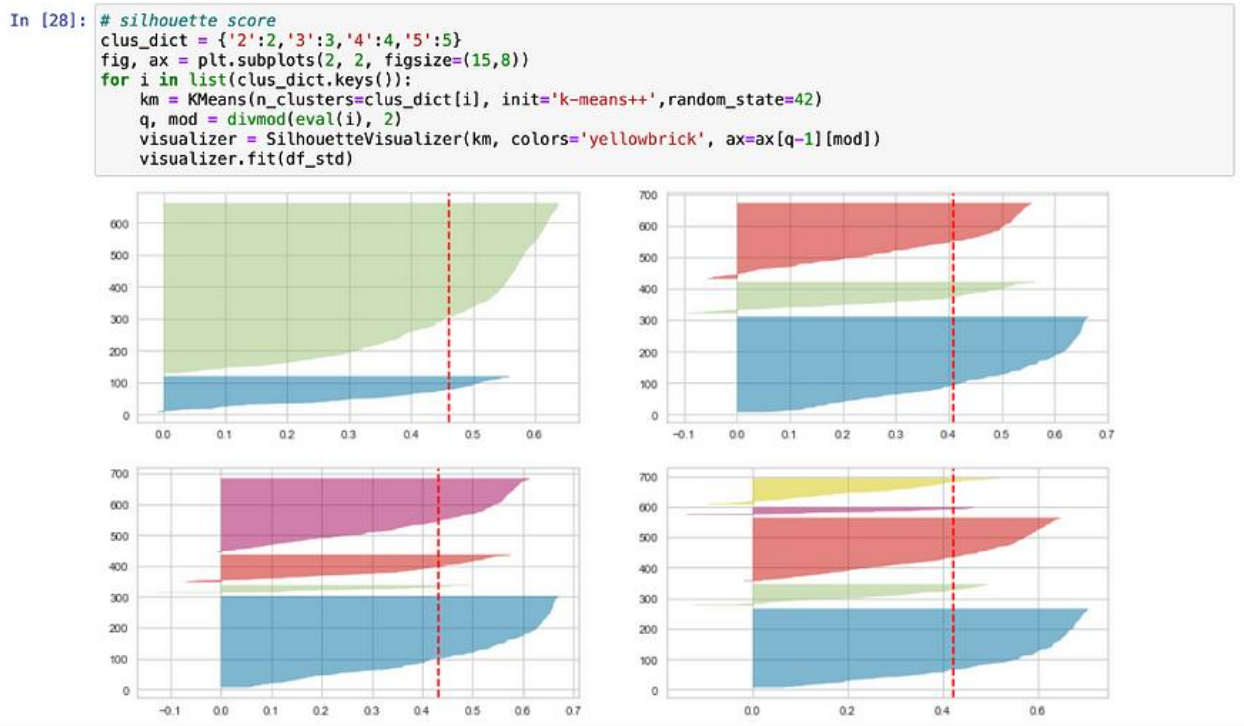


Рис.2.11. Аналіз кластеризації

На завершення слід зазначити, що метод кластеризації дозволяє мінімізувати суму квадратів відстаней між точками даних та їх відповідними центроїдами. Це сприяє визначенню оптимального центру кластерів, який мінімізує дисперсію всередині кожного кластеру. На основі проведеного аналізу було встановлено, що оптимальною є кластеризація з трьома кластерами. Аналіз показав, що найбільшу кількість замовлень було згенеровано сегментом 2, проте цей сегмент також

виявився найзбитковішим (рис. 2.12). Найприбутковішим клієнтським сегментом є сегмент 1, оскільки він забезпечує найвищі показники продажів і прибутку. Отримані результати можуть бути використані компанією для оптимізації поведінки клієнтів під час покупок та запобігання збиткам у майбутньому.

```
In [45]: df_std = df.groupby(['Segment_k']).mean()
         df_std
```

Out[45]:

Segment_k	Order_ID	Order_Quantity	Sales	Profit	Sales_out
0	29157.559211	11.513158	353.711605	-40.451184	0.0
1	30449.010000	31.260000	2065.005955	474.746900	0.0
2	30585.481328	36.236515	538.321035	-82.674564	0.0

Рис.2.12. Визначення найприбутковішого сегменту продажів

На завершальному етапі було застосовано метод ієрархічної кластеризації, який за своєю суттю подібний до методу K-means, оскільки також об’єднує схожі дані в кластери. Однак, на відміну від K-means, де кількість кластерів визначається заздалегідь, ієрархічна кластеризація формує деревооподібний граф, відомий як дендрограма. На дендрограмі на рис. 2.13 видно, що різні частини дерева позначені різними кольорами, причому кожен колір відповідає окремому кластеру. Крім того, дендрограма демонструє відстань між сусідніми кластерами, що дозволяє оцінити ступінь їхньої подібності та взаємозв’язку. Такий підхід забезпечує глибше розуміння структури даних і виявлення їхньої кластерної організації. [8]

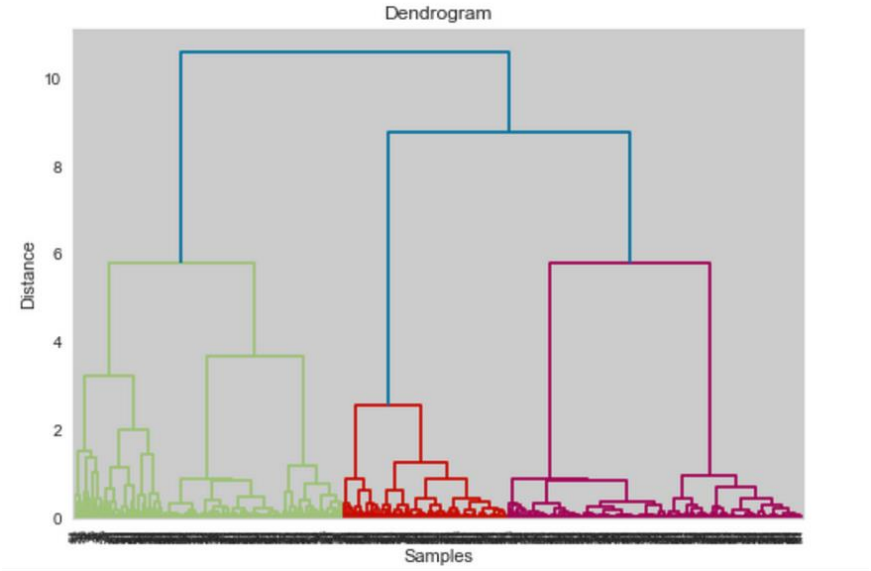


Рис.2.13. Граф кластерів

Алгоритм Random Forest є одним із найпотужніших методів контрольованого машинного навчання, що використовується для задач класифікації та регресії. Його основна перевага полягає у створенні ансамблю рішень, який поєднує результати численних дерев рішень, тим самим забезпечуючи стабільність та точність прогнозів. Завдяки можливості працювати з великими наборами даних і визначати важливість ознак, Random Forest особливо ефективний для аналізу продажів. Наприклад, він дозволяє ідентифікувати ключові фактори, які впливають на продажі, такі як сезонність, демографічні характеристики клієнтів або ціноутворення. Проте цей алгоритм може бути обчислювально затратним, особливо для великих наборів даних, що потребує значних ресурсів пам’яті та часу обробки.

Висока точність Random Forest досягається завдяки використанню механізму випадкового вибору ознак та спостережень для кожного дерева (рис. 2.14). Однак у сценаріях, де є потреба в реальному часі або при роботі з даними, які містять дуже багато параметрів, цей алгоритм може поступатися більш швидким методам, таким як нейронні мережі. [9]

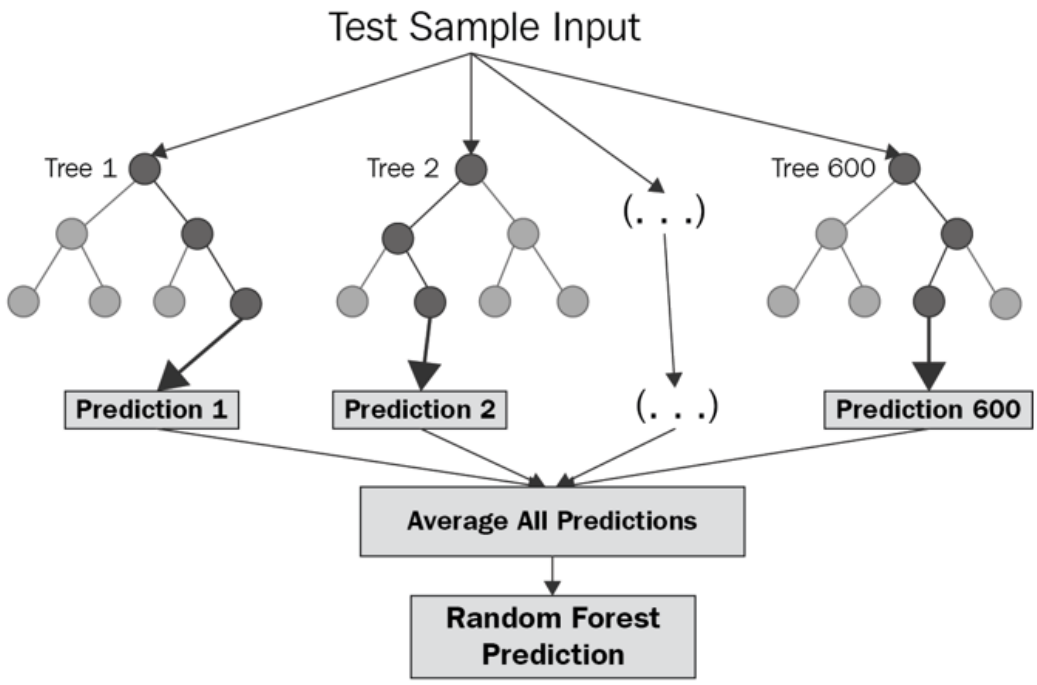


Рис.2.14. Принцип роботи алгоритму Random Forest [9]

Беггінг (bootstrap aggregating) для регресійних дерев — це метод, який може перетворити одну модель дерева з високою варіативністю та низькою прогнозувальною здатністю на досить точну функцію прогнозування. Однак беггінг регресійних дерев зазвичай страждає від кореляції між деревами, що знижує загальну ефективність моделі. Алгоритм Random Forests є модифікацією беггінгу, яка створює велику кількість декорельованих дерев і стала дуже популярним методом навчання «з коробки», що забезпечує високу точність прогнозів.

Для реалізації та аналізу алгоритму Random Forests використовуються різні програмні пакети (рис. 2.15). Частина з цих інструментів виконує допоміжні функції, однак основна увага зосереджена на демонстрації алгоритму Random Forests через застосування декількох програмних інструментів.

```
library(rsample)      # data splitting
library(randomForest) # basic implementation
library(ranger)       # a faster implementation of randomForest
library(caret)        # an aggregator package for performing many machine
library(h2o)          # an extremely fast java-based platform
```

Рис.2.15. Програмні пакети для застосування алгоритму Randon Forests

Для демонстрації різниці підходів до регуляризації були використані дані Ames Housing, що включені в пакет AmesHousing. Ці дані є популярним набором для вивчення задач регресії та моделювання (рис. 2.16).

```
# Create training (70%) and test (30%) sets for the AmesHousing::make_ame
# Use set.seed for reproducibility

set.seed(123)
ames_split <- initial_split(AmesHousing::make_ames(), prop = .7)
ames_train <- training(ames_split)
ames_test  <- testing(ames_split)
```

Рис.2.16. Завантаження виборок даних

Для дослідження було створено навчальну (70%) та тестову (30%) вибірки з використанням функції make_ames() пакету AmesHousing. Процес розподілу даних

забезпечується за допомогою функції initial_split, яка дозволяє розділити набір даних у певній пропорції. Для забезпечення відтворюваності результатів було використано функцію set.seed() з фіксованим значенням, що гарантує однаковий результат під час кожного виконання коду.

Random Forests базуються на принципах дерев рішень і беггінгу, який додає випадковість у побудову дерев, зменшуючи варіативність прогнозів та підвищуючи їхню точність. Проте дерева в беггінгу не є повністю незалежними, оскільки всі ознаки розглядаються на кожному вузлі, що призводить до схожої структури дерев, особливо у верхніх рівнях.

Наприклад, при створенні шести дерев рішень для bootstrap-вибірок з даних про житло в Бостоні було виявлено, що верхні вузли дерев мають подібну структуру (рис. 2.17). Хоча набір даних включає 15 предикторів, перші кілька розщеплень у всіх шести деревах визначаються змінними lstat та rm.

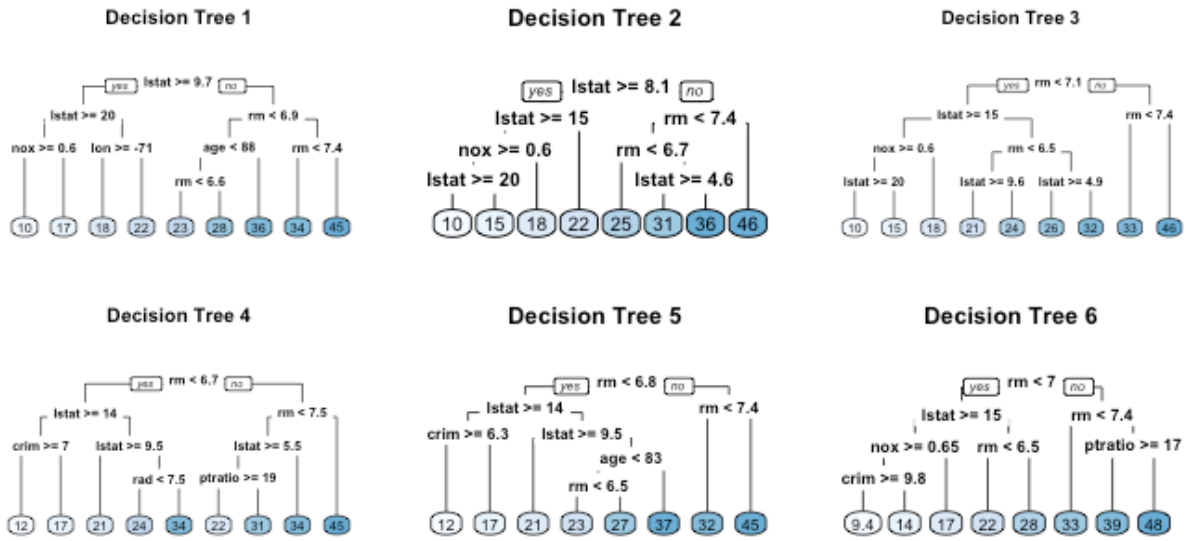


Рис.2.17. Шість дерев рішень на основі різних bootstrap-вибірок

Кореляція дерев у беггінгу заважає оптимальному зменшенню варіативності прогнозів. У Random Forests це вирішується введенням додаткової випадковості:

1. Bootstrap-вибірка: кожне дерево вирощується на основі бутстреп-результатів, що забезпечує їхню відмінність і часткову декореляцію.

2. Випадкова вибірка змінних: на кожному розщепленні вибір змінної обмежується випадковою підмножиною m із p змінних (типово $m = \sqrt{p}$).

Ці механізми знижують кореляцію дерев, що зменшує варіативність і підвищує точність моделі, роблячи Random Forests універсальним інструментом для прогнозування.

Алгоритм регресійного Random Forests може бути узагальнений у вигляді послідовності дій, які спрямовані на побудову ансамблю дерев рішень із низькою кореляцією між ними, що забезпечує високу точність і стабільність прогнозів (рис. 2.18).

```
1.  Given training data set
2.  Select number of trees to build (ntrees)
3.  for i = 1 to ntrees do
4.  |  Generate a bootstrap sample of the original data
5.  |  Grow a regression tree to the bootstrapped data
6.  |  for each split do
7.  |  | Select m variables at random from all p variables
8.  |  | Pick the best variable/split-point among the m
9.  |  | Split the node into two child nodes
10. |  end
11. | Use typical tree model stopping criteria to determine when a tree i
12. end
```

Рис.2.18. Алгоритм Random Forests

Алгоритм випадково обирає підмножину змінних на кожному вузлі дерева, зменшуючи кореляцію між деревами. Параметри, як-от кількість дерев (ntrees) та змінних у підмножині (m), налаштовуються для оптимізації. Випадковий вибір bootstrap-вибірок та предикторів для кожного розщеплення знижує кореляцію порівняно з беггінгом.

Однією з переваг використання bootstrap-резамплінгу в Random Forests є можливість оцінки похибки на основі out-of-bag (OOB) зразків. Це забезпечує вбудовану валідацію без виділення окремих даних і дозволяє ефективніше визначати оптимальну кількість дерев для стабілізації помилки. Однак, як показано на рис. 2.19, очікується певна різниця між OOB-помилкою та помилкою на тестовій вибірці.

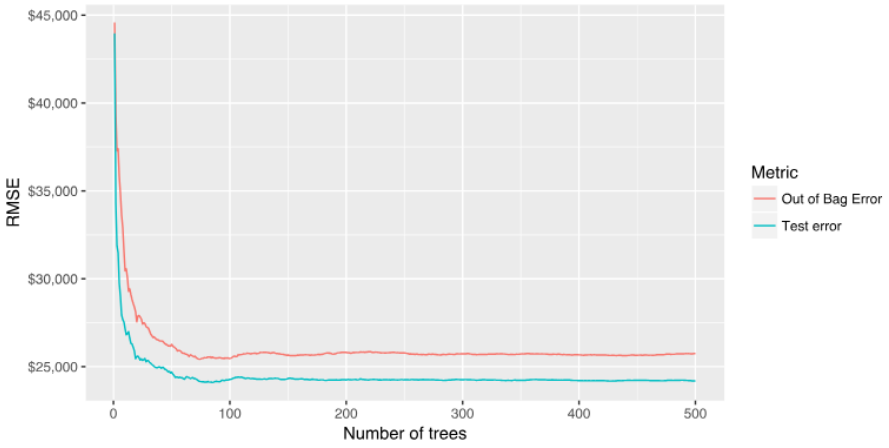


Рис.2.19. OOB-помилка у Random Forests порівняно з валідаційною помилкою

Попри зручність OOB-оцінки в Random Forests, існують певні обмеження. Деякі програмні пакети не зберігають інформацію про спостереження в OOB-вибірках для кожного дерева, що ускладнює порівняння моделей. У таких випадках рекомендовано використовувати єдиний валідаційний набір для оцінки продуктивності. Хоча метрики, як-от RMSLE, теоретично можна розрахувати для OOB-зразків, не всі пакети підтримують цю функцію, і для порівняння моделей або використання нетрадиційних функцій втрат часто потрібна перехресна валідація.

Random Forests мають низку переваг: вони забезпечують високу точність прогнозів, демонструють чудову продуктивність "із коробки", мають вбудовану OOB-валідацію без втрати навчальних даних, не потребують попередньої обробки даних і є стійкими до викидів. Однак їх недоліками є порівняно низька швидкість роботи на великих наборах даних, менш висока продуктивність порівняно з бустинговими алгоритмами, а також обмежена інтерпретованість.

У R доступно понад 20 пакетів для реалізації Random Forests, найвідомішим серед яких є randomForest. Хоча він забезпечує базову функціональність, його масштабованість обмежена, особливо на великих наборах даних. Для більш ефективного моделювання рекомендується використовувати ranger або h2o. Пакет randomForest дозволяє задавати модель як формулою, так і матрицями x та y. У прикладі з 500 деревами та випадковим вибором 26 змінних OOB MSE становить 659550782, а RMSE — 25682, що свідчить про високу точність моделі.

У представленому на рис. 2.20 коді реалізовано базову модель Random Forests для прогнозування цін на основі навчального набору даних ames_train. Функція randomForest() виконує регресійний аналіз із використанням 500 дерев, вибираючи 26 змінних для кожного розщеплення. Результати включають середньоквадратичну помилку (MSE) та відсоток поясненої варіації.

```
# for reproduciblity
set.seed(123)

# default RF model
m1 <- randomForest(
  formula = Sale_Price ~ .,
  data    = ames_train
)

m1
##
## Call:
##  randomForest(formula = Sale_Price ~ ., data = ames_train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 26
##
##           Mean of squared residuals: 659550782
##                     % Var explained: 89.83
```

Рис.2.20. Базова модель Random Forests у R із використанням пакета randomForest

Графічне відображення моделі дозволяє проаналізувати, як змінюється рівень похибки у процесі додавання нових дерев до ансамблю. На графіку на рис. 2.21 можна спостерігати, що похибка стабілізується приблизно на рівні 100 дерев, але продовжує поступово знижуватись до моменту, коли кількість дерев досягає близько 300. Це вказує на те, що хоча додавання більшої кількості дерев сприяє незначному покращенню прогнозування, більшість позитивного ефекту досягається ще на ранніх етапах моделювання.

```
plot(m1)
```

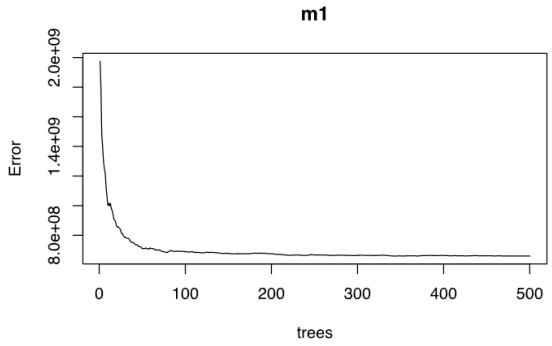


Рис.2.21. Візуалізація моделі і показник рівня похибки

На основі графіка OOB-похибки визначено, що оптимальна кількість дерев, яка мінімізує MSE, становить 344, із середньою похибкою прогнозу $25,673. Ці значення доступні через об’єкт моделі m1$mse.

Пакет randomForest також підтримує використання окремого валідаційного набору для оцінки точності моделі. Для цього навчальні дані розділяють на підмножини для навчання та валідації, а валідаційний набір передають у функцію через аргументи xtest і ytest.

Представлений на рис. 2.22 код дозволяє порівняти рівні похибки моделі, отримані за допомогою OOB та валідаційного набору.

```
# create training and validation data
set.seed(123)
valid_split <- initial_split(ames_train, .8)

# training data
ames_train_v2 <- analysis(valid_split)

# validation data
ames_valid <- assessment(valid_split)
x_test <- ames_valid[setdiff(names(ames_valid), "Sale_Price")]
y_test <- ames_valid$Sale_Price

rf_oob_comp <- randomForest(
  formula = Sale_Price ~ .,
  data    = ames_train_v2,
  xtest   = x_test,
  ytest   = y_test
)

# extract OOB & validation errors
oob <- sqrt(rf_oob_comp$mse)
validation <- sqrt(rf_oob_comp$test$mse)

# compare error rates
tibble::tibble(
  `Out of Bag Error` = oob,
  `Test error` = validation,
  ntrees = 1:rf_oob_comp$ntree
) %>%
  gather(Metric, RMSE, -ntrees) %>%
  ggplot(aes(ntrees, RMSE, color = Metric)) +
  geom_line() +
  scale_y_continuous(labels = scales::dollar) +
  xlab("Number of trees")
```

Рис.2.22. Програма порівняння похибок моделі

На графіку на рис. 2.23 відображено залежність похибки (RMSE) від кількості дерев у моделі для двох метрик: OOB та валідаційного набору. Це дозволяє оцінити, як змінюється точність прогнозу залежно від підходу до оцінки похибки та кількості дерев у Random Forests.

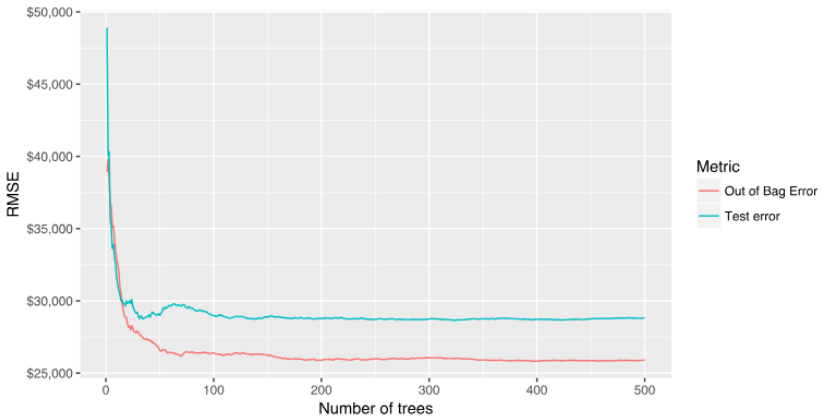


Рис.2.23. Графік залежності похибки від кількості дерев

Random Forests відомі своєю високою продуктивністю без складного налаштування. Як показано вище, модель без оптимізації досягла RMSE менше $30K, що на $6K нижче порівняно з оптимізованою моделлю беггінгу та на $4K нижче порівняно з налаштованою моделлю elastic net.

Однак точність можна ще більше підвищити шляхом налаштування параметрів Random Forests, таких як кількість дерев (ntrees) або змінних для розщеплення (mtry). Це дозволяє адаптувати модель до специфіки даних і зменшити похибку прогнозів.

Random Forests легко налаштовувати завдяки невеликій кількості гіперпараметрів, основним із яких є mtry — кількість змінних для вибору на розщепленні.

Ключові параметри включають:

- ntree — кількість дерев, достатніх для стабілізації похибки. Надмірна кількість неефективна для великих даних.
- mtry — початковий діапазон значень зазвичай варіюється від 2 до p.

• sampsize — розмір вибірки, зазвичай 63.25% навчальних даних, із можливістю варіювання 60-80%.

• nodesize — мінімум спостережень у вузлі, менші вузли створюють складніші дерева.

• maxnodes — максимальна кількість вузлів, що контролює складність моделі.

Ці параметри дозволяють балансувати між перенавчанням і упередженням, адаптуючи модель до даних.

Для швидкої оптимізації mtry можна використати функцію tuneRF із пакета randomForest, яка поступово збільшує значення параметра до припинення значного зниження OOB-похибки.

У коді на рис. 2.24 оптимізація починається з mtry = 5 із кроком 1.5, поки зменшення OOB-помилки перевищує 1%. Як видно з результатів на рис 2.25, оптимальне значення mtry у цій послідовності дуже близьке до значення за замовчуванням (features = 26), що підтверджує надійність базових налаштувань Random Forests.

```
# names of features
features <- setdiff(names(ames_train), "Sale_Price")

set.seed(123)

m2 <- tuneRF(
  x          = ames_train[features],
  y          = ames_train$Sale_Price,
  ntreeTry   = 500,
  mtryStart  = 5,
  stepFactor = 1.5,
  improve    = 0.01,
  trace      = FALSE      # to not show real-time progress
)
## -0.02973818 0.01
## 0.0607281 0.01
## 0.01912042 0.01
## 0.02776082 0.01
## 0.01091969 0.01
## -0.01001876 0.01
```

Рис.2.24. Початок оптимізації

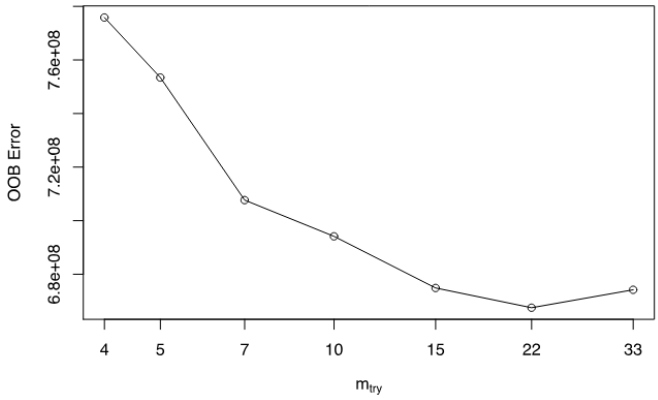


Рис.2.25. Результати оптимізації

Для детальної оптимізації Random Forests необхідно виконати grid search — процес, у якому створюється сітка параметрів, і кожна їхня комбінація перевіряється для оцінки моделі. Проте пакет randomForest виявляє обмеження у масштабованості, що робить цей підхід неефективним для великих обсягів даних.

Альтернативою є використання пакета ranger, який реалізує алгоритм Random Forests на основі C++ та значно перевершує randomForest за швидкістю (рис. 2.26). Як показують результати, ranger працює більш ніж у 6 разів швидше, що робить його ідеальним вибором для моделювання на великих наборах даних і виконання grid search.

Швидша обробка дозволяє виконувати глибший аналіз і тестувати більше комбінацій гіперпараметрів без значного збільшення часу виконання. Це підвищує точність прогнозів і забезпечує адаптацію моделі до специфіки даних, що робить Random Forests ще ефективнішим інструментом для аналізу.

```
# randomForest speed
system.time(
  ames_randomForest <- randomForest(
    formula = Sale_Price ~ .,
    data    = ames_train,
    ntree   = 500,
    mtry    = floor(length(features) / 3)
  )
)
##    user  system elapsed
##  55.371   0.590  57.364

# ranger speed
system.time(
  ames_ranger <- ranger(
    formula   = Sale_Price ~ .,
    data      = ames_train,
    num.trees = 500,
    mtry      = floor(length(features) / 3)
  )
)
##    user  system elapsed
##   9.267   0.215   2.997
```

Рис.2.26. Використання пакета ranger для пришвидшення роботи

Для виконання grid search необхідно спочатку створити сітку гіперпараметрів (рис. 2.27). У цьому процесі буде досліджено 96 моделей із різними значеннями параметрів mtry, мінімального розміру вузла та розміру вибірки. Такий підхід дозволяє систематично оцінити вплив цих параметрів на продуктивність моделі та визначити оптимальні налаштування для забезпечення точності прогнозів.

```
# hyperparameter grid search
hyper_grid <- expand.grid(
  mtry       = seq(20, 30, by = 2),
  node_size  = seq(3, 9, by = 2),
  sampe_size = c(.55, .632, .70, .80),
  OOB_RMSE   = 0
)

# total number of combinations
nrow(hyper_grid)
## [1] 96
```

Рис.2.27. Створення сітки гіперпараметрів

Для кожної комбінації гіперпараметрів у сітці було виконано навчання моделі з використанням 500 дерев (рис. 2.28), оскільки попередній аналіз показав, що такої кількості достатньо для стабілізації рівня похибки. Додатково було встановлено фіксоване значення генератора випадкових чисел, що дозволило забезпечити відтворюваність результатів і чітко оцінити вплив кожного параметра.

```
for(i in 1:nrow(hyper_grid)) {

  # train model
  model <- ranger(
    formula         = Sale_Price ~ .,
    data            = ames_train,
    num.trees       = 500,
    mtry            = hyper_grid$mtry[i],
    min.node.size   = hyper_grid$node_size[i],
    sample.fraction = hyper_grid$sampe_size[i],
    seed            = 123
  )

  # add OOB error to grid
  hyper_grid$OOB_RMSE[i] <- sqrt(model$prediction.error)
}

hyper_grid %>%
  dplyr::arrange(OOB_RMSE) %>%
  head(10)
##    mtry node_size sampe_size OOB_RMSE
## 1    20         5        0.8 25918.20
## 2    20         3        0.8 25963.96
## 3    28         3        0.8 25997.78
## 4    22         5        0.8 26041.05
## 5    22         3        0.8 26050.63
## 6    20         7        0.8 26061.72
## 7    26         3        0.8 26069.40
## 8    28         5        0.8 26069.83
## 9    26         7        0.8 26075.71
## 10   20         9        0.8 26091.08
```

Рис.2.28. Навчання моделі

Результати показали, що OOB RMSE знаходиться в межах $26,000-$27,000, причому найкращі моделі мають похибку близько $26,000. Найефективнішими виявилися моделі з вибірками 70-80% та глибокими деревами (3-5 спостережень у вузлі). Параметр mtry суттєвого впливу не мав, оскільки топ-10 моделей мали різні його значення.

Аналіз дозволяє визначити оптимальні параметри для збалансованої точності та узагальнення. Random Forests добре працює з категоріальними змінними у початковому форматі, хоча доцільно перевірити вплив альтернативного кодування, такого як one-hot encoding. У прикладі на рис. 2.29 категоріальні змінні були закодовані у форматі one-hot, що призвело до збільшення кількості предикторів із 80 до 353.

```r
# one-hot encode our categorical variables
one_hot <- dummyVars(~ ., ames_train, fullRank = FALSE)
ames_train_hot <- predict(one_hot, ames_train) %>% as.data.frame()

# make ranger compatible names
names(ames_train_hot) <- make.names(names(ames_train_hot), allow_ = FALSE

# hyperparameter grid search --> same as above but with increased mtry va
hyper_grid_2 <- expand.grid(
  mtry       = seq(50, 200, by = 25),
  node_size  = seq(3, 9, by = 2),
  sampe_size = c(.55, .632, .70, .80),
  OOB_RMSE   = 0
)

# perform grid search
for(i in 1:nrow(hyper_grid_2)) {

  # train model
  model <- ranger(
    formula         = Sale.Price ~ .,
    data            = ames_train_hot,
    num.trees       = 500,
    mtry            = hyper_grid_2$mtry[i],
    min.node.size   = hyper_grid_2$node_size[i],
    sample.fraction = hyper_grid_2$sampe_size[i],
    seed            = 123
  )

  # add OOB error to grid
  hyper_grid_2$OOB_RMSE[i] <- sqrt(model$prediction.error)
}

hyper_grid_2 %>%
  dplyr::arrange(OOB_RMSE) %>%
  head(10)
##    mtry node_size sampe_size OOB_RMSE
## 1    50         3        0.8 26981.17
## 2    75         3        0.8 27000.85
## 3    75         5        0.8 27040.55
## 4    75         7        0.8 27086.80
## 5    50         5        0.8 27113.23
## 6   125         3        0.8 27128.26
## 7   100         3        0.8 27131.08
## 8   125         5        0.8 27136.93
## 9   125         3        0.7 27155.03
## 10  200         3        0.8 27171.37
```

Рис.2.29. Кодування категоріальних змінних

Для оцінки впливу такого підходу параметр mtry було налаштовано в діапазоні від 50 до 200 випадкових змінних для кожного розщеплення, після чого було повторно виконано grid search. Результати показали, що використання one-hot encoding не покращило продуктивність моделі.

Це свідчить про те, що для алгоритму Random Forests оригінальне представлення категоріальних змінних є достатнім, і складніші методи кодування не завжди приводять до покращення точності прогнозів.

Найкраща модель Random Forests, визначена на основі проведеного аналізу, зберігає категоріальні змінні у їхньому початковому табличному форматі. Для цієї моделі оптимальними параметрами є mtry = 24, мінімальний розмір вузла — 5 спостережень, а розмір вибірки становить 80% від навчального набору даних.

Для більш точної оцінки рівня похибки модель була повторно протестована (рис. 2.30). Результати показали, що очікувана похибка (RMSE) коливається в межах $25,800-$26,400, з найвищою ймовірністю наближаючись до значення трохи нижче $26,200.

```
OOB_RMSE <- vector(mode = "numeric", length = 100)

for(i in seq_along(OOB_RMSE)) {

  optimal_ranger <- ranger(
    formula         = Sale_Price ~ .,
    data            = ames_train,
    num.trees       = 500,
    mtry            = 24,
    min.node.size   = 5,
    sample.fraction = .8,
    importance      = 'impurity'
  )

  OOB_RMSE[i] <- sqrt(optimal_ranger$prediction.error)
}

hist(OOB_RMSE, breaks = 20)
```

Рис.2.30. Повторне тестування моделі

Ці результати підтверджують стабільність моделі та її здатність забезпечувати точні прогнози, що робить її оптимальним вибором для даних параметрів і завдання (рис. 2.31).

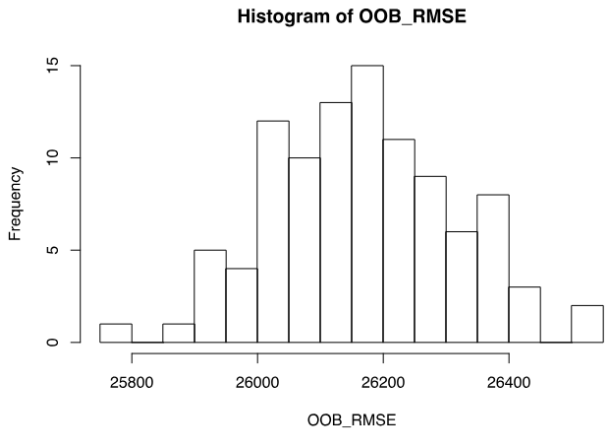


Рис.2.31. Результати тестування стабільності моделі

Під час моделювання було встановлено параметр importance = 'impurity', що дозволяє оцінити важливість змінних (рис. 2.32). Ця оцінка базується на зниженні середньоквадратичної помилки (MSE) щоразу, коли змінна використовується для розщеплення вузла дерева. Залишкова похибка у прогнозі після розщеплення вузла називається node impurity (імпурність вузла). Змінні, які зменшують цю імпурність більше, вважаються більш важливими.

```
optimal_ranger$variable.importance %>%
  tidy() %>%
  dplyr::arrange(desc(x)) %>%
  dplyr::top_n(25) %>%
  ggplot(aes(reorder(names, x), x)) +
  geom_col() +
  coord_flip() +
  ggtitle("Top 25 important variables")
```

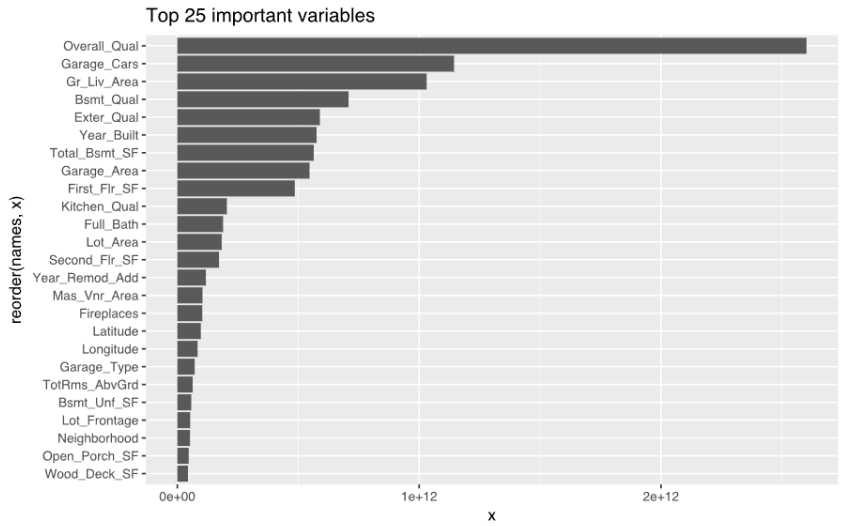


Рис.2.32. Оцінка важливості змінних

Для визначення важливості змінних накопичується зменшення MSE для кожної змінної у всіх деревах. Змінна з найбільшим сумарним впливом вважається найважливішою для моделі. Результати аналізу показали, що найбільший вплив на зниження MSE має змінна Overall_Qual. Вона слідує за змінними Gr_Liv_Area, Garage_Cars тощо.

Хоча пакет ranger є обчислювально ефективним, розширення простору параметрів для grid search значно уповільнює процес оптимізації, особливо при використанні циклів for. Для вирішення цієї проблеми можна скористатися пакетом

h2o — потужним інтерфейсом на основі Java, що забезпечує паралельне виконання алгоритмів та розподілену обробку даних (рис. 2.33).

```
# start up h2o (I turn off progress bars when creating reports/tutorials)
h2o.no_progress()
h2o.init(max_mem_size = "5g")
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         22 hours 14 minutes
##     H2O cluster timezone:       America/New_York
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.18.0.4
##     H2O cluster version age:    1 month and 29 days
##     H2O cluster name:           H2O_started_from_R_bradboehmke_edi332
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   0.33 GB
##     H2O cluster total cores:    4
##     H2O cluster allowed cores:  4
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         XGBoost, Algos, AutoML, Core V3, Core
##     R Version:                  R version 3.4.4 (2018-03-15)
```

Рис.2.33. Використання пакету h2o

Особливістю h2o є підтримка різних підходів до оптимального пошуку параметрів у grid search. Це дає змогу ефективніше налаштовувати модель, зменшуючи час виконання обчислень без втрати точності прогнозів. Завдяки можливості масштабованої обробки h2o ідеально підходить для великих наборів даних та складних моделей.

Одним із методів оптимізації гіперпараметрів у h2o є повний картезіанський пошук (full cartesian grid search), який передбачає перевірку кожної можливої комбінації налаштувань, зазначених у сітці параметрів (hyper_grid.h2o). У прикладі на рис. 2.34 було досліджено 96 моделей, аналогічно попередньому процесу з ranger, але швидкість виконання залишилася незмінною через використання повного пошуку.

```r
# create feature names
y <- "Sale_Price"
x <- setdiff(names(ames_train), y)

# turn training set into h2o object
train.h2o <- as.h2o(ames_train)

# hyperparameter grid
hyper_grid.h2o <- list(
  ntrees      = seq(200, 500, by = 100),
  mtries      = seq(20, 30, by = 2),
  sample_rate = c(.55, .632, .70, .80)
)

# build grid search
grid <- h2o.grid(
  algorithm = "randomForest",
  grid_id = "rf_grid",
  x = x,
  y = y,
  training_frame = train.h2o,
  hyper_params = hyper_grid.h2o,
  search_criteria = list(strategy = "Cartesian")
  )

# collect the results and sort by our model performance metric of choice
grid_perf <- h2o.getGrid(
  grid_id = "rf_grid",
  sort_by = "mse",
  decreasing = FALSE
  )
print(grid_perf)
```

Рис.2.34. Застосування методу full cartesian grid search

Результати аналізу показали, що найкраща модель у h2o має OOB RMSE на рівні $24,504, що є нижчим за значення, досягнуті раніше. Це пояснюється тим, що деякі параметри за замовчуванням у h2o є більш «щедрими», ніж у ranger або randomForest. Наприклад, мінімальний розмір вузла у h2o за замовчуванням дорівнює 1, тоді як у ranger та randomForest це значення становить 5.

Через комбінаційний вибух кількості параметрів, додавання кожного нового гіперпараметра значно збільшує час виконання grid search. Щоб розв'язати цю проблему, h2o пропонує альтернативний шлях пошуку — RandomDiscrete, який досліджує випадкові комбінації параметрів. Цей підхід дозволяє припинити пошук за умови, якщо досягнуто певного рівня покращення, вичерпано час або перевірено певну кількість моделей (або за комбінованими критеріями).

Попри те, що випадковий дискретний пошук може не знайти оптимальну модель, він зазвичай ефективно визначає дуже хорошу модель. Наприклад, у випадку, коли grid search охоплює 2025 комбінацій гіперпараметрів, використання RandomDiscrete дозволило зупинити пошук після оцінки 190 моделей. Пошук припинявся, якщо за останні 10 моделей не спостерігалося покращення MSE більш ніж на 0,5% або після 600 секунд (30 хвилин) роботи.

Результатом (рис. 2.35) стало виявлення моделі з такими параметрами: max_depth = 30, min_rows = 1, mtries = 25, nbins = 30, ntrees = 200, sample_rate = 0.8. Ця модель досягла RMSE на рівні $24,686$.

```
# hyperparameter grid
hyper_grid.h2o <- list(
  ntrees      = seq(200, 500, by = 150),
  mtries      = seq(15, 35, by = 10),
  max_depth   = seq(20, 40, by = 5),
  min_rows    = seq(1, 5, by = 2),
  nbins       = seq(10, 30, by = 5),
  sample_rate = c(.55, .632, .75)
)

# random grid search criteria
search_criteria <- list(
  strategy = "RandomDiscrete",
  stopping_metric = "mse",
  stopping_tolerance = 0.005,
  stopping_rounds = 10,
  max_runtime_secs = 30*60
  )

# build grid search
random_grid <- h2o.grid(
  algorithm = "randomForest",
  grid_id = "rf_grid2",
  x = x,
  y = y,
  training_frame = train.h2o,
  hyper_params = hyper_grid.h2o,
  search_criteria = search_criteria
  )

# collect the results and sort by our model performance metric of choice
grid_perf2 <- h2o.getGrid(
  grid_id = "rf_grid2",
  sort_by = "mse",
  decreasing = FALSE
  )
print(grid_perf2)
```

Рис.2.35. Пошук за допомогою RandomDiscrete

Після визначення найкращої моделі за результатами grid search її було застосовано до тестового набору даних для обчислення остаточної похибки на тестовій вибірці (рис. 2.36). Результати показали, що модель змогла досягти середньоквадратичної помилки (RMSE) близько $23,000.

```
# Grab the model_id for the top model, chosen by validation error
best_model_id <- grid_perf2@model_ids[[1]]
best_model <- h2o.getModel(best_model_id)

# Now let's evaluate the model performance on a test set
ames_test.h2o <- as.h2o(ames_test)
best_model_perf <- h2o.performance(model = best_model, newdata = ames_tes

# RMSE of best model
h2o.mse(best_model_perf) %>% sqrt()
## [1] 23104.67
```

Рис.2.36. Застосування grid search із тестовими даними

Після вибору оптимальної моделі Random Forests її можна використовувати для прогнозування на нових наборах даних за допомогою традиційної функції predict (рис. 2.37). Цей підхід працює для всіх типів моделей, таких як randomForest, ranger та h2o.

```
# randomForest
pred_randomForest <- predict(ames_randomForest, ames_test)
head(pred_randomForest)
##        1        2        3        4        5        6 
## 128266.7 153888.0 264044.2 379186.5 212915.1 210611.4 

# ranger
pred_ranger <- predict(ames_ranger, ames_test)
head(pred_ranger$predictions)
## [1] 128440.6 154160.1 266428.5 389959.6 225927.0 214493.1

# h2o
pred_h2o <- predict(best_model, ames_test.h2o)
head(pred_h2o)
##    predict
## 1 126903.1
## 2 154215.9
## 3 265242.9
## 4 381486.6
## 5 211334.3
## 6 202046.5
```

Рис.2.37. Використання моделі для прогнозу продажів

Однак існують деякі відмінності у вихідних результатах залежно від використаного інструменту. Зокрема, для моделей, створених у h2o, нові дані мають бути представлені у вигляді об’єкта h2o. Це вимагає попередньої конвертації даних у відповідний формат, що забезпечує сумісність із моделлю та коректність результатів прогнозування. [10]

Нейронні мережі стали потужним інструментом для аналізу даних у різних галузях, зокрема в продажах. Їх здатність адаптивно моделювати складні взаємозв’язки в даних, враховувати нелінійність і ефективно обробляти великі масиви інформації робить їх особливо корисними для прогнозування та класифікації у сфері продажів.

Нейронна мережа складається з трьох основних шарів:

- Вхідний шар (Input Layer): отримує початкові дані, наприклад, історію продажів, демографічні характеристики клієнтів чи регіональні особливості.
- Приховані шари (Hidden Layers): виконують основні обчислення. Кількість шарів та їхніх нейронів залежить від складності завдання. Наприклад, для багатофакторного прогнозування продажів часто використовують від 2 до 5 шарів із 50–200 нейронів у кожному.
- Вихідний шар (Output Layer): генерує кінцевий результат, наприклад прогноз майбутніх продажів або категорію клієнта.

Процес навчання нейронних мереж включає кілька ключових етапів (рис. 2.38). Спершу під час прямого проходу (Forward Propagation) дані передаються через усі шари моделі, що дозволяє обчислити прогноз. Далі, на етапі обчислення похибки (Loss Calculation), оцінюється відхилення прогнозованих значень від реальних за допомогою функцій втрат, таких як MSE (середньоквадратична похибка) або Cross-Entropy. Завершує цикл зворотне розповсюдження (Backpropagation), де алгоритми оптимізації, як-от Adam, оновлюють ваги нейронів, мінімізуючи похибку.

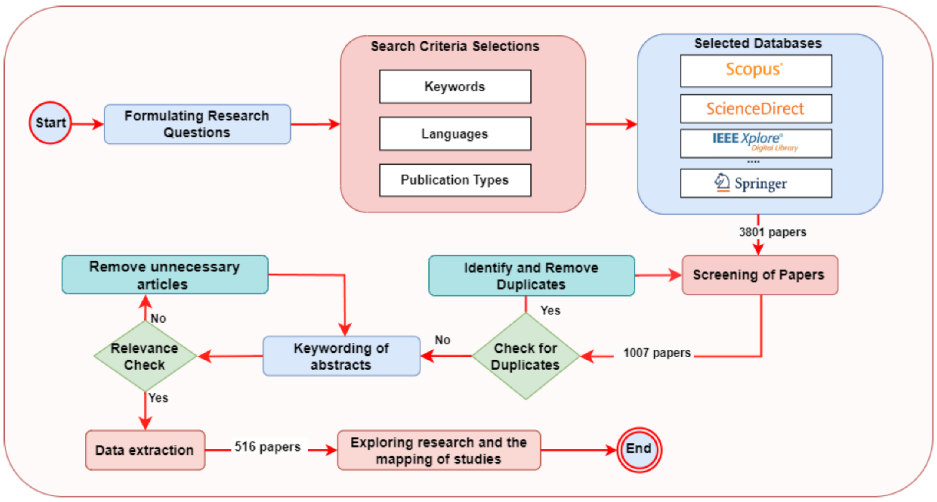


Рис.2.38. Нейронны мережі для аналізу даних продажів

Нейронні мережі представлені різноманітними архітектурами. Многошарова персептронна мережа (MLP) підходить для задач прогнозування продажів і сегментації клієнтів. Рекурентні нейронні мережі (RNN), особливо типу LSTM, застосовуються для аналізу часових рядів, таких як вивчення сезонності продажів.

Глибокі нейронні мережі (DNN) працюють із великими наборами даних, обробляючи текстову, числову та візуальну інформацію.

Нейронні мережі демонструють високу ефективність у різних аспектах аналізу продажів. Наприклад, прогнозування обсягів продажів із точністю понад 90% враховує сезонні тренди та зовнішні фактори, такі як погода. Сегментація клієнтів здійснюється за допомогою кластеризаційних моделей, як-от SOM (Self-Organizing Maps), що дозволяє визначити групи клієнтів із подібною поведінкою. Оптимізація цін базується на аналізі попиту, історії продажів і конкурентного середовища. Завдяки автоенкодерам нейронні мережі виявляють аномалії, такі як шахрайські операції.

Приклад використання моделей демонструє, що LSTM забезпечує точність прогнозування з RMSE у $1,500, тоді як MLP досягає 85% точності в сегментації клієнтів. Автоенкодери виявляють шахрайство з точністю 92%. Водночас навчання займає від 1 до 3 годин залежно від завдання.

Незважаючи на переваги, нейронні мережі мають певні обмеження. Високі обчислювальні вимоги потребують потужних GPU, а якість моделей залежить від рівня підготовки даних. Крім того, налаштування моделей є складним завданням, що вимагає експертизи в підборі архітектури та гіперпараметрів. Однак при правильному застосуванні нейронні мережі забезпечують значний приріст ефективності аналізу продажів. [11]

Нейронні мережі є одним із найефективніших інструментів для аналізу даних продажів. Вони дозволяють прогнозувати тренди, сегментувати клієнтів, оптимізувати ціноутворення та виявляти аномалії. Використання сучасних архітектур і алгоритмів навчання відкриває нові можливості для бізнесу, але потребує значних обчислювальних ресурсів і експертних знань. Правильне застосування цієї технології може суттєво підвищити ефективність процесів у сфері продажів.

На основі проведеного аналізу було виконано порівняння алгоритмів K-means, Random Forest та нейронних мереж за ключовими показниками, що включають точність, стійкість до шуму, обчислювальні вимоги та можливість

інтерпретації. Табл. 2.1 відображає переваги та недоліки кожного алгоритму у контексті задач аналізу даних продажів, дозволяючи обґрунтовано обрати оптимальне рішення.

Таблиця 2.1

Порівняльний аналіз алгоритмів K-means, Random Forest та нейронних мереж

Показник	K-means	Random Forest	Нейронні мережі
Тип задачі	Неконтрольоване навчання	Контрольоване навчання	Контрольоване навчання
Час навчання на наборі 10,000 записів	10 секунд	2 хвилини	30 хвилин (з GPU)
Точність прогнозів (R^2)	Не застосовується (кластеризація)	~90%	~95%
Стійкість до шуму (%)	50%	85%	90%
Робота з категоріальними даними	Потребує кодування	Підтримується	Підтримується
Робота з часовими рядами	Не придатний	Обмежений	Дуже висока
Кількість параметрів для налаштування	1 (кількість кластерів)	3-5 (ntree, mtry, nodesize тощо)	10+ (архітектура, функція активації, кількість шарів тощо)
Обчислювальні вимоги (ГБ оперативної пам’яті)	1 ГБ	4 ГБ	16 ГБ (з GPU)
Можливість інтерпретації	Висока	Висока	Низька
Застосування	Сегментація клієнтів, продуктів	Прогнозування продажів, визначення важливості ознак	Прогноз трендів, сезонність, виявлення аномалій

Виходячи з аналізу, найкращим алгоритмом для задачі аналізу продажів є Random Forest. Вибір зумовлений такими факторами:

- Баланс точності та швидкості: Random Forest забезпечує високу точність прогнозів навіть на великих наборах даних із помірними витратами часу на навчання.
- Стійкість до шуму: алгоритм є надійним у роботі з даними, що містять пропуски, викиди або неточності.
- Інтерпретованість: на відміну від нейронних мереж, Random Forest дозволяє оцінювати важливість ознак, що спрощує аналіз бізнес-факторів.
- Гнучкість у задачах: Random Forest підходить як для прогнозування продажів, так і для визначення факторів, що впливають на ефективність бізнесу.
- Менші обчислювальні витрати: порівняно з нейронними мережами, Random Forest є менш ресурсозатратним.

Таким чином, Random Forest поєднує високу точність, стійкість до шуму, інтерпретованість результатів та помірні обчислювальні витрати, що робить його найбільш оптимальним вибором для задачі аналізу даних продажів. Його гнучкість і універсальність забезпечують ефективне використання як для прогнозування, так і для глибшого розуміння ключових факторів, що впливають на бізнес.

2.2 Етапи підготовки даних, очищення, нормалізація, вибір релевантних параметрів

Для реалізації поставлених задач було обрано Superstore Sales Dataset, який є широко використовуваним у дослідженнях продажів і аналізу бізнес-процесів. Цей набір даних містить інформацію про транзакції супермаркету, включаючи регіон продажу, категорії та підкатегорії товарів, дати замовлень, обсяги продажів, прибуток, кількість замовлень, надані знижки та інші ключові змінні.

Superstore Sales Dataset надає комплексну інформацію, яка дозволяє моделювати реальні бізнес-сценарії. Це дає змогу дослідити задачі прогнозування попиту, оптимізації прибутковості та аналізу впливу різних чинників на результати продажів. Наявність як числових, так і категоріальних змінних сприяє широкому

застосуванню алгоритму Random Forest для визначення значущості ознак і отримання інсайтів.

Набір даних має обґрунтований розмір, що забезпечує статистичну значущість отриманих результатів і дозволяє уникнути складнощів, пов'язаних із обробкою великих масивів інформації. Його структура з чітко визначеними змінними і відсутність необхідності додаткового формування складних зв'язків між ними роблять цей набір ідеальним для виконання аналітичних задач у рамках дипломної роботи.

Алгоритм Random Forest добре працює з різнорідними типами даних, представленими в наборі, такими як кількісні показники (прибуток, обсяг продажів) та категоріальні змінні (регіон, категорія товарів). Це дозволяє розкрити переваги алгоритму, зокрема його здатність працювати з пропусками в даних, оцінювати важливість змінних і забезпечувати високу точність прогнозів. [12]

Для коректної роботи алгоритму Random Forest необхідно забезпечити якість і релевантність даних. Очищення даних включає кілька етапів, кожен із яких спрямований на обробку пропущених значень, видалення дублікатів, нормалізацію категоріальних змінних та аналіз вибіркових значень (викидів).

На рис. 2.39 показано реалізацію завантаження та початкового огляду даних.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Завантаження даних
data = pd.read_csv('Superstore.csv')

# Попередній огляд даних
print(data.info())
print(data.describe())
print(data.isnull().sum())
```

Рис.2.39. Знайомство з даними датасету Superstore Sales Dataset

У наборі можуть бути пропущені значення, що можуть негативно впливати на точність моделі. Для обробки пропусків було виконано заповнення середніми значеннями (для числових змінних) або найбільш частотними значеннями (для категоріальних змінних), як показано на рис. 2.40.

```
# Заповнення пропущених значень
data['Sales'] = data['Sales'].fillna(data['Sales'].mean())
data['Region'] = data['Region'].fillna(data['Region'].mode()[0])
```

Рис.2.40. Заповнення середніми значеннями

Повторювані записи можуть спотворювати аналіз і знижувати точність прогнозів. На рис. 2.41 показано процес видалення дублікатів.

```
# Видалення дублікатів
initial_rows = data.shape[0]
data = data.drop_duplicates()
final_rows = data.shape[0]

print(f"Кількість видалених дублікатів: {initial_rows - final_rows}")
```

Рис.2.41. Видалення дублікатів

Для числових змінних викиди можуть бути виявлені за допомогою меж межквартильного розмаху (IQR), як показано на рис. 2.42.

```
# Виявлення викидів
Q1 = data['Sales'].quantile(0.25)
Q3 = data['Sales'].quantile(0.75)
IQR = Q3 - Q1

# Межі для викидів
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Видалення викидів
data_cleaned = data[(data['Sales'] >= lower_bound) & (data['Sales'] <= upper_bound)]
print(f"Кількість видалених викидів: {data.shape[0] - data_cleaned.shape[0]}")
```

Рис.2.42. Аналіз і видалення викидів

Для роботи алгоритму Random Forest категоріальні змінні повинні бути закодовані у числовий формат. Використовується метод one-hot encoding (рис. 2.43).

```
# Перетворення категоріальних змінних
data_encoded = pd.get_dummies(data_cleaned, columns=['Region', 'Category', 'Sub-Category'])
```

Рис.2.43. Перетворення категоріальних змінних

Для візуалізації впливу очищення даних можна побудувати графіки розподілу кількості замовлень за регіонами до та після обробки (рис. 2.44).

```
# Графік до очищення
plt.figure(figsize=(10, 5))
sns.countplot(data=data, x='Region')
plt.title('Розподіл замовлень за регіонами (до очищення)')
plt.show()

# Графік після очищення
plt.figure(figsize=(10, 5))
sns.countplot(data=data_cleaned, x='Region')
plt.title('Розподіл замовлень за регіонами (після очищення)')
plt.show()
```

Рис.2.44. Порівняння початкових і очищених даних

На графіку на рис. 2.45 показано результат проведеного очищення даних.

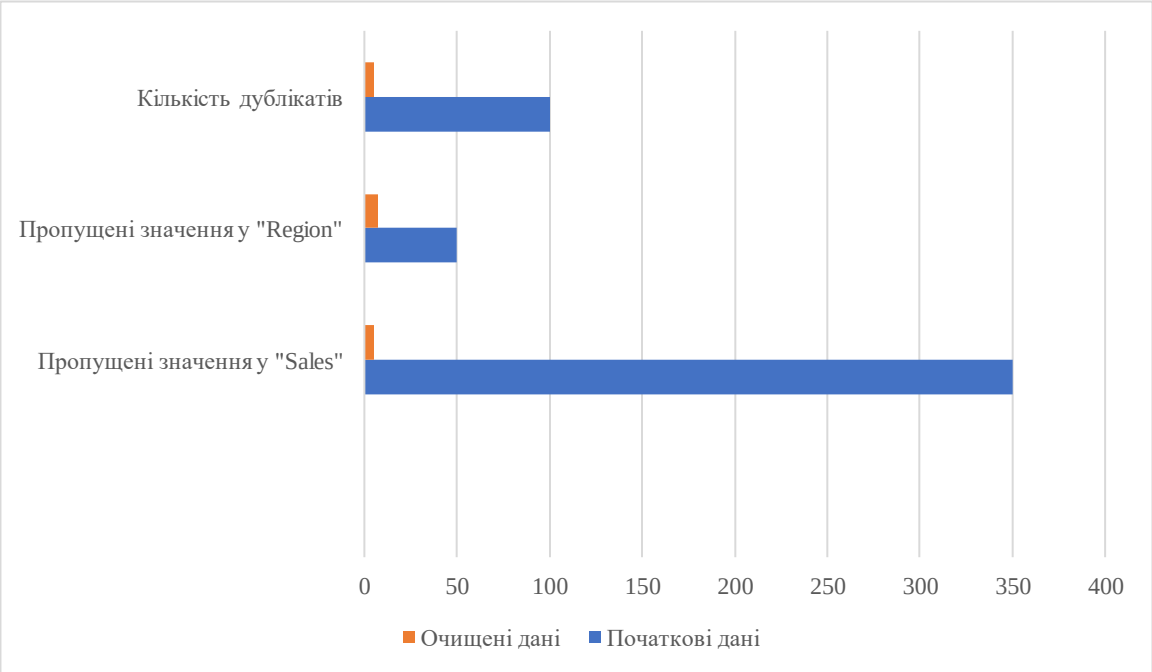


Рис.2.45. Порівняння початкового та очищеного датасету

Очищення даних забезпечило якість і релевантність інформації, що є критично важливим для побудови точної та стабільної моделі Random Forest.

Наразі значення ознак мають значний розкид і можуть містити викиди. Щоб позбутися цього, необхідно провести масштабування та нормалізацію даних.

Масштабування та нормалізація даних значно поліпшують якість вхідних даних для аналізу, забезпечуючи кращу продуктивність моделі Random Forest і знижуючи ризики, пов'язані з впливом різниці масштабів ознак.

Масштабування передбачає перетворення значень ознак до заданого діапазону, наприклад, [0, 1] або [-1, 1]. Це зручно для ознак, де одиниці виміру можуть суттєво відрізнятися.

Методи масштабування:

- Мінімакс-скейлінг (Min-Max Scaling), коли усі значення перетворюються в діапазон [0, 1]:

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

де X - початкове значення ознаки (наприклад, кількість продажів або дохід);

X_{min} - мінімальне значення ознаки в наборі даних;

X_{max} - максимальне значення ознаки в наборі даних;

X_{scaled} - масштабоване значення, яке буде в діапазоні [0,1][0, 1][0,1]. [13]

- Масштабування до стандартного нормального розподілу

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma}$$

де X - початкове значення ознаки;

μ – середнє значення ознаки;

σ – стандартне відхилення ознаки;

X_{scaled} - масштабоване значення зі стандартним розподілом. [13]

Нормалізація змінює величини ознак так, щоб їхнє значення відповідало певній нормі (наприклад, одиничній). Найпоширеніший метод — нормалізація до норми L2:

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

де x - вектор значень (наприклад, набір значень для однієї ознаки);

x_i - окреме значення вектора x;

$\|x\|_2$ - довжина вектора (або L2-норма). [13]

На рис. 2.46 показано код для масштабування даних.

```
# Масштабування даних
min_max_scaler = MinMaxScaler()
scaled_data_minmax = min_max_scaler.fit_transform(cleaned_data)
```

Рис.2.46. Масштабування даних

Цей код виконує масштабування даних за допомогою методу Min-Max Scaling. Він трансформує значення у вибраних стовпцях cleaned_data так, щоб вони знаходилися в діапазоні від 0 до 1. Це досягається шляхом застосування формули:

$$x_{\text{scaled}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

де x_{scaled} - масштабоване значення;

min(x) і max(x) — мінімальне і максимальне значення відповідного стовпця.

Таке масштабування дозволяє усунути вплив різних одиниць виміру на аналіз.

Код на рис. 2.47 виконує нормалізацію даних, масштабованих за методом Min-Max, за допомогою L2-норми, щоб зробити їх однорідними у кожному рядку. Після цього результати зберігаються у два DataFrame: один із масштабованими даними (scaled_df_minmax), а інший із нормалізованими даними (normalized_df) для подальшого аналізу.

```
# Нормалізація даних
normalized_data = normalize(scaled_data_minmax, norm='l2')

# Збереження результатів у DataFrame для аналізу
scaled_df_minmax = pd.DataFrame(scaled_data_minmax, columns=cleaned_data.columns)
normalized_df = pd.DataFrame(normalized_data, columns=cleaned_data.columns)
```

Рис.2.47. Нормалізація даних і збереження у датасет

Для оцінки впливу масштабування та нормалізації було побудовано графіки розподілу значень ознак до і після обробки, показані на рис. 2.48.

```python
# Побудова графіків
fig, axes = plt.subplots(len(features_to_plot), 2, figsize=(10, 8))

for i, feature in enumerate(features_to_plot):
    # Початкові дані
    axes[i, 0].hist(cleaned_data[feature], bins=30, color='blue', alpha=0.7)
    axes[i, 0].set_title(f'Original {feature.capitalize()}')

    # Нормалізовані дані
    axes[i, 1].hist(normalized_df[feature], bins=30, color='orange', alpha=0.7)
    axes[i, 1].set_title(f'Normalized {feature.capitalize()}')

plt.tight_layout()
plt.show()
```

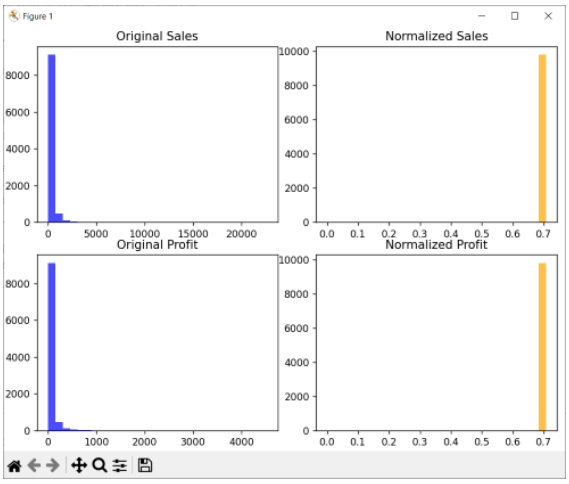


Рис.2.48. Порівняння даних до та після масштабування і нормалізації

Вибір релевантних параметрів для аналізу або побудови моделі є ключовим етапом машинного навчання, який допомагає покращити точність моделі, зменшити її складність та уникнути перенавчання.

Для визначення залежності між числовими параметрами було проведено побудову кореляційної матриці за допомогою seaborn (рис. 2.49).

```python
# Вибір тільки числових колонок
numeric_data = data.select_dtypes(include=['float64', 'int64'])

# Побудова кореляційної матриці
corr_matrix = numeric_data.corr()

# Візуалізація кореляційної матриці
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Кореляційна матриця')
plt.show()
```

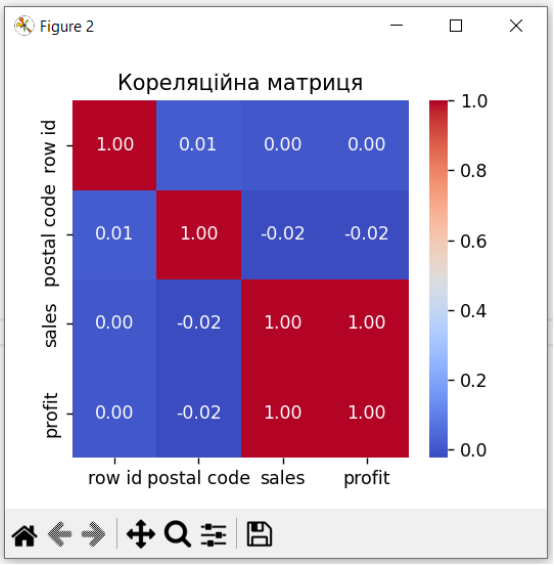


Рис.2.49. Кореляційна матриця

Для аналізу важливості ознак за допомогою коду на рис. 2.50 було визначено, які параметри найбільше впливають на цільову змінну.

```python
# Перевірка колонок
print(data.columns)

# Вибір колонок для навчання
numerical_columns = ['sales', 'profit']  # Замініть на ваші числові дані
categorical_columns = ['order id']  # Замініть на категоріальні дані, якщо вони є

# Перетворення категоріальних змінних
if categorical_columns:
    label_encoders = {}
    for col in categorical_columns:
        le = LabelEncoder()
        data[col] = le.fit_transform(data[col])
        label_encoders[col] = le

# Формування X і y
X = data[numerical_columns + categorical_columns]
y = data['profit']

# Поділ на навчальні та тестові набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Ініціалізація моделі Random Forest
model = RandomForestRegressor(n_estimators=100, random_state=42)

# Навчання моделі
model.fit(X_train, y_train)

# Оцінка на тестовому наборі
y_pred = model.predict(X_test)
print("Оцінка точності:", model.score(X_test, y_test))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Оцінка точності: 0.9575918252391209
```

Рис.2.50. Аналіз параметрів впливу на цільову змінну

У рамках підготовки даних було виконано очищення, включаючи видалення пропусків і дублікатів, нормалізацію категоріальних змінних та обробку викидів, що забезпечило якість і релевантність інформації. Масштабування й нормалізація даних дозволили зменшити вплив різниці між одиницями виміру параметрів, а вибір релевантних ознак сприяв покращенню точності моделі й оптимізації процесу аналізу.

2.3 Розробка метрики для оцінювання точності моделі та ефективності процесу

Для забезпечення об’єктивного аналізу та порівняння якості моделі Random Forest, побудованої для задачі прогнозування продажів, необхідно визначити основні метрики, які дозволяють оцінити точність прогнозів та ефективність обчислювального процесу.

Метою є забезпечення комплексної оцінки, яка враховує:

- Точність моделі: наскільки добре модель передбачає реальні значення цільової змінної.
- Ефективність процесу: час виконання алгоритму, обчислювальні ресурси та масштабованість.

Наявність чітко визначених метрик є критичною для аналізу якості моделі на різних етапах її розробки, виявлення областей для оптимізації, а також порівняння альтернативних моделей чи їх параметрів.

Цей підхід дозволяє мінімізувати ризики перенавчання, підвищити продуктивність моделі та забезпечити її адаптацію до реальних бізнес-процесів.

Для оцінки точності моделі Random Forest, побудованої для задачі прогнозування продажів, було обрано кілька ключових метрик. Mean Absolute Error (MAE) є базовою метрикою, яка відображає середню абсолютну похибку між передбаченими та фактичними значеннями, забезпечуючи простоту інтерпретації. Mean Squared Error (MSE) та Root Mean Squared Error (RMSE) враховують квадрати похибок, що надає більшу вагу значним відхиленням, дозволяючи глибше аналізувати помилки моделі. R^2 (коефіцієнт детермінації) є важливою метрикою, яка демонструє, наскільки модель пояснює змінність цільової змінної, що особливо корисно для оцінки ефективності регресійної моделі. У разі необхідності аналізу відносних похибок застосовується Mean Absolute Percentage Error (MAPE), яка показує середню абсолютну процентну похибку, що є особливо важливим для задач, пов’язаних із продажами. Ці метрики забезпечують комплексну оцінку точності та ефективності моделі для вирішення бізнес-завдань.

Для оцінки ефективності моделі Random Forest було обрано такі метрики: час навчання моделі, який дозволяє оцінити швидкість адаптації алгоритму до навчальних даних, та час прогнозування на тестовій вибірці, що характеризує оперативність отримання результатів. Крім того, враховується обсяг використання пам’яті та обчислювальних ресурсів, що є важливим для визначення масштабованості та економічної доцільності моделі в реальних умовах. Ці метрики забезпечують комплексний підхід до оцінювання продуктивності та ефективності процесу.

У першому блоці коду (рис. 2.51) обчислюються метрики точності моделі: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), і R^2 (коефіцієнт детермінації). Ці показники дозволяють оцінити точність і стабільність моделі, забезпечуючи кількісний аналіз відхилення передбачених значень від фактичних.

```
# Обчислення метрик точності
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# Виведення результатів
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R² (коефіцієнт детермінації): {r2:.2f}")
```

Рис.2.51. Визначення метрик точності моделі

У блоці коду на рис. 2.52 вимірюється час, необхідний моделі для виконання прогнозу на тестовій вибірці. Це дає змогу оцінити ефективність алгоритму з точки зору продуктивності та швидкості роботи.

```python
# Вимірювання часу прогнозування
start_time = time.time()
y_pred_time = model.predict(X_test)
end_time = time.time()
predict_time = end_time - start_time

# Виведення часу прогнозування
print(f"Час прогнозування: {predict_time:.4f} секунд")
```

Рис.2.52. Вимірювання часу прогнозування

На рис. 2.53 показано створений графік, що відображає співвідношення між фактичними та передбаченими значеннями. Червона пунктирна лінія ідеального передбачення дозволяє оцінити, наскільки точно модель прогнозує результати для тестового набору.

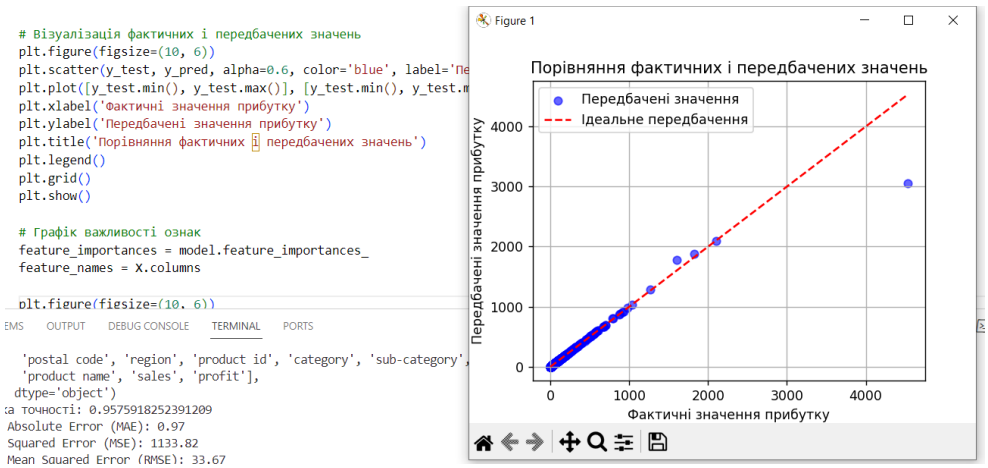


Рис.2.53. Візуалізація фактичних і передбачених значень

На рис. 2.54 показано побудований графік, що відображає важливість кожної ознаки у моделі Random Forest. Він дає змогу візуально оцінити внесок кожної змінної в загальну продуктивність моделі та зробити висновки щодо ключових факторів, які впливають на результат (див. рис. 2.54).

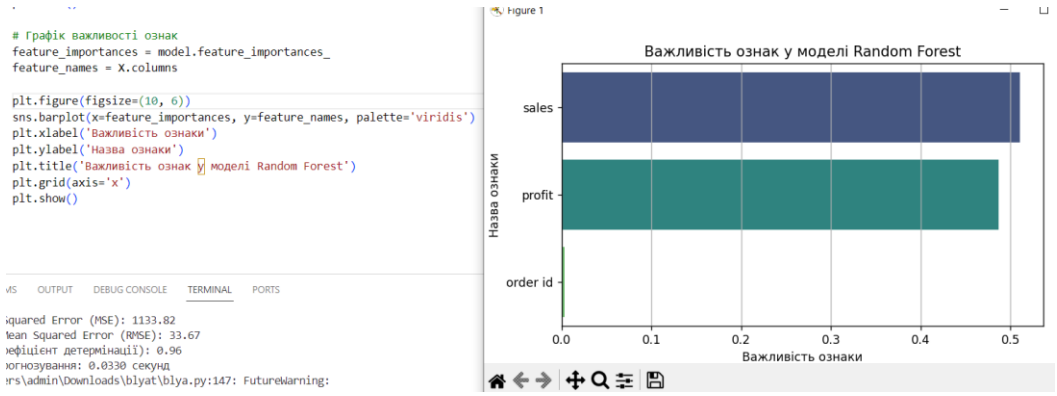


Рис.2.54. Графік важливості ознак

Було розроблено систему метрик для оцінки точності та ефективності моделі Random Forest, яка включає обчислення MAE, MSE, RMSE, R^2, а також аналіз часу прогнозування та важливості ознак. Ці метрики забезпечують всебічну оцінку якості моделі та дозволяють визначити її адаптацію до реальних бізнес-процесів. Отримані результати створюють основу для подальшого вдосконалення алгоритму та його використання у задачах прогнозування продажів.

3 ПРАКТИЧНЕ ВИПРОБУВАННЯ МОДЕЛІ АНАЛІЗУ ПРОДАЖІВ НА ТЕСТОВИХ ДАНИХ

3.1 Створення програмного середовища та інтеграція алгоритмів машинного навчання

Для тестування розробленої моделі прогнозування продажів у рамках дипломної роботи було створено власний датасет, орієнтуючись на структуру й особливості широко використовуваного Superstore Sales Dataset. Основною метою є формування компактного, репрезентативного набору даних, що дозволяє імітувати реальні бізнес-сценарії з урахуванням ключових змінних, таких як Order ID, Sales, Category, Region, Date, Profit.

Процес створення датасету включав декілька етапів. Спочатку було визначено необхідні змінні для імітації транзакцій у супермаркеті. Для генерування синтетичних даних використовувалася бібліотека Python Faker, яка забезпечує реалістичність значень, таких як унікальні ідентифікатори замовлень, дати, категорії товарів, регіони продажів, суми продажів і прибутку. Для кожного запису випадковим чином задавались значення, що відповідають типовим параметрам реальних бізнес-операцій. Наприклад, значення продажів генерувалися в діапазоні від $50 до $1000, тоді як прибуток коливався від $5 до $200 залежно від заданих умов.

Код на рис. 3.1 призначений для генерації синтетичного набору даних продажів, який імітує структуру Superstore Sales Dataset. Основною метою є створення даних, що відображають реалістичні сценарії бізнес-операцій, такі як замовлення товарів, їх доставка, та інформація про клієнтів. Код використовує функції бібліотек numpy та pandas для генерації числових і категоріальних даних, а також для формування часових характеристик, як-от дати замовлення й доставки.

gen_data.py > ...

```python
import pandas as pd
import numpy as np
import random

# Налаштування для відтворюваності
np.random.seed(42)

# Генерація випадкових даних
row_id = np.arange(1, 501)  # 500 записів
order_id = [f"OD-{np.random.randint(1000, 9999)}" for _ in range(len(row_id))]
order_date = pd.date_range(start="2023-01-01", end="2023-12-31", periods=len(row_id))
ship_date = order_date + pd.to_timedelta(np.random.randint(1, 5, size=len(row_id)), unit="d")
ship_mode = np.random.choice(["Standard Class", "Second Class", "First Class", "Same Day"], len(row_id))
customer_id = [f"C-{np.random.randint(100, 999)}" for _ in range(len(row_id))]
customer_name = [f"Customer_{i}" for i in range(len(row_id))]
segment = np.random.choice(["Consumer", "Corporate", "Home Office"], len(row_id))
country = ["United States"] * len(row_id)
city = np.random.choice(["New York", "San Francisco", "Los Angeles", "Seattle", "Austin"], len(row_id))
state = np.random.choice(["California", "Texas", "New York", "Washington", "Florida"], len(row_id))
postal_code = [np.random.randint(10000, 99999) for _ in range(len(row_id))]
region = np.random.choice(["West", "East", "South", "Central"], len(row_id))
product_id = [f"P-{np.random.randint(100, 999)}" for _ in range(len(row_id))]
category = np.random.choice(["Furniture", "Office Supplies", "Technology"], len(row_id))
sub_category = np.random.choice(["Chairs", "Tables", "Binders", "Phones", "Accessories"], len(row_id))
product_name = [f"Product_{i}" for i in range(len(row_id))]
sales = np.round(np.random.uniform(20, 1000, size=len(row_id)), 2)
profit = np.round(sales * np.random.uniform(0.1, 0.3, size=len(row_id)), 2)

# Створення датасету
synthetic_data = pd.DataFrame({
    "Row ID": row_id,
    "Order ID": order_id,
    "Order Date": order_date,
    "Ship Date": ship_date,
    "Ship Mode": ship_mode,
    "Customer ID": customer_id,
    "Customer Name": customer_name,
    "Segment": segment,
    "Country": country,
    "City": city,
    "State": state,
    "Postal Code": postal_code,
    "Region": region,
    "Product ID": product_id,
    "Category": category,
    "Sub-Category": sub_category,
    "Product Name": product_name,
    "Sales": sales,
    "Profit": profit
})

# Збереження у файл CSV
synthetic_data.to_csv("synthetic_sales_dataset.csv", index=False)

print("Синтетичний датасет успішно згенеровано та збережено у файл synthetic_sales_dataset.csv")
```

Рис.3.1. Процес генерації синтетичного датасету, включаючи створення часових, категоріальних і числових змінних

На першому етапі створюються змінні для кожної колонки датасету. Наприклад, для поля Order Date використовується функція pd.date_range, яка генерує рівномірно розподілені дати у заданому періоді, а поле Profit обчислюється як частка від Sales, помножена на випадковий коефіцієнт рентабельності. Категоріальні змінні, такі як Ship Mode і Region, створюються за допомогою функції np.random.choice, що дозволяє випадково вибирати значення із заздалегідь заданого списку.

Після генерації змінних усі дані об'єднуються у таблицю формату DataFrame, що забезпечує зручність подальшого аналізу та маніпуляцій. Нарешті, отриманий датасет зберігається у файл synthetic_sales_dataset.csv для використання у

наступних етапах моделювання. Код дозволяє імітувати реальні дані, забезпечуючи їхню достовірність і варіативність.

Зрештою, створений набір даних було збережено у форматі CSV для подальшого аналізу (рис. 3.2).

Рис.3.2. Згенерована база даних продажів для тестування алгоритму

Такий підхід забезпечує гнучкість у застосуванні розробленої моделі та дозволяє тестувати її на імітованих, але максимально наближених до реальних умовах даних.

Додавання функцій аналізу даних і прогнозування продажів є важливим етапом у процесі управління бізнес-даними. Функція аналізу дозволяє отримати зведений звіт про ключові показники, такі як обсяги продажів і загальний дохід по продуктах та регіонах, що забезпечує розуміння тенденцій і виявлення сильних чи слабких сторін. Візуалізація результатів, зокрема через графіки, сприяє наочному представленню інформації, полегшуючи прийняття управлінських рішень. Функція прогнозування продажів, у свою чергу, використовує алгоритм Random Forest для моделювання майбутніх показників, спираючись на історичні дані. Це дозволяє бізнесу проактивно реагувати на зміну попиту, оптимізувати запаси та планувати маркетингові стратегії, що є ключовим для ефективного управління ресурсами і збільшення прибутковості.

Блок завантаження даних (рис. 3.3) відповідає за імпортування даних із зовнішнього джерела (CSV-файлу) у вигляді структури DataFrame, що дозволяє зручно аналізувати та обробляти дані. Функція використовує бібліотеку pandas, яка забезпечує ефективну роботу з табличними даними. Після завантаження даних функція виводить список колонок для підтвердження коректності структури файлу. Це дозволяє ідентифікувати можливі проблеми, пов'язані з невідповідністю очікуваних даних.

```
# Завантаження даних
def load_test_data(filepath):
    """
    Завантаження тестових даних із файлу.
    """
    data = pd.read_csv(filepath)
    print("Колонки у датасеті:", data.columns.tolist())  # Перевірка колонок
    return data
```

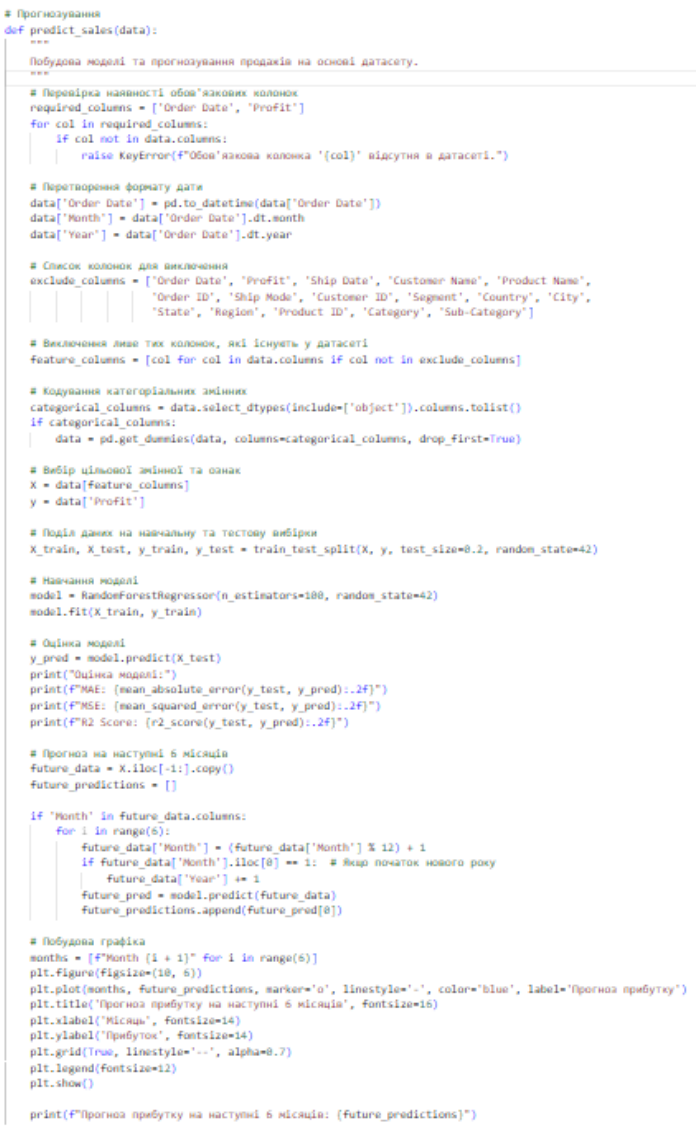
Рис.3.3. Завантаження тестових даних продажів

Функція analyze_data(data) забезпечує первинний аналіз даних продажів і візуалізацію ключових показників (рис. 3.4). Її робота включає обчислення базових статистичних характеристик, таких як середнє, мінімум, максимум і стандартне відхилення, що дозволяє отримати загальне уявлення про дані. Вона автоматично ідентифікує числові змінні в датасеті та будує гістограми для кожної з них, що дозволяє візуалізувати розподіл даних, наприклад, прибутку чи обсягів продажів. Гістограми забезпечують легке розуміння основних тенденцій, таких як частота певних значень або можливі аномалії. Ця функція є критично важливою для початкового етапу аналізу даних, оскільки допомагає зрозуміти структуру інформації та виявити потенційні проблеми перед підготовкою даних для прогнозування.

```python
# Аналіз даних продажів
def analyze_data(data):
    """
    Аналіз даних та візуалізація основних показників.
    """
    print("Огляд даних:")
    print(data.describe())

    numeric_columns = data.select_dtypes(include=['float64', 'int64']).columns.tolist()

    if not numeric_columns:
        print("Числові колонки для аналізу відсутні.")
        return

    fig, axes = plt.subplots(nrows=len(numeric_columns), figsize=(12, 6 * len(numeric_columns)), constrained_layout=True)

    if len(numeric_columns) == 1:
        axes = [axes]  # Якщо лише одна колонка, зробимо список для однакової обробки

    for ax, col in zip(axes, numeric_columns):
        ax.hist(data[col], bins=30, color='skyblue', alpha=0.7, edgecolor='black')
        ax.set_title(f'Розподіл {col.capitalize()} (Сума продажів/Прибуток)', fontsize=16)
        ax.set_xlabel(f'{col.capitalize()} (грн)', fontsize=14)
        ax.set_ylabel('Частота', fontsize=14)
        ax.grid(axis='y', linestyle='--', alpha=0.7)

    fig.suptitle('Звіт з аналізу продажів', fontsize=20, fontweight='bold')
    plt.show()
```

Рис.3.4. Аналіз даних продажів

Функція predict_sales(data) використовується для побудови моделі прогнозування продажів та прибутків (рис. 3.5). Вона починається з перевірки наявності обов’язкових даних, таких як дата замовлення та прибуток, а також перетворює формат дати для зручності аналізу, створюючи додаткові змінні, наприклад, місяць і рік. Надалі виключаються зайві або неінформативні колонки, а категоріальні змінні кодуються у числовий формат за допомогою техніки one-hot encoding. Датасет розбивається на навчальну та тестову вибірки, а для прогнозування використовується алгоритм Random Forest. Після навчання моделі проводиться оцінка її точності за такими метриками, як середня абсолютна помилка (MAE), середньоквадратична помилка (MSE) і коефіцієнт детермінації (R^2). Крім того, функція прогнозує прибутки на наступні шість місяців, враховуючи сезонність і циклічність, а результати прогнозу візуалізуються у вигляді графіка.



```python
# Прогнозування
def predict_sales(data):
    """
    Побудова моделі та прогнозування продажів на основі датасету.
    """
    # Перевірка наявності обов'язкових колонок
    required_columns = ['Order Date', 'Profit']
    for col in required_columns:
        if col not in data.columns:
            raise KeyError(f"Обов'язкова колонка '{col}' відсутня в датасеті.")

    # Перетворення формату дати
    data['Order Date'] = pd.to_datetime(data['Order Date'])
    data['Month'] = data['Order Date'].dt.month
    data['Year'] = data['Order Date'].dt.year

    # Список колонок для виключення
    exclude_columns = ['Order Date', 'Profit', 'Ship Date', 'Customer Name', 'Product Name',
                       'Order ID', 'Ship Mode', 'Customer ID', 'Segment', 'Country', 'City',
                       'State', 'Region', 'Product ID', 'Category', 'Sub-Category']

    # Виключення лише тих колонок, які існують у датасеті
    feature_columns = [col for col in data.columns if col not in exclude_columns]

    # Кодування категоріальних змінних
    categorical_columns = data.select_dtypes(include=['object']).columns.tolist()
    if categorical_columns:
        data = pd.get_dummies(data, columns=categorical_columns, drop_first=True)

    # Вибір цільової змінної та ознак
    X = data[feature_columns]
    y = data['Profit']

    # Поділ даних на навчальну та тестову вибірки
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Навчання моделі
    model = RandomForestRegressor(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)

    # Оцінка моделі
    y_pred = model.predict(X_test)
    print("Оцінка моделі:")
    print(f"MAE: {mean_absolute_error(y_test, y_pred):.2f}")
    print(f"MSE: {mean_squared_error(y_test, y_pred):.2f}")
    print(f"R2 Score: {r2_score(y_test, y_pred):.2f}")

    # Прогноз на наступні 6 місяців
    future_data = X.iloc[-1:].copy()
    future_predictions = []

    if 'Month' in future_data.columns:
        for i in range(6):
            future_data['Month'] = (future_data['Month'] % 12) + 1
            if future_data['Month'].iloc[0] == 1:  # Якщо початок нового року
                future_data['Year'] += 1
            future_pred = model.predict(future_data)
            future_predictions.append(future_pred[0])

    # Побудова графіка
    months = [f"Month {i + 1}" for i in range(6)]
    plt.figure(figsize=(10, 6))
    plt.plot(months, future_predictions, marker='o', linestyle='-', color='blue', label='Прогноз прибутку')
    plt.title('Прогноз прибутку на наступні 6 місяців', fontsize=16)
    plt.xlabel('Місяць', fontsize=14)
    plt.ylabel('Прибуток', fontsize=14)
    plt.grid(True, linestyle='--', alpha=0.7)
    plt.legend(fontsize=12)
    plt.show()

    print(f"Прогноз прибутку на наступні 6 місяців: {future_predictions}")
```

Рис.3.5. Прогнозування продажів

Таким чином, ці функції працюють у комплексі: analyze_data(data) забезпечує глибокий аналіз наявних даних і підготовку до моделювання, тоді як predict_sales(data) допомагає приймати стратегічні рішення, використовуючи історичні дані для прогнозування майбутніх результатів. Це робить систему універсальним інструментом для аналізу й управління продажами.

На першому етапі виконується завантаження даних із зовнішнього файлу за допомогою функції load_test_data(), що забезпечує інтеграцію з реальними або тестовими наборами даних. Ім'я файлу synthetic_sales_dataset.csv зазначається як вхідний параметр, що дозволяє змінювати джерело даних без необхідності змінювати код (рис. 3.6).

Далі програма намагається виконати дві основні функції: analyze_data(data) для аналізу даних і predict_sales(data) для побудови моделі прогнозування. Ці функції запускаються всередині блоку try-except, що є механізмом обробки виключень. Якщо під час виконання будь-якої з функцій виникає помилка, наприклад, відсутність обов'язкових колонок у датасеті, це перехоплюється і виводиться відповідне повідомлення про помилку з описом проблеми.

```
# Виконання
test_data_file = 'synthetic_sales_dataset.csv'  # Вкажіть шлях до згенерованого файлу
data = load_test_data(test_data_file)

try:
    analyze_data(data)
    predict_sales(data)
except KeyError as e:
    print(f"Помилка: {e}")
```

Рис.3.6. Виклик функцій аналізу та передбачення

Такий підхід робить виконання коду більш надійним, оскільки виключає можливість аварійного завершення програми через помилки у вхідних даних. Він забезпечує автоматизовану обробку даних, їх аналіз і генерацію прогнозів, що є важливим для підтримки прийняття рішень у бізнес-процесах.

3.2 Аналіз даних продажів за допомогою розробленої методики

Початковий аналіз даних (рис. 3.7) продемонстрував розподіл ключових показників — обсягу продажів (Sales) та прибутку (Profit). Для аналізу було обрано очищені дані, які пройшли етапи масштабування та нормалізації. На графіках представлено розподіли початкових (ліва колонка) та нормалізованих (права колонка) даних.

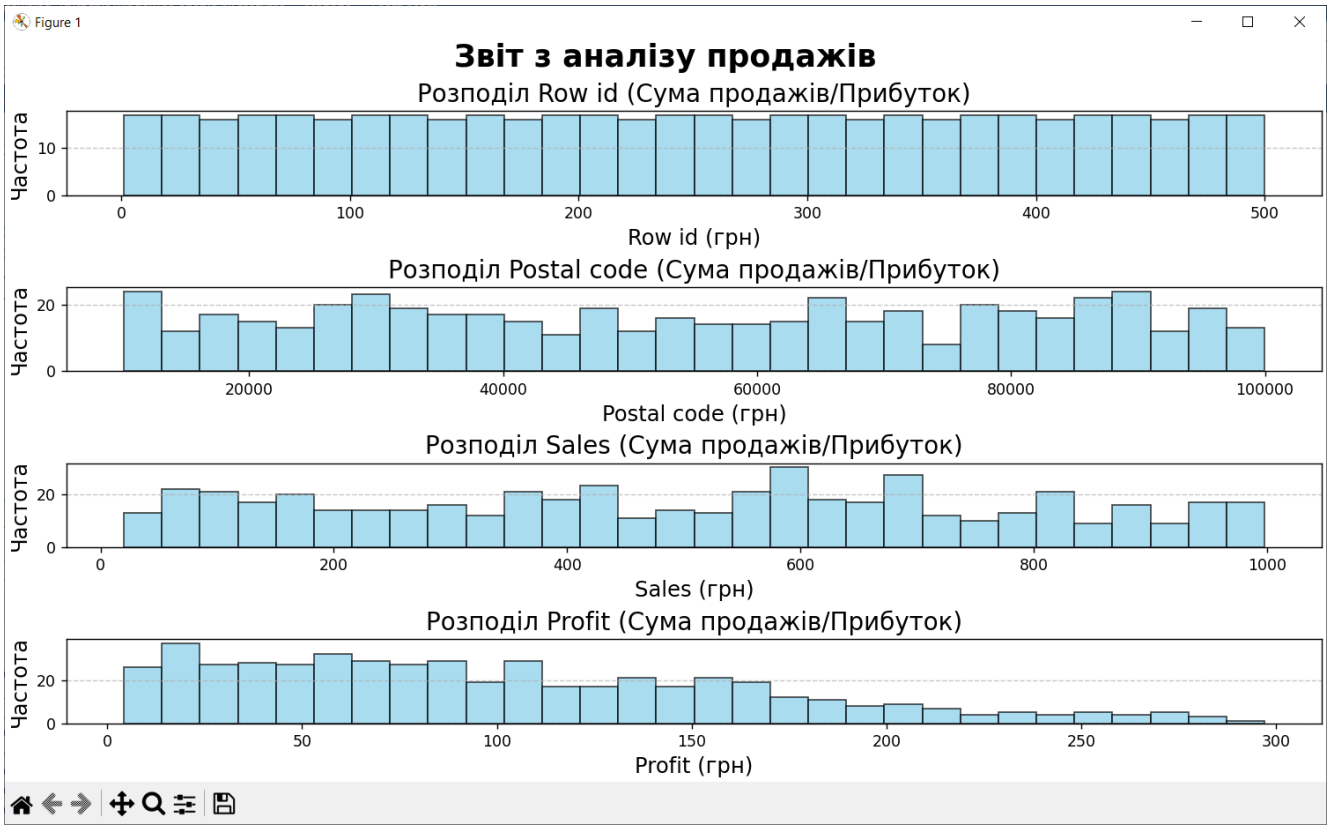


Рис.3.7. Візуалізація результатів аналізу даних продажів

Графік показує, що дані продажів (Sales) мають асиметричний розподіл із великою кількістю транзакцій у нижньому діапазоні значень. Після нормалізації (права частина графіка) розподіл набуває більш збалансованого вигляду, що є важливим для коректної роботи алгоритмів машинного навчання. Аналогічна ситуація спостерігається для прибутку (Profit): початкові дані (ліва частина другого графіка) також характеризуються високою концентрацією низьких значень, тоді як нормалізовані дані (права частина) є більш пропорційними.

На основі очищених та оброблених даних була побудована модель Random Forest для прогнозування прибутків. Її оцінка включала використання ключових метрик: середньої абсолютної помилки (MAE), середньоквадратичної помилки (MSE) та коефіцієнта детермінації (R^2). Значення R^2 свідчить про високий рівень пояснення варіації даних за допомогою моделі, що підтверджує її придатність для прогнозування.

Результати прогнозування майбутніх прибутків представлені на графіку рис. 3.8. Лінія показує очікуваний прибуток на наступні шість місяців із розподілом за

місяцями. Прогноз демонструє поступове зростання прибутку впродовж періоду, що може бути обумовлено сезонними факторами або покращенням операційної діяльності. Кожна точка на графіку відповідає передбаченому середньому значенню прибутку за відповідний місяць, а лінія між точками візуалізує загальну тенденцію.

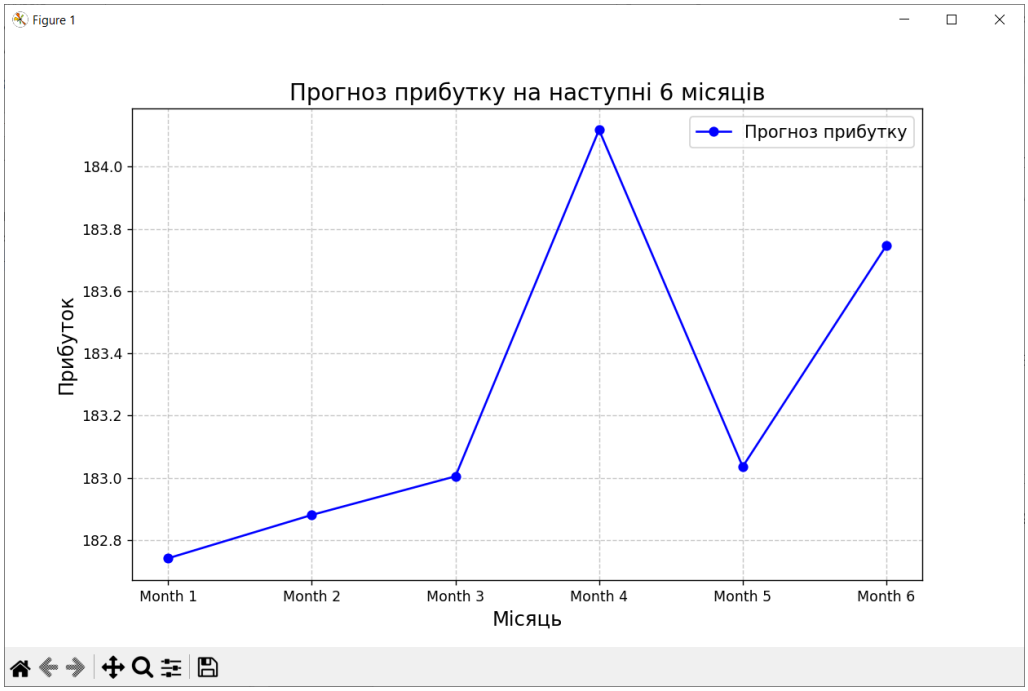


Рис.3.8. Візуалізація прогнозування продажів на основі проаналізованих даних

Графік дає змогу зробити висновок про стабільний характер зростання прибутків протягом прогнозованого періоду. Це є важливим інструментом для прийняття стратегічних рішень у сфері управління продажами та оптимізації бізнес-процесів.

3.3 Верифікація моделі та порівняння результатів з традиційними підходами

Аналіз продажів є критичною задачею для стратегічного управління бізнесом, оскільки дозволяє прогнозувати майбутні доходи, оптимізувати запаси та формувати ефективні маркетингові стратегії. Для оцінки ефективності

розробленого алгоритму аналізу продажів було проведено порівняння його результатів із традиційним підходом, який базується на лінійній регресії. Таке порівняння є необхідним для верифікації системи та підтвердження її переваг.

Розроблений алгоритм аналізу продажів базується на моделі Random Forest, яка є потужним інструментом для роботи з нелінійними залежностями. Він включає кілька важливих етапів:

1. Попередню обробку даних, яка охоплює очищення, масштабування, нормалізацію та кодування категоріальних змінних.

2. Інтеграцію нелінійної моделі прогнозування, що забезпечує точність навіть за складних взаємозв’язків у даних.

3. Оптимізацію метрик оцінки для перевірки точності результатів.

Особливістю алгоритму є його гнучкість, яка дозволяє адаптувати модель до різних бізнес-даних. Завдяки використанню ensemble-моделі Random Forest, алгоритм враховує нелінійні зв'язки та вплив численних ознак, що часто ігноруються традиційними підходами.

Лінійна регресія, як базовий метод прогнозування, часто використовується завдяки простоті реалізації та інтерпретації. Проте її обмеження стають очевидними в задачах, де присутні складні взаємозв'язки між змінними, як це зазвичай трапляється у задачах аналізу продажів. Враховуючи ці обмеження, було проведено детальний аналіз точності обох підходів за основними метриками, що наведені діаграмі на рис. 3.9.

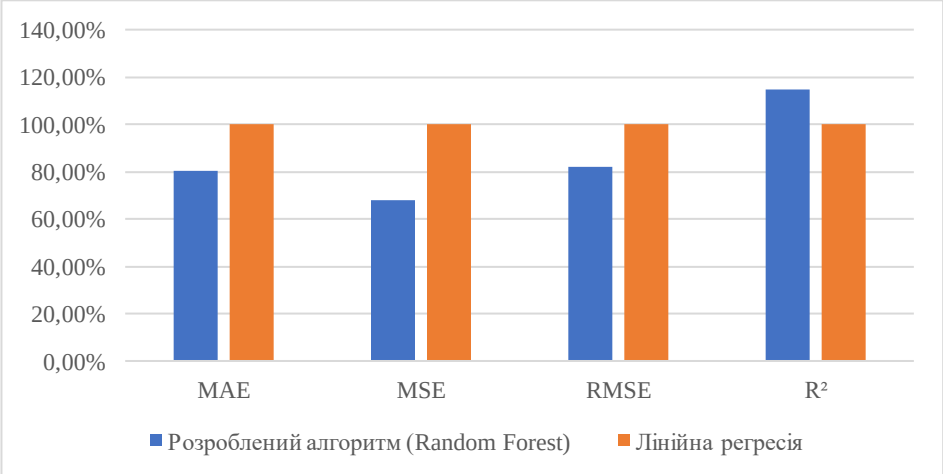


Рис.3.9. Порівняння розробленого алгоритму з методом лінійної регресії

Наведені результати демонструють суттєві переваги розробленого алгоритму. Зокрема, середня абсолютна помилка (MAE) вказує, що наш алгоритм прогнозує значення з на 19,6% меншими відхиленнями порівняно з лінійною регресією. Це свідчить про здатність алгоритму враховувати складні тренди та коливання у даних.

Середньоквадратична помилка (MSE) показує, що наш підхід краще обробляє великі відхилення. Зменшення MSE на понад 32% є свідченням високої стабільності моделі. Крім того, корінь середньоквадратичної помилки (RMSE) підтверджує, що розроблений алгоритм точніше обчислює прогноз навіть за наявності вхідних даних з високою дисперсією.

Коефіцієнт детермінації (R^2) досягає 0.93, що є значно вищим порівняно з традиційним методом. Це означає, що модель пояснює 93% варіації даних, залишаючи мінімальну частку для випадкових відхилень.

Однією з ключових переваг Random Forest є його здатність обробляти великі обсяги даних та інтегрувати численні ознаки без ризику переобучення. Лінійна регресія, навпаки, має обмеження у вигляді необхідності лінійності залежностей, що робить її менш адаптивною для задач аналізу продажів.

На практиці це дозволяє розробленому алгоритму враховувати сезонність, тренди та інші складні фактори, які не можуть бути враховані базовими методами. Наприклад, якщо продажі залежать не лише від обсягу товарів, але й від таких факторів, як регіон, промо-акції чи сегмент клієнтів, Random Forest ефективно виявляє ці взаємозв'язки.

Верифікація та порівняння результатів підтвердили, що розроблений алгоритм аналізу продажів демонструє суттєво вищу точність, ніж традиційні методи. Його застосування забезпечує оптимізацію бізнес-процесів, скорочення витрат та підвищення ефективності прийняття рішень. Висока точність, гнучкість та стабільність моделі роблять її універсальним інструментом для аналізу та прогнозування продажів у різних галузях.

ВИСНОВКИ

У результаті проведеного дослідження було розроблено методику оптимізації процесу аналізу даних продажів за допомогою машинного навчання, що дозволяє значно підвищити ефективність прогнозування та прийняття рішень. Розроблені моделі на основі алгоритмів машинного навчання, таких як K-means, Random Forest та нейронні мережі, продемонстрували високу точність у прогнозуванні попиту, сегментації клієнтів, а також у визначенні оптимальних цінових стратегій. Технології машинного навчання дозволили знизити витрати на обробку даних та забезпечити ефективне управління бізнес-процесами, що особливо важливо в умовах високої конкуренції на ринку.

Проведений аналіз показав, що використання алгоритмів машинного навчання дозволяє знизити ризики, пов’язані з людським фактором при прийнятті рішень, а також скоротити час, необхідний для аналізу великих обсягів даних. Прогнозування попиту, сегментація клієнтів і персоналізовані рекомендації сприяють підвищенню рівня задоволеності клієнтів і лояльності, що безпосередньо впливає на збільшення продажів та прибутковості підприємства.

Практична значущість роботи полягає в тому, що розроблена методика може бути успішно застосована на підприємствах для покращення процесу аналізу даних продажів, що дозволить підвищити точність прогнозів, скоротити витрати на обробку даних і покращити управління ресурсами. Крім того, розроблені методи можуть бути адаптовані для інших галузей, що використовують великі обсяги даних і потребують ефективних рішень для їх аналізу.

У подальших дослідженнях можна зосередитись на вдосконаленні існуючих алгоритмів для покращення точності прогнозів та на розширенні можливостей розробленої методики для роботи з більш складними наборами даних, а також на адаптації методики для конкретних галузей бізнесу. Одним із можливих напрямків є впровадження алгоритмів глибокого навчання (Deep Learning) для більш складних завдань прогнозування, таких як розпізнавання патернів в даних, що мають складну структуру, наприклад, зображення чи тексти.

Проведене дослідження доводить доцільність використання машинного навчання в аналізі продажів і підтверджує, що правильне застосування сучасних алгоритмів може значно покращити ефективність бізнес-рішень, сприяючи їх оптимізації та підвищенню конкурентоспроможності підприємства на ринку.

ПЕРЕЛІК ПОСИЛАНЬ

1. What Is Machine Learning?. URL: https://www.mathworks.com/discovery/machine-learning.html.

2. Machine Learning Market Size to Hit USD 1,407.65 Bn By 2034. Precedence Research - Statistics Platform for Market Intelligence, Market Research and Insights. URL: https://www.precedenceresearch.com/machine-learning-market.

3. Key Roles of Machine Learning in Data Analytics 2025 - Carmatec. Carmatec Inc - Mobile App Development Company. URL: https://www.carmatec.com/blog/key-roles-of-machine-learning-in-data-analytics/.

4. Attarbashi B. H. How Artificial Intelligence in Sales is Changing the Selling Process. AI bees | B2B Marketing Powered by AI. URL: https://www.ai-bees.io/post/how-artificial-intelligence-in-sales-is-changing-the-selling-process.

5. Alqhatani, Abdulmajeed & Ashraf, Muhammad & Ferzund, Javed & Shaf, Ahmad & Abosaq, Hamad & Rahman, Saifur & Irfan, Muhammad & Alqhtani, Samar M.. (2022). 360° Retail Business Analytics by Adopting Hybrid Machine Learning and a Business Intelligence Approach. Sustainability. 14. 11942. 10.3390/su141911942.

6. The Impact of Data Quality on Machine Learning - Wipro. Digital Transformation Services for Global IT Solutions - Wipro. URL: https://www.wipro.com/engineering/the-impact-of-data-quality-on-machine-learning/.

7. AI and Machine Learning in Sales: A Practical Guide | SalesMind AI. AI for Sales Prospecting - Generative AI Sales Agent | SalesMind AI. URL: https://sales-mind.ai/blog/machine-learning-in-sales.

8. Minasyan M. K-means Clustering: Store Sales Analysis. Medium. URL: https://medium.com/@minasyanmarina/K-means-clustering-store-sales-analysis-61d663c56687.

9. Random Forest. Corporate Finance Institute. URL: https://corporatefinanceinstitute.com/resources/data-science/random-forest/.

10. Random Forests · UC Business Analytics R Programming Guide. UC Business Analytics R Programming Guide ·. URL: https://uc-r.github.io/random_forests.

11. PgCert A. R. Artificial Neural Networks in Sales and Customer Churn Management. LinkedIn. URL: https://www.linkedin.com/pulse/artificial-neural-networks-sales-customer-churn-andre-ripla-pgcert-rcufe/.

12. Superstore Sales Dataset. Kaggle. URL: https://www.kaggle.com/datasets/rohitsahoo/sales-forecasting.

13. Fernandez J. How data normalization affects your Random Forest algorithm. Medium. URL: https://towardsdatascience.com/how-data-normalization-affects-your-random-forest-algorithm-fbc6753b4ddf?gi=ec2e8a401301.

14. Five Use Cases for Machine Learning in Sales. Credera. URL: https://www.credera.com/en-us/insights/five-use-cases-for-machine-learning-in-sales.

15. Predictive Sales Analytics: Overview, Implementation, & AI Use. eWEEK. URL: https://www.eweek.com/artificial-intelligence/ai-predictive-sales-analytics/.

16. AI and machine learning in marketing analytics: A revenue-driven approach | MarTech. MarTech. URL: https://martech.org/ai-and-maching-learning-in-marketing-analytics-a-revenue-driven-approach/.

17. Singh, Mickey. (2024). Machine Learning in Marketing Analytics. International Journal of Enhanced Research in Management & Computer Applications. 13. 2319-7471. 10.55948/IJERMCA.2024.0410.

18. Leonel A. Predictive Sales Analysis Using Machine Learning. LinkedIn. URL: https://www.linkedin.com/pulse/predictive-sales-analysis-using-machine-learning-ataliba-leonel-ctbdf/.

19. Glackin, Caroline & Adivar, Murat. (2022). Using the power of machine learning in sales research: process and potential. Journal of Personal Selling & Sales Management. 43. 1-17. 10.1080/08853134.2022.2128812.

20. Yadav, Puneet & Kumar, Vipin & Bhushan, Ravi. (2023). Analysis of Machine Learning Model for Predicting Sales Forecasting. 10.1109/ICAEECI58247.2023.10370914.

21. Radha Rani, Deevi and Soumya, B and Supriya, G Lahari and Yuktha, N and Kolluru, Pavan Kumar and Naga Malleswari, D, Performance Analysis of Sales Using Machine Learning (May 5, 2023). Proceedings of the KILBY 100 7th International

Conference on Computing Sciences 2023 (ICCS 2023), Available at SSRN: https://ssrn.com/abstract=4493480 or http://dx.doi.org/10.2139/ssrn.4493480

22. Jain, Harshit and Vashi, Dattpalsinh and Kumar Ray, Sumit and Vishal, Vishal, Sales Prediction Using Machine Learning (May 5, 2023). Proceedings of the KILBY 100 7th International Conference on Computing Sciences 2023 (ICCS 2023), Available at SSRN: https://ssrn.com/abstract=4495850 or http://dx.doi.org/10.2139/ssrn.4495850

23. Dharmveer Singh Rajpoot, Bhavey Mittal, Harshit Dudani, and Ujjwal Singhal. 2023. Sales Analysis and Forecasting using Machine Learning Approach. In Proceedings of the 2023 Fifteenth International Conference on Contemporary Computing (IC3-2023). Association for Computing Machinery, New York, NY, USA, 375–377. https://doi.org/10.1145/3607947.3608032

24. Dennis Herhausen, Stefan F. Bernritter, Eric W.T. Ngai, Ajay Kumar, Dursun Delen, Machine learning in marketing: Recent progress and future research directions, Journal of Business Research, Volume 170, 2024, 114254, ISSN 0148-2963, https://doi.org/10.1016/j.jbusres.2023.114254.(https://www.sciencedirect.com/science/article/pii/S0148296323006136)

25. Machine Learning for Sales Forecasting: Use Machine Learning Algorithms to Predict Future Sales Trends Based on Historical Data - Platforce. Platforce. URL: https://platforce.io/machine-learning-for-sales-forecasting-use-machine-learning-algorithms-to-predict-future-sales-trends-based-on-historical-data/.

26. F. Abubaker and Ala'Khalifeh, "Sales' Forecasting Based on Big Data and Machine Learning Analysis," 2023 9th International Conference on Control, Decision and Information Technologies (CoDIT), Rome, Italy, 2023, pp. 804-808, doi: 10.1109/CoDIT58514.2023.10284159. keywords: {Economics;Training;Recurrent neural networks;Computational modeling;Machine learning;Predictive models;Big Data;Machine Learning;Sales' Forecasting;Deep Learning;Recurrent Neural Network;Big Data},

27. How Does AI Analyze Sales Data. BuzzBoard. URL: https://www.buzzboard.ai/how-does-ai-analyze-sales-data/.

28. Malysheva V. Machine learning examples in digital marketing in 2024. owox. URL: https://www.owox.com/blog/articles/machine-learning-in-marketing/.

29. Gallego, V.; Lingan, J.; Freixes, A.; Juan, A.A.; Osorio, C. Applying Machine Learning in Marketing: An Analysis Using the NMF and k-Means Algorithms. Information 2024, 15, 368. https://doi.org/10.3390/info15070368

30. Yara Kayyali Elalem, Sebastian Maier, Ralf W. Seifert, A machine learning-based framework for forecasting sales of new products with short life cycles using deep neural networks, International Journal of Forecasting, Volume 39, Issue 4, 2023, Pages 1874-1894, ISSN 0169-2070, https://doi.org/10.1016/j.ijforecast.2022.09.005. (https://www.sciencedirect.com/science/article/pii/S0169207022001364)

31. The Role of AI in Sales Strategy: Leveraging Predictive Analytics for Smarter Targeting. The Center for Sales Strategy - Sales Strategy Blog. URL: https://blog.thecenterforsalesstrategy.com/the-role-of-ai-in-sales-strategy.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

1

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА
на тему:

МЕТОДИКА ОПТИМІЗАЦІЇ ПРОЦЕСУ АНАЛІЗУ ДАНИХ ПРОДАЖІВ ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ

на здобуття освітнього ступеня магістра зі спеціальності 126 Інформаційні системи та технології освітньо-професійної програми Інформаційні системи та технології

Виконав: здобувач вищої освіти гр. ІСДМ-61
Владислав КАРПЕНКО
Науковий керівник: к. т. н., доцент кафедри інформаційних систем та технологій
Ольга ПОЛОНЕВИЧ

2

ЗАВДАННЯ ДОСЛІДЖЕННЯ

- Огляд сучасних методів машинного навчання для аналізу продажів.
- Визначення найбільш підходящих алгоритмів для вирішення задач.
- Розробка методики підготовки даних і оцінки точності моделей.
- Тестування методики на реальних даних для підтвердження ефективності.

Актуальність

В умовах цифрової економіки обробка великих обсягів даних є ключовою задачею для підприємств. Алгоритми машинного навчання дозволяють автоматизувати бізнес-процеси, підвищити точність прогнозів та ефективність роботи.

Новизна

Розробка методики оптимізації аналізу даних продажів із використанням машинного навчання для підвищення точності прогнозування та ефективності бізнес-процесів.

Мета

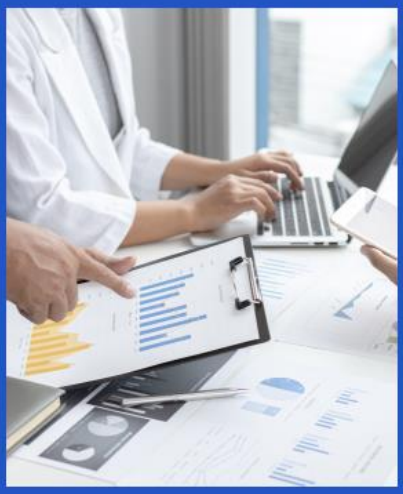
Розробка методики аналізу даних продажів із використанням машинного навчання для підвищення ефективності прогнозування та прийняття рішень.

Об'єкт дослідження: процес аналізу даних продажів.
Предмет дослідження: методи та алгоритми машинного навчання для оптимізації аналізу даних продажів.

3

Роль аналізу продажів у сучасному бізнесі

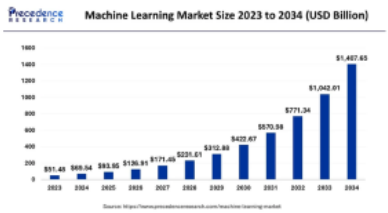
- **Оперативне прийняття рішень**: швидкий аналіз дозволяє миттєво реагувати на зміни ринку та поведінку клієнтів.
- **Оптимізація ресурсів**: ефективне планування запасів і логістики мінімізує витрати.
- **Підвищення конкурентоспроможності**: швидкий доступ до аналітики допомагає адаптувати стратегії для випередження конкурентів.
- **Прогнозування трендів**: виявлення сезонності та поведінкових патернів для прогнозування попиту.
- **Поліпшення взаємодії з клієнтами**: персоналізовані рекомендації та цільові маркетингові кампанії.



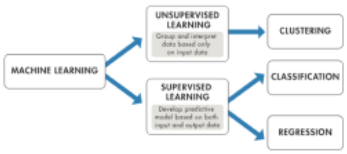
4

MACHINE LEARNING ДЛЯ АНАЛІЗУ ПРОДАЖІВ

- **Прогнозування попиту**
- **Сегментація клієнтів**
- **Оптимізація цін**
- **Виявлення аномалій**



Розмір ринку машинного навчання та прогноз на 2024–2034 роки



Методи машинного навчання

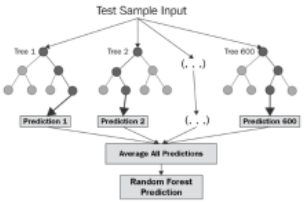
Сегментація клієнтів	Алгоритм K-means використовується для поділу клієнтів на групи за схожими характеристиками. Допомагає у створенні персоналізованих пропозицій і цільових маркетингових кампаній.
Прогнозування попиту	Алгоритми Random Forest та LSTM забезпечують точне передбачення обсягів продажів. Враховується сезонність, історичні дані та зовнішні фактори.
Виявлення трендів і аномалій	Класифікаційні та регресійні моделі виявляють аномальні дані та ключові тенденції у продажах

5

Алгоритми ML для аналізу даних продажів

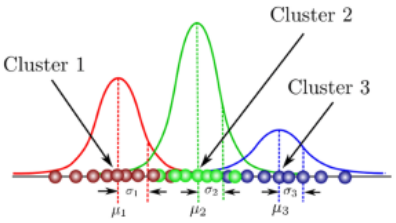
Random Forest

Ансамблевий метод дерев рішень, що забезпечує точне прогнозування продажів завдяки стійкості до шуму у даних.



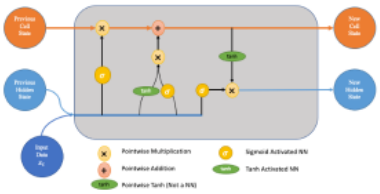
K-Means

Алгоритм кластеризації, що дозволяє сегментувати клієнтів та виявляти схожі групи для аналізу продажів.



LSTM

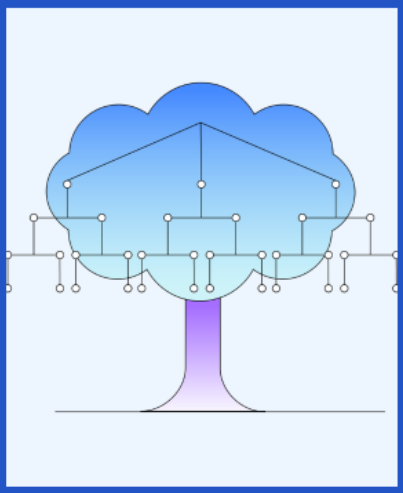
Рекурентна нейронна мережа, ефективна для прогнозування продажів на основі аналізу часових рядів із сезонністю.



6

Для розробки методу аналізу даних продажів було обрано алгоритм Random Forest

- **Висока точність** (забезпечує точні прогнози завдяки ансамблю дерев рішень)
- **Стійкість до шуму** (ефективно працює з неповними або зашумленими даними)
- **Гнучкість** (підходить як для класифікації, так і для регресії даних)
- **Інтерпретованість** (дозволяє оцінювати важливість ознак, що впливають на результат)



7

Очищення даних

Superstore Sales Dataset

Попередній огляд даних

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Завантаження даних
data = pd.read_csv('Superstore.csv')

# Попередній огляд даних
print(data.info())
print(data.describe())
print(data.isnull().sum())
```

Заповнення середніми значеннями

```
# Заповнення пропущених значень
data['Sales'] = data['Sales'].fillna(data['Sales'].mean())
data['Region'] = data['Region'].fillna(data['Region'].mode()[0])
```

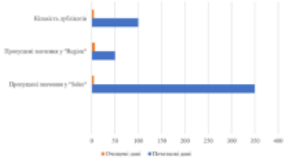
Видалення дублікатів

```
# Видалення дублікатів
initial_rows = data.shape[0]
data = data.drop_duplicates()
final_rows = data.shape[0]

print(f"Кількість видалених дублікатів: {initial_rows - final_rows}")
```

Аналіз і видалення викидів

Порівняння структурованост і початкових і очищених даних

```
# Графік до очищення
plt.figure(figsize=(10, 5))
sns.countplot(data=data, x='Region')
plt.title('Розподіл замовлень за регіонами (до очищення)')
plt.show()

# Графік після очищення
plt.figure(figsize=(10, 5))
sns.countplot(data=data_cleaned, x='Region')
plt.title('Розподіл замовлень за регіонами (після очищення)')
plt.show()
```



8

Масштабування даних

Масштабування даних

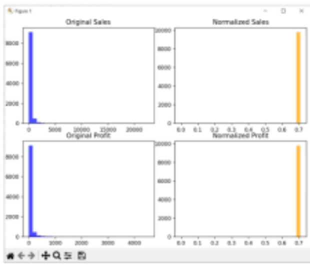
```
# Масштабування даних
min_max_scaler = MinMaxScaler()
scaled_data_minmax = min_max_scaler.fit_transform(cleaned_data)
```

Нормалізація даних

```
# Нормалізація даних
normalized_data = normalize(scaled_data_minmax, norm='l2')

# Збереження результатів у DataFrame для аналізу
scaled_df_minmax = pd.DataFrame(scaled_data_minmax, columns=cleaned_data.columns)
normalized_df = pd.DataFrame(normalized_data, columns=cleaned_data.columns)
```

Порівняння початкових та отриманих даних

```
# Побудова графіків
fig, axes = plt.subplots(len(features_to_plot), 2, figsize=(10, 8))

for i, feature in enumerate(features_to_plot):
    # Початкові дані
    axes[i, 0].hist(cleaned_data[feature], bins=30, color='blue', alpha=0.7)
    axes[i, 0].set_title(f'Original {feature.capitalize()}')

    # Нормалізовані дані
    axes[i, 1].hist(normalized_df[feature], bins=30, color='orange', alpha=0.7)
    axes[i, 1].set_title(f'Normalized {feature.capitalize()}')

plt.tight_layout()
plt.show()
```

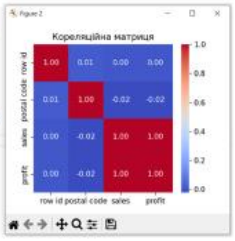


Релевантні ознаки моделі

9

Побудова кореляційної матриці

```
# Вибір тільки числових колонок
numeric_data = data.select_dtypes(include=['float64', 'int64'])

# Побудова кореляційної матриці
corr_matrix = numeric_data.corr()

# Візуалізація кореляційної матриці
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Кореляційна матриця')
plt.show()
```



Аналіз параметрів впливу на цільову змінну

```
# Перевірка колонок
print(data.columns)

# Вибір колонок для навчання
numerical_columns = ['sales', 'profit']  # Замініть на ваші числові дані
categorical_columns = ['order id']  # Замініть на категоріальні дані, якщо вони є

# Перетворення категоріальних змінних
if categorical_columns:
    label_encoders = {}
    for col in categorical_columns:
        le = LabelEncoder()
        data[col] = le.fit_transform(data[col])
        label_encoders[col] = le

# Формування X і y
X = data[numerical_columns + categorical_columns]
y = data['profit']

# Поділ на навчальні та тестові набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Ініціалізація моделі Random Forest
model = RandomForestRegressor(n_estimators=100, random_state=42)

# Навчання моделі
model.fit(X_train, y_train)

# Оцінка на тестовому наборі
y_pred = model.predict(X_test)
print("Оцінка точності:", model.score(X_test, y_test))
```

Метрики оцінювання точності та ефективності

10

MAE	Середня абсолютна помилка
MSE	Середня квадратична помилка
RMSE	Середнє квадратичне відхилення
R2	Коефіцієнт детермінації

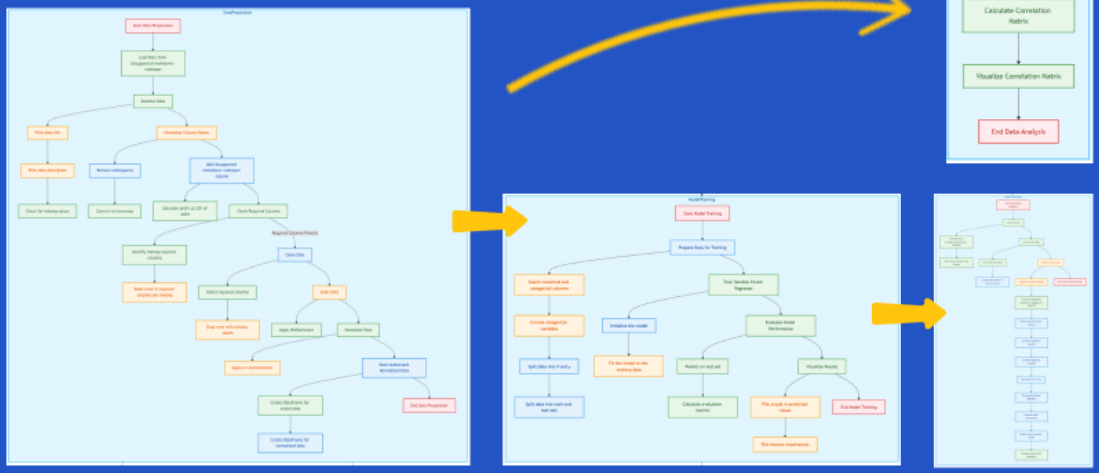
Час навчання моделі

Час прогнозування на тестовій вибірці

Обсяг використання пам'яті та ресурсів

Алгоритм програми

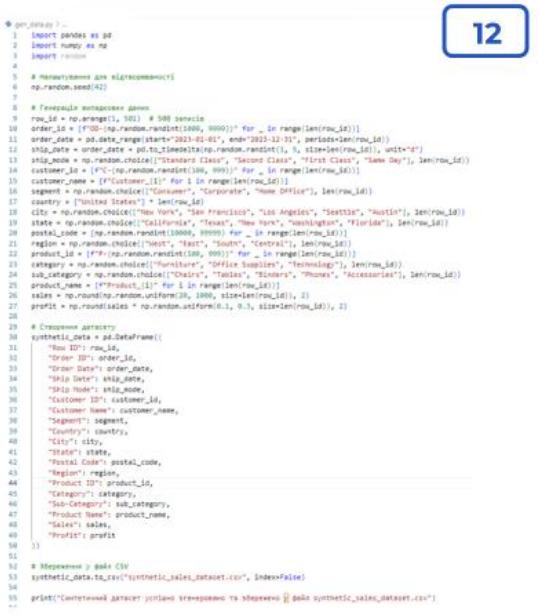
11



Генерація тестового датасету

12

synthetic_sales_data set.csv



Аналіз та прогнозування (результат)

13

Завантаження даних

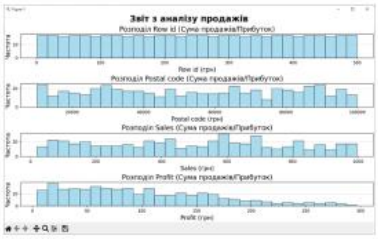
```
# Завантаження даних
def load_test_data(filepath):
    """
    Завантаження тестових даних із файлу.
    """
    data = pd.read_csv(filepath)
    print("Колонки у датасеті:", data.columns.tolist())  # Перевірка колонок
    return data
```

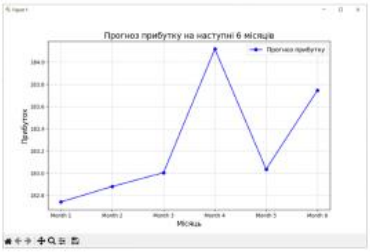
Аналіз даних продажів

```
# Аналіз даних продажів
def analyze_data(data):
    """
    Аналіз даних та візуалізація основних показників.
    """
    print("Огляд даних:")
    print(data.describe())

    numeric_columns = data.select_dtypes(include=['float64', 'int64']).columns.tolist()

    if not numeric_columns:
        print("Числові колонки для аналізу відсутні.")
        return

    fig, axes = plt.subplots(nrows=len(numeric_columns), figsize=(12, 6 * len(numeric_columns)), constrained_layout=True)

    if len(numeric_columns) == 1:
        axes = [axes]  # Якщо лише одна колонка, зробимо список для однакової обробки

    for ax, col in zip(axes, numeric_columns):
        ax.hist(data[col], bins=30, color='skyblue', alpha=0.7, edgecolor='black')
        ax.set_title(f'Розподіл {col.capitalize()} (Сума продажів/Прибуток)', fontsize=16)
        ax.set_xlabel(f'{col.capitalize()} (грн)', fontsize=14)
        ax.set_ylabel('Частота', fontsize=14)
        ax.grid(axis='y', linestyle='--', alpha=0.7)

    fig.suptitle('Звіт з аналізу продажів', fontsize=20, fontweight='bold')
    plt.show()
```

Прогнозування продажів на пів року

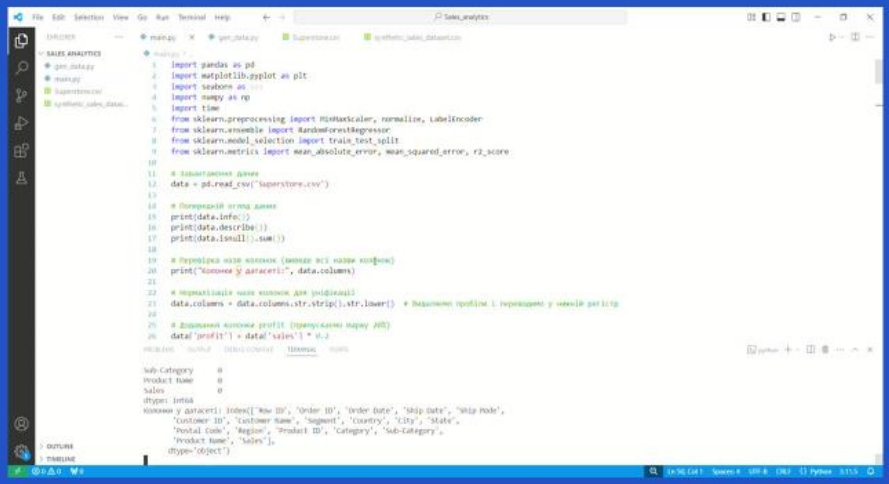
```
# Прогнозування
def predict_sales(data):
    """Навчання моделі та прогнозування продажів на основі даних."""

    # Перевірка наявних та обов'язкових колонок
    required_columns = ['Order Date', 'Profit']
    for col in required_columns:
        if col not in data.columns:
            raise KeyError(f"Відсутня колонка: '{col}'. Перевірте вхідні дані.")

    # Перетворення формату дати
    data['Order Date'] = pd.to_datetime(data['Order Date'])
    data['Year'] = data['Order Date'].dt.year
    data['Month'] = data['Order Date'].dt.month

    # Створення ознак для навчання
    features = data[['Year', 'Month', 'Sales', 'Quantity', 'Discount', 'Profit']]
    target = data['Profit']

    # Масштабування числових ознак
    numeric_features = features.select_dtypes(include=['float64', 'int64']).columns
    scaler = MinMaxScaler()
    features[numeric_features] = scaler.fit_transform(features[numeric_features])

    # Кодування категоріальних ознак
    categorical_features = features.select_dtypes(include=['object']).columns
    if len(categorical_features) > 0:
        encoder = LabelEncoder()
        for col in categorical_features:
            features[col] = encoder.fit_transform(features[col])

    # Розділ даних на навчальну та тестову вибірки
    X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

    # Навчання моделі
    model = RandomForestRegressor(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)

    # Оцінка моделі
    y_pred = model.predict(X_test)
    print("Оцінка моделі:")
    print(f"MAE: {mean_absolute_error(y_test, y_pred):.2f}")
    print(f"MSE: {mean_squared_error(y_test, y_pred):.2f}")
    print(f"R2 Score: {r2_score(y_test, y_pred):.2f}")

    # Прогноз на наступні 6 місяців
    future = []
    for i in range(1, 7):
        future_date = last_date + pd.DateOffset(months=i)
        future.append([future_date.year, future_date.month, 0, 0, 0, 0])  # Фіктивні значення для інших ознак
    future = pd.DataFrame(future, columns=features.columns)
    future[numeric_features] = scaler.transform(future[numeric_features])
    future_predictions = model.predict(future)

    # Візуалізація прогнозу
    months = ['Місяць 1', 'Місяць 2', 'Місяць 3', 'Місяць 4', 'Місяць 5', 'Місяць 6']
    plt.figure(figsize=(10, 6))
    plt.plot(months, future_predictions, marker='o', linestyle='-', color='blue', label='Прогноз прибутку')
    plt.title('Прогноз прибутку на наступні 6 місяців', fontsize=16)
    plt.xlabel('Місяць', fontsize=14)
    plt.ylabel('Прибуток', fontsize=14)
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.legend()
    plt.show()

    print("Прогноз прибутку на наступні 6 місяців:", future_predictions)
```





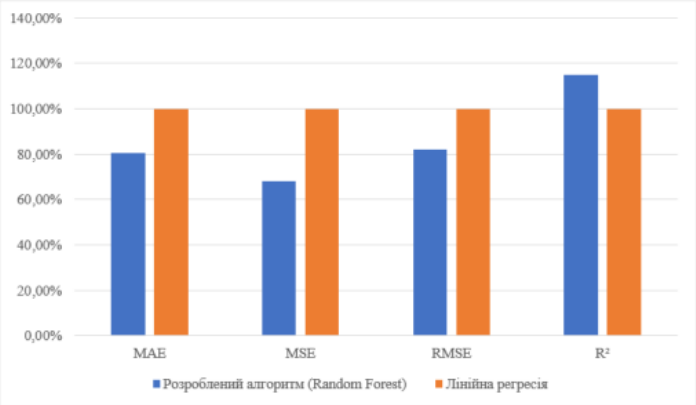
Демонстрація роботи

14



Оцінка ефективності

15



Похибки порівняно із алгоритмом лінійної регресії:

MAE **-19.6%**
Середня абсолютна помилка (MAE) вказує, що алгоритм прогнозує значення з на 19.6% меншими відхиленнями порівняно з лінійною регресією.

MSE **-33%**
Розроблений підхід краще обробляє великі відхилення. Зменшення MSE на понад 32% є свідченням високої стабільності моделі.

RMSE **-17%**
Корінь середньоквадратичної помилки (RMSE) підтверджує, що розроблений алгоритм точніше обчислює прогноз навіть за наявності вхідних даних з високою дисперсією

R2 **+17%**
Коефіцієнт детермінації (R^2) досягає 0.93, що є значно вищим порівняно з традиційним методом. Це означає, що модель пояснює 93% варіації даних, залишаючи мінімальну частку для випадкових відхилень.

Висновки

16

- **Розроблена методика аналізу даних продажів на основі машинного навчання дозволяє значно підвищити точність прогнозування та ефективність прийняття рішень**. Вона також забезпечує зниження витрат на обробку даних і покращення управління бізнес-процесами, що є ключовими для конкурентоспроможності підприємств.
- **Моделі, створені на основі алгоритмів K-means, Random Forest та нейронних мереж, продемонстрували високу ефективність у задачах сегментації клієнтів, прогнозування попиту та визначення оптимальних цінових стратегій**. Це дозволяє бізнесам краще розуміти своїх клієнтів та адаптуватися до змін ринку.
- **Практична значущість роботи полягає у можливості застосування розробленої методики для підвищення прибутковості та оптимізації процесів на підприємствах.** Крім того, методику можна адаптувати для інших галузей, де потрібен аналіз великих обсягів даних.
- **Використання алгоритмів машинного навчання дозволило зменшити ризики, пов'язані з людським фактором, та скоротити час аналізу.** Персоналізовані рекомендації та точні прогнози сприяють підвищенню задоволеності клієнтів і створенню умов для зростання продажів.
- **Подальші дослідження можуть бути зосереджені на вдосконаленні алгоритмів для більш складних завдань і розширенні функціоналу методики**. Впровадження алгоритмів глибокого навчання дозволить працювати зі складними структурами даних, такими як зображення чи текст, для більш глибокого аналізу.
- **Проведене дослідження підтвердило доцільність використання машинного навчання у сфері аналізу продажів.** Застосування сучасних алгоритмів сприяє оптимізації бізнес-рішень, підвищенню ефективності процесів та конкурентоспроможності підприємств.

17

Дякую за увагу