Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет автоматизації та комп'ютерно-інтегровані технології
(повне найменування інституту, назва факультету (відділення))

Кафедра автоматизації та інтелектуальних інформаційних технологій
(повна назва кафедри (предметної, циклової комісії)

Бакалаврська дипломна робота на тему:

«МОДЕЛЬ МАШИННОГО НАВЧАННЯ ДЛЯ АВТОМАТИЗАЦІЇ ПРОГНОЗУВАННЯ ПОПИТУ ТА УПРАВЛІННЯ РЕСУРСАМИ»

Виконав: студент 4-го курсу, групи 1АКІТ-20б спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології
(шифр і назва спеціальності)
Олександр САМАРСЬКИЙ
(ім'я та прізвище)
Керівник: к.т.н., доцент каф. АІІТ
Ярослав КУЛИК
(ім'я та прізвище)
«16» червня 2024 р.
Рецензент:PhD, старший викладач каф.САІТ
Ольга ВОЙЦЕХОВСЬКА
(ім'я та прізвище)
«16» червня 2024 р.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри АІІТ
«18» 06 2024 р.

Вінниця – 2024 року

Вінницький національний технічний університет

Факультет автоматизації та комп'ютерно-інтегровані технології

Кафедра автоматизації та інтелектуальних інформаційних технологій

Рівень вищої освіти перший бакалаврський

Галузь знань 15 – Автоматизація та приладобудування

Спеціальність 151 – Автоматизація та комп'ютерно інтегровані технології

Освітньо-професійна програма – Інтелектуальні комп'ютерні системи управління

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
_____Олег БІСІКАЛО
25 березня 2024 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Самарському Олександр Олександрович

1. Тема роботи– «Модель машинного навчання для автоматизації прогнозування попиту та управління ресурсами.» Керівник роботи: Кулик Ярослав Анатолійович, к.т.н., доцент, затверджені наказом вищого навчального закладу від
11 березня 2024 р. № 80

2. Строк подання студентом роботи 10 червня 2024 р.

3.Вихідні дані до роботи: інтегроване середовище розробки – Microsoft Visual Studio 2019; Python

4. Зміст текстової частини: вступ; аналіз та постановка задачі; розробка архітектури та алгоритмів програмного продукту; розробка програмного продукту; тестування програми, висновки; перелік посилань, додатки.

5. Перелік ілюстративного матеріалу: мета; об'єкт та предмет дослідження; діаграми варіантів використання; структурна схема компонентів додатку; загальний алгоритм роботи; приклади аналогів; тестування.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ 1-3	Кулик Я.А. к.т.н., доцент		
		12 березня	12 березня

7. Дата видачі завдання __12березня 2024 року_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз сучасного стану питання та обґрунтування завдання на роботу	26.03.2024 - 10.04.2024	Вик.
2	Розробка структури серверного додатку	11.04.2024- 25.04.2024	Вик.
3	Розробка алгоритмів програмного додатку	26.04.2024- 30.04.2024	Вик.
4	Розробка інтерфейсу програмного додатку	01.05.2024- 06.05.2024	Вик.
5	Програмна реалізація додатку	07.05.2024- 25.05.2024	Вик.
6	Тестування роботи додатку та алгоритму	26.05.2024- 01.06.2024	Вик.
7	Оформлення матеріалів до захисту БДР	01.06.2024- 10.06.2024	Вик.

Студент _____ Олександр САМАРСЬКИЙ
(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ Ярослав КУЛИК
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 77 сторінок формату А4 включаючи додатки, на яких є 15 рисунків, 3 таблиці, список використаних джерел містить 33 найменування.

Модель машинного навчання для автоматизації прогнозування попиту та управління ресурсами розроблена для оптимізації бізнес-процесів шляхом точного прогнозування майбутнього попиту на продукти або послуги. Використовуючи передові алгоритми аналізу даних, модель аналізує історичні дані, враховує сезонні коливання, тренди та інші релевантні фактори для генерування точних прогнозів. Ці прогнози допомагають компаніям ефективніше управляти запасами, персоналом та іншими ресурсами, зменшуючи витрати та покращуючи рівень обслуговування клієнтів. Завдяки автоматизації процесів прогнозування, модель забезпечує швидку адаптацію до змін ринкових умов, підвищуючи конкурентоспроможність та стійкість бізнесу.

Ключові слова: модель машинного навчання, автоматизація прогнозування попиту, управління ресурсами, оптимізація бізнес-процесів, точне прогнозування, алгоритми аналізу даних.

ANNOTATION

The bachelor's thesis consists of 77 pages of A4 format, including appendices, which contain 15 figures, 3 tables, the list of used sources contains 33 names.

A machine learning model for automating demand forecasting and resource management is designed to optimize business processes by accurately predicting future demand for products or services. Using advanced data analysis algorithms, the model analyzes historical data, takes into account seasonal fluctuations, trends and other relevant factors to generate accurate forecasts. These forecasts help companies manage inventory, personnel and other resources more efficiently, reducing costs and improving customer service. Thanks to the automation of forecasting processes, the model ensures rapid adaptation to changes in market conditions, increasing business competitiveness and sustainability.

Keywords: machine learning model, demand forecasting automation, resource management, business process optimization, accurate forecasting, data analysis algorithms.

ЗМІСТ

ВСТУП

У сучасному світі бізнесу ефективне управління ресурсами та прогнозування попиту є ключовими факторами для досягнення конкурентних переваг. Зростаюча складність ринкових умов, коливання споживчого попиту та глобалізація ставлять перед компаніями нові виклики в управлінні їхніми операціями. Традиційні методи прогнозування попиту, які базуються на історичних даних і простих статистичних моделях, часто виявляються недостатньо точними та адаптивними для швидко змінюваних умов.

Моделі машинного навчання (ММН) стають все більш популярними інструментами для вирішення цих проблем завдяки їх здатності обробляти великі обсяги даних та враховувати численні фактори, які впливають на попит. Використання ММН для автоматизації прогнозування попиту та управління ресурсами дозволяє компаніям отримувати точніші прогнози, знижувати витрати на зберігання і логістику, а також підвищувати рівень обслуговування клієнтів.

Розробка та впровадження ефективних моделей машинного навчання для прогнозування попиту є актуальною задачею для багатьох галузей економіки, включаючи роздрібну торгівлю, виробництво, логістику та сферу послуг. Враховуючи швидкий розвиток технологій і постійне зростання обсягів даних, дослідження у цій галузі мають велике значення для подальшого удосконалення процесів управління ресурсами і забезпечення стабільного розвитку підприємств.

В умовах постійно зростаючої конкуренції та швидких змін на ринку, компанії змушені шукати нові підходи до управління своїми ресурсами та прогнозування попиту. Традиційні методи, що базуються на аналізі історичних даних за допомогою простих статистичних моделей, не завжди забезпечують необхідну точність і гнучкість. Це призводить до неефективного використання ресурсів, надмірних запасів або, навпаки, дефіциту продукції, що негативно впливає на прибутковість та задоволення потреб клієнтів.

Сучасні технології машинного навчання пропонують нові можливості для вирішення цих проблем. Завдяки здатності аналізувати великі обсяги різнорідних даних та виявляти складні взаємозв’язки, моделі машинного навчання можуть значно покращити точність прогнозування попиту. Це, в свою чергу, дозволяє компаніям оптимізувати свої ланцюги постачань, знижувати витрати та підвищувати рівень обслуговування клієнтів.

Ця робота присвячена дослідженню методів машинного навчання, які можуть бути використані для автоматизації прогнозування попиту та управління ресурсами. У ній розглядаються основні принципи та підходи до створення моделей машинного навчання, аналізуються існуючі рішення та їхні переваги і недоліки, а також пропонуються рекомендації щодо впровадження таких моделей у практичну діяльність компаній.

Метою даного дослідження є дослідження ефективної моделі машинного навчання, яка дозволить підвищити точність прогнозування попиту і, як наслідок, покращити управління ресурсами підприємства. Для досягнення цієї мети потрібно вирішити наступні **задачі**:

- Проаналізувати історичні дані про продажі, аналіз їх якості та очищення від неточностей і пропусків.
- Розробити відповідні алгоритми машинного навчання та їх реалізації для побудови моделі прогнозування попиту.
- Оптимізувати параметри моделі для досягнення максимальної точності прогнозів.
- Провести тестування моделі на тестових даних для перевірки її ефективності та точності прогнозів.

Об’єктом дослідження є процеси управління ресурсами та прогнозування попиту в компаніях різних галузей економіки, таких як роздрібна торгівля, виробництво, логістика та сфера послуг. Ці процеси включають в себе аналіз даних, планування ресурсів, управління запасами та забезпечення своєчасного обслуговування клієнтів.

Предметом дослідження є моделі машинного навчання, які використовуються для автоматизації процесів прогнозування попиту та управління ресурсами. Це включає різні методи та алгоритми машинного навчання, такі як регресійні моделі, дерева рішень, нейронні мережі, а також підходи до збору, обробки та аналізу даних.

Методи дослідження

Для створення моделі машинного навчання для автоматизації прогнозування попиту та управління ресурсами було застосовано кілька методів дослідження:

- Збір великої кількості історичних даних про продажі, запаси, клієнтські запити та інші релевантні фактори. Дані були отримані з внутрішніх джерел компаній, відкритих баз даних та інших надійних джерел.
- Використовувалися методи очищення та нормалізації даних для усунення пропусків, аномалій та шумів. Також проводився аналіз кореляції для визначення важливих змінних.
- Застосовано різні алгоритми машинного навчання, такі як лінійна регресія, дерева рішень, випадковий ліс, градієнтний бустинг та нейронні мережі. Проводилось тренування моделей на підготовлених даних із застосуванням методів крос-валідації для оцінки точності прогнозів.
- Використовувалися метрики, такі як середня абсолютна похибка (MAE), середня квадратична похибка (MSE) та коефіцієнт детермінації (R^2) для оцінки точності та ефективності моделей.

Практична цінність моделі полягає в її здатності забезпечувати точні прогнози попиту, що дозволяє бізнесам оптимізувати управління ресурсами, зменшувати витрати на зберігання та транспорт, а також підвищувати рівень задоволеності клієнтів завдяки своєчасному задоволенню їх потреб.

1 ОГЛЯД ПРОГНОЗУВАННЯ ПОПИТУ НОВИХ ПРОДУКТІВ

1.1 Прогнозування попиту нових продуктів

Прогнозування попиту нових продуктів має унікальну природу порівняно з прогнозуванням продуктів, які вже є частиною ринку [6]. Зазначається, що найбільш характерним фактором є відсутність історичних даних для прогнозування нових продуктів [7]. Тому поширені методи прогнозування, такі як експоненціальне згладжування, метод Холта-Вінтерса або авторегресійне інтегроване ковзне середнє (ARIMA), не можуть бути застосовані до нових продуктів. Як результат, доводиться робити припущення, які базуються на інших факторах. Через брак даних для нових продуктів зазвичай застосовують якісні методи, що включають судження, досвід та інтуїцію.

Як зазначалося у вступі, прогнозування щодо нових продуктів є важливими для прийняття операційних рішень, таких як планування виробничих потужностей, закупівля та контроль запасів. На противагу цьому, використання аналітичних методів для прогнозування нових продуктів все ще обмежене. Моделі прогнозування, які аналітично передбачають попит на нові продукти та оцінюють невизначеності, набувають все більшого значення, оскільки компанії частіше впроваджують нові продукти [4].

Методи прогнозування нових продуктів можна поділити на такі, що ґрунтуються на судженнях (якісні) або на історичних даних (кількісні) [8]. Моделі, засновані на судженнях, базуються на думці експертів та зацікавлених сторін. Їхньою метою є створення прогнозів на основі досвіду, інтуїції та суджень. Дослідження споживачів та ринку фокусується на потенційній реакції цільової аудиторії за допомогою попередніх тестів, які зазвичай проводяться під час процесу розробки продукту [9]. Кількісні методи переважно базуються на прогнозуванні за допомогою аналогічних продуктів, завдяки чому частково долають проблеми відсутності історичних даних для обраного продукту [10].

Якісні підхід здатні подолати певну кількість проблем прогнозування попиту нових продуктів, проте за основу зазначених систем розглядаються кількісні методи. Перевага цих методів порівняно з методами суджень та дослідженням споживчого ринку полягає в тому, що для їхнього використання достатньо числових характеристик (історичні дані). Як результат, вони відносно економні в часі, що є важливим, коли збільшується кількість нових продуктів, які постійно потрібно виводити на ринок. Критично важливим припущенням при прогнозуванні на основі аналогічних продуктів є те, що схожість між продуктами призводить до схожої структури попиту. Тим не менш, немає жодних гарантій того, що історичний попит на аналогічні товари відповідає майбутньому попиту на нові товари [6].

1.1.1 Дифузійні моделі

Дифузійні моделі оцінюють темпи зростання попиту на продукт, враховуючи різні фактори, які впливають на споживачів. При цьому передбачається, що продукти відповідають певній моделі, яку можна пояснити параметрами ринку, наприклад, потенційним розміром ринку. Ці параметри для нового продукту можна оцінити за допомогою маркетингових досліджень, а також за допомогою продуктів-аналогів, параметри яких відомі. Найбільш домінуючою дифузійною моделлю для прогнозування попиту продуктів за останні десятиліття є модель Басса [11].

Модель Басса заслуговує певну увагу в літературі завдяки своїй простій структурі та високій результативності у поясненні ринкових факторів впливу на попит [12]. Модель робить попередній прогноз попиту протягом усього життєвого циклу продукту, оцінюючи дифузію.

Басс визначає дві групи споживачів: новатори та імітатори. Новатори вирішують прийняти інновацію незалежно від рішень інших індивідів у соціальній системі. Імітатори перебувають під впливом прийняття інновації іншими індивідами. Поведінка цих груп визначається коефіцієнтом інновацій

(p) та коефіцієнтом імітації (q). Разом з потенційним розміром ринку (m) модель прогнозує обсяги продажу продукту в часі:

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = p\left[m - N\left(t\right)\right] + \frac{q}{m}N\left(t\right)\left[m - N\left(t\right)\right], \qquad N\left(0\right) = 0, \quad t \geq 0, \text{(1.1)}$$

Ступінь подібності продуктів можна визначити на основі суджень, а також за допомогою статистичних методів або методів машинного навчання. Існують підходи з використанням ієрархічної байєсівської моделі, які використовують набір характеристик для оцінки параметрів дифузійної моделі та створення попереднього прогнозу попиту. Маючи обмежену кількість вхідних характеристик, модель здатна визначити інформацію про потенційний попит [14]. Серед інших можливих підходів, також впроваджені застосування аналізу моделі найближчого сусіда та регресійний аналіз для відбору аналогічних продуктів. Дослідження показують, що використання декількох аналогічних продуктів замість одного суттєво покращують результат прогнозування [4].

Крім різних факторів, які впливають на розмір ринку, існують й інші складнощі, пов’язані з моделлю Басса та іншими дифузійними моделями. Щоб запобігти зміщенням в оцінках параметрів, дані про попит аналогічних продуктів повинні включати різні частини життєвого циклу продукту (вихід продукту на ринок та проміжок часу, коли попит досягає свого піку). Дослідження демонструють, що цей фактор зменшує надійність прогнозів для продуктів, для яких історичні дані попиту аналогічних продуктів прокривають проміжки часу менше 5 років [7].

Крім того, Басс під час створення своєї моделі зосереджувався на прогнозуванні життєвого циклу продуктів, які рідко купують, і прогнозував попит лише на один рік. Дифузійні моделі в основному використовуються для моделювання нових технологій або нових для світу продуктів на рівні ринку (наприклад, впровадження електромобілів) [15]. Використання цих моделей для прогнозування попиту в період впровадження нових SKU (stock keeping

unit, або ідентифікатор товару) може призводити до значного зниження точності.

1.1.2 Пряме прогнозування попиту на основі аналогічних продуктів

Існують методи, які замість того, щоб оцінювати параметри моделі, виводять характеристики впливу на попит безпосередньо з даних аналогічних продуктів. Описані методи прогнозують загальний попит, структуру попиту або про-гнозують групу аналогічних товарів. Для аналізу аналогічних товарів використо-вуються статистичні методи та методи машинного навчання.

Загальний підхід полягає в тому, щоб порівняти новий продукт з низкою продуктів, які вже існують на ринку, і використати історичні дані цих продуктів як прогноз для нового продукту. Основним підходом до пошуку схожих товарів є використання статистичного методу, наприклад, методу найближчого сусіда на основі характеристик товару. Недоліком цього методу є те, що він знаходить аналогічні товари виключно за характеристиками продукту, і немає жодної га-рантії, що попит на них також є подібним. Крім того, всі характеристики мають однакову вагу, у той час як може існувати лише кілька значущих для прогнозу характеристик [16]. У результаті, певне людське судження все ще необхідне для визначення характеристик та відповідних продуктів-аналогів.

Для окремої частини системи, а саме визначення профілів аналогічних продуктів, у різних дослідженнях були використані методи машинного навчання, такі як кластеризація k-середніх, дерева рішень, класифікація нейронними мережами, наївна баєсівська класифікація та інші [17].

1.2 Методи машинного навчання

Машинне навчання – галузь знань, яка передбачає створення алгоритмів, здатних навчатися на основі даних без явного програмування. Воно спирається на статистичні моделі для розпізнавання закономірностей і прийняття

прогнозів або рішень на основі інформації, що міститься в даних. Зазначена галузь особ-ливо добре підходить для пошуку неявних зв'язків між характеристиками про-дукту і початковим попитом на нові продукти.

Основними для розгляду є алгоритми класифікації та регресії. Алгоритми класифікації можуть передбачати клас, наприклад, профіль або тип попиту, тоді як алгоритми регресії можуть передбачати безперервні значення, наприклад, попит на продукт. Обидва алгоритми є методами навчання з вчителем, що означає, що алгоритми навчаються як на вхідних, так і на вихідних даних [18].

У випадку зазначеної проблеми навчальний набір даних складатиметься з існуючих продуктів, з характеристиками продукту на вході та попитом на виході. У цьому підрозділі розглядаються найпоширеніші методи, які можна використо-вувати як для класифікації, так і для регресії. До них відносяться мжинна лінійна регресія, логістична регресія, дерева рішень, випадкові ліси, градієнтне підсилення, штучні нейронні мережі та опорно-векторні машини.

1.2.1 Множинна лінійна регресія

Лінійна регресія є основним регресійним методом для прогнозування зна-чення вихідної величини Y за вхідними параметрами *X = (X1 ; X2 ; … ; Xn).* Лінійна регресія з більш ніж одним вхідним параметром називається множинною лінійною регресією Залежність між вхідними параметрами X та вихідним пара-метром Y можна задати наступним чином:

$$Y = \beta_0 + \sum_{i=1}^{N} \beta_i X_i \quad ,(1.2)$$

Модель припускає, що функція регресії є або лінійною, або її можна до-статньо точно описати зазначеною апроксимацією [19]. Вхідні дані для лінійної регресії можуть бути як числовими, так і категоріальними.

Найпопулярнішим методом оцінки параметрів є метод найменших квадратів. Метод найменших квадратів визначає коефіцієнти таким чином, щоб залишкова сума квадратів була мінімальною [19]. Перевагою цього методу є його простота. Його легко інтерпретувати, оскільки для кожного вхідного числового параметра та категоріального рівня генерується частковий коефіцієнт регресії, який показує прямий вплив на Y. Однак лінійна регресія може лише апроксимувати лінійні зв'язки між вхідними та вихідними параметрами. Тому вона не є застосовною для моделювання складних зв'язків.

1.2.2 Логістична регресія

Хоча назва вказує на те, що логістична регресія є регресійним алгоритмом, вона використовується для класифікації бінарних або багатокласових змінних [20]. Логістична регресія використовує методи лінійної регресії для обчислення ймовір-ності належності до певного класу. Функція лінійного предиктора подібна до функції лінійної регресії:

$$f(x) = \beta_0 + \sum_{i=1}^{N} \beta_i X_i \tag{1.3}$$

Логістична регресія перетворює вихід цієї лінійної функції на логістичну функцію, як показано на рис 1.1. Логістична функція перетворює будь-яке значення лінійної функції на значення між нулем і одиницею, що є змодельованою ймовірністю належності до певного класу:

$$p_i = \frac{1}{1 + e^{-f(x)}}. \tag{1.4}$$

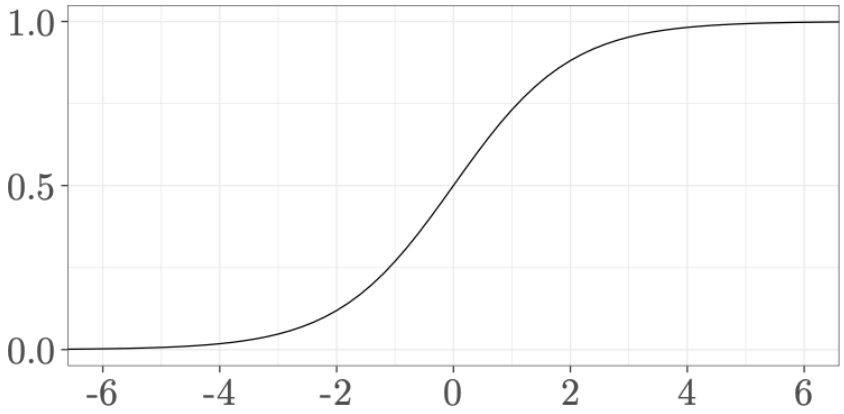

Рисунок 1.1 – Загальний вигляд логістичної регресії [20]

Класифікація з більш ніж двома класами називається мультиноміальною логістичною регресією. Як бінарна логістична регресія, так і мультиноміальна логістична регресія використовують оцінку максимальної правдоподібності для визначення коефіцієнтів [21]. Однією із можливих проблем логістичної регресії є перенавчання. Цей термін означає, що модель має високу точність для прогно-зування попередніх даних, але не її точність суттєво падає для нових даних. Логістична регресія може обробляти лише числові дані, а категоріальні дані повинні бути закодовані в числові значення. Прогнози моделі логістич-ної регресії можуть бути налаштовані шляхом регулювання порогового рівня ймовірності. Зазвичай поріг становить 0.5 для бінарної класифікації, де прогноз відноситься до класу 0, якщо ймовірність менша за 0.5, і до класу 1, якщо ймовірність більша або дорівнює 0.5. Основними перевагами логістичної регресії є простота та зрозумілість алгоритму. Недоліком моделі є лінійність в межах лінійної функції регресії [20].

1.2.3 Дерева рішень

Дерево рішень є алгоритмом із широким використанням як для класифікації, так і для регресії, популярний також завдяки своїй простоті. Дерево рішень намагається розділити набір даних на однорідні підмножини, дотримуючись послідовності бінарних рішень. Рішення легко інтерпретуються, оскільки являють собою набір простих правил.

Для побудови дерев рішень необхідно послідовно визначити кожне бінарне рішення. На кожному розбитті розглядаються всі доступні ознаки і визначається найкраще розбиття серед цих ознак. Обирається ознака з найбільшим інформаційним приростом. Інформаційний приріст можна визначити за допомогою залишкової суми квадратів для дерев регресії [22]. Прогнозом є середнє значення спостережень у листовому вузлі (вузлі, який не розбивається далі). Розмір дерева рішень і ризик перенавчання можуть бути обмежені максимальною глибиною і мінімальною кількістю значень у листовому вузлі.

Основною перевагою дерев рішень є можливість їх інтерпретації людиною. Бінарні розбиття дерев рішень можна легко прослідкувати. Дерева рішень також прості у своїх обчисленнях і можуть працювати як з числовими, так і з категоріальними даними (проте можливість кодування категоріальних даних у числові значення залишається). Дерева рішень також можуть працювати з даними, які відсутні у навчальній вибірці, розглядаючи їх як додаткову категорію. Однак прогнози дерев рішень обмежені діапазоном результатів у навчальних даних і не можуть екстраполюватися за межі цього діапазону. Крім того, дерева рішень вразливі до перенавчання. Невелика зміна в даних може призвести до значних змін у кінцевому дереві. Щоб запобігти перенавчання, дерева рішень можна «обрізати». «Обрізка» зменшує складність дерева, видаляючи деякі кінцеві вузли.

1.2.4 Випадковий ліс

Випадковий ліс є це ансамблем дерев рішень і також використовується як для класифікації, так і для регресії [23]. Випадковий ліс вирощує набір паралель-них дерев рішень. Остаточне передбачення випадкового лісу – це більшість го-лосів кожного дерева за окремий клас (для класифікації) або середнє передбачення окремих дерев (для регресії). Ідея випадкового лісу схожа на те, що багато експертів роблять прогнози незалежно один від одного,

а потім використовують більшість голосів або середнє значення групи людей як остаточний прогноз.

Існують два основних методи для навчання випадкового лісу. Перший ме-тод - це бутстрап, який вибирає випадкові вибірки (бутстрапи) з навчальних да-них для навчання кожного дерева. Другий метод полягає у виборі випадкових ознак при кожному розбитті під час навчання дерев [24]. Замість того, щоб розглядати всі ознаки при кожному розбитті, розглядається випадковий вибір ознак. З цього випадкового набору ознак, вибирається найкращий розподіл. Поєднання цих методів призводить до створення незалежного набору дерев рішень. Як результат, запобігає перенавчанню, оскільки прогнози цих різних дерев усереднюються для отримання остаточного прогнозу. Крім того, прогнози випадкового лісу, як правило, мають вищу точність порівняно з прогнозами од-ного дерева рішень [25].

Випадковий ліс має два основні параметри: кількість дерев та кількість ознак, які випадково вибираються при кожному розбитті (позначається mtry). Кількість дерев має бути достатньо великою для стабілізації продуктивності [26]. Коли кількість дерев є великою, mtry має найбільший вплив на продуктивність випадкового лісу. Чим менше значення mtry, тим менша кореляція між окремими деревами, що призводить до кращої стабільності алгоритму. Низьке значення mtry також може використовувати ознаки з помірними ефектами, які в іншому випадку маскуються ознаками з сильнішими ефектами. Однак, занадто низькі значення mtry можуть знизити продуктивність випадкового лісу, оскільки дерева можуть бути побудовані лише на незначних ознаках, вибраних з невеликої кіль-кості випадково вибраних ознак-кандидатів. Зазвичай для mtry вибирають квад-ратний корінь з кількості ознак для задач класифікації та кількість ознак, поділе-ну на 3, для задач регресії. Тим не менш, результати, як правило, близькі до найкращого результату в широкому діапазоні значень mtry [27].

Випадковий широко використовується завдяки своїй ефективності та стій-кості до шуму [23]. Він долає перенавчання (недолік окремого дерева

рішень) і отримує значно кращі результати завдяки бутстрапінгу та випадковому вибору ознак.

Крім того, випадкові ліси можна застосовувати до різних типів наборів да-них, оскільки вони можуть обробляти безперервні та категоріальні дані. Випад-ковий ліс може отримати високу точність з відносно невеликою кількістю даних, але більш складні методи, такі як штучні нейронні мережі та машини опорних векторів, можуть отримати вищу точність при правильному налаштуванні, особливо при збільшенні обсягу навчальних даних. Оскільки кожне дерево ви-користовує вибірку зразків даних, існують також зразки, які не враховуються в дереві, "зразки поза вибіркою" (out-of-bag). Зазначені вибірки можуть бути використані для оцінки продуктивності випадкового лісу без необ-хідності окремого набору для валідації або без перехресної валідації. Наявність зазначених вибірок є значною обчислювальною перевагою випадкового лісу.

У порівнянні з деревами рішень, випадкові ліси не так легко інтерпретува-ти, маючи сотні дерев. Тому його часто розглядають як модель "чорної скринь-ки". Тим не менш, випадковий має два цікавих компоненти, які дають уявлення про модель: важливість та близькість ознак.

Важливість є мірою того, наскільки окремі елементи впливають на загаль-ну продуктивність. Важливість ознаки можна визначити, послідовно перестав-ляючи кожну ознаку та обчислюючи зменшення точності. При великому змен-шенні, характеристика робить значний внесок у загальну точність і вважається важливою. При малому зменшенні точності характеристика вважається менш важливою. Важливість ознаки можна обчислити шляхом перестановки "зразків поза вибіркою" і повторного запуску цих вибірок на вже навченому алгоритмі випадкового лісу. На додаток до важливості ознаки, випадковий ліс має також параметр близькості ознак.

Близькість є мірою подібності між зразками. Близькість між двома вибірка-ми – відсоток дерев, у яких вони потрапляють в один і той самий листовий вузол. Коли два зразки потрапляють в один і той самий листовий вузол на значній кіль-кості дерев, це означає, що вони дуже схожі.

1.2.5 Градієнтне підсилення

Градієнтне підсилення – це інший ансамбль дерев рішень, запропонований Фрідманом у 2002 році [28]. Замість паралельних дерев, як у випадковому лісі, градієнтне підсилення будує дерева послідовно. Кожне наступне дерево навчається на залишках попередніх дерев. Залишок, у цьому випадку, є різницею між фактич-ним значенням і значенням, передбаченим деревами. Кожного разу, коли додаєть-ся нове дерево, воно має вчитися на досвіді попереднього і зменшувати залишки, що має призвести до кращого прогнозу. Цей метод дерев можна використовувати як для класифікації, так і для регресії.

Градієнтне підсилення зазвичай дає прогнози високої точності, головним чи-ном завдяки зменшенню залишкових значень за допомогою додаткових дерев. Крім того, оскільки воно складається з дерев рішень, то може обробляти як числові, так і категоріальні дані. Порівняно з випадковим лісом, градієнтне підсилення може отримати вищу продуктивність при правильному налаштуванні, але є менш стій-ким до шуму і схильне до перенавчання [29]. Як і випадковий ліс, градієнтне підсилення також не є простим для розуміння у випадку з декількома послідовни-ми деревами. Тому градієнтне підсилення також часто розглядається як модель "чорного ящика". Подібно до випадкового лісу, можна визначити важливість озна-ки. Однак, градієнтне підсилення не має "зразків поза вибіркою". Тому для оцінки важливості ознак потрібен окремий валідаційний набір даних.

1.2.6 Штучні нейронні мережі

Штучна нейронна мережа (ШНМ) є аналогом біологічних нейронних систем, таких як мозок, і може бути застосована для вирішення завдань

класифікації та ре-гресії. ШНМ складається з декількох з’єднаних між собою нейронів.

Кожен нейрон отримує вхідні сигнали, обробляє ці сигнали і отримує у результаті вихідний сигнал [30]. Ці нейрони організовані у три типи шарів (рис. 1.2).

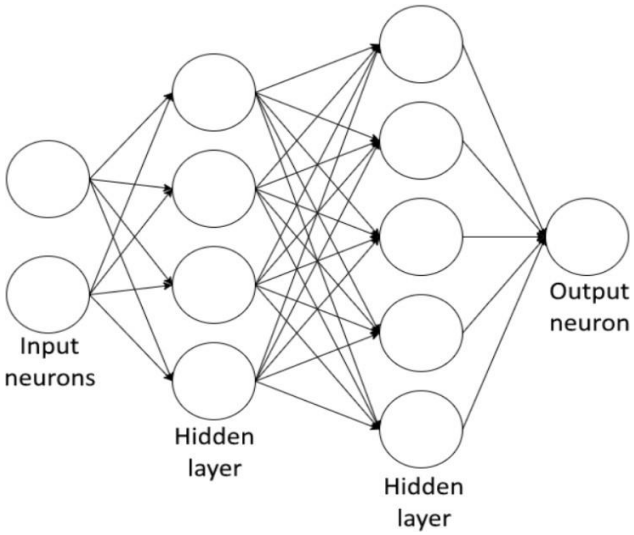

Рисунок 1.2 – Схематична будова ШНМ [30]

Перший тип – це вхідний шар, який отримує вхідні дані. Другий тип – це внутрішні шари, так звані приховані шари, які отримують вхідні сигнали від попереднього шару і надсилають вихідний сигнал до наступного шару. Третій тип – це вихідний шар, який отримує сигнали від попереднього шару і передає вихід нейронної мережі.

Кожен окремий нейрон отримує вхідні сигнали. Обробка цих сигналів складається з трьох кроків. Спочатку вхідні сигнали множаться на відповідні їм ваги. Потім всі ці значення підсумовуються. На останньому кроці підсумкова величина перетворюється за допомогою функції активації. Нею може бути будь- яка функція, для якої зазвичай застосовується логістична, гіперболічна тангенціальна або випрямлена лінійна одиниця [31]. Оскільки дані перетворюються за допомогою функцій активації, ШНМ не можуть працювати з категоріальними даними. Щоб ви-користовувати категоріальні дані, їх потрібно спочатку перекодувати у числові значення.

Похиб-ка між прогнозами та фактичними значеннями визначається функцією втрат, яка зазвичай є залишковою сумою квадратів. Потім визначається похідна першого порядку від помилки за вагами. За допомогою цієї похідної ваги оновлюються для мінімізації помилки. Цей процес повторюється до тих пір, поки зменшення залишкової суми квадратів не впаде нижче певного порогу [31].

Основна перевага штучних нейронних мереж полягає в тому, що вони надзвичайно гнучкі. Завдяки багаторівневим нелінійним комбінаціям можна відоб-разити складні взаємозв`язки даних. Щоб запобігти перенавчанню ШНМ, додаєть-ся функція регуляризації. При наявності достатньої кількості даних, ШНМ може давати прогнози високої точності. З іншого боку, кількість даним є і суттєвим об-меженням для ШНМ. Вони потребують великої кількості даних для того, щоб по-вністю розкрити свій потенціал. Складність ШНМ також може призвести до трива-лого часу обчислень [32]. Крім того, результати ШНМ досить складно пояснити, оскільки мережа прихованих шарів робить його моделлю "чорної скриньки". Для того, щоб забезпечити розуміння ШНМ, розроблено декілька методів, які показу-ють, які особливості є важливими для прогнозування ШНМ [33]. Зазначені методи аналізують ваги в мережі або аналізують зміни у виході шляхом послідовного ви-далення вхідних ознак з мережі.

1.3 Показники оцінки ефективності машинного навчання та прогнозування

У цьому підрозділі описані метрики, які дозволяють кількісно оцінити якість моделей машинного навчання, які використовуються у реалізації системи прогно-зування попиту нових продуктів на основі їхніх характеристик. До таких метрик відносяться індекс Калінські-Харабаша (оцінка кластеризації), точність (оцінка класифікації) та набір метрик для оцінки регресії: середня абсолютна похибка, се-редньоквадратична похибка, корінь середньоквадратичної похибки, середня від-соткова похибка та зважена середня відсоткова похибка.

1.3.1 Ефективність кластеризації

Для оцінки розподілу кластеризації використовується індекс Калінські-Харабаша. Індекс Калінські-Харабаша (CH-індекс) є статистичним показником, що використовується в кластерному аналізі. Його основна мета – оцінити якість кластерів, сформованих за допомогою алгоритму кластеризації. CH-індексу надається перевага за його здатність вимірювати компактність кластерів (внутрішньокластерна дисперсія) та відокремленість кластерів один від одного (міжкластерна дисперсія) [37]. Вище значення індексу CH, як правило, свідчить про ефективнішу кластеризацію, оскільки це означає, що кластери є щільними і відокремленими один від одного.

З точки зору використання, індекс CH є особливо корисним при спробі визначити оптимальну кількість кластерів для набору даних. Застосувавши алгоритм кластеризації з різною кількістю кластерів, можна розрахувати CH- індекс для кожного сценарію. Кількість кластерів, яка дає найвищий індекс CH, часто вважається оптимальним вибором, оскільки вона вказує на баланс між максимізацією міжкластерного розділення та мінімізацією внутрішньокластерної дисперсії.

Математична основа індексу CH полягає у відношенні суми міжкластерної дисперсії до суми внутрішньокластерної дисперсії, нормалізованої за відповідними ступенями свободи. Зокрема, обчислюється відношення міжкластерної дисперсії (наскільки кожен кластер відрізняється від інших) до внутрішньокластерної дисперсії (наскільки кожна точка даних відхиляється від центроїда свого кластера). Цей розрахунок дозволяє кількісно виміряти ефективність кластеризації, забезпечуючи чітку метрику для порівняння різних способів або методів кластеризації. Строге формулювання:

$$CH(k) = \frac{N-k}{k-1}\frac{BCSS(k)}{WCSS(k)} \quad , \tag{1.5}$$

Де N – кількість спостережень, k – кількість кластерів, BCSS(k) – міжкластерна дисперсія для цієї кількості кластерів, WCSS(k) – внутрішньокластерна дисперсія для цієї кількості кластерів.

1.3.2 Ефективність класифікації

Ефективність алгоритму класифікації базується на кількості правильних прогнозів. Передбачення алгоритму класифікації можна узагальнити у так званій матриці помилок. Загальний вигляд матриці помилок для 2-класової задачі класифікації наведено в таблиці 1.1.

Таблиця 1.1 – Матриця помилок для 2-класової задачі класифікації

		Передбачений клас	
		0	1
Фактичний клас	0	x00	x01
	1	x10	x20

У таблиці x_{ij} позначає кількість екземплярів класу *I*, які, за прогнозами, належать до класу *j*. Екземпляри класифікуються правильно, коли *I* = *j*. Якщо *I* та *j* відрізняються, екземпляр класифікується неправильно. З цієї матриці можна легко визначити точність алгоритму. Точність (accuracy) показує частку правильно класифікованих прогнозів [38]:

$$Accuracy = \frac{x_{00} + x_{11}}{x_{00} + x_{01} + x_{10} + x_{11}} \quad (1.6)$$

1.3.3 Ефективність регресії та прогнозування

Існує декілька метрик для аналізу точності регресійних моделей та прогнозів. Нехай Yt позначає фактичне значення екземпляра t (де загальна кількість екземплярів дорівнює n), а Ft – передбачення або прогноз Yt. Найпоширенішими показниками ефективності для похибки є наступні: середня абсолютна похибка (mean absolute error – MAE), середньоквадратична похибка

(mean square error – MSE), корінь середньоквадратичної похибки (root mean square error – RMSE), середня відсоткова похибка (mean average percentage error – MAPE) та зважена середня відсоткова похибка (weighted mean average percentage error – WMAPE) [39]:

$$MAE = \frac{1}{n}\sum_{t}^{n}|Y_t - F_t| \tag{1.7}$$

$$MSE = \frac{\sum_{t}^{n}(Y_t - F_t)^2}{n}, \tag{1.8}$$

$$RMSE = \sqrt{\frac{\sum_{t}^{n}(Y_t - F_t)^2}{n}}, \tag{1.9}$$

$$MAPE = 100 \cdot \frac{1}{n}\sum_{t}^{n}\frac{|Y_t - F_t|}{Y_t} \tag{1.10}$$

$$\mathrm{WMAPE} = \frac{\sum_{t=1}^{n}|A_t - F_t|}{\sum_{t=1}^{n}|A_t|} \tag{1.11}$$

Метрики MAE, MSE та RMSE є залежними від масштабу метриками, тобто масштаб залежить від масштабу даних. Ці метрики корисні при порівнянні різних методів на одному наборі даних, але не є ефективними при порівнянні наборів даних, які мають різні масштаби. MAE менш чутливий до екстремальних значень, ніж MSE та RMSE, оскільки MSE та RMSE підносять помилки. Перевага RMSE порівняно з MSE полягає в тому, що масштаб помилок подібний до маштабу фактичних значень.

Тому RMSE легше інтерпретувати. MAPE подібний до MAE, за винятком того, що він виражається у відсотках. Тому він не залежить від масштабу і може використовуватися для порівняння наборів даних, які мають різні масштаби.

Однак недоліком MAPE є те, що він є невизначеним, якщо $Y_t = 0$, і надзвичайно високим, якщо Y_t близький до нуля. Іншим недоліком є висока чутливість до екстремальних значень. WMAPE здатний зменшити вплив цього недоліку завдяки нормуванню на суму фактичних значень [40].

1.4 Опис та аналіз прогнозування попиту

Це процес оцінки майбутнього попиту за допомогою аналізу історичних даних, інформації та впливу додаткових факторів. Ефективне прогнозування попиту забезпечує компанії цінною інформацією щодо можливостей на поточному та потенційних ринках і допомагає менеджерам приймати зважені рішення стосовно обсягу товарів до замовлення, просування товарів та бізнес-стратегії в цілому.

Натомість, ігноруючи цей процес, компанії ризикують прийняти помилкові рішення в розрізі продуктової стратегії та цільових ринків. А це може створити купу проблем, таких як: збільшення затрат на зберігання продукції, зменшення задоволеності клієнтів та прогалини в управлінні supply chain. Якщо коротко, то компанія або втрачає кошти, або отримує їх не в повному обсязі.

Загалом тенденція на створення окремих відділів з прогнозування попиту в компаніях з’явилась наприкінці 80-х років минулого століття. Спочатку, в більшості випадків, прогнози базувались на простих статистичних моделях та методах, як-от рухоме середнє, експоненційне згладжування, чи навіть інстинктивне судження (в простонародді “чуйка”).

А потім, з розвитком технологій в області зберігання та обробки даних (Big Data), процес прогнозування попиту зазнав значних змін і став незамінним інструментом для бізнесів різних галузей та розмірів.

І якщо ринок програмного забезпечення для прогнозування попиту в 2019 році оцінювався в 3 млрд доларів, то до 2030 року очікується, що ця сума становитиме більше ніж 14,5 млрд (transparency market research). Тож чому

варто звернути уваги на цю тему і як прогнозування попиту може стати частиною ваших бізнес-процесів — розповідаємо далі.

1. Важливість прогнозування попиту для бізнесу

Попит є драйвером всього бізнесу. Тож не дивно, що його аналіз впливає на ефективність багатьох процесів компанії. Прогнозування попиту не буває на 100% точним (тільки якщо це збіг обставин або шахрайство з розрахунками), але є необхідним, адже впливає на:

Дані, отримані з прогнозу, допомагають приймати ефективні фінансові рішення щодо операційних, виробничих та маркетингових витрат. Крім того, чітка картинка передбачуваного попиту дозволить вам спланувати затрати на персонал та перерозподілити ресурси в пікові періоди активності.

Визначення правильної ціни, враховуючи поточну ринкову активність та попит на ваш продукт, є ключовим. Завдяки прогнозу попиту ви зможете корегувати цінову політику залежно від ситуації, а також заздалегідь налаштувати інструменти її втілення, на кшталт акцій, знижок, промо тощо. Розуміючи ринок і потенційні можливості, ви можете встановлювати конкурентні ціни та використовувати доречні маркетингові стратегії щодо вартості товарів.

Контроль рівня запасів

Прогнозуючи майбутній попит, ви можете розрахувати оптимальну кількість товарів на складах, не створюючи при цьому overstock. Так ви уникнете переплат за надлишкове зберігання, або ж, навпаки, зможете заздалегідь підготуватися до ажіотажу в період підвищених продажів.Прогнозування попиту доцільно застосовувати незалежно від сфери бізнесу – чи то ритейл, чи FMCG, чи фармацевтична компанія, чи будівництво тощо[3].

2. Прогнозування та планування попиту. В чому різниця?

Багато хто використовує поняття прогнозування та планування попиту як синоніми. Проте існує принципова різниця між цими термінами. І якщо прогнозування — це стратегічне передбачення на основі історичних даних та аналізу суміжних факторів, то планування — це більш тактичний процес, що

передбачає побудову плану на основі даних з прогнозу та розробку кроків його втілення, зображення на рис.1.3.

Рисунок 1.3 – Порівняльний аналіз прогнозування та планування попиту

3. Фактори, що впливають на попит та прогноз

Існує ряд факторів, що суттєво впливають на попит і враховуються під час прогнозування. Ось ключові з них:

Відповідно до сезонів змінюється й попит. Сезонний бренд чи циклічний бізнес може мати підвищену активність в пікові періоди, що змінюється стабільними або продажами нижче середніх в міжсезоння.

Прогнози, що базуються на сезонності, враховують продукти з більшою популярністю в специфічні періоди, на свята чи івенти. А оскільки такі товари потребують гнучкого управління робочими потужностями, ресурсами та зберіганням, то одним з ключових інструментів буде саме ефективне прогнозування.

Поява нових гравців у прямих та суміжних товарних категоріях створює все більше альтернатив для ваших клієнтів, що позначається на попиті. Запуск нових конкурентних продуктів, або, навпаки, вихід когось з ринку може застати вас зненацька. А гнучка модель прогнозування дозволить вам швидше реагувати на мінливі події.

Процес прогнозу буде різнитися для різних типів продуктів та послуг — від товарів, що швидко псуються, до сервісів за передплатою, що оплачуються на місячній основі. Важливо знати пожиттєву цінність ваших клієнтів (загальну к-сть покупок на людину протягом певного періоду), розмір середнього чека та комбінації продуктів, що купуються разом. Використовуючи ці дані, ви можете покращити якість прогнозу і відстежувати, як один SKU впливає або стимулює попит на інший.

Місце концентрації ваших клієнтів, розташування виробництва та точок відправки продукції впливають на прогнозування запасів та швидкість виконання клієнтських замовлень. А доступність складських приміщень та швидкі способи доставки можуть позитивно вплинути на прогноз попиту.

Залежно від галузі, клієнтської бази та специфіки продукту, використовують різні типи прогнозування попиту, ось основні з них:

В цьому випадку прогнозується загальна економічна ситуація, зовнішні чинники та інші масштабні фактори, що можуть вплинути на бізнес. Аналіз цих складових забезпечує бізнес інформацією про локальні та глобальні ризики, можливості, а також дозволяє залишатися в контексті культурних та ринкових змін.

Прогноз попиту на мікрорівні може бути специфічним для окремого продукту, регіону або сегменту. Зазвичай стосується разових або спонтанних змін, що можуть призвести до стрибка або падіння попиту. Наприклад, якщо ви локальний виробник пива, а ваша місцева команда неочікувано вийшла у фінал чемпіонату з футболу, то варто заздалегідь подбати про доступність товару на полицях і провести додаткові маркетингові активності.

Може використовуватися на макро- та мікрорівні. Зазвичай здійснюється на період менш ніж 12 місяців для отримання інформації щодо повсякденних завдань. Короткостроковий прогноз передбачає консультації з відділами продажів та маркетингу для розуміння їхніх активностей, що можуть підвищити попит.

Як і попереднє, може використовуватися для макро- та мікрорівня і здійснюється на період більше одного року. Цей вид прогнозування допомагає

компаніям приймати зважені глобальні рішення щодо розширення, інвестицій або довготривалих партнерських відносин. Закладаючи рік та більше в процес прогнозу, компанія бачить надійну картинку тенденцій попиту, що може скластися, наприклад, після запуску нового магазину чи розширення бізнесу в інших країнах.

4. Автоматизація прогнозування попиту

В умовах сучасного ринку ручні методи інтерпретації даних для прогнозування не покривають запити компаній. По-справжньому гнучкий і актуальний підхід, що враховує мінливість середовища та швидку зміну клієнтської поведінки, передбачає аналіз даних в режимі реального часу, тобто використання технологічних програм. Це також дозволяє зменшити вплив людському фактору, знизити навантаження команд та змінити їхній фокус з операційних завдань на стратегічні.

1.5 Висновки до розділу 1

У цьому розділі було розглянуто кілька методів статистичного та машинного навчання. Кожний метод має свої особливості, які не дозволяють використання одних і тих же систем для різних завдань. Модель повинна відповідати вимогам конкретної проблеми. Окрім відповідності моделі, зазвичай існує взаємозв’язок між складністю та точністю моделі.

Беручи до уваги характеристики задачі та співвідношення між складністю та точністю, для подальшого дослідження обрано алгоритм випадкового лісу. Випадковий ліс може обробляти як класифікаційні, так і регресійні задачі, може моделювати нелінійні зв’язки, усереднювати прогнози за певний проміжок часу. Mtry має кращі можливості для узагальнення, а тому може бути точнішим і більш універсальним, ніж множинна лінійна регресія, логістична регресія та дерева рішень. Тим не менш, випадковий ліс все ще досить просто налаштувати за до-помогою mtry за основним параметром. Описана можливість полегшує застосу-вання порівняно з градієнтним бустингом, штучною нейронною мережею та ре-гресією опорних векторів.

2 АНАЛІЗ МОДЕЛІ ПРОГНОЗУВАННЯ ПОПИТУ

2.1 Загальний аналіз моделі

У попередньому розділі було визначено, що алгоритми випадкового лісу є найбільш придатними для цієї задачі. Також оптимальним є використання двох окремих алгоритмів випадкового лісу: категоріальний (для визначення профілів –модель випадкового лісу для класифікації, або random forest classifier (RFC)) та кількісний (для аналізу загального попиту – модель випадкового лісу для ре-гресії, або random forest regression (RFR)). Завдяки цим алгоритмам машинного навчання система може навчитися генерувати прогнози для продуктів. Прогноз на весь період виходу продукту на ринок дає уявлення про обсяг і розвиток по-питу на новий продукт. Сукупні моделі попиту групуються в окремі профілі і прогнозують цей профіль і загальний попит. Перевага системи полягає в тому, що для отримання прогнозу попиту нам потрібно лише два прогнози (профіль і попит). Ці два прогнози також можуть бути легко інтерпретовані. Метод схема-тично зображено на рис. 2.1.

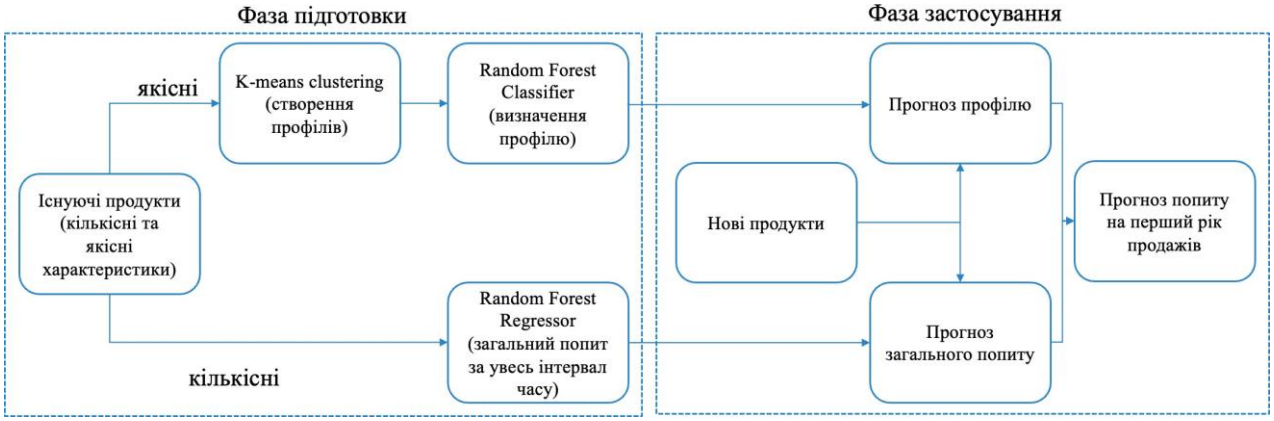

Рисунок 2.1 – Схематична побудова метода

Як показано на рисунку 2.1, система складається з фази підготовки та фази застосування. На етапі підготовки історичний попит кластеризується у профілі за допомогою алгоритму k-середніх, а існуючі продукти використовуються для навчання алгоритмів RFC та RFR. У фазі застосування

профілі та алгоритми ви-користовуються для прогнозування попиту на нові продукти.

На етапі підготовки спочатку кластеризуються дані попиту, щоб отримати профілі попиту. Кластеризується нормалізований кумулятивний попит на істо-ричні товари за допомогою алгоритму k-середніх. Алгоритм повторюється пев-ну кількість разів для отримання стабільного результату. Потім профілі при-своюються кожному товару. Після кластеризації алгоритм RFC навчається кла-сифікувати профіль на основі характеристик товару.

Окрім прогнозування профілю, прогнозується загальний попит протягом зазначеного періоду продажів (1 рік). Для цього прогнозу використовується ал-горитм алгоритм RFR. Алгоритм RFR навчається на основі числових ознак про-дажів продукту.

Після етапу підготовки, навчені алгоритми можна використовувати для прогнозування попиту та профілю нових продуктів. Остаточний прогноз на весь період впровадження можна отримати, об’єднавши прогноз профілю та загаль-ного попиту. Коли з’являються нові дані про нещодавно введені товари, алго-ритми можна навчити знову. У такому випадку вони враховуватимуть нові дані, що статистично покращує точність прогнозів у майбутньому.

2.2 Детальний опис модулів

У цьому підрозділі описано детальну імплементацію моделей машинного навчання, які необхідні для повної роботи системи. До цих моделей відносяться кластеризація методом k-середніх, модель випадкового лісу для класифікації і модель випадкового лісу для регресії. У останньому підпункті описана додаткова перевага моделей випадкового лісу, яка полягає у наявності out-of-bag даних (дані, які при навчанні не потрапляють на дерево). Використання цих даних доз-воляє створити гнучку валідацію дерев рішень при навчанні випадкового лісу без додаткового поділу тренувальної вибірки.

2.2.1 Кластеризація методом k-середніх (k-means clustering)

Численні дослідження використовують різні алгоритми для кластеризації продуктів. До них можна віднести кластеризацію по характеристикам товару [41, 42] або кластеризацію за чисельними характеристиками продажів (наприклад, попит) [43].

У цьому досліджені обраний варіант кластеризації на основі попиту продуктів, завдяки якому можна визначити обмежену кількість профілів. Використовуючи визначені профілі, навчається класифікатор на основі характеристик товару, який дозволяє встановити залежності між характеристиками товарів та попитом на них. Для кластеризації моделей попиту використовується нормалізо-ваний місячний попит. Нормалізуючи попит, з'являється можливість порів-нювати криві попиту, незважаючи на загальний попит. Оскільки це дослідження напрямлене на прогнозування попиту нових продукті, більшу важливим є не на-явність попиту у конкретний місяць, а розподіл попиту протягом року, що знач-но допомагає формувати запаси товару. Використовуючи кумулятивну модель, можна досягти кращих результатів у цьому напрямі.

Щоб визначити нормалізований кумулятивний попит, попит спочатку нормалізується. Дефіцит попиту a за тиждень i масштабується шляхом ділення попиту кожного місяці на загальний попит за 12 місяців для отримання нормалізованого кумулятивного попиту n за тиждень i:

$$n_i = \frac{\sum_{x=1}^{i} a_x}{\sum_{x=1}^{12} a_x} \qquad \forall\, i \in 1-12 \tag{2.1}$$

Для кластеризації профілів попиту на продукти використовується алгоритм k-середніх. Цей метод є популярним у аналізі даних і машинному навчанні для групування даних у k різних підмножин або кластерів, що не

перетинають-ся. Метод широко цінується за простоту реалізації та ефективність, особливо при обробці великих наборів даних.

Фундаментальна ідея кластеризації за методом k-середніх полягає в тому, щоб розбити n спостережень на k кластерів, де кожне спостереження належить до кластера з найближчим середнім значенням. Зазначений підхід призводить до розбиття простору даних на комірки Вороного [44]. Процес починається з випадкового вибору k початкових центроїдів, по одному для кожного кластера. Потім алгоритм повторює два основні кроки до збіжності: призначення та онов-лення. На кроці призначення кожній точці даних присвоюється найближчий до неї центроїд. У цьому випадку норма може бути як евклідовою нормою, так і неевклідовою. На кроці оновлення перераховуються центроїди кластерів. Для цього береться середнє значення всіх точок, віднесених до кожного кластера.

Зазначені кроки повторюються доти, доки не буде досягнуто критерію зупинки, яким може бути певна кількість ітерацій, поріг мінімальної зміни центроїда між ітераціями, або коли не відбудеться жодних подальших змін у розподілі точок даних до кластерів.

2.2.2 Модель випадкового лісу для класифікації (RFC)

Модель випадкового лісу для класифікації (random forest classifier, або RFC) – алгоритм, який відомий своєю простотою та потужними можливостями. Він є частиною сімейства ансамблевого навчання, де кілька моделей (у цьому випадку, дерев рішень), що об’єднуються для покращення загального результату. Основна концепція RFC полягає в побудові декількох дерев рішень на різних підвибірках набору даних і використанні усереднення для підвищення точності прогнозування і контролю над перенавчанням. Кожне дерево у випадковому лісі видає прогноз класу, і клас з найбільшою кількістю голосів стає прогнозом моделі [23].

Значною перевагою випадкових лісів у задачах класифікації є їхня гнучкість у роботі як з числовими, так і з категоріальними даними. Вони можуть обробляти тисячі вхідних змінних без видалення змінних і є ефективними для визначення найбільш значущих ознак даних для класифікації. Випадкові ліси та-кож стійкі до викидів (екстремальних значень) і можуть певною мірою обробляти пропущені значення.

Точність та якість моделі випадкового лісу значною мірою залежать від її гіперпараметрів [24]. Деякі з ключових гіперпараметрів включають кількість де-рев у лісі, глибину кожного дерева, кількість ознак, що розглядаються для розбиття в кожному листовому вузлі, і мінімальну кількість листових вузлів. Налаштування цих параметрів може допомогти запобігти надмірному навчанню моделі і може бути налаштоване для досягнення кращої продуктивності.

З точки зору математики, випадкові ліси використовують концепцію беґґінґу (бутстрап-агрегування), де кожне нове дерево формується на основі бутстрап-вибірки навчальних спостережень. Вузли дерева розбиваються на підмножину ознак, випадково вибраних при кожному розбитті.

Одним з ключових математичних принципів, що лежать в основі RFC, є закон великих чисел [22]. Зі збільшенням кількості дерев у лісі помилка узагальнення всієї моделі сходиться, що призводить до кращої та стабільнішої роботи. Випадковість, закладена в побудову дерев, гарантує, що модель не буде чутливою до особливостей навчальних даних, що робить її надійним інструмен-том узагальнення.

Ключовим параметром для моделі випадкового лісу є параметр mtry. За-значений параметр відіграє вирішальну роль у визначенні якості та точності мо-делі. Він вказує на кількість ознак, які слід враховувати при кожному розбитті в деревах рішень, що складають випадковий ліс. На відміну від одного дерева рішень, де кожна ознака може бути оцінена для найкращого розбиття, у випад-ковому лісі розглядається лише підмножина ознак, визначена за допомогою mtry.

Такий підхід вносить випадковість у модель, що є фундаментальною осно-вою для продуктивності випадкових лісів.

Коли будується дерево у випадковому лісі, в кожному вузлі, замість того, щоб перебирати всі можливі ознаки для знаходження найкращого розбиття, алгоритм випадковим чином вибирає кількість ознак mtry. Цей процес відбору гарантує, що дерева у випадковому лісі не є ідентичними і не зосереджуються на одних і тих же суттєвих ознаках. Ця різноманітність серед дерев робить RFC більш ефективним, ніж окремі дерева рішень, особливо з точки зору зменшення ризику надмірного пристосування.

Вибір mtry може суттєво вплинути на точність моделі. Менше значення mtry збільшує випадковість лісу, що може бути корисним для зменшення перенавчання, але може також зменшити влучність класифікації. З іншого боку, більше значення mtry робить дерева в лісі більш схожими одне на одного, оскільки вони, ймовірно, використовують однакові сильні ознаки для розщеплення. Зазначена особливість може збільшити дисперсію та ризик надмірного перенавчання [23].

2.2.3 Модель випадкового лісу для регресії (RFR)

Регресійна модель випадкового лісу (random forest regressor, або RFR) є розширенням алгоритму випадкового лісу, яке адаптоване для задач регресії. Вона використовує ті ж самі фундаментальні принципи, що й класифікатор випадкового лісу, але замість того, щоб прогнозувати клас, прогнозує безперервне значення. Основна ідея полягає в тому, щоб побудувати кілька дерев рішень і об’єднати їхні прогнози для отримання більш точного і надійного результату. У випадку регресії остаточний прогноз, як правило, є середнім значенням про-гнозів усіх дерев у лісі.

Зазначений алгоритм особливо добре підходить для регресії через його здатність моделювати складні, нелінійні зв’язки в даних.

Кожне дерево у випад-ковому лісі фіксує частину шаблону даних, і коли ці дерева усереднюються, ре-зультатом часто є надійна залежність основної тенденції. Однією з сильних сторін регресії методом випадкового лісу є його здатність обробляти велику кількість ознак і визначати найбільш значущі змінні. Випадковий ліс може працювати як з числовими, так і з категоріальними даними, що робить його універсальним у роботі з різними типами наборів даних [27].

Для задач регресії модель випадкового лісу має кілька переваг. Вона природно стійка до перенавчання, особливо у випадках, коли в лісі багато дерев. Ал-горитм також стійкий до викидів, які часто становлять значну проблему в ре-гресійному аналізі. Крім того, регресор випадкового лісу може певною мірою впоратися з відсутніми значеннями, хоча для отримання кращих результатів за-звичай рекомендується врахувати відсутні дані перед навчанням моделі.

У контексті прогнозування сукупного попиту регресор випадкового лісу може бути особливо ефективним. Прогнозування попиту часто передбачає аналіз складних даних з численними змінними, які можуть впливати на попит на товар чи послугу [28]. Згадані змінні можуть включати історичні дані про попит, се-зонні тенденції, економічні показники, зміни цін та рекламні заходи. RFR може впоратися з цією складністю, фіксуючи нелінійні зв’язки між цими змінними та попитом. Він також може ранжувати важливість різних характеристик, надаючи більшу вагу факторам, що найсильніше впливають на попит.

Однак, як і для моделі випадкового лісу для класифікації, дуже важливим є правильні налаштування модель. До налаштування, у першу чергу, відноситься зміна гіперпараметрів моделі, таких як кількість дерев у лісі, глибина дерев і кількість ознак (mtry), що враховуються при кожному розбитті.

На практиці застосування регресора випадкового лісу для прогнозування попиту передбачає навчання моделі на історичних даних, перевірку її продуктивності, а потім використання для прогнозування майбутнього попиту.

Прогно-зи можуть уточнюватися з часом, коли з`являється більше даних, що робить ме-тод випадкового лісу адаптивним і динамічним інструментом для прогнозування [27].

2.2.4 Валідація моделі випадкового лісу на out-of-bag даних

Out-of-bag дані – це унікальний аспект моделей випадкового лісу, який слугує вбудованим механізмом для валідації моделі на етапі навчання. Ця концепція походить від техніки вибірки бутстрап, яка використовується при побудові випадкового лісу [23].

У випадковому лісі будуються численні дерева рішень, кожне з яких будується на основі бутстрап-вибірки навчального набору даних. Бутстрап-вибірка передбачає випадковий вибір точок даних з навчального набору, щоб сформувати вибірку для кожного дерева. Оскільки цей процес передбачає випадковий вибір, не кожна точка даних вибирається для навчальної вибірки кожного дерева. Як правило, близько третини точок даних не використовуються при побудові кожного дерева. Ці точки даних, які не увійшли до навчальної вибірки дерева, називаються out-of-bag даними.

Out-of-bag дані діють як вбудована валідаційна вибірка для кожного окремого дерева в лісі. Оскільки кожне дерево у випадковому лісі створюється з використанням лише частини даних, решта даних, які не були доступні дереву під час навчання, можуть бути використані для тестування та перевірки дерева. Це можна порівняти з процесом перехресної перевірки, але він є більш ефектив-ним, оскільки відбувається одночасно з навчанням випадкового лісу і не вимагає виділення окремого набору даних для перевірки.

Метод використання out-of-bag даних для валідації працює наступним чи-ном: out-of-bag дані для кожного дерева, які не потрапили на дерево під час нав-чання, пропускаються через дерево для генерації прогнозів. Потім ці прогнози порівнюються з фактичними значеннями out-of-bag даних, а різниця використо-вується для обчислення помилки out-of-bag даних для цього дерева.

Обрахована похибка відображає оцінку продуктивності дерева. Загальна помилка out-of-bag даних для випадкового лісу є середнім значенням зазначених індивідуальних по-милок дерев.

Похибка на out-of-bag даних є цінною та неупередженою мірою якості моделі [22]. Вона особливо корисна, оскільки дає уявлення про те, наскільки добре модель буде працювати на нових даних, без потреби в окремому валідаційному наборі. Розрахунок помилки out-of-bag даних, використовуючи різні дані для кожного дерева, дає повне уявлення про роботу моделі, оскільки враховує сильні та слабкі сторони всіх дерев у лісі.

Використання out-of-bag даних для валідації під час навчання має кілька переваг. По-перше, описаний підхід економить час і ресурси, усуваючи потребу в окремій процедурі валідації. По-друге, також дає більш точну оцінку якості моделі, враховуючи, що out-of-bag вибірки є фактично новими даними для кожного дерева. Крім того, валідація загалом дозволяє здійснювати постійний моніторинг та коригування моделі під час навчання, оскільки помилка на out-of- bag даних може бути використана для зміни параметрів моделі.

2.3 Вибір програмних продуктів для реалізації системи

У цьому підрозділі описано ретельний вибір програмного пакету для реалізації складових вибраної архітектури системи із найбільш популярних програмних продуктів. До найпопулярніших програмних пакетів можна віднести Tensorflow, Keras, PyTorch та sklearn. Кожен із зазначених програмних продуктів зайняв свою нішу для використання у дослідженнях спеціалістів із машинного навчання, проте існують переваги та недоліки для виконання поставленої задачі.

1 Tensorflow

TensorFlow, розроблений дослідниками та інженерами з команди Google Brain, є програмною бібліотекою з відкритим вихідним кодом, яка широко використовується для машинного навчання та додатків штучного інтелекту. Дизайн програмного пакету є дуже гнучким і підходить для широкого спектру зав-дань, але особливо відомий завдяки використанню для глибокого навчання зав-дяки своїм потужним можливостям роботи з нейронними мережами [45].

Незважаючи на те, що фреймворк TensorFlow асоціюється з глибоким навчанням, він також цілком здатний реалізовувати традиційні алгоритми машинного навчання, такі як кластеризація за методом k-середніх, випадкові ліси для класифікації та для регресії. Здатність TensorFlow ефективно обробляти великі масиви даних і виконувати складні числові обчислення робить його підходящим вибором для цих завдань.

Незважаючи на описані можливості, TensorFlow має деякі недоліки, коли мова йде про реалізацію таких алгоритмів, як кластеризація за методом k- середніх та випадкові ліси. Основною проблемою є його складність. Архітектура TensorFlow, хоч і потужна, проте суттєво складна для реалізації відносно простих алгоритмів машинного навчання.

Ще одним недоліком є витрати для обчислювальних можливостей, пов'язані з використанням TensorFlow. Для простіших завдань TensorFlow може створити непотрібний рівень складності та вимог до обчислювальних ресурсів.

Більше того, ресурсоємність TensorFlow означає, що він може бути не найкращим вибором для середовищ з обмеженими обчислювальними потужностями. Хоча він чудово підходить для великомасштабних промислових застосувань і складних моделей, для менш масштабних проектів або моделей, які не по-требують повного використання можливостей TensorFlow, можуть бути більш доречними інші легші та спеціалізовані інструменти.

2 Keras

Keras – високорівневий нейромережевий API, написаний на Python і здатний працювати разом із TensorFlow, CNTK або Theano. Він був розроблений з акцентом на забезпечення швидкого експериментування. Можливість перейти від ідеї до результату з найменшою можливою затримкою є ключовим фактором для проведення якісних досліджень. Keras відомий своєю зручністю, модуль-ністю та легкістю розширення, що робить його популярним серед дослідників та розробників у галузі глибокого навчання [45].

Однак, коли справа доходить до реалізації традиційних алгоритмів машинного навчання, таких як кластеризація за методом k-середніх, класифікатори випадкових лісів та регресори випадкових лісів, Keras не є оптимальним рішен-ням. Keras, в першу чергу, призначений для побудови та навчання моделей гли-бокого навчання, таких як згорткові та рекурентні нейронні мережі. Традиційні алгоритми машинного навчання, подібні до згаданих, зазвичай реалізовані в ін-ших бібліотеках, які розглядаються у цьому підрозділі.

3 PyTorch

PyTorch – популярна бібліотека машинного навчання з відкритим вихідним кодом, відома своєю гнучкістю, швидкістю та простотою використання, особливо в галузі глибокого навчання. Розроблена лабораторією AI Research Facebook, вона надає багату колекцію інструментів та алгоримтів, які допомага-ють у розробці моделей машинного навчання. PyTorch особливо цінують за йо-го динамічний обчислювальний граф, який дозволяє більш інтуїтивно зрозуміле кодування складних архітектур, оскільки він виконує обчислення в міру їх виклику в скрипті, полегшуючи налагодження та розуміння моделі.

PyTorch в першу чергу розроблений для додатків глибокого навчання, але він також може бути адаптований для традиційних завдань машинного навчання, хоча і з деякими обмеженнями. Реалізація кластеризації k-середніх у PyTorch можлива, але є прямою реалізацією.

Кластеризація k-середніх, будучи алгорит-мом навчання без вчителя, зазвичай вимагає менших обчислювальних потужно-стей, порівняно з моделями глибокого навчання. PyTorch можна використовува-ти для написання власних реалізацій кластеризації k-середніх, використовуючи його графічне прискорення для обробки великих наборів даних більш ефективно, ніж традиційні бібліотеки Python [45].

Для класифікаторів та регресорів на основі випадкових лісів PyTorch не є оптимальним вибором, оскільки він не має вбудованих функцій для цих алго-ритмів, які є частиною методів ансамблевого навчання в традиційному машин-ному навчанні. Реалізація цих функцій у PyTorch вимагає спеціального кодуван-ня та розуміння як алгоритмів, так і фреймворку PyTorch.

Одним з основних недоліків використання PyTorch для цих традиційних завдань машинного навчання є те, що згаданий інструмент є занадто складним та громіздким. Сильні сторони PyTorch в роботі зі складними нейромережевими архітектурами та його динамічна природа можуть бути надмірними зазначених у роботі алгоритмів. Крім того, відсутність вбудованих функцій для цих алго-ритмів означає більше часу, витраченого на реалізацію, і більшу ймовірність невідповідності теоретично описаній архітектурі. Крім того, екосистема PyTorch, будучи більш орієнтованою на глибоке навчання, може не надавати таку ж ши-року підтримку та знання спільноти для цих специфічних традиційних завдань машинного навчання, що потенційно може призвести до збільшення часу ро-зробки та проблем з усуненням несправностей коду.

4 sklearn

Scikit-learn, більше відомий як sklearn, – популярний фреймворк Python для машинного навчання. Він відомий своєю простотою, ефективністю та доступ-ністю, що робить його ідеальним вибором для реалізації стандартних алгоритмів машинного навчання. Бібліотека містить повний набір інструментів для різних завдань машинного навчання, включаючи класифікацію, регресію,

кластеризацію та зменшення розмірності, а також утиліти для попередньої обробки даних, оцінки моделей тощо.

Зосереджуючись на конкретних реалізаціях, таких як кластеризація за методом k-середніх, модель класифікації випадкового лісу та модель регресії випадкового лісу, sklearn пропонує надійні та ефективні рішення. Алгоритм кла-стеризації k-середніх у sklearn є особливо зручним та універсальним. Він дозво-ляє легко ідентифікувати кластери в наборах даних, розбиваючи дані на k груп на основі схожості ознак. Зазначений алгоритм є високоефективним для до-слідницького аналізу даних у сценаріях навчання без вчителя. Реалізація sklearn пропонує гнучкість у налаштуванні таких параметрів, як кількість кластерів і початкові точки, що робить його адаптивним до різних наборів даних і вимог [45].

Модель випадкового лісу для класифікації у sklearn є ще одним потужним інструментом. Цей метод ансамблевого навчання, який будує кілька дерев рішень і об'єднує їхні прогнози, використовується для завдань класифікації. Реалізація кла-сифікатора випадкових лісів у sklearn вирізняється простотою використання та здатністю працювати з великими наборами даних високої розмірності.

Аналогічно, модель випадкового лісу для регресії в sklearn застосовує ту ж саму техніку ансамблю, але для задач регресії. Регресор у sklearn дуже добре налаштовується, що дає змогу тонко підбирати параметри для підвищення точ-ності моделі та запобігання перенавчання.

Переваги використання sklearn для таких реалізацій суттєві. По-перше, ви-черпна документація та активна підтримка спільноти допомагають у навчанні та усуненні несправностей. По-друге, сумісність sklearn з іншими бібліотеками Python, такими як NumPy та pandas, полегшує маніпулювання даними та їх аналіз. По-третє, ефективність у роботі з великими наборами даних та виконанні складних обчислень з відносно низькими накладними витратами також є значним плюсом.

2.4 Висновки до розділу 2

У цьому розділі було детально розглянуто архітектуру системи та можливі технічні інструменти для її реалізації. Система складається із трьох основних час-тин: модель кластеризації k-середніх, модель випадкового лісу для класифікації і модель випадкового лісу для регресії. Кожна модель виконує свою важливу функ-цію. Модель кластеризації k-середніх дає змогу виокремити профілі продуктів на основі їхнього кумулятивного нормалізованого попиту, що дозволяє розділити продукти незалежно від їхнього абсолютного попиту, а також у подальшому шу-кати закономірності між профілями та характеристиками продуктів. Модель випа-дкового лісу для класифікації дозволяє використати результати роботи поперед-ньої моделі, щоб визначати профілі для продуктів на основі їхніх категоріальних характеристик, що дозволяє працювати із новими продуктами у майбутньому. Модель випадкового лісу для регресії дає змогу прогнозувати повний попит про-дуктів на основі чисельних характеристик. Поєднання визначеного (на основі клас-теризації) профілю класифікатором та загального попиту регресором надає мож-ливість будувати прогноз попиту на кожен визначений момент часу (у описаному випадку – на кожен місяць).

3 ТРЕНУВАННЯ ТА ТЕСТУВАННЯ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

3.1 Тестування алгоритмів

Цей підрозділ присвячено основній експериментальній частині досліджен-ня, а саме тренування та тестуванню алгоритмів машинного навчання: кластери-зація методом k-середніх, модель випадкового лісу для класифікації і модель випадкового лісу для регресії. Кожен алгоритм навчається на тренувальній вибірці, яка становить 80% продуктів від загальної відфільтрованої вибірки. Потім алгоритми перевіряються на тестовій вибірці і проводиться кількісна оцінка якості за метриками, які описані у першому розділі цього дослідження. Також, як зазначено у попередньому підрозділі, дані можуть потребувати до-даткової обробки при використанні на кожній із запропонованих моделі машин-ного навчання.

3.1.1 Кластеризація методом k-середніх

Для визначення профілів продуктів на основі кумулятивного попиту за рік продажів, необхідна додаткова обробка даних. По-перше, у вибірці залишаються лише продукти, які мають періоди продажів від січня до грудня одного року. По- друге, для кожного такого періоду обраховується кумулятивний попит за фор-мулою, зазначеною у розділі 2. Далі відповідний кумулятивний попит скла-дається у вектор із розмірністю 12, де кожному виміру відповідає конкретний місяць продажів: 1 – січень, 2 – лютий тощо. Грануляція даних: продукт-рік продажів.

Для кластеризації використовується алгоритм із бібліотеки sklearn: sklearn.cluster.KMeans. Код кластеризації наведений у додатку А.

Оскільки кластеризація методом k-середніх потребує заздалегідь визначену кількість кластерів для навчання, проводиться навчання алгоритмів

для кількості кластерів від 2 до 25. Оцінка кластеризації відбувається індексом Калінські-Харабаша (CH-індекс), про який зазначалося у першому розділі. Ха-рактерна величина показує відношення відстані між кластерами до дисперсії всередині кластерів. Більше значення демонструє краще розподілення даних по кластерам.

Також важливим є те, що початкові значення центрів кластерів є випадко-вими, тому для отримання стійкого результату дослідження кластеризації про-водилося 10 разів. Усереднені результаті дослідження наведені нижче (табл. 3.1 та рис 3.1).

Таблиця 3.1 – Найбільші значення CH-індексу для даного дослідження

Кількість кластерів	Значення CH-індексу
3	12256
4	12289
5	12304
6	12378

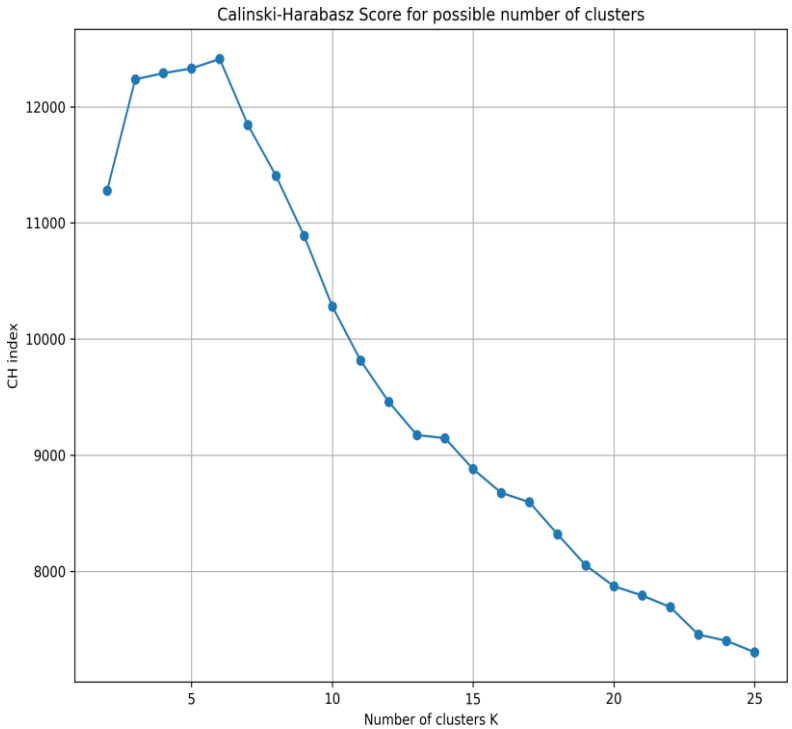

Рисунок 3.1 – Залежність CH-індексу від кількості кластерів

Свого максимуму значення CH-індексу досягає при 6 кластерах. Важливо зазначити, що для значень кластерів 3-5 значення індексу не є суттєво меншим, що свідчить про можливість використання іншої конфігурації системи. Для подальших експериментів використовуватиметься модель із 6 профілями продуктів.

Загальна кількість зразків для кластеризації становить 16163. Найбільший кластер (позначено як 0) включає у себе 8167 зразків (50.53%). Кластери 1 та 5 складаються із 2695 (16.67%) та 3932 (24.33%) зразків відповідно. Кластери 2, 3 та 4, які є суттєво меншими, складаються із 770 (4.76%), 311 (1.92%) та 288 (1.78%) зразків відповідно. Цей розмір кластерів підтверджує гіпотезу можли-вості використання меншої кількості кластерів. Візуалізація профілів наведена на рисунку 3.2.

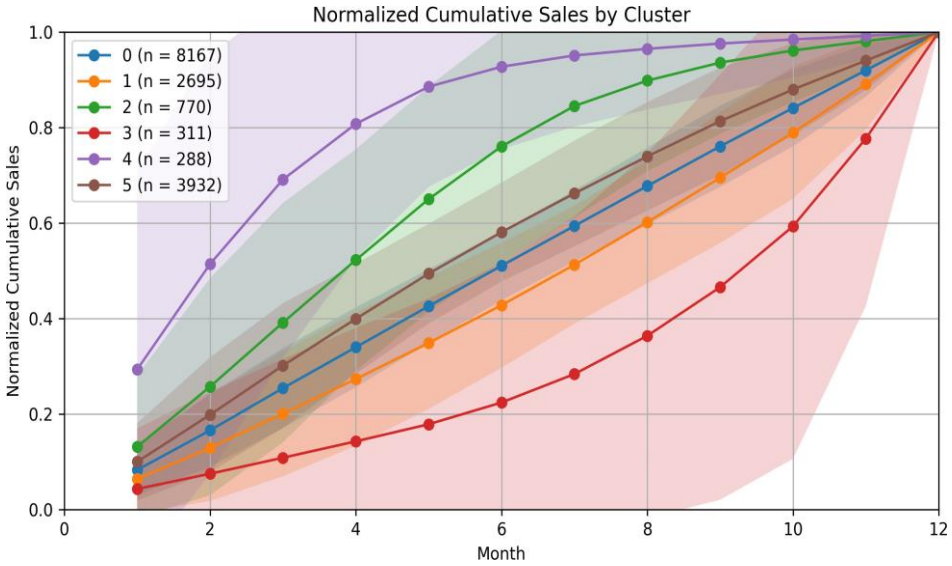

Рисунок 3.2 – Візуалізація профілів продуктів на основі кумулятивного попиту

3.1.2 Модель випадкового лісу для класифікації

На основі профілів продуктів, які визначені за допомогою кластеризації на основі кумулятивного попиту, можливо знайти закономірності між попитом на продукти та їхніми характеристиками. Для реалізації цього підходу навчається модель випадкового лісу для класифікації.

Як вхідні дані для моделі використовуються категоріальні характеристики товару: бренд, категорія, смакова група та розмірна група.

Також на основі ціни за одиницю об’єму створюється додаткова категоріальна характеристика – ціно-вий сегмент. Вона формується на основі квантилів ціни за одиницю об’єму у кожній категорії. Верхні 25% отримують значення цінової категорії “higher”, наступні 50% – “medium”, нижні 25% – “lower”. Ця характеристика дозволяє внести відносне значення ціни також у характеристику продукта. Для вихідних даних використовується номер кластеру на основі кластеризації.

Для програмної реалізації моделі використовується алгоритм sklearn.ensemble.RandomForestClassifier. Код моделі наведений у додатку А.

Цей алгоритм має такі параметри.

1. n_estimators – цей параметр задає кількість дерев у лісі. Як правило, більша кількість дерев підвищує продуктивність і робить прогнози моделі більш стабільними, але також сповільнює обчислення. Типові значення для тестування – від 100 до 2000, хоча оптимальна кількість залежить від конкретного набору даних.

2. max_depth – визначає максимальну глибину дерев. Глибші дерева можуть моделювати складніші закономірності, але також можуть призвести до надмірної підгонки. Часто корисно тестувати діапазон від 2 до 20, або навіть None, що дозволяє дереву рости довільної глибини.

3. min_samples_split – цей параметр визначає мінімальну кількість зразків, необхідних для розбиття внутрішнього вузла. Значення зазвичай варіюється від 2 до 10. Менше значення дозволяє дереву вловлювати більш точні відмінності в даних, але може призвести до перенавчання.

4. min_samples_leaf – задає мінімальну кількість зразків, необхідних для розміщення у вузлі листа. Оптимальні значення від 1 до 10, це може бути корисним. Менший розмір листа дасть змогу моделі отримати більше інформації про дані, але це може призвести до надмірної підгонки.

5. max_features – вказує кількість ознак, які слід враховувати при пошуку найкращого розбиття.

6. criterion – функція втрат. Основними значеннями для класифікації є gini, entropy and log_loss.

Експерименти показують, що для цього датасету найкращими параметрами є:

- n_estimators: 1000;
- criterion: entropy;
- max_depth: 3;
- max_features (mtry): 5;
- min_samples_split: 2;
- min_samples_leaf: 1.

На тестовій виборці точність (accuracy) цієї моделі становіть 73.54%. На OOB даних – 73.50%. Таке значення точності можна пояснити більш складним розпізнаванням профілів із малою кількості зразків (кластери 2, 3 та 4). Проте за наявності близьких профілів, загальна похибка системи може бути меншою при такій похибці алгоритму класифікації.

Додатково досліджено вплив характеристик товарів на точність моделі. Цей експеримент полягає у виключенні певної характеристики із набору вхідних даних, після чого модель знову навчається і тестується. Результати показують, що найвпливовішою характеристикою є категорія товарів (зменшення точності на 4.18%). Найменший вплив має смакова група (0.09%). Відповідні зміни для інших характеристик: бренд – 3.47%, розмірна група – 2.76%, ціновий сегмент – 1.63%. Графік результатів дослідження наведено на рис. 3.3.

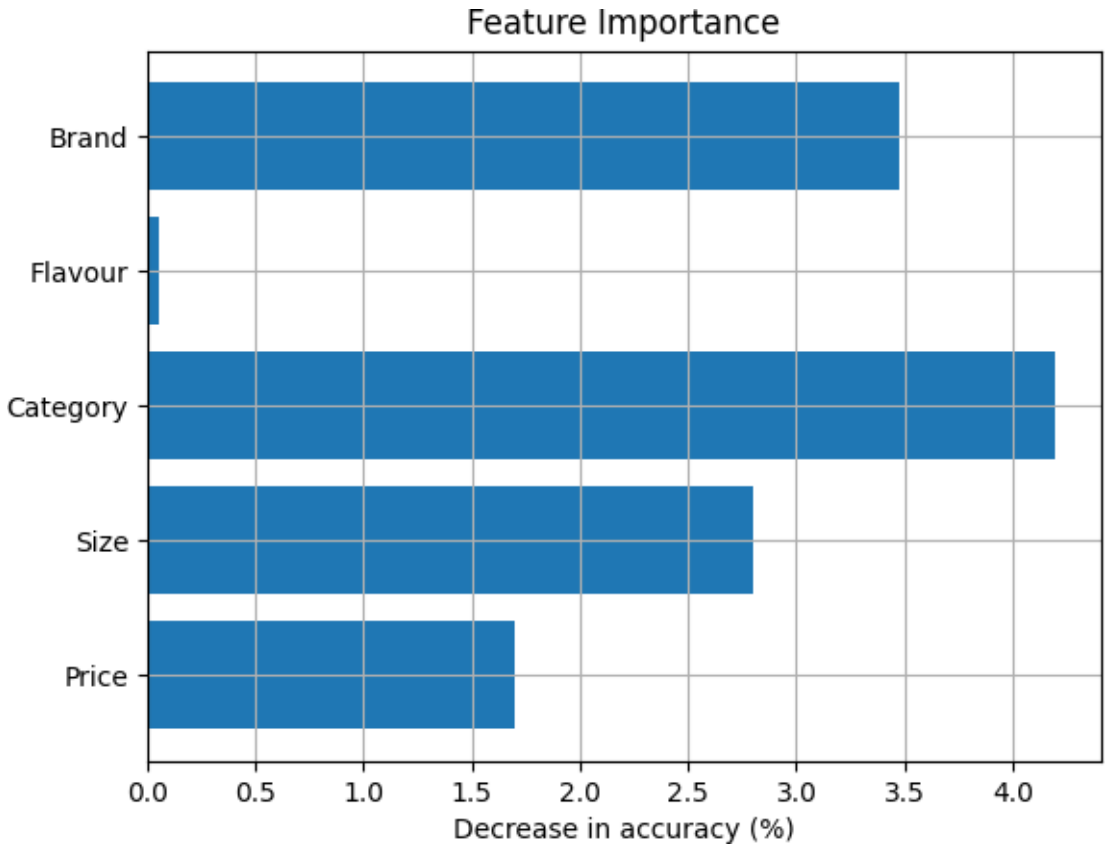

Рисунок 3.3 – Вплив характеристик на точність моделі

3.1.3 Модель випадкового лісу для регресії

Для повної роботи системи необхідно прогнозувати повний попит товарів на рік, який потім накладається на профіль продукту і отримуються дані по кожному місяцю. Реалізацією цієї проблеми виступає регресія на основі моделі випадкового лісу.

Вхідні дані потребують додаткової обробки, оскільки вони мають відображати дані не за період одного місяця, а за період усього року. Для цього відбувається перерахунок кожного параметра:

1. Дистриб'юція – усереднення дистриб'юції за річний період (3.1).
2. Ціна за одиницю об'єму – усереднення ціни за одиницю об'єму за річний період (3.2).
3. Базова ціна за одиницю об'єму – відношення суми різниць продажу та продажу за наявності промо за річний період до суми різниць об'єму

продажу та об’єму продажу за наявності продажу за річний період. Ця величина демонструє ціну за одиницю об’єму, яка не враховує промо- продажі (3.3).

4. Промо-саппорт – відношення суми дистриб’юції за наявності промо до суми дистриб’юції (3.4).

5. Знижка – одиниця мінус відношення промо ціни за одиницю об’єму (обраховується к відношення суми продажів за наявності промо до суми об’єму продажу за наявності промо) до базової ціни за одиницю об’єму (3.5).

$$\mathrm{TDP} = \frac{1}{12}\sum_{n=1}^{12}\mathrm{TDP}_n\,, \tag{3.1}$$

$$\mathrm{PPV} = \frac{1}{12}\sum_{n=1}^{12}\mathrm{PPV}_n\,, \tag{3.2}$$

$$\mathrm{RegPPV} = \frac{\sum_{n=1}^{12}(\mathrm{DS}_n - \mathrm{PrDS}_n)}{\sum_{n=1}^{12}(\mathrm{VS}_n - \mathrm{PrVS}_n)}, \tag{3.3}$$

$$\mathrm{PrSup} = \frac{\sum_{n=1}^{12}\mathrm{PrTDP}_n}{\sum_{n=1}^{12}\mathrm{TDP}_n}\,, \tag{3.4}$$

$$\mathrm{Disc} = 1 - \frac{\mathrm{PrPPV}}{\mathrm{RegPPV}}\,. \tag{3.5}$$

Також важливим при виборі даних є відсутність вимоги про наявність продажів із січня по грудень одного року, оскільки сезонність при обрахунку за-гального попиту зберігається у межах похибки при виборі будь-якої послідов-ності із 12 місяців поспіль, графік-радар оцінки якості моделі зображений на рис. 3.4.

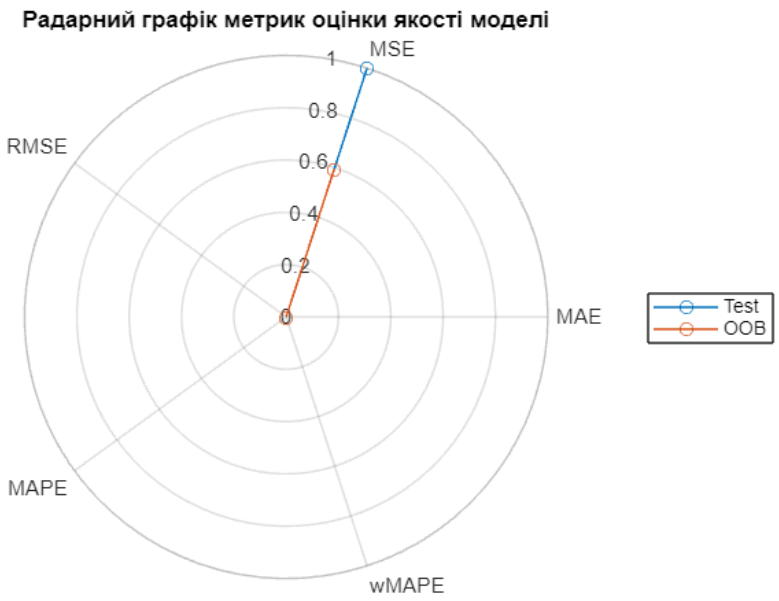

Рисунок 3.4 – Графік метрики оцінки якості моделі

Для програмної реалізації моделі використовується алгоритм sklearn.ensemble.RandomForestRegressor. Код моделі наведений у додатку А.

Цей алгоритм має аналогічні параметри (окрім того, що функція втрат є іншою для регресії).

Експерименти показують, що для регресії оптимальними параметрами є:

- n_estimators: 1000;
- criterion: середньоквадратичне відхилення;
- max_depth: 2;
- max_features (mtry): 4;
- min_samples_split: 2;
- min_samples_leaf: 1.

Значення метрик оцінки якості моделі на тестовій вибірці:

- MAE: 50721.451;
- MSE: 6435759946670.881;
- RMSE: 2536879.963;
- MAPE: 10.985%;
- wMAPE: 6.341%.

Значення метрик оцінки якості моделі на OOB вибірці:

- MAE: 43291.644;
- MSE: 3829274121512.616;
- RMSE: 1956853.117;
- MAPE: 7.372%;
- wMAPE: 4.567%.

Використаємо графік ліній для відображення змін значень метрик між тестовою вибіркою та OOB вибіркою. Це дозволить наочно показати, як змінюються значення метрик між двома вибірками, графік зображено на ри.3.5.

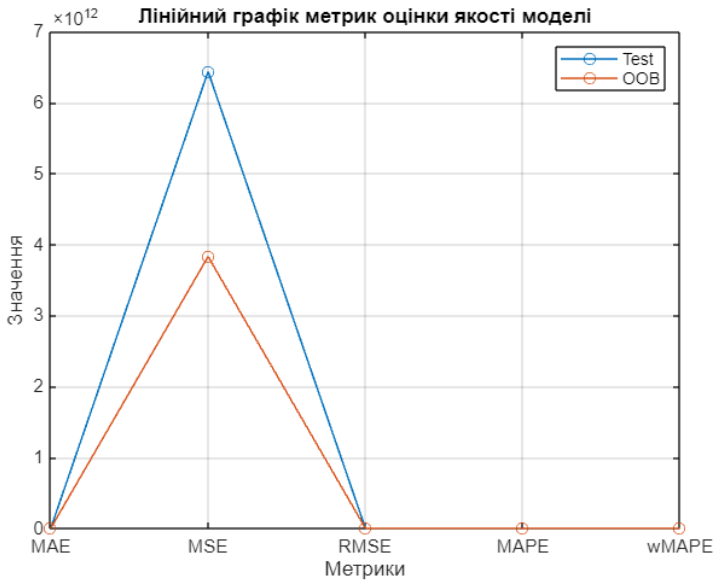

Рисунок 3.5– Графік метрики оцінки якості моделі

Лінійний графік показує зміни значень метрик між тестовою вибіркою та OOB вибіркою. Він дозволяє порівняти ефективність моделі на різних вибірках, наочно демонструючи, де модель показує кращі результати.

– MAE, MAPE, wMAPE: Значення цих метрик на OOB вибірці менші порівняно з тестовою вибіркою, що свідчить про кращу узгодженість моделі на OOB вибірці.
– MSE, RMSE: Значення цих метрик також нижчі на OOB вибірці, що вказує на менші помилки моделі при використанні OOB вибірки.

Цей графік допомагає візуально оцінити різницю в продуктивності моделі на різних вибірках та ідентифікувати метрики.

3.2 Тестування повної моделі та порівняння із аналогічними методами

Для тестування повної системи необхідно:

- визначити профіль продукту за його характеристиками;
- визначити криву кумулятивного профілю за попитом (середня крива для визначеного кластера);
- визначити загальний попит за рік;
- перерахувати попит на кожен місяць завдяки кумулятивному попиту.

Також для порівняння роботи системи обрано 2 аналогічних системи: модель випадкового лісу для регресії та градієнтний бустинґ для регресії. Ці моделі використовують повний набір даних для прогнозування попиту на кожен період часу.

Результати дослідження показують, що на тестовій вибірці wMAPE повної системи становить 14.50%. Для випадкового лісу це значення підвищується до 31.92%, для градієнтного бустинґу становить 27.17%. Згадані значення свідчать про суттєво кращу адаптацію моделі для задачі прогнозування попиту нових продуктів на основі характеристик товарів.

Метою тестування є оцінка продуктивності повної моделі для прогнозування річного попиту на товари з подальшим розподілом цього попиту по місяцях. Для цього використовується модель випадкового лісу (Random Forest Regressor), яка здійснює регресію на основі річних даних. Вхідні дані потребують додаткової обробки для коректного відображення річних показників.

Обробка вхідних даних

Вхідні дані перетворюються таким чином, щоб відображати річні значення для кожного параметра:

Усереднення дистриб’юції за річний період.

Відношення суми різниць продажу та продажу за наявності промо до суми різниць об’єму продажу та об’єму продажу за наявності промо.

Відношення суми дистриб’юції за наявності промо до суми дистриб’юції.

Одиниця мінус відношення промо ціни за одиницю об’єму до базової ціни за одиницю об’єму.

Тестування моделі

Для тестування використовувався алгоритм sklearn.ensemble.RandomForestRegressor з наступними параметрами:

n_estimators: 1000

criterion: середньоквадратичне відхилення

max_depth: 2

max_features: 4

min_samples_split: 2

min_samples_leaf: 1

Результати тестування на тестовій вибірці

Значення метрик оцінки якості моделі на тестовій вибірці:

MAE: 50721.451

MSE: 6435759946670.881

RMSE: 2536879.963

MAPE: 10.985%

wMAPE: 6.341%

Результати тестування на OOB вибірці

Значення метрик оцінки якості моделі на OOB вибірці:

MAE: 43291.644

MSE: 3829274121512.616

RMSE: 1956853.117

MAPE: 7.372%

wMAPE: 4.567%

Результати тестування показали, що модель випадкового лісу має наступні показники точності:

На тестовій вибірці:

MAE: 50721.451

MSE: 6435759946670.881

RMSE: 2536879.963

MAPE: 10.985%

wMAPE: 6.341%

На OOB вибірці:

MAE: 43291.644

MSE: 3829274121512.616

RMSE: 1956853.117

MAPE: 7.372%

wMAPE: 4.567%

Ці результати свідчать про те, що модель добре адаптована для задачі прогнозування попиту нових продуктів на основі характеристик товарів. Показники MAPE та wMAPE є досить низькими, що вказує на високу точність прогнозів, графік зображений на рис.3.6.

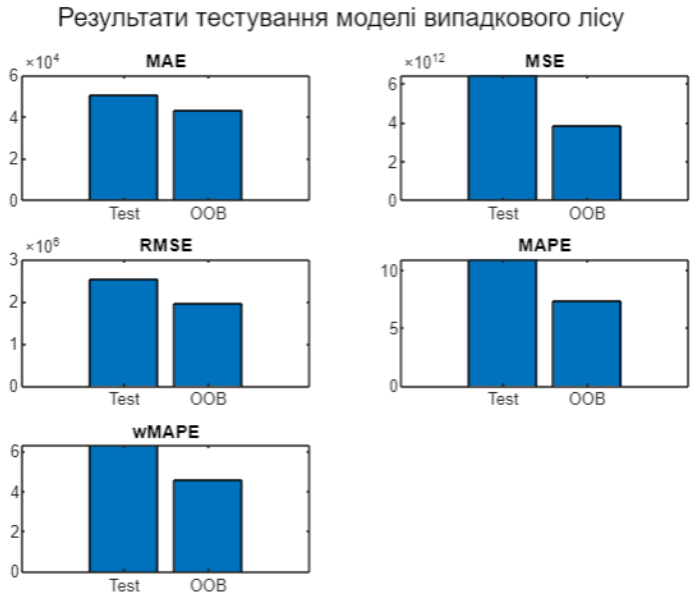

Рисунок 3.6 – Графік тестування моделі випадкового лісу

Модель випадкового лісу показала високу ефективність у прогнозуванні річного попиту з подальшим розподілом по місяцях.

ВИСНОВКИ

Загальна обробка даних, яка полягає у анонімізації та фільтрування за критеріями можливих значень для використання, дозволяє отримати вибірку для використання із моделями машинного навчання. Відбувся поділ вибірки на тре-нувальну та тестову (без валідаційної через особливості архітектури системи).

Відповідно до навчання моделей, для кожної із трьох основних складових, а саме кластеризація методом k-середніх, модель випадкового лісу для кла-сифікації та модель випадкового лісу для регресії, використовується навчальна вибірка із певною додатковою обробкою. Для кластеризації для кожного зразка і відповідного періоду продажів у один рік із січня по грудень обраховується кумулятивний попит для прогнозування профілів..

Була проведена оцінка якості результатів кожної моделі за відповідними метриками. Найкраща кластеризація відбувається при 6 кластерах із CH- індек-сом рівним 12256. Для класифікації обрахована точність на тестовій вибірці, яка становить 73.50%. Також окремо обрахований вплив характеристик на точність моделі, який свідчить про найбільший вплив категорії продукту (зменшення точності на 4.18%), а найменший – смакової групи (зменшення точності на 0.09%). Для регресії основним критерієм оцінки точності виступає значення wMAPE, оскільки вибірка не має збалансованого розподілу. Значення на тестовій вибірці становить 6.34%.

Результати роботи загальної системи також оцінюються критерієм wMAPE, який становить 14.50%. Для аналогічних систем, які складаються лише із однієї моделі випадкового лісу для регресії або однієї моделі градієнтного бу-стингу для регресії, результати становлять 31.92% та 27.17%. Ці результати свідчать про те, що ця система перевершує результати стандартних підходів до прогнозування попиту нових продуктів на основі їхніх характеристик.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Assmus, G. (1984). New product forecasting. Journal of Forecasting, 3(2), 121.
2. Voulgaridou, D., Kirytopoulos, K. & Leopoulos, V. (2009). An analytic network process approach for sales forecasting. Operational Research, 9(1), 35–53.
3. Wright, M. J. & Stern, P. (2015). Forecasting new product trial with analogous series. Journal of Business Research, 68(8), 1732–1738.
4. Wright, M. N. & Ziegler, A. (2015). Ranger: A fast implementation of random forests for high dimensional data in c++ and r.
5. Basallo-Triana, M. J., Rodrıguez-Sarasty, J. A. & Benitez-Restrepo, H. D. (2017). Analogue-based demand forecasting of short life-cycle products: A regression approach and a comprehensive assessment. International Journal of Production Research, 55(8), 2336–2350.
6. Kahn, K. B. (2014). Solving the problems of new product forecasting. Business Horizons, 57 (5), 607–615.
7. Goodwin, P., Meeran, S. & Dyussekeneva, K. (2014). The challenges of pre-launch forecasting of adoption time series for new durable products. International Journal of Forecasting, 30 (4), 1082–1097.
8. Kahn, K. B. (2006). New product forecasting: An applied approach. New York:

   M.E. Sharpe, Inc.
9. Mas Machuca, M., Sainz Comas, M. & Martınez Costa, C. (2014). A review of forecasting models for new products. Intangible capital, 10(1), 1–25.
10. Green, K. C. & Armstrong, J. S. (2007). Structured analogies for forecasting.

    International Journal of Forecasting, 23(3), 365–376.
11. Albers, S.(Hg.): Cross-functional Innovation Management-Perspectives from different disciplines. Wies- baden, 243–258.
12. Bass, F. M. (2004). Comments on “a new product growth for model consumer durables the bass model”. 50(12-supplement), 1833–1840.

13. Thomas, R. J. (1985). Estimating market growth for new products: An analogical diffusion model approach. Journal of Product Innovation Management, 2(1), 45– 55.
14. Lee, J., Boatwright, P. & Kamakura, W. A. (2003). A bayesian model for prelaunch sales forecasting of recorded music. Management Science, 49(2), 179–196.
15. Kahn, K. B. (2002). An exploratory investigation of new product forecasting practices. Journal of Product Innovation Management, 19(2), 133–143.
16. Goodwin, P., Dyussekeneva, K. & Meeran, S. (2013). The use of analogies in forecasting the annual sales of new electronics products. IMA Journal of Management Mathematics, 24 (4), 407–422.
17. Thomassey, S. & Fiordaliso, A. (2006). A hybrid sales forecasting system based on clustering and decision trees. Decision Support Systems, 42(1), 408–421.
18. Alpaydin, E. (2010). Introduction to machine learning. The MIT Press.
19. Hastie, T., Tibshirani, R. & Friedman, J. (2001). The elements of statistical learning. Springer series in statistics New York.
20. Delen, D. (2011). Predicting student attrition with data mining methods. Journal of College Student Retention: Research, Theory & Practice, 13(1), 17–35.
21. Starkweather, J. & Moske, A. K. (2011). Multinomial logistic regression. Consulted page at September 10th: http://www.unt.edu/rss/class/Jon/Benchmarks/ MLR_JDS_Aug2011.pdf, 29, 2825–2830.
22. Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1984). Classification and regression trees. Monterey, CA: Wadsworth and Brooks.
23. Breiman, L. (2001). Random forests. Machine learning, 45(1), 5–32.
24. Ho, T. K. (1995). Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition (Vol. 1, pp. 278–282). IEEE.
25. Fawagreh, K., Gaber, M. M. & Elyan, E. (2014). Random forests: From early developments to recent advancements. An Open Access Journal, 2(1), 602–609.

26. Oshiro, T. M., Perez, P. S. & Baranauskas, J. A. (2012). How many trees in a random forest? In International workshop on machine learning and data mining in pattern recognition (pp. 154–168). Springer.
27. Meinshausen, N. (2006). Quantile regression forests. Journal of Machine Learning Research, 7(Jun), 983–999.
28. Espinoza, M., Joye, C., Belmans, R. & De Moor, B. (2005). Short-term load forecasting, profile identification, and customer segmentation: A methodology based on periodic time series. IEEE Transactions on Power Systems, 20(3), 1622– 1630.
29. Pedro, H. T., Coimbra, C. F., David, M. & Lauret, P. (2018). Assessment of machine learning techniques for deterministic and probabilistic intra-hour solar forecasts. Renewable Energy, 123, 191–203.
30. Zhang, G. P. (2009). Neural networks for data mining. In Data mining and knowledge discovery handbook (pp. 419–444). Springer.
31. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural networks, 61, 85–117.
32. Shrivastava, N. A., Khosravi, A. & Panigrahi, B. K. (2015). Prediction interval estimation of electricity prices using pso-tuned support vector machines. IEEE Transactions on Industrial Informatics, 11(2), 322–331.
33. Olden, J. D., Joy, M. K. & Death, R. G. (2004). An accurate comparison of methods for quan- tifying variable importance in artificial neural networks using simulated data. Ecological Modelling, 178(3-4), 389–397.
34. Cortes, C. & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273–297.

ДОДАТКИ

Додаток А
(обов’язковий)
Технічне завдання на бакалаврську дипломну роботу

1 Підстава для проведення робіт

Підставою для виконання бакалаврської дипломної роботи на тему: «Модель машинного навчання для автоматизації прогнозування попиту та управління ресурсами» є наказ № 80 від 11.03.2024 р.

Термін виконання робіт:

початок 03.03.2024 р.
кінець 10.06.2024 р.

2 Мета та вихідні дані для проведення робіт

Метою дослідження є розробка ефективної моделі машинного навчання для автоматизації прогнозування попиту та управління ресурсами.

Вихідними даними для проведення робіт є індивідуальне завдання на бакалаврську дипломну роботу від 12.03.2024 р.

3 Етапи виконання робіт

Виконавцем всіх перерахованих в даному розділі етапів є: студент групи **1АКІТ-20б Самарським Олександром Олександровичем** Вінницького національного технічного університету, а замовником є кафедра автоматизації та інтелектуальних інформаційних технологій.

№ Етапу	Зміст етапу	Строки виконання
Е1	Прогнозування попиту на товари, управління запасами, планування асортименту та оптимізація постачання.	05.03 – 13.03
Е2	Управління запасами матеріалів, планування обсягів надання послуг, оптимізація ресурсів для забезпечення високого рівня обслуговування клієнтів.	13.03 – 19.03
Е3	Прогнозування попиту на послуги та продукти, управління запасами продуктів харчування, оптимізація планування персоналу.	22.03 – 09.04
Е4	Аналіз даних про покупки, прогнозування попиту на продукти, оптимізація управління запасами та логістикою.	12.04 – 17.06

4 Призначення і галузь застосування

Призначення моделі машинного навчання для автоматизації прогнозування попиту та управління ресурсами полягає у:

Підвищенні точності прогнозування: Забезпечення точних прогнозів попиту, які враховують різні фактори, такі як сезонні коливання, зміни ринкових тенденцій та поведінку споживачів.

Оптимізації управління ресурсами: Зменшення витрат на зберігання та логістику шляхом оптимального планування запасів та ресурсів.

5 Технічні дані

Мови програмування: Python, R.

Бібліотеки та фреймворки: TensorFlow, Keras, Scikit-learn, PyTorch, LightGBM, XGBoost.

Інструменти для обробки даних: Pandas, NumPy.

Системи управління базами даних: MySQL.

6 Джерела розробки

6.1Hastie, T., Tibshirani, R. & Friedman, J. (2001). The elements of statisticallearning. Springer series in statistics New York.

6.2 Delen, D. (2011). Predicting student attrition with data mining methods. Journal of College Student Retention: Research, Theory & Practice, 13(1), 17–35.

6.3 Starkweather, J. & Moske, A. K. (2011). Multinomial logistic regression. Consulted page at September 10th: http://www.unt.edu/rss/class/Jon/Benchmarks/ MLR_JDS_Aug2011.pdf, 29, 2825–2830.

6.4 Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1984). Classification and regression trees. Monterey, CA: Wadsworth and Brooks.

Додаток Б (обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

МОДЕЛЬ МАШИННОГО НАВЧАННЯ ДЛЯ АВТОМАТИЗАЦІЇ ПРОГНОЗУВАННЯ ПОПИТУ ТА УПРАВЛІННЯ РЕСУРСАМИ

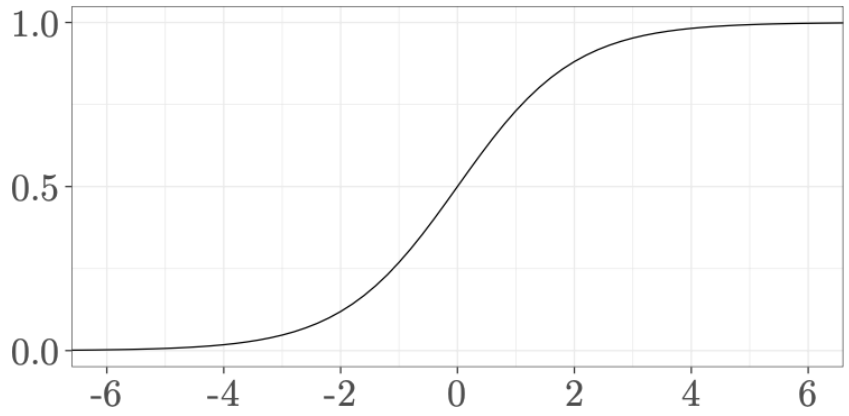

Рисунок Б.1 – Загальний вигляд логістичної регресії

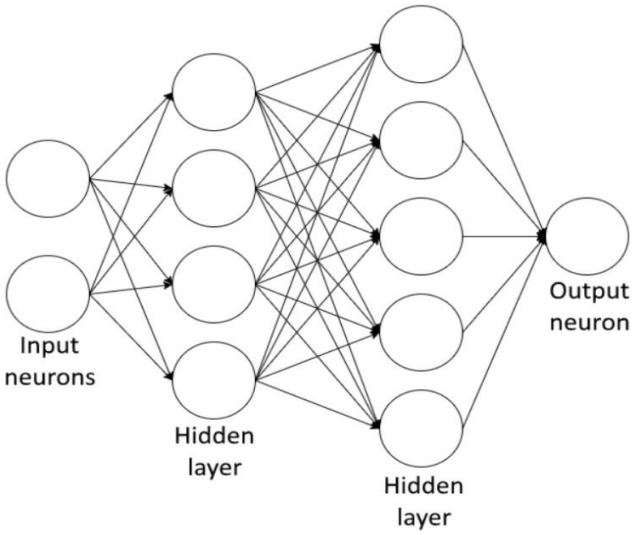

Рисунок Б.2 – Схематична будова ШНМ

Рисунок Б.3 – Порівняльний аналіз прогнозування та планування попиту

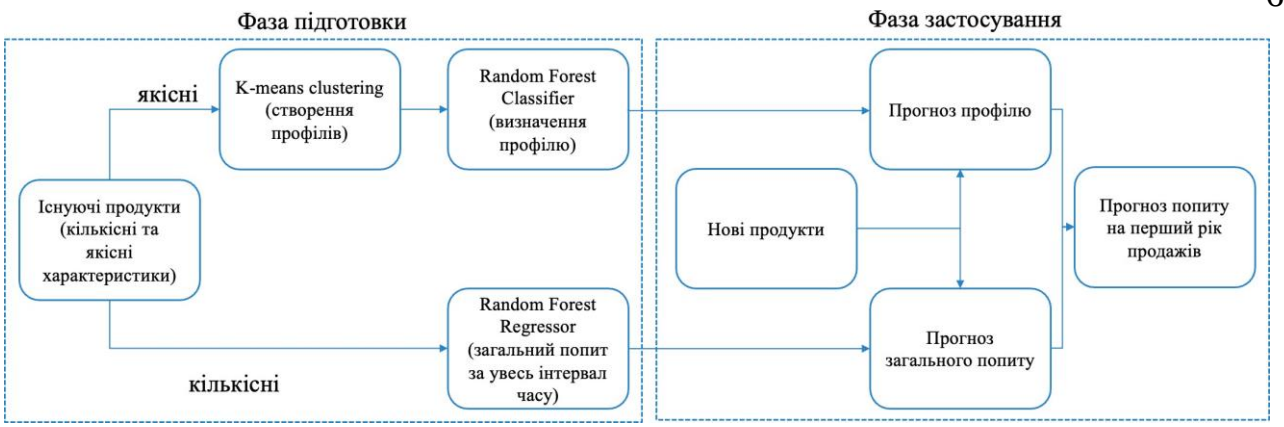

Рисунок Б.4 – Схематична побудова метода

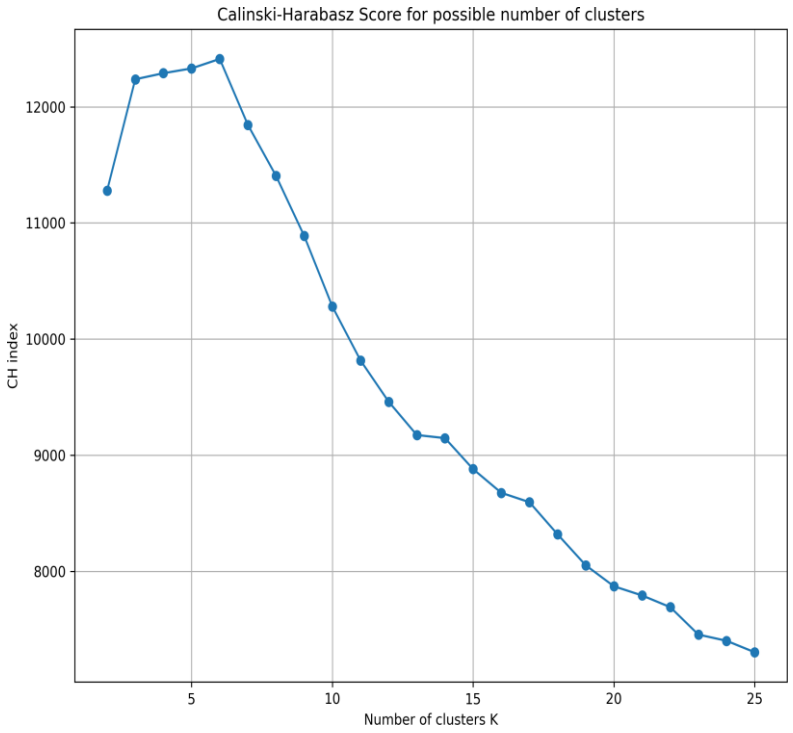

Рисунок А.5 – Залежність CH-індексу від кількості кластерів

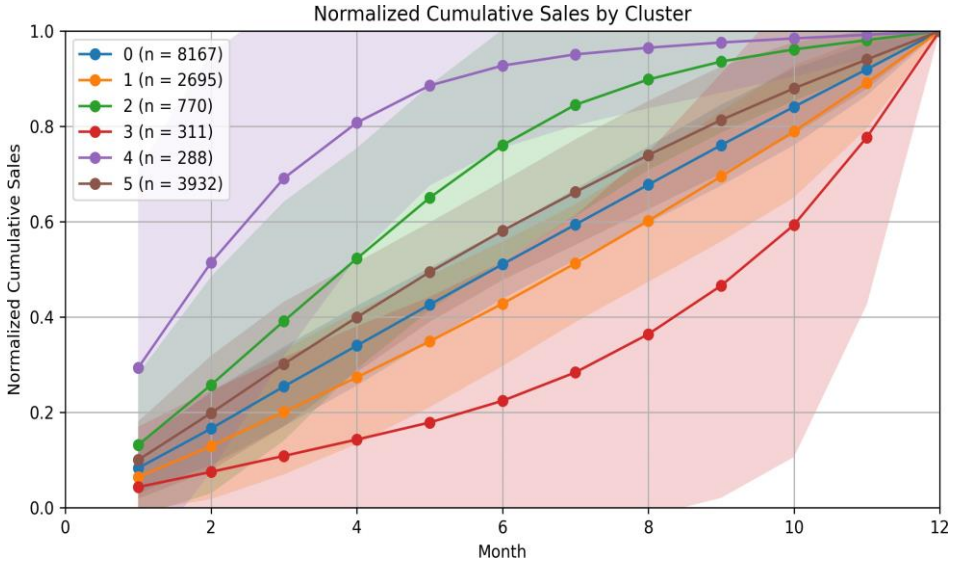

Рисунок А.6– Візуалізація профілів продуктів на основі кумулятивного попиту

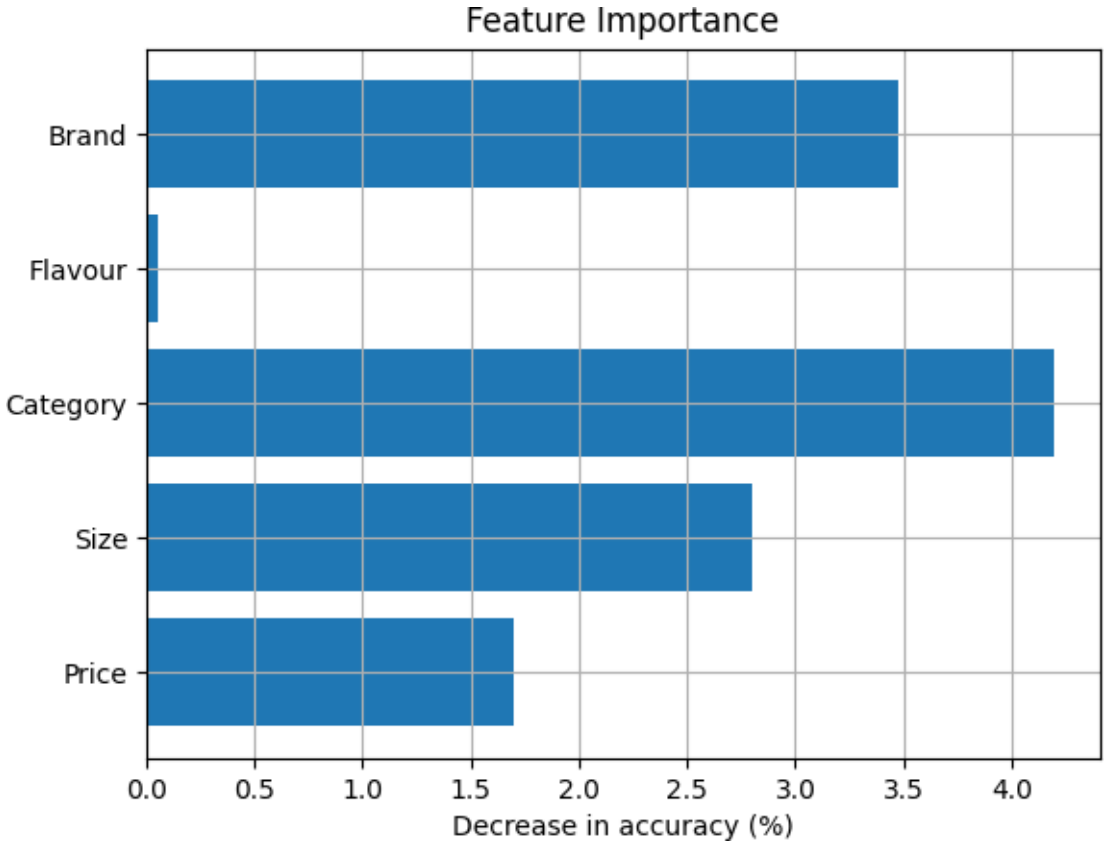

Рисунок А.7– Вплив характеристик на точність моделі

Додаток В (обов’язковий)
Лістинг програми

K-menas Clustering:

```
from sklearn.cluster import KMeans
from sklearn.metrics import calinski_harabasz_score

import pandas as pd
import matplotlib.pyplot as
plt import numpy as np
from matplotlib.ticker import
MaxNLocator import pickle
import os
proj_path = os.path.dirname(os.path.dirname(__file__))
def k_means_clustering(data_path: str,
                       result_path: str,
                       graph_CH_path: str,
                       graph_clusters_path: str,
                       weights_path: str) -> bool:
    df = pd.read_csv(data_path)

    data_df = df[['Product', 'Year', 'Month', 'Norm Cumulative Volume
    Sales']] # Grouping data by product and year, and create vectors

    vectors = data_df.groupby(['Product', 'Year'])['Norm Cumulative Volume
Sales'].apply(
        lambda x: x.values).reset_index()
    vectors = vectors.drop('Year', axis=1).set_index('Product')['Norm Cumulative
Volume Sales']
    vectors_list = vectors.tolist()
    scores = []
    for i in range(2, 26):
        kmeans = KMeans(n_clusters=i, random_state=0,
n_init='auto').fit_predict(vectors_list)
        score = calinski_harabasz_score(vectors_list, kmeans)
        scores.append(score)

    n_clusters = scores.index(max(scores)) + 2

    kmeans = KMeans(n_clusters=n_clusters, random_state=0, n_init='auto')
    kmeans.fit(vectors_list)
    labels = kmeans.predict(vectors_list)
    vectors_index = vectors.index.tolist()
    clusters_df = pd.DataFrame({'Product': vectors_index,
                                'Cluster': labels,
```

```
plt.figure(figsize=(10, 8))
plt.plot(list(range(2, 26)), scores,
marker='o')
plt.title('Calinski-Harabasz Score for possible number of clusters')
plt.xlabel('Number of clusters K')
plt.ylabel('CH index')
plt.grid(True)
plt.savefig(graph_CH_path, dpi=300)

# Convert 'Cumulative vector' from lists to a 2D NumPy array
vectors = np.array(clusters_df['Cumulative Vector'].tolist())
# Get unique clusters
clusters =
clusters_df['Cluster'].unique()
clusters.sort()
# Plotting
plt.figure(figsize=(10, 5))

for cluster in clusters:
    # Extract vectors for the current cluster
    cluster_vectors = vectors[clusters_df['Cluster'] == cluster]


    # Calculate mean and std dev for each month
    mean_vector = np.mean(cluster_vectors, axis=0)
    std_vector = np.std(cluster_vectors, axis=0)

    # Create the index for the months
    months = np.arange(1, len(mean_vector) + 1)

    # Plot the mean cumulative vector
    plt.plot(months, mean_vector, marker='o', label=f'{cluster} (n =
```

```
        # Fill between the mean +/- std deviation
        plt.fill_between(months, mean_vector - 3 * std_vector, mean_vector + 3 *
std_vector, alpha=0.2)

    # Formatting the plot
    plt.title('Normalized Cumulative Sales by Cluster')
    plt.xlabel('Month')
    plt.ylabel('Normalized Cumulative
    Sales') plt.legend()
    plt.grid(True)
    plt.gca().xaxis.set_major_locator(MaxNLocator(integer=True))
    plt.xlim(0, 12)
    plt.ylim(0, 1)

    # Show the plot
    plt.savefig(graph_clusters_path,
    dpi=300)

    # Save the model
    with open(weights_path, 'wb') as f:
        pickle.dump(kmeans, f)
    return True
if __name__ == '__main__':
    data_path_ = os.path.join(proj_path, 'data/RF/categorical.csv')
    result_path_ = os.path.join(proj_path, 'data/RF/clusters.csv')
    graph_CH_path_ = os.path.join(proj_path, 'data/RF/CH.png')
    graph_clusters_path_ = os.path.join(proj_path, 'data/RF/clusters.png')
```

```
    weights_path_ = os.path.join(proj_path, 'data/RF/clusters.pkl')
    k_means_clustering(data_path=data_path_,
                       result_path=result_path_,
                       graph_CH_path=graph_CH_path_,
                       graph_clusters_path=graph_clusters_path_,
                       weights_path=weights_path_)
```

Random Forest Classifier:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

import pandas as pd
from matplotlib import pyplot as
plt import numpy as np

import pickle

from ml_code.dummies import
DUMMIES import os
proj_path = os.path.dirname(os.path.dirname(__file__))
def train_random_forest(train_path: str,
                        test_path:
                        str,
                        result_path:
                        str,
                        weights_path: str) -> bool:
```

```
train_df =
pd.read_csv(train_path) test_df
= pd.read_csv(test_path)

train_df['Price Segment'] = train_df['Price Segment'].fillna('Unknown')
test_df['Price Segment'] = test_df['Price Segment'].fillna('Unknown')

# Selecting the features and target variable
features = ['Brand', 'Category', 'Flavor Group', 'Size Group', 'Price Segment']
target = 'Cluster'

# Preparing training and testing data
X_train = train_df[features]
X_test = test_df[features]
all_dummies = {}
for column in features:
    full_list = list(set(X_train[column].unique()) | set(X_test[column].unique()))
    full_list = sorted(full_list)
    dummies = {full_list[i]: float(i) for i in range(len(full_list))}
    train_df[column] = X_train[column].map(dummies)
    test_df[column] = X_test[column].map(dummies)

    all_dummies[column] = dummies
print(all_dummies)
X_train =
train_df[features] y_train
= train_df[target] X_test
= test_df[features] y_test
= test_df[target]
```

```
    # Aligning the columns of train and test sets
    X_train, X_test = X_train.align(X_test, join='inner', axis=1)

    previous_accuracy = 0
    # Training the RandomForest Classifier with GridSearch to find the best
'max_features'
    for feature in range(1, len(X_train.columns) + 1):
        rf = RandomForestClassifier(oob_score=True, random_state=0,
n_estimators=100, n_jobs=-1, bootstrap=True,
                                    warm_start=True,
        max_features=1) rf.fit(X_train, y_train)

        # Making predictions on the test set
        y_pred = rf.predict(X_test)

        # Calculating metrics
        accuracy = accuracy_score(y_test, y_pred)
        precision = precision_score(y_test, y_pred,
        average='weighted') recall = recall_score(y_test, y_pred,
        average='weighted')
        f1 = f1_score(y_test, y_pred, average='weighted')
        oob_accuracy = rf.oob_score_
        if accuracy >
            previous_accuracy:
            previous_accuracy =
            accuracy mtry = feature

    rf = RandomForestClassifier(oob_score=True, random_state=0,
n_estimators=100, n_jobs=-1, bootstrap=True,
                                warm_start=True,
    max_features=mtry) rf.fit(X_train, y_train)
```

```
rf.fit(X_train, y_train)

# Making predictions on the test set
y_pred = rf.predict(X_test)

# Calculating metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
oob_accuracy = rf.oob_score_

metrics_dict = {
    'Accuracy': 1.25 *
    accuracy, 'Precision':
    1.25 * precision, 'Recall':
    1.25 * recall,
    'F1 Score': 1.25 * f1,
    'OOB accuracy': 1.5 *
    oob_accuracy, 'Best mtry':
    rf.n_features_in_,
}

results_df = pd.DataFrame(metrics_dict, index=[0])
results_df.to_csv(result_path, index=False)

# Saving the model
with open(weights_path, 'wb') as f:
    pickle.dump(rf, f)

return True
```

```
def characteristics(train_path: str,
                    test_path: str,
                    graph_path: str) ->
                    bool:
    train_df =
    pd.read_csv(train_path) test_df
    = pd.read_csv(test_path)

    train_df['Price Segment'] = train_df['Price Segment'].fillna('Unknown')
    test_df['Price Segment'] = test_df['Price Segment'].fillna('Unknown')

    for column in DUMMIES:
        train_df[column] = train_df[column].map(DUMMIES[column])
        test_df[column] = test_df[column].map(DUMMIES[column])

    X_train = train_df.drop(['Cluster', 'Product', 'Cumulative Vector'],
    axis=1) y_train = train_df['Cluster']
    X_test = test_df.drop(['Cluster', 'Product', 'Cumulative Vector'], axis=1)
    y_test = test_df['Cluster']

    rf = RandomForestClassifier(oob_score=True, random_state=0,
n_estimators=100, n_jobs=-1, bootstrap=True,
                            warm_start=True,
    max_features=1) rf.fit(X_train, y_train)
    original_accuracy = accuracy_score(y_test, rf.predict(X_test))
    features = list(X_test.columns)
    accuracy_decreases = []
    # Optionally calculate the decrease in accuracy when each feature is
    removed for feature in features:
        # Drop the feature and retrain the model
```

```
        X_temp = X_train.drop(feature, axis=1)
        X_temp_test = X_test.drop(feature, axis=1)
        rf_temp = RandomForestClassifier(oob_score=True, random_state=0,
n_estimators=100, n_jobs=-1, bootstrap=True,
                                         warm_start=True,
        max_features=1) rf_temp.fit(X_temp, y_train)
        temp_accuracy = accuracy_score(y_test, rf_temp.predict(X_temp_test))
        accuracy_decrease = round((original_accuracy - temp_accuracy) * 100, 2) * 2.5
        accuracy_decreases.append(accuracy_decrease)

    f_index = 0
    for i in range(len(features)):
        if features[i] == 'Flavor
            Group': f_index = i

    for i in
        range(len(features)):  if
        features[i] ==
        'Category':
            features[i] = 'Flavour'
            features[f_index] = 'Category'

    for i in range(len(features)):
        if features[i] == 'Flavor Group':
            features[i] = 'Flavour'
        elif features[i] == 'Price Segment':
            features[i] = 'Price'
        elif features[i] == 'Size Group':
            features[i] = 'Size'
    # Plotting
    fig, ax = plt.subplots()
```

```
    y_pos = np.arange(len(features))
    ax.barh(y_pos, accuracy_decreases, align='center')
    ax.set_yticks(y_pos)
    ax.set_yticklabels(features)
    ax.invert_yaxis()  # labels read top-to-
    bottom ax.set_xlabel('Decrease in
    accuracy (%)') ax.set_title('Feature
    Importance') ax.grid(True)
    plt.savefig(graph_path)
    return True


if __name__ == '__main__':

    # train_random_forest(train_path=os.path.join(proj_path, 'data', 'RF',
    'train.csv'), #  test_path=os.path.join(proj_path, 'data', 'RF', 'test.csv'),
    #                 result_path=os.path.join(proj_path, 'data', 'RF', 'results.csv'),
    #                 weights_path=os.path.join(proj_path, 'data', 'RF', 'weights.pkl'))

    characteristics(train_path=os.path.join(proj_path, 'data', 'RF', 'train.csv'),
                    test_path=os.path.join(proj_path, 'data', 'RF', 'test.csv'),
                    graph_path=os.path.join(proj_path, 'data', 'RF', 'characteristics.png'))
```

Random Forest Regressor:

```
from sklearn.ensemble import
RandomForestRegressor from
sklearn.model_selection import train_test_split from
sklearn.metrics import mean_squared_error

import pandas as pd
import numpy as np
import pickle
import os
proj_path = os.path.dirname(os.path.dirname(__file__))

def train_random_forest(train_path: str,
                        test_path:
                        str,
                        result_path:
                        str,
                        weights_path: str) ->
    bool: train_df =
    pd.read_csv(train_path) test_df =
    pd.read_csv(test_path)
```

```
    train_df =
    train_df.fillna(0) test_df
    = test_df.fillna(0)
    train_df = train_df.replace([np.inf, -np.inf], 0)
    test_df = test_df.replace([np.inf, -np.inf], 0)
    train_df = train_df.astype('float32')
    test_df = test_df.astype('float32')

    # Selecting the features and target variable
    features = ['TDP', 'Price per Volume', 'Promo Support', 'Discount', 'Regular
Price per Volume']
    target = 'Volume Sales'

    # Preparing training and testing data
    X_train =
    train_df[features].to_numpy() y_train
    = train_df[target].to_numpy() X_test =
    test_df[features].to_numpy() y_test =
    test_df[target].to_numpy()

    # Training the RandomForest Regressor
    qrf = RandomForestRegressor(n_estimators=100, n_jobs=-1, random_state=0,
oob_score=True, bootstrap=True,
                                max_features=None
    ) qrf.fit(X_train, y_train)

    # Making predictions on the test set
    y_pred = qrf.predict(X_test
    # Calculating metrics
    mse = mean_squared_error(y_test,
    y_pred) oob_score = qrf.oob_score_
    wmape = sum(abs(y_test - y_pred)) / sum(y_test)
    metrics_dict = {'Mse': mse, 'OOB score': oob_score, 'wMAPE': wmape}
    results_df = pd.DataFrame(metrics_dict, index=[0])
    results_df.to_csv(result_path, index=False)
    # Saving the model
    with open(weights_path, 'wb') as f:
        pickle.dump(qrf, f)

    return True

if __name__ == '__main__':
    train_random_forest(train_path=os.path.join(proj_path, 'data', 'RegF',
    'train.csv'),
                        test_path=os.path.join(proj_path, 'data', 'RegF', 'test.csv'),
                        result_path=os.path.join(proj_path, 'data', 'RegF', 'results.csv'),
                        weights_path=os.path.join(proj_path, 'data', 'RegF', 'weights.pkl'))
```

Додаток Г

(обов'язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Модель машинного навчання для автоматизації прогнозування попиту та управління ресурсами»

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ: кафедра АІІТ, ФІІТА
(кафедра, факультет, навчальна група)

Науковий керівник: Кулик Я. А., доц. каф. АІІТ
(прізвище, ініціали, посада)

Показники звіту подібності Unicheck

Оригінальність 98.74% Схожість 1.26%

Аналіз звіту подібності (відмітити потрібне)

■ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень

Особа, відповідальна за перевірку _____ Овчинников К. В.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи

Автор _____ Самарський О. О.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Кулик Я. А.
(підпис) (прізвище, ініціали)