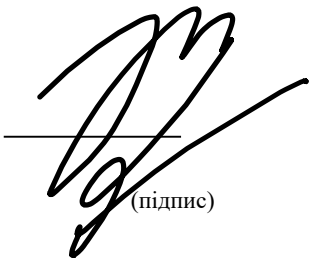
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота на здобуття ступеня бакалавра
за спеціальністю 121 Інженерія Програмного Забезпечення
на тему:

РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ АНАЛІЗУ КУПІВЕЛЬНОЇ ПОВЕДІНКИ КОРИСТУВАЧІВ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

Виконав студент 4-го курсу
Юрій ГОРОБЕЦЬ



(підпис)

Науковий керівник:
доцент
Лариса КАТЕРИНИЧ

_____

(підпис)

Засвідчую, що в цій курсовій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент

_____

(підпис)

Київ-2023

РЕФЕРАТ

Обсяг роботи - 44 сторінки, 26 ілюстрацій, 8 джерел посилань.

Ключові слова: машинне навчання, аналіз даних, сегментація клієнтів, асоціативні правила, прогностична модель, класифікація, веб-додаток.

Об'єктом дослідження є процес використання технік машинного навчання і аналізу даних для сегментування клієнтів та прогнозування поведінки клієнтів.

Предметом роботи є набір даних про клієнтів і їх покупки, який обробляється за допомогою різних технік машинного навчання і аналізу даних, включаючи методи сегментації, алгоритм Apriori для виявлення асоціативних правил та модель логістичної регресії для класифікації.

Метою роботи є аналіз даних користувачів та створення моделі машинного навчання, яка допоможе підприємству краще розуміти своїх клієнтів, передбачати їх поведінку і таким чином покращувати свої продукти та послуги.

Інструментами розробки обрано Python - мову програмування високого рівня, яка широко використовується для наукових обчислень, аналізу даних та машинного навчання, і бібліотеки Python, такі як pandas, scikit-learn і Matplotlib, для обробки даних, моделювання і візуалізації.

Методи розробки: модель розробки “Waterfall”

Результати роботи: проведено загальний аналіз набору даних про клієнтів, розроблено сегментацію клієнтів на основі їх властивостей і поведінки, виявлено асоціативні правила, що описують взаємозв'язки між різними продуктами та групами клієнтів, та розроблено прогностичну модель, яка дозволяє класифікувати клієнтів на основі їх потенційної активності. Система дозволяє компанії впроваджувати більш цілеспрямовані стратегії маркетингу та продажів, а також надає важливий інструмент для зрозуміння

та оптимізації поведінки клієнтів. Впроваджено і використано машинне навчання та техніки аналізу даних для створення цієї системи. Цей процес включає обробку і аналіз вхідних даних, визначення важливих характеристик, побудову і валідацію моделей машинного навчання та оцінку їх продуктивності та точності. Окрім того, виконано загальний огляд поточних методів і технік аналізу даних та машинного навчання, які використовуються для сегментації клієнтів і прогнозування поведінки клієнтів, та проаналізовано їх переваги та обмеження. Описано можливі напрямки подальшого розвитку системи, включаючи розширення набору даних для покращення точності моделі, використання більш складних моделей машинного навчання для виявлення більш тонких взаємозв'язків у даних, та інтеграцію системи з іншими джерелами даних або бізнес-процесами для ширшого застосування.

ЗМІСТ

Київ-2023

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

ML (Machine Learning) - машинне навчання.

AI (Artificial Intelligence) - штучний інтелект.

CSV (Comma-Separated Values) - значення, розділені комами (формат файлу).

Pandas - бібліотека Python для обробки і аналізу даних.

NumPy (Numerical Python) - бібліотека Python для обчислювальних операцій.

Sklearn(Scikit-learn) - бібліотека Python для машинного навчання.

LR (Logistic Regression) - логістична регресія.

SVM (Support Vector Machines) - метод опорних векторів.

RF (Random Forest) - випадковий ліс.

IQR (Interquartile range) – метод міжквартильного розмаху

KNN (K-Nearest Neighbors) - метод найближчих сусідів.

PCA (Principal Component Analysis) - аналіз головних компонент.

ROC (Receiver Operating Characteristic) - характеристика роботи приймача.

AUC (Area Under the Curve) - площа під кривою.

AC (Accuracy) - точність.

F1 (F1 Score) - метрика F1.

CM (Confusion Matrix) - матриця помилок.

ВСТУП

Оцінка сучасного стану об'єкта розробки. Сучасний ринок споживачів, особливо в контексті онлайн-покупок, постійно змінюється і розвивається. Це включає в себе не тільки зміну в попиті на різні товари, але і зміну в поведінці та виборі клієнтів. Дослідження та аналіз цих даних має величезне значення для підприємств, що прагнуть оптимізувати свої стратегії продажу та маркетингу, адаптуватись до нових тенденцій та враховувати мінливість вибору споживачів.

Актуальність роботи та підстави її виконання. З ростом великих даних і поширенням технологій машинного навчання, можливості для аналізу поведінки споживачів та прогнозування їх вибору стали небувало актуальними. Різноманітні алгоритми та моделі машинного навчання можуть допомогти в дослідженні даних та виявленні цінних винсовків, що можуть бути використані для покращення стратегій маркетингу та продажу. Відповідно, дана робота виявляється актуальною і важливою в контексті сучасних тенденцій у сфері аналізу даних і машинного навчання. Мета - використати ці технології для дослідження даних про покупки клієнтів, побудови асоціативних правил та класифікації клієнтів, що сприятиме кращому розумінню потреб споживачів і допоможе підприємствам в прийнятті обґрунтованих та ефективних рішень.

Мета та завдання роботи. Метою даної роботи є проведення детального аналізу даних та виявлення важливих закономірностей у вибірці, яка включає інформацію про покупки. Специфічні задачі, що виходять з цієї мети:

- Дослідження основних характеристик даних, таких як вік, дохід, дні прив'язаності до бренду та витрати на різні товари.

- Сегментація клієнтів на основі їх характеристик, що дозволяє зрозуміти взаємозв'язки між різними групами.
- Побудова моделі асоціативних правил для визначення паттернів покупок та поведінки клієнтів.
- Використання моделі логістичної регресії для класифікації клієнтів на основі їх характеристик та поведінки.

Об'єкт та методи дослідження. Об'єктом дослідження є дані про покупки клієнтів у різних товарних категоріях, а також їх демографічні характеристики і характеристики поведінки. Для аналізу даних використовувалися різноманітні методи та технології машинного навчання, включаючи сегментацію клієнтів, побудову асоціативних правил та моделювання логістичної регресії. Сегментація клієнтів проводилась за допомогою методу вирізання Pandas для створення категорій віку, доходу та днів прив'язаності. Асоціативні правила були побудовані за допомогою алгоритму Apriori, який є одним з найпопулярніших методів виявлення закономірностей в наборах даних. Для класифікації клієнтів використовувалась модель логістичної регресії, яка була навчена та протестована на основі набору даних. Нарешті, для представлення результатів роботи було побудовано інтерактивний веб-додаток, що дозволяє завантаження даних для отримання результатів передбачення. Ця робота включає в себе важливий аналіз великої кількості даних і є важливим внеском в розуміння шаблонів поведінки клієнтів, що може допомогти підприємствам у прийнятті ефективних рішень щодо стратегії продажів і маркетингу.

РОЗДІЛ 1. ЗАГАЛЬНИЙ ОГЛЯД ТЕХНОЛОГІЙ ВИКОРИСТОВУВАНИХ У МАШИННОМУ НАВЧАННІ ТА ЇХ ОСОБЛИВОСТІ

1.1. Методи машинного навчання

Машинне навчання — це процес, який дає комп'ютерам здатність вчитися з досвіду та вдосконалювати свою роботу без безпосереднього програмування. Методи машинного навчання засновані на розпізнаванні шаблонів та закономірностей в наборах даних, що надаються на вхід. Застосування цих методів змінює спосіб, яким ми аналізуємо дані, розробляємо моделі та приймаємо рішення в різних областях, включаючи, але не обмежуючись, медицину, науку, фінанси, роздрібну торгівлю.

Основні методи машинного навчання поділяються на категорії за такими критеріями[2]:

а) За типом навчання:

1) Навчання з учителем: моделі навчаються на основі вхідних даних, для яких відомі правильні відповіді. Метою є побудова моделі, яка зможе робити точні прогнози для нових, невідомих даних.

2) Навчання без учителя: моделі навчаються на основі вхідних даних, для яких не відомі правильні відповіді. Метою є виявлення закономірностей та шаблонів в даних.

3) Посилене навчання: модель навчається шляхом взаємодії з середовищем та отримання винагороди або кари за вжиті дії.

б) За типом завдання:

1) Класифікація: передбачення дискретних категорій для нових даних.

2) Регресія: передбачення неперервних значень для нових даних.

3) Кластеризація: групування подібних об'єктів.

4) Зменшення розмірності: спрощення даних без втрати важливої інформації.

в) За типом моделі:

1) Лінійна регресія, логістична регресія.

2) Дерева рішень, випадковий ліс.

3) Нейронні мережі.

4) Опорні вектори.

5) K-найближчі сусіди.

6) Наївний баєсівський класифікатор.

Машинне навчання відкриває безліч можливостей для використання даних на новому рівні, проте вибір методу машинного навчання залежить від специфіки задачі, доступності даних, а також вимог до точності та прозорості моделі.

1.2. Специфіка технологій машинного навчання

Застосування машинного навчання дозволяє комп'ютерам вдосконалювати свою роботу, вчучись використовувати великі набори даних, з якими працюють люди. Однак, на відміну від людей, машини можуть обробляти ці дані набагато швидше і точніше. Ось декілька прикладів, де використовуються технології машинного навчання[2]:

- Вибір продуктів: використовуючи алгоритми машинного навчання, можна визначити, які продукти вподобають певні сегменти покупців. Це дозволяє зрозуміти, які товари мають найвищий рівень впевненості при рекомендації для кожного сегмента.

- Кластеризація: Методи машинного навчання дозволяють групувати користувачів згідно з їхніми особливостями та вподобаннями, що в свою чергу дозволяє більш точно цілитися на певні сегменти покупців.

- Прогнозування: Застосування моделей машинного навчання дозволяє комп'ютерам прогнозувати майбутній попит на товари на основі історичних даних.

- Стандартизація: Важливим етапом у будь-якому процесі машинного навчання є підготовка даних. Застосування масштабування допомагає нам зробити дані більш порівнянними та релевантними для моделі.

- Класифікація: Використовуючи алгоритми машинного навчання, можна розробити моделі, які здатні класифікувати нові дані на основі навчання з використанням існуючих даних.

Використання таких технологій, як логістична регресія[2], дозволяє оцінювати та прогнозувати відповідні змінні, а також міркувати про можливі взаємозв'язки між різними факторами. Важливо врахувати, що будь-яка модель машинного навчання може бути покращена через періодичне навчання та оновлення.

Крім того, розуміння важливості оцінки ефективності моделі[2] є ключовим для успішного застосування машинного навчання. Для цього використовуються такі метрики, як точність, F1-оцінка та матриця невизначенності, що допомагає нам зрозуміти якість прогнозування нашої моделі та її здатність впоратися з новими даними.

1.3 Мова програмування

Python є однією з найпопулярніших мов програмування, особливо в областях наукових досліджень, обробки даних, веб-розробки, автоматизації та, звичайно ж, машинного навчання. Python є відкритим програмним

забезпеченням, що означає, що він вільно доступний для використання та розповсюдження, а його код можна вільно змінювати.

Основною перевагою Python є його простота і зручність для читання та написання коду. Він відомий своєю чистою та зрозумілою синтаксичною структурою, що дозволяє розробникам зосередитися на розв'язанні проблем, а не на розбірливості самої мови програмування. Також, Python має велику стандартну бібліотеку, що включає багато корисних функцій, які роблять розробку програмного забезпечення простішою та швидшою.

Python також є динамічно типізованою мовою, що дозволяє більше гнучкості під час написання коду. Використання динамічної типізації в комбінації з об'єктно-орієнтованим програмуванням дозволяє розробникам ефективно створювати швидкі прототипи для алгоритмів машинного навчання.

Зрештою, Python володіє вбудованими засобами тестування, які сприяють ефективному виявленню та виправленню помилок. Ці інструменти допомагають підтримувати високу якість коду, що є важливим фактором для наукових досліджень та комерційного використання.

У сфері машинного навчання Python[7] є основним вибором переважної більшості розробників. Це зумовлено його простотою та гнучкістю, а також неймовірною кількістю бібліотек та інструментів, що значно спрощують розробку і виконання складних алгоритмів машинного навчання. З рисунку 1.1 можна виділити такі бібліотеки, як Scikit-learn для загального машинного навчання, TensorFlow і PyTorch для глибокого навчання, а також Pandas і NumPy для обробки та аналізу даних.

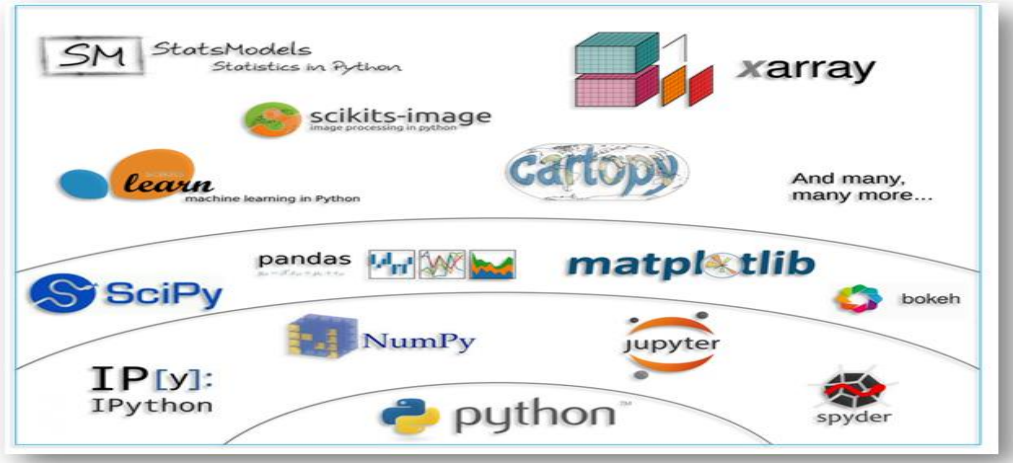


Рис 1.1 - частковий список доступних бібліотек[6].

Ще однією причиною широкої популярності Python у цій сфері є відкрита модель розробки, яка підтримує активне співтовариство користувачів. Це створює величезну кількість ресурсів для навчання та допомоги, доступних онлайн, що робить Python привабливим для вчених та інженерів, що працюють над новими ідеями або проектами в області машинного навчання.

Python був обраний для цього дослідження через його простоту, гнучкість, потужність, а також широку підтримку у спільноті розробників. Його легкість у вивченні та використанні, а також величезний набір інструментів машинного навчання робить його ідеальним вибором для аналізу даних про користувачів та їх покупки, та розробки моделі, що передбачає, чи здійснить певний користувач покупку в магазині, чи ні.

1.4 Технологія Streamlit

Streamlit є інноваційною технологією, що революціонізує процес створення веб-застосунків для машинного навчання і аналізу даних. Цей інструмент з відкритим надає потужну платформу, що дозволяє розробникам і дослідникам швидко перетворювати Python скрипти в готові веб-застосунки, що робить Streamlit ідеальним вибором в для нашої роботі

Streamlit має виразно відмінні характеристики від інших фреймворків для веб-розробки. Він вбудовує аналіз даних та машинне навчання прямо в процес створення застосунків, надаючи унікальний вплив на маніпулювання даними. Streamlit мінімізує часові витрати на створення інтерфейсу, що дозволяє фахівцям сконцентруватися на самих даних і їх аналізі. Однією з ключових особливостей Streamlit є його простота використання. З його допомогою можна легко створювати візуалізації, контролювати інтерактивність та публікувати застосунки, використовуючи Python. Незалежно від складності проекту - від простого додатку для візуалізації даних до складного застосунку для машинного навчання - Streamlit забезпечує простий, але потужний інструмент для досягнення будь-якої мети.

РОЗДІЛ 2. ПРОЦЕС АНАЛІЗУ ТА ОБРОБКИ ВХІДНИХ ДАНИХ

2.1.Загальний огляд вхідних даних

Одним з ключових аспектів успішного застосування машинного навчання є ретельна підготовка та аналіз вхідних даних[3]. Вихідний результат будь-якої моделі, навіть найбільш вдосконаленої, в більшості випадків залежить від якості вхідних даних. Використання ненадійних або неповних даних може призвести до помилкових висновків та неправильного прогнозування. Саме тому, детальний огляд та аналіз вхідних даних є обов'язковим кроком у будь-якому проекті з машинного навчання.

Перш ніж перейти до аналізу вхідних даних для даного проекту, важливо зрозуміти, що кожен набір даних є унікальним, і тому потребує індивідуального підходу. Зокрема, важливо врахувати такі чинники, як обсяг даних, їхню структуру, рівень шуму, наявність відсутніх значень та викидів[3]. Всі ці аспекти впливають на вибір підходу до підготовки даних та моделі машинного навчання.

Набір даних, що наведений на рисунку 2.1[1], подається на вхід до нашої програми. Цей датасет є збором результатів декількох маркетингових кампаній, та містить інформацію про користувачів, їх особисті характеристики та історію покупок.

1	id	birth_year	education_level	relationship_status	salary	kids_count	teens_count	customer_since	last_bought	wines_amount
2	9999	1965	Graduation	Together	75276	0	0	27-09-2012	2	610
3	999	1991	Graduation	Single	86037	0	0	02-01-2013	95	490
4	9988	1976	Master	Single	70379	0	1	03-03-2013	84	553
5	9986	1982	Graduation	Together	19444	1	0	22-02-2014	8	16
6	9984	1981	2n Cycle	Married	56337	1	1	27-03-2013	25	349

Рис 2.1 – фрагмент представлення вхідних даних

Дані представлені в наступних стовпцях[1]:

- id: Унікальний ідентифікатор користувача.
- birth_year: Рік народження користувача.

- education_level: Рівень освіти користувача.
- relationship_status: Сімейний статус користувача.
- salary: Річний дохід користувача.
- kids_count: Кількість дітей у віці до 12 років у користувача.
- teens_count: Кількість дітей у віці від 12 до 18 років у користувача.
- customer_since: Дата з якої користувач є клієнтом.
- last_bought: Кількість днів від останньої покупки.
- wines_amount, fruits_amount, meat_amount, fish_amount, dessert_amount, gold_amount: Кількість куплених продуктів за останні два роки за категоріями.
- deal_purchases, web_purchases, catalog_purchases, instore_purchases: Кількість покупок, зроблених через різні канали.
- web_visists_per_month: Кількість візитів веб-сайту за місяць.
- is_campaign_accepted_1, is_campaign_accepted_2, is_campaign_accepted_3, is_campaign_accepted_4, is_campaign_accepted_5: Бінарні поля, що вказують, чи прийняв користувач пропозицію під час проведення кампанії.
- complained: Бінарне поле, що вказує, чи подавав користувач скаргу протягом останніх двох років.
- z_cost_to_contact, z_profit: Змінні, пов'язані з вартістю контакту та доходом відповідно.
- last_campaign_accepted: Бінарне поле, що вказує, чи прийняв користувач останню маркетингову пропозицію.

Цей набір даних є досить різноманітним та багатогранним, що дозволяє нам провести глибокий аналіз покупацької поведінки та її кореляції з різними факторами. Наступним кроком у нашому аналізі

буде детальний розгляд цих даних, їхнє перетворення та попередня обробка перед використанням для моделей машинного навчання.

2.2 Огляд використаних бібліотек та модулів

В даній роботі використовується широкий набір бібліотек Python, кожна з яких призначена для вирішення певних завдань, пов'язаних з аналізом даних, машинним навчанням та візуалізацією.

Ось детальний огляд кожної з них[7]:

- NumPy (Numerical Python) - це фундаментальна бібліотека для наукових обчислень в Python. Вона надає високопродуктивний багатовимірний масив об'єктів і інструменти для роботи з цими масивами. Вона широко використовується для проведення математичних і логічних операцій на масивах.
- Pandas[3] - це гнучка бібліотека для обробки та аналізу даних. Pandas використовується для обробки структурованих даних і підтримує різні типи даних: часові ряди, таблиці з гетерогенними стовпцями і т.д. Pandas дозволяє зручно читати, записувати та маніпулювати дані.
- Seaborn[4] - це бібліотека візуалізації даних Python на основі Matplotlib, яка надає вищий рівень інтерфейсу для створення детальних та інформативних статистичних графіків.
- Plotly - це високорівнева, відкрита бібліотека для створення інтерактивних та статистичних графіків. plotly.graph_objects включає в себе високорівневий інтерфейс для створення графіків Plotly.
- Matplotlib - це найпопулярніша бібліотека для візуалізації даних в Python. Вона надає повний контроль над стилями, форматами і деталями графіків, і використовується для створення статичних, анімованих та інтерактивних візуалізацій в Python.

- Scikit-learn[5] (sklearn) - це найпопулярніша бібліотека машинного навчання для Python. Вона містить широкий набір моделей машинного навчання, а також інструменти для предпроцесингу даних, вибору і оцінки моделей. В даній роботі використовуються різні модулі Scikit-learn: preprocessing для предпроцесингу даних, cluster.KMeans для кластеризації, metrics для вимірювання точності моделей, linear_model.LogisticRegression для реалізації логістичної регресії, model_selection.train_test_split для розбиття даних на навчальну та тестову вибірки.
- LabelEncoder - це інструмент з sklearn, який використовується для перетворення категоріальних або текстових даних в числа, що є більш зручним форматом для моделей машинного навчання.
- StandardScaler - це інструмент з sklearn для стандартизації ознак, видаляючи середнє і масштабуючи до одиниці дисперсії.
- Apyori[8] - це проста і легка бібліотека Python для реалізації алгоритму Apriori, що є основним алгоритмом для виявлення асоціативних правил в даних.
- Mlxtend (machine learning extensions) - це корисна бібліотека для розширення функціональності Scikit-learn, Pandas та Matplotlib. Модулі frequent_patterns.apriori та frequent_patterns.association_rules використовуються для виявлення часто вживаних шаблонів та правил асоціації в даних.
-

2.3 Попередній аналіз та обробка вхідних даних

В цій роботі було виконано декілька кроків з метою підготувати дані до подальшого аналізу та моделювання. Коректний попередній аналіз та обробка

інформації, що подається на вхід до програми є запорукою[3] побудування ефективної моделі для досягнення мети нашого завдання.

Перший крок в цьому процесі – зчитування даних(рисунок 2.2). Для цього використаємо метод read_csv бібліотеки Pandas.

```
data = pandas.read_csv("consumer_data.csv", sep="\t")
data.head()
```

	id	birth_year	education_level	relationship_status	salary	kids_count	teens_count	customer_since	last_bought	wines_amount	...	web_visits_per_mon
0	5524	1957	Graduation	Single	58138.0	0	0	04-09-2012	58	635	...	
1	2174	1954	Graduation	Single	46344.0	1	1	08-03-2014	38	11	...	
2	4141	1965	Graduation	Together	71613.0	0	0	21-08-2013	26	426	...	
3	6182	1984	Graduation	Together	26646.0	1	0	10-02-2014	26	11	...	
4	5324	1981	PhD	Married	58293.0	1	0	19-01-2014	94	173	...	

5 rows × 29 columns

Рис 2.2 Зчитування вхідних даних

Далі робимо загальний огляд прочитаної інформації. Для цього використаємо[7] методи info та describe на об’єкті data, що зберігає представлення наших даних. Метод info надає короткий опис даних, включаючи інформацію про типи даних, кількість відсутніх значень, кількість рядків та стовпців і т.д. Це дозволяє швидко оцінити якість даних і виявити потенційні проблеми, такі як відсутні значення. Метод describe[7] використовуємо, щоб отримати статистичний опис числових стовпців в даних. Цей метод надає таку інформацію, як середнє значення, стандартне відхилення, мінімум, максимум, першу (25%), другу (50%, або медіану) та третю (75%) квартилі. Ця інформація може допомогти зрозуміти розподіл даних і виявити аномалії або викиди. Результати виконання info та describe показано на рисунках 2.3 та 2.4 відповідно.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    2240 non-null   int64  
 1   birth_year            2240 non-null   int64  
 2   education_level       2240 non-null   object 
 3   relationship_status   2240 non-null   object 
 4   salary                2216 non-null   float64
 5   kids_count            2240 non-null   int64  
 6   teens_count           2240 non-null   int64  
 7   customer_since        2240 non-null   object 
 8   last_bought           2240 non-null   int64  
 9   wines_amount          2240 non-null   int64  
 10  fruits_amount         2240 non-null   int64  
 11  meat_amount           2240 non-null   int64  
 12  fish_amount           2240 non-null   int64  
 13  dessert_amount        2240 non-null   int64  
 14  gold_amount           2240 non-null   int64  
 15  deal_purchases        2240 non-null   int64  
 16  web_purchases         2240 non-null   int64  
 17  catalog_purchases     2240 non-null   int64  
 18  instore_purchases     2240 non-null   int64  
 19  web_visits_per_month  2240 non-null   int64  
 20  is_campaign_accepted_3  2240 non-null   int64  
 21  is_campaign_accepted_4  2240 non-null   int64  
 22  is_campaign_accepted_5  2240 non-null   int64  
 23  is_campaign_accepted_1  2240 non-null   int64  
 24  is_campaign_accepted_2  2240 non-null   int64  
 25  complained            2240 non-null   int64  
 26  z_cost_to_contact     2240 non-null   int64  
 27  z_profit              2240 non-null   int64  
 28  last_campaign_accepted  2240 non-null   int64  
dtypes: float64(1), int64(25), object(3)
memory usage: 507.6+ KB
```

Рис 2.3 – результат виконання команди info

```
data.describe()
```

	id	birth_year	salary	kids_count	teens_count	last_bought	wines_amount	fruits_amount	meat_amount	fish_amount	...	web_vis
count	2240.000000	2240.000000	2216.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	2240.000000	...	
mean	5592.159821	1968.805804	52247.251354	0.444196	0.506250	49.109375	303.935714	26.302232	166.950000	37.525446	...	
std	3246.662198	11.984069	25173.076661	0.538398	0.544538	28.962453	336.597393	39.773434	225.715373	54.628979	...	
min	0.000000	1893.000000	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
25%	2828.250000	1959.000000	35303.000000	0.000000	0.000000	24.000000	23.750000	1.000000	16.000000	3.000000	...	
50%	5458.500000	1970.000000	51381.500000	0.000000	0.000000	49.000000	173.500000	8.000000	67.000000	12.000000	...	
75%	8427.750000	1977.000000	68522.000000	1.000000	1.000000	74.000000	504.250000	33.000000	232.000000	50.000000	...	
max	11191.000000	1996.000000	666666.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000	...	

8 rows × 26 columns

Рис 2.4 – результат виконання команди info

Після проведеного первинного аналізу було вирішено видалити з набору даних стовпці “z_cost_to_contact” та “z_profit”. Дані в цих стовпчиках не були описані в наборі[1], та значення в полях цих стовпців повторюються, тому вони не нестимуть ніякої цінності для побудови нашої моделі.

```
data = data.drop(['z_cost_to_contact', 'z_profit'],axis=1)
data.head()
```

campaign_accepted_3	is_campaign_accepted_4	is_campaign_accepted_5	is_campaign_accepted_1	is_campaign_accepted_2	complained	last_campaign_accepted
0	0	0	0	0	0	1
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Рис 2.5 – процес видалення стовпців “z_cost_to_contact” та “z_profit”

З рисунку 2.5 можемо побачити, що процес видалення даних за допомогою методу drop був успішним, та дані більше не представляються при використанні команди head.

Далі встановлюємо для matplotlib параметри за замовчуванням, як показано на рисунку 2.6. Цю бібліотеку ми в подальшому будемо використовувати побудови графіків.

```
plot.rcParams.update(plot.rcParamsDefault)
```

Рис 2.6 – встановлення для matplotlib параметрів за замовчуванням.

Важливим аспектом аналізу даних є розуміння взаємозв'язків між різними характеристиками. Одним з інструментів, що використовуються для цього, є матриця кореляції.

На початку визначаємо числові стовпці в датафреймі за допомогою методу select_dtypes. Цей метод використовується для вибору стовпців в датафреймі на основі типу даних. В даному випадку вибрані стовпці, що містять числові дані.

Після цього, для цих числових стовпців було розраховано матрицю кореляції за допомогою методу corr в pandas[7]. Матриця кореляції - це таблиця, в якій кожний елемент (i, j) відображає ступінь кореляції між і-м та j-м стовпцями в наборі даних. Значення кореляції варіюють від -1 до 1, де -1

означає повну обернену кореляцію, 0 означає відсутність кореляції, а 1 означає повну пряму кореляцію.

Для візуалізації матриці кореляції було використано теплокарту[7] з бібліотеки seaborn. Теплокарта - це двовимірне графічне представлення даних, де різні відтінки кольорів відповідають різним значенням характеристик. В цьому випадку, кольори відповідають різним значенням кореляції.

Теплокарта допомагає зрозуміти взаємозв'язки між різними характеристиками на одному графіку, визначити які характеристики сильно корелюють одна з одною, і відповідно, можуть впливати одна на одну, що допоможе нам в виборі характеристик для моделі машинного навчання.

Візуалізацію теплокарти показано на рисунку 2.7:

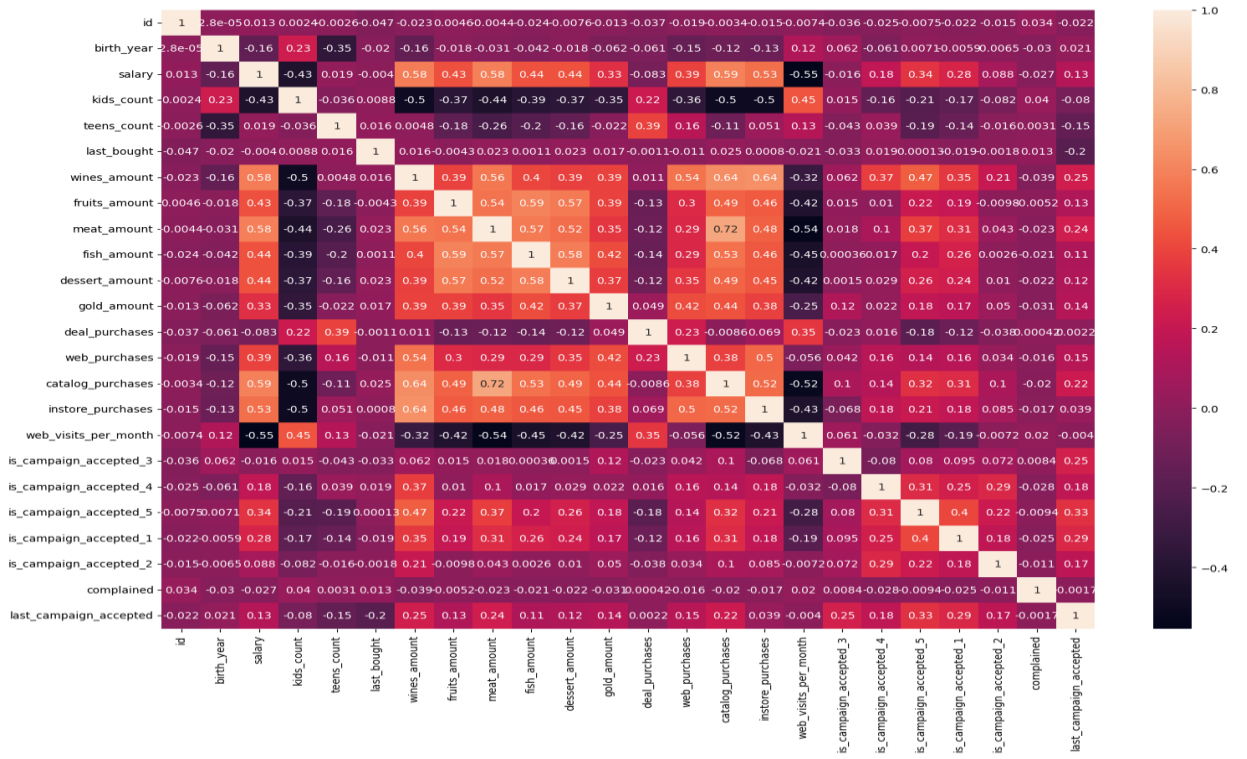


Рис 2.7 – теплокарта

Ступінь кореляції між змінними має велике значення, особливо при розробці моделей машинного навчання[3], адже це впливає на точність передбачення моделі. Сильна кореляція між змінними може свідчити про те,

що одна змінна може бути визначена через іншу, що може призвести до проблеми мультиколінеарності. Мультиколінеарність виникає, коли дві або більше незалежних змінних в регресійній моделі сильно корелюють, що може зіпсувати оцінки параметрів та погіршити здатність моделі передбачувати результат.

У цій частині роботи, спочатку знову було виділено числові стовпці в датафреймі за допомогою методу select_dtypes. Після цього, була розрахована матриця кореляції для цих стовпців. Щоб знайти пари змінних з найвищою кореляцією, матриця кореляції була перетворена в одномірний масив за допомогою методу, а потім відсортована за спаданням за допомогою sort_values. Значення кореляції були взяті по модулю, щоб зосередитися на силі взаємозв'язку, а не на його напрямку. Наведені кроки показано на рисунку 2.8.

```
numeric_columns = data.select_dtypes(include=[numpy.number]).columns
corr_data = data[numeric_columns].corr().abs().unstack().sort_values(ascending=False)[24:50:2]

print(corr_data)
```

```
meat_amount        catalog_purchases       0.723827
instore_purchases  wines_amount            0.642100
catalog_purchases  wines_amount            0.635226
fruits_amount      fish_amount             0.594804
salary             catalog_purchases       0.589162
                   meat_amount             0.584633
fish_amount        dessert_amount          0.579870
salary             wines_amount            0.578650
fish_amount        meat_amount             0.568402
dessert_amount     fruits_amount           0.567164
meat_amount        wines_amount            0.562667
salary             web_visits_per_month    0.553088
meat_amount        fruits_amount           0.543105
dtype: float64
```

Рис 2.8 знаходження пар змінних з найвищою кореляцією

Після сортування, було вибрано 13 пар змінних з найвищою кореляцією. Це було зроблено шляхом вибору кожного другого елементу з 24 по 50 (включно) з відсортованого списку. Вибір кожного другого елементу забезпечує, що кожна пара змінних не включається двічі, так як кореляція між A та B та між B та A є однаковою.

На практиці, дані рідко бувають чистими і готовими до аналізу. Вони часто містять пропущені значення, які потребують опрацювання[3]. Пропущені дані можуть викривити результати аналізу або змусити статистичні програми припинити роботу. Однак існує багато способів опрацювання пропущених даних, в залежності від природи даних і конкретного дослідницького запиту.

У цьому кроці, пропущені значення в стовпці “salary” були замінені, як показано на рисунку 2.9, на середнє значення цього стовпця. Цей метод відноситься до категорії методів введення значень, тобто пропущені значення замінюються екстрапольованими значеннями. Заміна пропущених значень на середнє - це простий і ефективний спосіб заповнення пропусків, який не вимагає складних обчислень або додаткових даних.

```
data['salary'] = data['salary'].fillna(data['salary'].mean())
data.isna().any()
```

Рис 2.9 – заміна пустих значень в стовпчику доходу на медіану

Втім, цей метод також має свої недоліки. Він використовує середнє значення, і не враховує можливі зміни в даних. Крім того, він може змінити розподіл даних та вплинути на варіацію даних, що може змінити результати статистичного аналізу. Тому цей метод варто використовувати тільки після ретельного огляду даних та розуміння значень та їх тенденцій.

У багатьох випадках, даними можуть бути категорійні змінні, які представляють різні групи чи класи. Одним із типів категорійних даних є номінальні дані[3], де немає порядку чи ієрархії між різними категоріями. В цьому випадку, однією з номінальних змінних в датасеті є “relationship_status”, який відображає статус шлюбу споживача. В цьому дослідженні, деякі категорії в цьому стовпці були об'єднані(рисунок 2.10) в більш загальні категорії, 'relationship' і 'Single' для спрощення моделі та підвищення точності

прогнозування.

```
data['relationship_status'] = data['relationship_status'].replace(['Married', 'Together'],'relationship')
data['relationship_status'] = data['relationship_status'].replace(['Divorced', 'Widow', 'Alone', 'YOLO', 'Absurd'],'Single')
```

```
data['relationship_status'].value_counts()
```

```
relationship_status
relationship    1444
Single           796
Name: count, dtype: int64
```

```
marital_counts = data['relationship_status'].value_counts()

plot.figure(figsize=(4, 4))
seaborn.barplot(x=marital_counts.index, y=marital_counts)
plot.show()
```

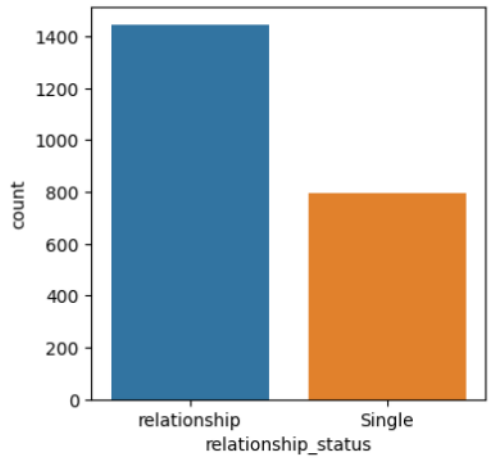


Рис 2.10 – об’єднання категорій в стовпці “relationship_status”

Витрати споживачів на різні товари є важливою інформацією, яка допомагає нам розуміти їхні вподобання та звички покупки. Це, в свою чергу, може відігравати ключову роль в плануванні маркетингових стратегій, оптимізації інвентаризації та покращенні рівня обслуговування клієнтів.

З цією метою було створено новий датафрейм “products_data”, що включає витрати споживачів на різні товари, такі як вина, фрукти, м'ясні продукти, рибні продукти, солодощі та золоті вироби. Кожен рядок в цьому датафреймі відповідає окремому споживачу, а стовпці відображають витрати цього споживача на кожний з товарів. В подальшому цей набір даних будемо використовувати для аналізу вподобань користувачів та їх впливу на вибір товарів. Процес створення набору описано на рисунку 2.11.

```
product_data = []
for i in range(0, len(data)):
  productdata = [data['wines_amount'][i], data['fruits_amount'][i],
                 data['meat_amount'][i], data['fish_amount'][i],
                 data['dessert_amount'][i], data['gold_amount'][i]]
  product_data.append(productdata)
products_data = pandas.DataFrame(product_data, columns = ['Wines', 'Fruits', 'Meat', 'Fish', 'Sweets', 'Gold'])
products_data.head()
```

	Wines	Fruits	Meat	Fish	Sweets	Gold
0	635	88	546	172	88	88
1	11	1	6	2	1	6
2	426	49	127	111	21	42
3	11	4	20	10	3	5
4	173	43	118	46	27	15

Рис 2.11 – створення набору даних з витратами на різні види товарів

Далі, для зменшення розміру датафрейму та підвищення ефективності подальшого аналізу згрупуємо(рисунок 2.12) деякі стовпці в сумарні стовпці[3]. Стовпцями, що підходять для такого групування є “kids_count”, “teens_count” – об’єднуємо в “Kids”; “wines_amount’, “fruits_amount”, “meat_amount”, “fish_amount", “dessert_amount", "gold_amount” – в “Expenses” ; “is_campaign_accepted_1”, “is_campaign_accepted_2”, “is_campaign_accepted_3”, “is_campaign_accepted_4”, “is_campaign_accepted_5”, “last_campaign_accepted” – в “total_accepted_campaigns”; “web_purchases”, “catalog_purchases”, instore_purchases”, “deal_purchases” в “total_purchases”, та видаляємо незгруповані стовці з набору даних.

```
data['Kids'] = data['kids_count'] + data['teens_count']
data['Expenses'] = data['wines_amount'] + data['fruits_amount'] + data['meat_amount'] + data['fish_amount'] + data['dessert_amou
data['total_accepted_campaings'] = data['is_campaign_accepted_1'] + data['is_campaign_accepted_2'] + data['is_campaign_accepted_3
data['total_purchases'] = data['web_purchases'] + data['catalog_purchases'] + data['instore_purchases'] + data['deal_purchases']
```

```
col_del = ["is_campaign_accepted_1" , "is_campaign_accepted_2", "is_campaign_accepted_3" , "is_campaign_accepted_4","is_campaign_
data=data.drop(columns=col_del,axis=1)
data.head()
```

	id	birth_year	education_level	relationship_status	salary	customer_since	last_bought	complained	Kids	Expenses	total_accepted_campaings	total_pu
0	5524	1957	Graduation	Single	58138.0	04-09-2012	58	0	0	1617	1	
1	2174	1954	Graduation	Single	46344.0	08-03-2014	38	0	2	27	0	
2	4141	1965	Graduation	relationship	71613.0	21-08-2013	26	0	0	776	0	
3	6182	1984	Graduation	relationship	26646.0	10-02-2014	26	0	1	53	0	
4	5324	1981	PhD	relationship	58293.0	19-01-2014	94	0	1	422	0	

Рис 2.12 – групування стовпців

Наступним кроком є консолідація освітнього статусу, як показано на рисунку 2.13. Замість використання великої кількості різних категорій, таких як 'PhD', '2n Cycle', 'Graduation', 'Master' та 'Basic', їх можна розділити на дві групи: 'PG' (післядипломна освіта) та 'UG' (додипломна освіта). Знову ж таки, цей крок є важливим для покращення загальної ясності та виразності набору даних.

```
data['education_level'] = data['education_level'].replace(['PhD','2n Cycle','Graduation', 'Master'],'PG')
data['education_level'] = data['education_level'].replace(['Basic'], 'UG')
```

Рис 2.13 – консолідація навчального статусу

У дослідницькому аналізі даних також важливо ретельно оцінювати, які стовпці даних є необхідними, а які можна видалити без втрати значущої інформації[3]. Аналізуючи наш датафрейм, ми виявили можливість оптимізації шляхом перетворення формату даних та видалення непотрібних стовпців.

На рисунку 2.14 показані проведені для цього процедури.

```
data['customer_since'] = pandas.to_datetime(data['customer_since'], format='%d-%m-%Y')
data['first_day'] = '01-01-2015'
data['first_day'] = pandas.to_datetime(data['first_day'], format='%d-%m-%Y')
data['day_engaged'] = (data['first_day'] - data['customer_since']).dt.days
```

```
data=data.drop(columns=["id", "customer_since", "first_day", "birth_year", "customer_since", "last_bought", "complained"],a
```

Рис 2.14 – оптимізація стовпчиків даних

Далі, використовуючи візуалізацію, ми досліджуємо взаємозв'язок між шлюбним статусом та витратами в контексті рівня освіти. На рисунку 2.15 ці дані представлені у вигляді вертикальної стовпчатої діаграми, що дозволяє нам відобразити витрати на осі Y і шлюбний статус на осі X.

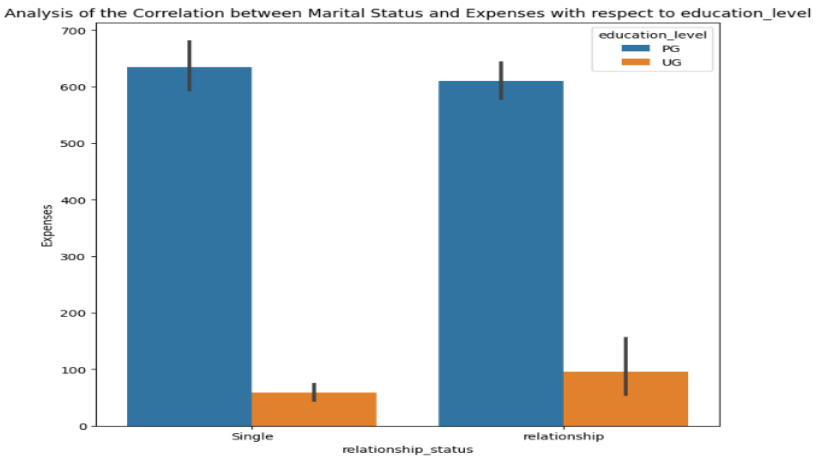


Рис 2.15 – відношення витрат до сімейного статусу в контексті рівня освіти

На рисуку легко можна помітити, що люди з післядипломною освітою та в стосунках витрачають суттєво більше. Це може свідчити про високий ступінь довіри до маркетингових кампаній або більшу схильність до покупок, особливо в сім'ях, де обидва партнери мають вищу освіту. Також, з рисунку 2.16 можна зрозуміти, що люди з вищим рівнем освіти мають зачно вищий дохід, що є важливим фактором, що впливає на доходи, та може зумовлювати більші витрати.

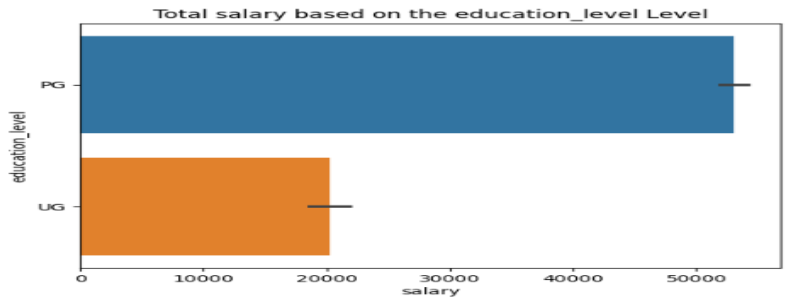


Рис 2.16 – кореляція рівню доходу та отриманої освіти

У деяких аналітичних задачах, зокрема в машинному навчанні, потребується перетворення категоріальних даних в числові[3], оскільки багато моделей можуть працювати лише з числовими даними.

По-перше, нам потрібно визначити, які з наших стовпців є категоріальними. Для цього ми використовуємо цикл, який проходить через всі стовпці в датафреймі, і перевіряє тип даних кожного стовпця. Якщо тип

даних стовпця вказує на "об'єкт", це зазвичай означає, що це категоріальний стовпець, і ми додаємо його в список категоріальних стовпців.

Після того, як ми ідентифікували всі категоріальні стовпці, ми можемо перейти до їх кодування. Для цього ми використовуємо інструмент LabelEncoder з бібліотеки Scikit-learn[7]. Цей інструмент працює шляхом призначення кожному унікальному значенню в категоріальному стовпці унікального числового ідентифікатора.

Це ключовий крок в підготовці даних, що дозволяє нам провести більш складні аналітичні методи на наших даних у подальшому. Ми маємо гарантувати, що ці перетворення виконуються коректно, оскільки вони можуть суттєво вплинути на результати нашого аналізу. Водночас, важливо пам'ятати про потребу інтерпретації цих кодованих значень у контексті наших вхідних даних. Завдяки цим крокам, ми забезпечуємо сумісність нашого набору даних з широким спектром аналітичних інструментів та методів. Виконання наведених вище кроків показано на рисунку 2.17.

```
cate = []
for i in data.columns:
    if (data[i].dtypes == "object"):
        cate.append(i)

print(cate)
```

```
['education_level', 'relationship_status']
```

```
lbl_encode = LabelEncoder()
for i in cate:
    data[i]=data[[i]].apply(lbl_encode.fit_transform)
```

Рис 2.17 – перетворення категоріальних значень у числові

Наступним кроком в нашому дослідженні є обробка викидів. Викиди - це значення, які суттєво відрізняються від інших значень в наборі даних[3]. Вони можуть бути результатом помилок введення даних, або можуть відображати реальні, але нехарактерні відхилення в даних. Якщо викиди є результатом помилок, їх слід виправити або вилучити; якщо вони є нехарактерними, але дійсними спостереженнями, вони можуть вплинути на оцінки параметрів статистичних моделей або на інші аспекти аналізу даних.

Для виявлення викидів ми використовуємо boxplot[7] (ящик з вусами), це потужний графічний інструмент для візуального виявлення викидів та оцінки розподілу даних. На цьому графіку "ящик" представляє межі між першим та третім квартилями даних, з "вусами", що вказують на мінімум та максимум даних, виключаючи викиди. Викиди зображуються окремими точками або зірочками, що виходять за межі вусів.

У цьому випадку, ми створюємо boxplot, як показано на рисунку 2.18, для всього нашого набору даних, використовуючи seaborn. Ми робимо це для кожного стовпця в датафреймі, що дає нам можливість одночасно переглянути розподіл кожного атрибуту та виявити можливі викиди.

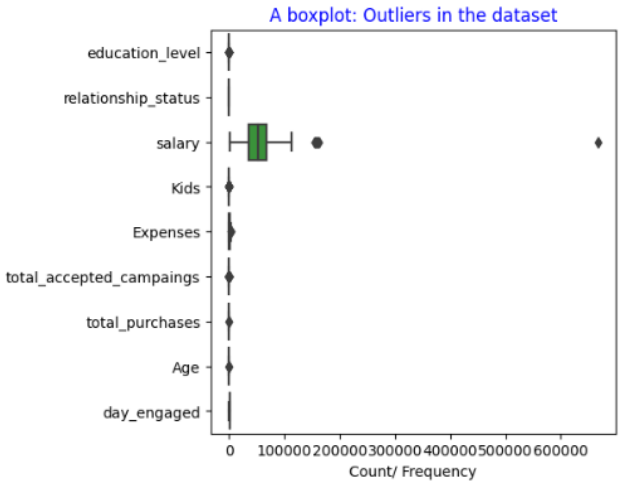


Рис 2.18 – візуалізація викидів в наборі даних

Хоча boxplot дає нам багато інформації, важливо пам'ятати, що він не дає конкретних значень для викидів. Саме тому в наступному кроці, для того щоб ідентифікувати викиди ми використаємо метод міжквартильного розмаху (IQR)[7], щоб ідентифікувати та вилучити викиди з нашого набору даних. IQR - це статистичний параметр, який вимірює статистичний розкид значень в наборі даних, який визначається як різниця між третім квартилем (Q3) та першим квартилем (Q1).

Для цього на першому кроці ми ідентифікуємо всі числові стовпці в нашому наборі даних. Застосовуючи метод select_dtypes, ми вибираємо тільки ті стовпці, які мають числовий тип даних. Потім ми обчислюємо IQR для кожного із цих числових стовпців, використовуючи метод quantile з pandas[7]. Метод quantile використовується для обчислення значення квартиля в відповідному стовпці даних. Ми обчислюємо Q1 та Q3 для кожного стовпця та визначаємо IQR як їх різницю. Потім використовуємо IQR для визначення нижнього та верхнього діапазонів викидів для кожного стовпця. Значення, які знаходяться за межами цих діапазонів, вважаються викидами. На рисунку 2.19 показане виконання вище описаних кроків.

```
numeric_columns = data_copy.select_dtypes(include=numpy.number).columns
q3 = data_copy[numeric_columns].quantile(0.75)
q1 = data_copy[numeric_columns].quantile(0.25)
iqr = q3 - q1

lower_range = q1 - (1.5 * iqr)
upper_range = q3 + (1.5 * iqr)

data_copy = data_copy[~((data_copy[numeric_columns] < lower_range) | (data_copy[numeric_columns] > upper_range)).any(axis=1)]
```

Рис 2.19 – обробка викидів в датафреймі

Таким чином, ми очищаємо наш набір даних від викидів та збільшуємо його релевантність та відповідність для подальшого аналізу та моделювання.

2.4 Кластеризація даних

У цьому розділі нашої роботи буде розглядатись безпосередньо процес тренування моделі. Однак, щоб отримати оптимальний результат, нам потрібно пройти ще декілька підготовчих кроків. Перш за все, нам потрібно провести кластеризацію даних[2].

Кластеризація - це процес групування даних у взаємозв'язані підгрупи, що називаються кластерами. Цей процес дозволяє нам розглядати великі набори даних у формі кластерів, що спрощує процес інтерпретації та аналізу. Ключовою метою кластеризації є забезпечення внутрішньої однорідності

(об'єкти всередині кластеру є подібними) та зовнішньої різноманітності (об'єкти з різних кластерів є різними).

Існують різні підходи до кластерного аналізу, розглянемо декілька з найбільш поширеніших методів[2]:

- K-середніх (K-Means) - цей алгоритм працює шляхом ініціалізації K "центрів кластерів" в даних, а потім ітеративно переміщує ці центри до тих пір, поки вони не стануть оптимальними. K-середніх є простим та ефективним алгоритмом для розмірних даних, але його основним недоліком є потреба вибрати кількість кластерів заздалегідь, а також він може бути чутливим до вихідних точок.

- Ієрархічна кластеризація - цей метод створює дерево кластерів, яке можна візуалізувати за допомогою дендрограми. Ієрархічна кластеризація може бути агломеративною (починається з окремих елементів та поступово об'єднує кластери) або дивізивною (починається з одного кластеру, що охоплює всі точки, і поступово розбиває кластери). Цей метод дозволяє нам візуалізувати структуру даних, але може бути обчислювально важким для великих наборів даних.

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise) - цей метод визначає кластери як області високої щільності, розділені областями низької щільності. DBSCAN є ефективним алгоритмом для виявлення форм кластерів будь-якого типу, і він не вимагає визначення кількості кластерів заздалегідь. Однак він може мати труднощі при знаходженні кластерів різної щільності.

- Метод розміщення очікувань (Expectation-Maximization Clustering) - цей метод є статистичним підходом, який використовує вероятностні моделі. Він зазвичай використовується з Гауссовими моделями кластерів і працює шляхом максимізації правдоподібності даних за умови моделі кластерів. Цей

метод може бути ефективним для оцінки параметрів моделі кластеру, але він може бути складним для використання та інтерпретації.

У цій роботі було обрано метод K-середніх з декількох причин. По-перше, він є простим у використанні і розумінні. По-друге, він швидкий і ефективний при обробці великих наборів даних, що робить його хорошим вибором для досліджень великих обсягів даних. Нарешті, результати методу K-середніх легко візуалізувати та інтерпретувати, що дозволяє дає нам краще розуміння набору даних.

Однак, перед використанням методу K-середніх потрібно знайти оптимальну кількість кластерів для нашого датасету. Для цього, знову ж таки, є вибір методів, також розглянемо декілька з них[2]:

- Метод ліктя - цей метод використовує варіацію всередині кластеру (WCSS) для різних значень K (кількості кластерів) і вибирає значення K, при якому варіація починає швидко зменшуватися, формуючи "лікть". Цей метод легко зрозуміти і використовувати, але його основним недоліком є те, що "лікть" не завжди очевидний або унікальний, особливо при складних або неоднорідних наборах даних.

- Аналіз силуету – метод, що оцінює якість кластеризації шляхом вимірювання відстані між кластерами та внутрішньої щільності кластерів. Він надає кількісну оцінку того, наскільки добре кожна точка була кластеризована. Цей метод може бути корисним при визначенні оптимальної кількості кластерів, але він може займати велику кількість часу для великих наборів даних.

- Метод щільності піку - вирішує проблему вибору K шляхом визначення центрів кластерів як піків щільності даних. Основним недоліком цього методу є його вимога до форми кластерів, і він може погано працювати, якщо реальні кластери в даних мають нерегулярні або складні форми.

- Метод власних значень матриці розсіювання - метод використовує власні значення матриці розсіювання для визначення кількості кластерів. Це може бути корисно для виявлення неочевидних структур в даних, однак метод вимагає глибокого розуміння лінійної алгебри і може бути складним для використання у практичних задачах.

Загалом, немає універсального методу знаходження оптимальної кількості кластерів. Вибір залежить від природи даних, поставлених цілей, доступних ресурсів та наявних знань. В цій роботі було обрано метод ліктя та аналіз силуету через їх простоту і зручність візуалізації. Вони обидва надають нам інтуїтивно зрозумілі критерії для визначення кількості кластерів.

Отож, повертаючись до нашого завдання, спочатку ми обчислюємо суму квадратів відстаней від кожної точки до її центроїду в межах кластеру (WCSS) для різної кількості кластерів від 1 до 10. Потім ми обчислюємо WCSS, використовуючи атрибут 'inertia_' об'єкта KMeans з бібліотеки sklearn та зберігаємо його в список. Рисунок 2.20 візуалізує зміну WCSS з ростом кількості кластерів. Якщо графік зміни має форму 'ліктя', оптимальна кількість кластерів знаходиться на 'згині' графіку, де зміна WCSS стає менш вираженою.

```
data_numeric = data_copy[numeric_columns]

wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(data_numeric)
    wcss.append(kmeans.inertia_)

plot.figure(figsize=(16, 8))
plot.plot(range(1, 11), wcss, 'bx-')
plot.title('The Elbow Method')
plot.xlabel('Number of clusters')
plot.ylabel('WCSS')
plot.show()
```

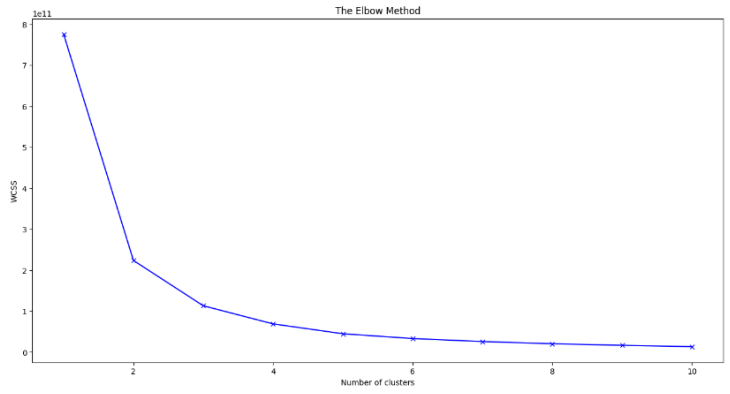


Рис 2.20 – метод ліктя

Оскільки з результату виконання методу кількість потрібних кластерів не є цілком очевидною, ми використаємо аналіз силуету для додаткової оцінки оптимальної кількості кластерів. Цей аналіз вимірює відстань між кластерами і відстань між точками всередині кластера. Чим більше значення силуету, тим краще точки згруповані всередині кластерів і відокремлені від суміжних кластерів. Ми визначаємо кількість кластерів як один плюс індекс найбільшого значення силуету у списку. Візуалізацію методу показано на рисунку 2.21.

Рис 2.21 – метод аналізу силуету

У контексті виконаної роботи, метою кластеризації було виявлення прихованих шаблонів та структур у даних про споживачів, що допомагає в ідентифікації різних сегментів аудиторії та надає глибше розуміння їх властивостей та взаємозв'язків, а також гратиме важливу роль в покращенні точності вихідної моделі.

Наступним кроком є використання натренованої моделі K-середніх для власне розподілу даних на кластери. Ми використовуємо метод predict[7] моделі методу, що приймає числовий датасет і повертає масив з мітками кластерів для кожного спостереження в датасеті.

Остаточним кроком є додавання міток кластерів до нашого початкового датафрейму. Ми створюємо новий стовпець cluster в data_copy, де кожному

рядку присвоюється мітка його кластера. Таким чином, кожне спостереження в нашому датафреймі тепер асоціюється з певним кластером. Результат кластеризації візуалізовано на рисунку 2.22.

```
kmeans = KMeans(n_clusters=number_of_clusters, random_state=42).fit(data_numeric)
pred = kmeans.predict(data_numeric)
data_copy['cluster'] = pred + 1
```

```
data_copy['cluster'].value_counts()
```

```
cluster
1    1018
2    1004
Name: count, dtype: int64
```

```
pl = seaborn.countplot(x=data_copy["cluster"])
pl.set_title("Distribution Of The Clusters")
plot.show()
```

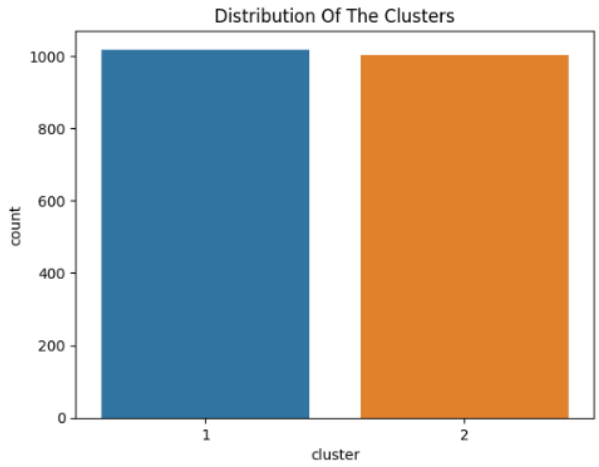


Рис 2.22 – результат кластеризації

Щоб оцінити якість нашої моделі кластеризації, використаємо матрицю невідповідностей (confusion matrix) та звіт про класифікацію[2]. Перш за все, матриця невідповідностей використовується для порівняння міток, присвоєних моделлю K-середніх, із реальними мітками. На рисунку 2.23 приведено результати вище наведених оцінок.

```
print("ConfusionMatrix \n",confusion_matrix(kmeans.labels_, pred))
print("classification report \n", classification_report(kmeans.labels_, pred))
```

```
ConfusionMatrix
 [[1018    0]
 [   0 1004]]
classification report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1018
           1       1.00      1.00      1.00      1004

    accuracy                           1.00      2022
   macro avg       1.00      1.00      1.00      2022
weighted avg       1.00      1.00      1.00      2022
```

Рис 2.23 – оцінка моделі кластеризації

Як можна побачити, всі оцінки дають нам значення в 1.00, що говорить про вкрай високу ефективність відпрацювання моделі методу K-середніх.

2.5 Асоціативний аналіз

Асоціативний аналіз – це процес дослідження набору даних, що вивчає взаємозв'язки між різними елементами в наборі[3]. Цей аналіз дозволяє виявляти шаблони, або асоціації, між різними продуктами або поведінкою користувачів, що може допомогти в управлінні продажами, маркетингом та стратегією продукту.

Для асоціативного аналізу існуть різні підходи та алгоритми, розглянемо декілька з них[3]:

- Алгоритм апріорі - цей метод базується на принципі, що підмножина частого набору предметів також є частою. Застосовуючи цей принцип, алгоритм апріорі значно зменшує простір пошуку, виключаючи набори предметів, що не задовольняють вимогу мінімальної підтримки.

- Алгоритм FP-росту (Frequent Pattern Growth) - цей метод є альтернативою алгоритму апріорі, оскільки він ефективніше обходить проблему розрідженості даних та великої кількості пошукових операцій, характерних для апріорі. FP-росту використовує структуру даних, відому як дерево FP, для зберігання контрольного списку транзакцій у стислому форматі.

- ECLAT (Equivalence Class Clustering and bottom-up Lattice Traversal) - цей метод використовує горизонтальний формат набору даних (у порівнянні з вертикальним форматом апріорі та FP-росту), що полегшує швидкі операції перетину. ECLAT швидший, але вимагає більше пам'яті для зберігання наборів даних.

Найпоширенішим методом асоціативного аналізу є алгоритм апріорі, який використовується для виявлення наборів елементів з найбільшою частотою у транзакційних наборах даних. Саме його ми і будемо використовувати в нашому дослідженні завдяки його високій ефективності та легкістю використання.

Асоціативний аналіз особливо корисний в областях, де потрібно аналізувати великі набори даних, таких як роздрібна торгівля, банківська сфера, медицина та електронна комерція. У контексті роздрібної торгівлі, наприклад, він може бути використаний для виявлення продуктів, які часто купуються разом, що може допомогти у плануванні маркетингових кампаній, упаковки продуктів або у впорядкуванні продуктів на полицях магазину.

Основна мета асоціативного аналізу полягає в тому, щоб надати організаціям глибше розуміння поведінки їх клієнтів, дозволивши їм краще відповідати на потреби своїх клієнтів та покращувати загальний досвід користувача.

В цій роботі, для проведення асоціативного аналізу було використано алгоритм апріорі, реалізований бібліотекою aryori[8]. На основі елементів з найбільшою частотою було встановлено правила асоціації. Застосовуючи ці правила, було здійснено пошук по даним для виявлення активних споживачів у різних категоріях продуктів, таких як вино, фрукти, м'ясо, риба, солодощі та золото. Для кожного з цих продуктів було знайдено відповідні набори предметів, що відображають асоціації з активними споживачами. Візуалізацію результатів показано на рисунку 2.24.

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhar
5875	(Age_group_Mature, salary_group_High salary, Fruits_segment_Highly Active Customer, Meat_segment_Highly Active Customer)	(Wines_segment_Highly Active Customer)	0.102	0.497	0.101	0.986	1.983	0.050	34.705	
8748	(Fruits_segment_Highly Active Customer, Age_group_Mature, Fish_segment_Highly Active Customer, salary_group_High salary, Meat_segment_Highly Active Customer)	(Wines_segment_Highly Active Customer)	0.093	0.497	0.092	0.984	1.980	0.046	31.687	

Рис 2.24 – активні споживачі в категорії «вино»

Ці процедури допомагають отримати корисний огляд поведінки споживачів, та є, у разі правильного використання, чудовим інструментом для удосконалення стратегій маркетингу та продажів. Зокрема, вони дозволяють визначити, які групи клієнтів найактивніше споживають певні продукти, що може слугувати основою для сегментації клієнтів та персоналізації маркетингових кампаній.

РОЗДІЛ 3. ПОБУДОВА МОДЕЛІ МАШИННОГО НАВЧАННЯ ДЛЯ ПЕРЕДБАЧЕННЯ ПОКУПАЦЬКОЇ ПОВЕДІНКИ КОРИСТУВАЧІВ

3.1 Моделі машинного навчання

При виборі моделі машинного навчання для вирішення конкретної проблеми, варто розглянути різні аспекти, такі як природа проблеми, тип даних, обчислювальні ресурси та потреба в інтерпретації. Давайте розглянемо деякі популярні моделі машинного навчання, що використовуються у класифікаційних проблемах[2]:

- Метод опорних векторів (SVM) - це дуже потужна та гнучка модель, яка використовує ядро для перетворення даних у вищі розмірності, де їх можна легко розділити. Вона здатна забезпечити високу точність класифікації для багатьох проблем, проте вона може бути досить вимогливою до ресурсів та важкою для інтерпретації, особливо при використанні не лінійних ядер.

- Дерева рішень та випадковий ліс - ці моделі використовують деревоподібні структури, щоб приймати рішення на основі даних. Вони відносно прості для інтерпретації, оскільки вони можуть бути відображені як набір "якщо-то" правил. Однак, вони можуть бути схильні до перенавчання, особливо якщо глибина дерева не регулюється. Випадковий ліс - це ансамблевий метод, що допомагає попередити цю проблему за рахунок усереднення результатів великої кількості дерев рішень.

- Градієнтний бустинг - це ще один ансамблевий метод, який об'єднує слабких учнів для створення сильного. Він використовує градієнтний спуск для мінімізації помилок і може бути дуже точним. Однак, він також може бути вимогливим до ресурсів та складним для налаштування, оскільки він має багато гіперпараметрів.

- Нейронні мережі - це особливо потужні моделі, які можуть моделювати складні не лінійні відносини. Вони використовуються в глибокому навчанні і можуть обробляти великі обсяги даних. Вони також можуть бути досить вимогливими до обчислювальних ресурсів та складними для інтерпретації.

- Логістична регресія - це добре відомий та широко використовуваний алгоритм машинного навчання, особливо підходящий для задач бінарної класифікації. Вона використовує логістичну функцію для моделювання ймовірності певного класу, тому вона може надавати не тільки прогнози класу, але й ймовірності, які можуть бути корисними для визначення ступеня впевненості в прогнозах.

У нашому випадку, логістична регресія має декілька переваг у порівнянні з іншими моделями, згаданими вище. По-перше, вона менш складна та менш обчислювально вимоглива, що, враховуючи природу наших даних, є кращим вибором. По-друге, вона відносно проста для інтерпретації, оскільки вона надає прямі ймовірності для кожного класу. По-третє, вона має великий набів ресурсів та інструментів, доступних для використання та налаштування, що є важливим аспектом в будь-якому дослідженні. Вчетверте, найголовніше, має високу ефективність при розробці моделей, що повертають бінарний результат, що співпадає з нашою метою.

Важливо відзначити, що незважаючи на переваги логістичної регресії в цьому конкретному випадку, інші моделі можуть мати вищу ефективність в залежності від конкретних обставин. Наприклад, якщо до задачі на вхід подається складний набір даних з великою кількістю нелінійних відносин, можливо, краще використовувати нейронну мережу. Якщо в наявності є ресурси та час для налаштування моделі, градієнтний бустинг може бути хорошим вибором. Однак для цієї конкретної задачі логістична регресія виявилась достатньою та ефективною.

3.2 Побудова моделі для передбачення поведінки клієнтів

Наступним кроком є власне розробка моделі машинного навчання, що здатна з високою точністю передбачати покупацьку поведінку користувачів. Ми розглянемо процедури, які були виконані для досягнення цієї мети.

Першим чином, виконаємо поділ наших даних на фактори (x) та цільову змінну (y)[7], в якій 'cluster' відіграє роль цільової змінної. Фактори - це змінні, на основі яких ми хотіли б передбачити покупацьку поведінку користувачів, тоді як 'cluster' представляє групу, до якої належить користувач на основі його покупацької поведінки.

Далі розділяємо дані на тренувальний та тестовий набори. За допомогою функції train_test_split з набору sklearn, ми забезпечили розподіл наших даних на тренувальний набір (80% від загальної кількості даних) та тестовий набір (20% від загальної кількості даних). Застосування параметра random_state=2[7] гарантує відтворюваність результатів.

Потім було виконано масштабування даних за допомогою StandardScaler з тої ж бібліотеки sklearn. Цей процес приводить всі ознаки до схожого масштабу, з середнім значенням 0 та стандартним відхиленням 1, що є важливою передумовою для багатьох алгоритмів машинного навчання.

Після цього, було створено модель логістичної регресії за допомогою LogisticRegression, та навчено її на тренувальних даних (x_train, y_train).

Після тренування моделі, ми використали її для передбачення на тестовому наборі даних. Отримані передбачення (y_predicted) були потім порівняні з дійсними цільовими значеннями за допомогою різних метрик.

За допомогою функції classification_report, ми отримали детальний звіт про продуктивність нашої моделі, включаючи точність, повноту та f1-оцінку для кожного класу. Додатково, ми обчислили загальну точність та F1-оцінку за допомогою accuracy_score та f1_score відповідно.

Нарешті, ми створюємо матрицю невідповідностей для візуалізації помилок нашої моделі. Такий вигляд дозволяє нам бачити, які класи модель переплутує, та допомагає визначити можливі напрямки для подальшого удосконалення моделі.

Оцінку передбачення моделі наведено на рисунку 3.1.

```
Classification Report:
              precision    recall  f1-score   support

           1       0.99      0.99      0.99       200
           2       1.00      1.00      1.00       205

    accuracy                           1.00       405
   macro avg       1.00      1.00      1.00       405
weighted avg       1.00      1.00      1.00       405

--------------------------------------------------------------------------------------------------------

Accuracy Score:  0.9950617283950617
--------------------------------------------------------------------------------------------------------

F1 Score:  0.995
--------------------------------------------------------------------------------------------------------

Confusion Matrix:
```

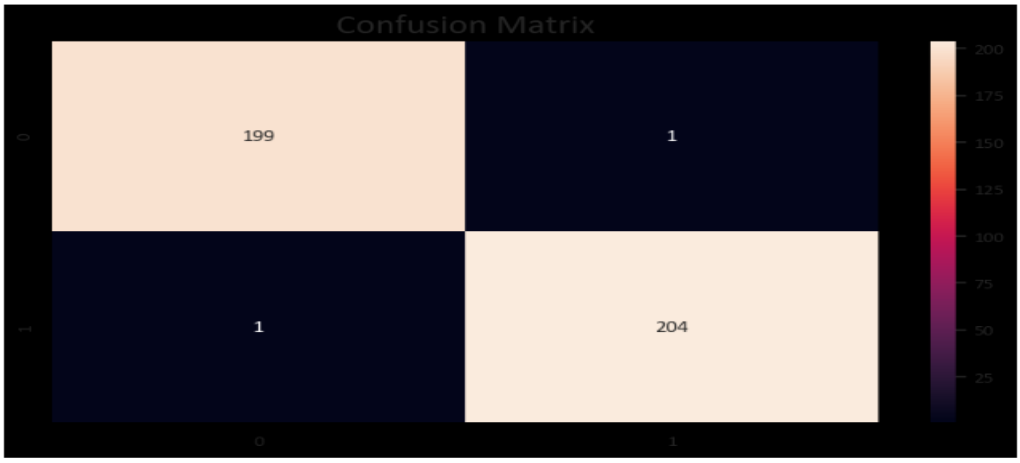


Рис 3.1 – оцінка передбачення моделі

З рисунку можна побачити, що показники близькі до одиниці, що є чудовим результатом та доводить, що ця модель здатна з високою точністю оцінювати покупацьку поведінку та блискуче підходить для задач оптимізації цільової аудиторії маркетингових кампаній, або в інших завданнях, які може мати підприємство, що потребують звуження кола потенційних клієнтів.

ВИСНОВКИ

Проаналізувавши набір даних і застосувавши на ньому різноманітні техніки обробки даних, включаючи кластерний аналіз, машинне навчання та аналіз асоціаційних правил, було отримано цінні висновки, що допоможуть підприємству краще розуміти та обслуговувати своїх клієнтів.

- Проведено сегментацію клієнтів за різними категоріями, що дозволило краще розуміти профіль клієнтів та їх вподобання.

- Досліджено попит на різного роду продукти з використанням алгоритмів кластеризації. Метою було виокремити схожість в поведінці покупців та виявити групи з високим та низьким рівнем активності у певних сегментах продуктів.

- Застосовано аналіз асоціативних правил для виявлення взаємозв'язків між різними продуктами. Це дало можливість зрозуміти, які товари часто купуються разом, що може використовуватися для розробки стратегій маркетингу та прогнозування об’єму продажів.

- Застосовано алгоритми машинного навчання для прогнозування поведінки клієнтів з високою точністю.

- Розроблено інтерактивний веб-додаток що з легкістю дозволяє підприємтсву отримати згрупований за купівельою поведінкою список користувачів

Результати проведеного дослідження дозволяють ефективну розробку спеціалізованих таргетованих маркетингових стратегій з високим рівнем реагування користувачів. Такі стратегії можуть включати персоналізовані рекомендації продуктів, таргетингову рекламу та розвиток продуктового асортименту, що відповідає вподобанням клієнтів.

Джерела

[1] Dr. Omar Romero-Hernandez, University of California, Berkeley. Marketing campaign dataset, 2015.

[2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT press.

[3] Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, 2016.

[3] https://pandas.pydata.org/ Pandas documentation

[4] https://seaborn.pydata.org/ Seaborn documentation

[5] https://scikit-learn.org/ Sklearn documentation

[6] https://aws.amazon.com/marketplace/pp/prodview-eslw7m6vhendg Python AI & Machine learning kit

[7]Liu Y. “Python Machine Learning By Example”, 2020.

[8] https://pypi.org/project/apyori/ - Apyori