НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

До захисту допущено:
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____2023 р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»
на тему: «Формалізація та моделювання бізнес-процесів в задачах управління складними об‘єктами**»**

Виконав:
студентка IV курсу, групи КА-91
Уманець Марія Олександрівна _____

Керівник:
доцент, к.ф.-м.н., доцент кафедри ММСА Яковлева А. П. _____

Консультант з економічного розділу:
доцент, к.е.н., доцент кафедри ТПЕ Рощина Н.В. _____

Консультант з нормоконтролю:
к.ф.-м.н., доцент кафедри ММСА Статкевич В. М. _____

Рецензент:
доцент кафедри ФІОТ, к.т.н., доц. Ткач М. М. _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студентка_____

Київ – 2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
НН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський) Спеціальність –
124 «Системний аналіз» Освітньо-професійна програма
«Системний аналіз і управління»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Оксана ТИМОЩУК

«___» _____ 2023 р.

ЗАВДАННЯ

на дипломну роботу студентці

Уманець Марії Олександрівні

1. Тема роботи: «Формалізація та моделювання бізнес-процесів в задачах управління складними об‘єктами», керівник роботи доцент кафедри ММСА, к.ф.м.н. Яковлева А. П., затверджені наказом по університету від «30» травня 2023 р. №2065-с.
2. Термін подання студентом роботи __.06.2023
3. Вихідні дані до роботи: дані про відвідуваність та резерви ресторанів в Японії.
4. Зміст роботи: дослідження та порівняння моделей машинного навчання прогнозування відвідувачів в ресторанному бізнесі.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація
6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Економічний	Рощина Н. В., доцент кафедри ТПЕ		

7. Дата видачі завдання 17.04.2023 р.

Календарний план

№ з/п	Назва етапів виконання роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми ДР	17.04.23-21.04.23	Виконано
2	Ознайомлення зі структурою БДР згідно з Положенням про державну атестацію студентів НТУУ «КПІ ім. І. Сікорського»	17.04.23-21.04.23	Виконано
3	Ознайомлення з ДСТУ 3008:2015 та стандарти ЄСПД	21.04.23-28.04.23	Виконано
4	Проведення дослідження за темою БДР під керівництвом керівника	28.04.23-05.05.23	Виконано
5	Завершення роботи над першим варіантом частини БДР	05.05.23-12.05.23	Виконано
6	Проведення роботи над експериментальною частиною БДР	05.05.23-12.05.23	Виконано
7	Проведення роботи над програмним продуктом	12.05.23-19.05.23	Виконано
8	Оформлення БДР та аналіз отриманих результатів	12.05.23-21.05.23	Виконано

Студентка Марія УМАНЕЦЬ

Керівник Алла ЯКОВЛЕВА

РЕФЕРАТ

Дипломна робота містить 119 сторінок, 53 рисунки, 7 таблиць, 2 додатки, 16 джерел.

Ключові слова: БІЗНЕС-ПРОЦЕСИ, МОДЕЛІ МАШИННОГО НАВЧАННЯ, ПРОГНОЗУВАННЯ В СФЕРІ РЕСТОРАННОГО БІЗНЕСУ, ЗАДАЧА ПРОГНОЗУВАННЯ.

Об‘єкт дослідження: методи моделювання та формалізації бізнес-процесів в задачах управління складними об‘єктами.

Предмет дослідження: моделі машинного навчання, методи оцінки якості моделі та їх застосування в ресторанному бізнесі.

Мета роботи: дослідити методи моделювання бізнес-процесів, реалізувати прогноз відвідування закладів харчування в ресторанному бізнесі з використанням методів машинного навчання.

Використані моделі: лінійна регресія, випадковий ліс, ARIMA (Autoregressive integrated moving average) з сезонною складовою, LSTM (Long Short-Term Memory) мережі.

Отримані результати: програмна реалізація моделей на мові Python для прогнозування середньої кількості відвідувачів, що допомагатиме бізнесу зі стратегічним плануванням, управлінням ресурсами, маркетинговими рішеннями, бенчмаркінгом та аналізом.

В якості подальших досліджень можна вдосконалювати побудовані моделі, застосовувати інші підходи, будувати моделі з урахуванням розширених даних (погодні умови, події), аналізувати та запобігати аномаліям в системах ведення обліку ресторанів.

ABSTRACT

Diploma thesis: 119 p., 53 figures, 7 tables, 2 appendices, 16 references.

BUSINESS PROCESSES, MACHINE LEARNING MODELS, FORECASTING IN THE SPHERE OF RESTAURANT BUSINESS, FORECASTING PROBLEM.

Research object: methods of modeling and formalization of business processes in complex object management tasks.

Research subject: machine learning models, methods of model evaluation, and their application in the restaurant business.

Research purpose: to investigate methods of modeling business processes,implement the prediction of restaurant attendance using machine learning methods.

Models used: linear regression, random forest, ARIMA (Autoregressive integrated moving average) with seasonal component, LSTM (Long Short-Term Memory) networks.

Results obtained: implementation of models in Python for predicting the average number of visitors, which will assist businesses in strategic planning, resource management, marketing decisions, benchmarking, and analysis.

As further research, the constructed models can be improved, other approaches can be applied, models can take into consideration additional data (weather conditions, events), and anomalies in restaurant accounting systems can be analyzed and prevented.

ЗМІСТ

ВСТУП

В сучасному світі процеси управління складними об‘єктами є ключовим напрямком розвитку бізнесу. Взагалі для успішного функціонування підприємства необхідні злагоджені та оптимізовані бізнес-процеси, які здатні адаптуватися до змін, чи то всередині компанії, чи то навколишнього світу. Важливими інструментами у цій справі стає формалізація та моделювання бізнес-процесів. Вони дозволяють розкрити потенціал підприємства, виявити його сильні та слабкі сторони, впроваджувати покращення та бути готовими до вчасного реагування на зміни.

Метою даного дипломного проекту є показати важливість та необхідність моделювання бізнес-процесів, а також дослідити методи для прогнозування цих процесів на прикладі прогнозу кількості відвідувачів у сфері ресторанного бізнесу.

В рамках даного дослідження будуть розглянуті різноманітні методи моделювання, такі як UML (Unified Modeling Language), BPMN (Business Process Model and Notation), метод Монте-Карло а також метод функціонального моделювання SADT (Structured Analysis and Design Technique). Для прогнозування будуть застосовані методи машинного навчання, а саме: лінійна регресія, випадковий ліс (Random Forest), ARIMA (Autoregressive integrated moving average) та LSTM (Long Short-Term Memory). Під час дослідження було використано мову програмування Python.

Дана дипломна робота має на меті сприяння вдосконаленню бізнес-процесів та підвищенню ефективності підприємств у сучасному динамічному бізнес-середовищі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність теми дослідження

За останні декілька років життя з пандемією та повномасштабним вторгненням чимало бізнесів та підприємств зазнали краху та банкрутства. Врятувались компанії, які оперативно налагоджували та вдосконалювали свої бізнес процеси .

Життя показує, що навіть в дуже добре організованих підприємствах існують ризики, які спричиняються непередбачуваними подіями навколишнього світу. Більшість складних об‘єктів, такі як виробничі підприємства, логістичні мережі і тд. характеризуються складною внутрішньою будовою та наявністю великої кількістю взаємозв‘язків та взаємодій. Тому сьогодні важливо мати чіткі бізнес-процеси, які можна швидко адаптувати до змін навколишньому середовищі.

Аналізуючи випадки з власного досвіду, я можу пригадати значну кількість випадків взаємодії з підприємствами, у яких були порушені процеси комунікації і наскільки це глибоко впливало на їхню успішність та ефективність ведення бізнесу.

Згідно результатів опитування APQC про топ складнощів управління процесами за 2019-2021 роки в п‘ятірку проблем 2019 року входили подолання організаційного опору змінам та залучення керівництва в управлінні процесами. Починаючи з 2020 року, ці складнощі зникли з порядку денного, що свідчить про необхідність налагоджених процесів не тільки для вирізнення компанії серед конкурентів, але і для виживання на ринку, коли відбуваються інтенсивні зміни середовища.

Мною була обрана тема роботи Моделювання та формалізація бізнес процесів управління складними об‘єктами для дослідження можливих методів та моделей для опису бізнес процесів та застосування цих технік на прикладі сфери ресторанного бізнесу, яка на даний час залишається левовою часткою в економіці країни.

Підсумовуючи, я можу виділити головні аспекти актуальності цієї теми дослідження, які існують на сьогодні:

1. Зростання ефективності управління складними об‘єктами.
2. Зменшення витрат підприємств.
3. Оптимізація наявних робочих процесів.
4. Передбачення можливих проблем та ризиків, та завчасне їх усунення.

Об‘єктом дослідження є методи моделювання і прогнозування процесів у ресторанному бізнесі.

Предметом дослідження є моделювання бізнес процесів управління складними об‘єктами.

1.2 Ознайомлення з поняттям бізнес процесу та його основними елементами

В кожній сфері людської діяльності, чи то на підприємстві, чи то внашому організмі, чи то в стосунках між людьми закладені певні процеси. Взагалі, процес - це деяка сукупність та послідовність дій, які відбуваються одна за одною, спрямована на досягнення конкретного результату. Зокрема, в своєму дослідженні я зосереджуватимусь на бізнес процесах. Для цього необхідно

надати основні поняття, типи, концепції та елементи, які визначають структуру і функціонування бізнес процесів.

Можна виділити дві великі групи тлумачень поняття бізнес процесу. Згідно зі стандартом ISO 9000:2000 «бізнес-процес - сукупність взаємопов’язаних дій, що перетворюють входи у виходи». Тобто своєрідна трансформація входів у виходи, яка розділена на декілька видів: фізична трансформація (перетворення сиорвини у виріб), локаційна транасформація (перенесення фабрики з одного міста в інше), транзакційна трансформація (перетворення фінансових ресурсів з одного виду в інший) та інформаційна трансформація (перетворення даних з одного вигляду на інший).

Інше поширене формулювання цього поняття запропонували Майкл Хаммер та Джеймс Чампі, автори концепції реінженірингу бізнес-процесів: бізнес процес - організований комплекс взаємозв'язаних дій, які в сукупності дають цінний для клієнта результат. Вони допомагають кожному співробітнику зрозуміти особливості своїх обов‘язків.

Якщо підсумувати, то бізнес процес - це потужний інструмент для вирішення питань бізнесу. Він може бути як формалізований, так і ні. Наприклад, в компанії на позиції працює менеджер по роботі з клієнтами, якому не було чітко прописано послідовність дій для співпраці з кожним типом клієнта , але усно обговорено цей процес, то в такому випадку цей бізнес процес налаштовується самостійно працівником і його інтерпретація може ширитись між колегами. Зазвичай у випадках такої реалізації і відсутності формалізації виникає суттєва різниця між теорією та практикою ведення діяльності, що в свою чергу погіршує ефективність бізнесу і прийняття рішень.

За структурою виділяють наступні види процесів:

- горизонтальні,
- вертикальні [1].

Перші за своєю суттю спрямовані на задоволення потреб клієнтів.

Вони можуть містити різні етапи, починаючи з замовлення товарів та послуг до їх доставки та підтримки обслуговування після продажу.

Вертикальні процеси відповідають за управлінську діяльність компанії та охоплюють різні її ієрархічні рівні від верхнього рівня - керівництва, до нижнього - виконавців.

Крім цього, існує велика кількість класів, на які поділяються БП в залежності від класифікаційної ознаки [2].

За участю у створенні цінності БП поділяють на:

- основні,
- допоміжні,
- управлінські.

Основні БП безпосередньо супроводжують продукт або послугу на її життєвому циклі, від аналітики зовнішнього середовища до збуту продукції, а також формують додану вартість. Допоміжні процеси підтримують основні і на пряму не мають впливу на формування продукту. Це можуть бути процеси ведення документообігу підприємства, інформаційної та технічної підтримки і тд. Управлінські процеси аналогічно не мають прямого відношення до основної діяльності підприємства, і також забезпечують підтримку основним і сприяють закриттю бізнес потреб зацікавлених груп. Прикладом таких процесів є управління персоналом, ризиками планування ресурсів підприємства.

За рівнем деталізації виділяють:

1. Процеси верхнього рівня (великі процеси) - зазвичай охоплюють увесь процес управління;
2. Підпроцеси - є деталізованими структурами процесів, які містять певні етапи і виконуються в межах процесів верхнього рівня. Їхня основна мета - спростити та деталізувати процеси верхнього рівня

3. Операції - структурні елементи підпроцесів, які найбільш деталізовані. Наприклад операція перевірки наявності товару на складі в межах підпроцесу обробки замовлення клієнтів.

За характером протікання в часі виділяють:

- циклічні БП,
- періодичні БП,
- однократні БП.

В теорії управління бізнес процесами виділяють 5 основних компонент: входи та виходи, елемент потоку (flow unit), мережа дій та буферів (the network of activities and buffers), ресурси та інформаційна структура [3].

Розуміння та подальше моделювання бізнес процесів починається з ідентифікації вхідних та вихідних точок. Варто зазначити, що входи та виходи бувають матеріальні, такі як сировина, фінанси, людський ресурс, та нематеріальні - інформація, енергія та час. Наприклад, салон краси приймає клієнтку без манікюру і випускає її вже з готовим лаковим покриттям.

Елемент потоку є досить гнучкою сутністю. Він може проходити через різні види діяльності та в результаті бути готовим продуктом. Контекст БП виступає квінтесенцією ідентичності елементу потоку: він може виконувати роль вхідних даних, проміжних продуктів та одиниці готової продукції. Також елемент потоку широко відомий під терміном робота.

Мережа дій та буферів - елемент БП, який відображає послідовність дій та буферів, які необхідні для досягнення певної мети. Крім того він виконує декілька важливих функцій, а саме: мінімізацію часу виконання процесу за рахунок можливості зберігання даних в буфері та керування їхнім потоком, покращення керованості процесу, підвищення гнучкості та оптимізацію ресурсів.

Ресурси, як елемент БП, використовуються для здійснення різних процедур та операцій в рамках процесу. На відміну від вхідних даних ресурси

використовуються, а не споживаються і до того ж, вони можуть бути як внутрішніми, так і зовнішніми. Ресурси діляться на 2 категорії: капітальні активи (технічне обладнання, нерухомість і тд.) та праця (працівники підприємства та їхні знання).

Для реалізації БП необхідна інформаційна структура, яка визначає потрібну та доступну інформацію для прийняття рішень. Така інформаційна структура може містити бази даних, звіти, інструкції та інші джерела інформації. Вона також допомагає в забезпеченні інформацією на кожному етапі БП.

Таким чином, ми маємо перед собою повну картину структури БП та його склад (рис. 1.1).

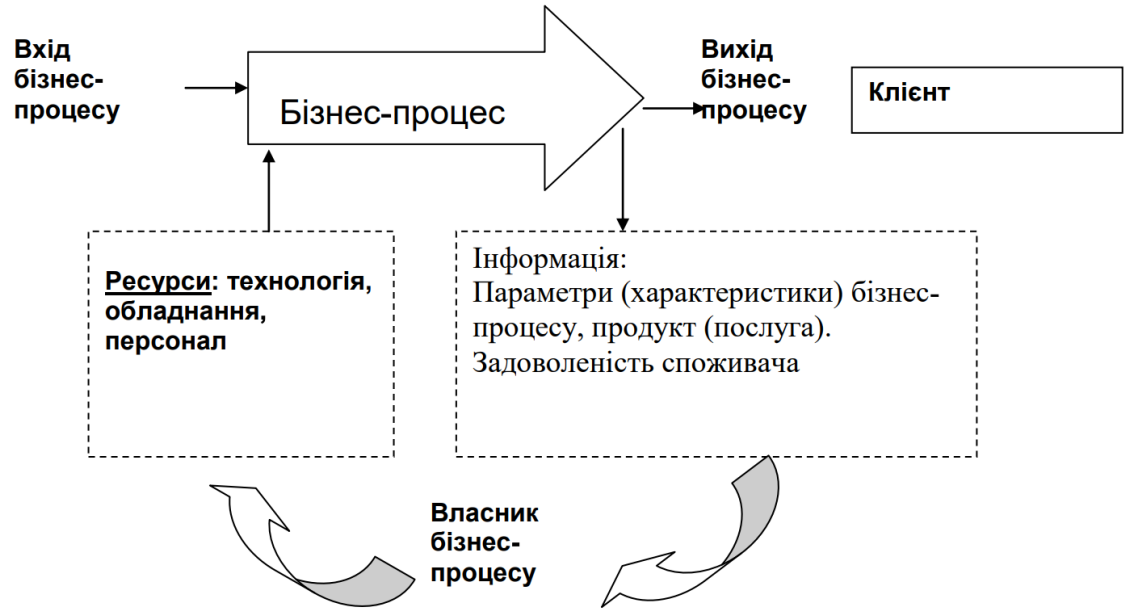


Рис. 1.1 – Елементи бізнес процесу [4]

1.3 Дослідження методів моделювання бізнес процесів

Важливим етапом в управлінні бізнесом або підприємством є моделювання БП, це дозволить глибше дослідити, проаналізувати та оптимізувати поточні

процеси в компанії, що може значно покращити ефективність бізнесу та зменшити витрати.

Побудова моделей БП є складним завданням і є предметом багатьох наукових досліджень та робіт. Для цього існують різноманітні методології, техніки та інструменти: графічні нотації, мережі Петрі, метод Монте Карло, об‘єктно-орієнтована мова моделювання UML та багато інших.

Відомі три основних підходи до моделювання БП.

1. Функціональне моделювання. В такому підході процес визначається через функції, а часова послідовність відсутня. Нам відомі вхідні і вихідні точки, але послідовність операцій та дій є об’єктом для розробки. В результаті такого моделювання ми отримаємо діаграму, яка ілюструє функціональну структуру організації.
2. Процесне моделювання. Цей підхід базується на створенні діаграми БП, що ілюструє послідовність дій та операцій, їхні взаємозв‘язки в рамках процесу.
3. Ментальне моделювання. Ідея в створенні карти довільної форми, яка має на меті структурувати знання фахівця в зрозумілий йому формат.

Повний перелік методів моделювання та їхнє призначення можна побачити на рис. 1.2 [5].

Аналіз застосування методів моделювання бізнес-процесів

Гнучкість моделі (здатність вносити зміни)		Цілі моделювання			
	Активні моделі	Опис та вивчення бізнес-процесів підприємства	Аналіз бізнес-процесів з метою їх оптимізації	Моніторинг та контроль бізнес-процесів	Автоматизація бізнес-процесів
		Інтегрована методологія GRAI (Graph with Results and Activities Interrelated)			
			Workflow		
			UML (Unified Modeling Language) – уніфікована мова моделювання		
			OOT (OO Technique) – об’єктно-орієнтована техніка		
			OMT (Object Modelling Technique) – техніка об’єктного моделювання		
			OOA/OOD (OOAnalysis / OODesign) – об’єктно-орієнтований дизайн та аналіз		
			OOD (Object Oriented Design) – об’єктно-орієнтований дизайн		
		IDEF3		CPN (Coloured Petri nets) – кольорові мережі Петрі	
	Пасивні моделі	Діаграма Ганта	SSM (Soft System Methodology) – методологія м’яких систем		
			IDEF3		
		IDEF0			
		RID (Role Interaction Diagrams) – діаграми взаємодії ролей			
		DFD (Data Flow Diagramming) – схеми інформаційних потоків			
		RAD (Role Activity Diagrams) – діаграми рольових дій			
		Блок-схема		Діаграма Ганта	

Рис. 1.2 – Методи моделювання бізнес-процесів

1.3.1 Графічні методи моделювання

На великих підприємствах без належної формалізації та опису бізнес процесів досить важко досягати значних успіхів. Для моделювання БП зазвичай залучаються бізнес аналітики і менеджери, які будують комплексну систему процесів. Моделювання дозволяє адаптувати БП незалежно від числа працівників на підприємстві, при цьому реалізовується можливість закріплення певних функцій та задач за окремими групами виконавців або працівників. Тому розвиток компанії легко супроводжується адаптивними та гнучкими змінами у бізнес процесах, що дозволить оперативно перерозподіляти функції і завдання підрозділів.

Графічні методи займають фундаментальну позицію в моделюванні БП через багаторічну та перевірену практику їх використання фахівцями. Найвідомішими серед них є діаграми потоків (DFD), нотації UML, BPMN, IDEF та ARIS. Детальніше розглянемо UML та BPMN.

BPMN (Business Process Model and Notation) - графічна нотація, мова моделювання БП. ЇЇ можна віднести до 2-го підходу моделювання БП - процесного моделювання.

Якщо розглядати BPMN в історичному контексті, то перша нотація з‘явилась близько 100 років тому, у вигляді звичайної блок схеми. Але цього виявилось недостатньо, бо необхідно було будувати ще й ієрархію процесу та працювати з подіями. В 2004 році була розроблена BPMN 1.0 як комплексна мова візуалізації бізнес процесів, саме на цей період припадає швидкий розвиток інформаційних технологій та автоматизації БП, яка набирала шалених обертів.

Перші ніж розглядати як використовується BPMN, потрібно ознайомитись з основними елементами нотації.

1. Категорія об‘єктів потоку (flow objects) - містить графічні елементи такі як Подія (Event): початкова, проміжна, завершальна; завдання (Task), підпроцес (Sub-Process), Виклик Дії (Call Activity), шлюз (Gateway): логічні оператори. Вони відтворюють послідовність та структуру БП.
2. Категорія з‘єднувальних об‘єктів (connecting objects) - потік послідовності (Sequence Flow), потік повідомлень (Message Flow), асоціація (Association), Зв‘язок Даних (Data Association). Вони використовуються для встановлення залежностей між компонентами БП.
3. Категорія Зон відповідальності (Swimlanes) - пул (Pool): відображає окремі організації, доріжка (Lane): відображає різні ролі у БП. Вони забезпечують організацію та представлення ролей у БП
4. Категорія артефактів (artifacts) - об‘єкт даних (Data Object), текстова анотація (Text Annotation), група (Group). Вони надають додаткову інформацію стосовно БП.

У математичному сенсі діаграма сценарію - спрямований граф, який складається з елементів нотації BPMN і описується кортежем [6]

$$Gr(Bp_N) = V(Bp_N), E(Gr), L(v), Id(v), Rul^{(Bp)}, p(v)$$

$V(Bp_N) -$ множина вузлів спрямованого графа $Gr(Bp_N)$

$E(Gr) -$ множина ребер

$L(v) -$ множина міток вершин

$Id(v) -$ функція розмітки вершин для надання унікальних імен

$Rul^{(Bp)} -$ множина правил зв‘язування графічних елементів нотації

$p(v) -$ булева функція, що зіставляє значення імовірності ребрам, які поєднують вершини спрямованого графа

Кожен елемент нотації має своє певне графічне зображення.

Подія (Event) відображається у вигляді круга, який може мати певну іконку в середині, що вказуватиме на тип Події (рис. 1.3).

Рис. 1.3 – Основні типи Подій в нотації BPMN

Дія (Activity) позначається у вигляді прямокутника з закругленими кутами, в середині якого може бути вказано ім’я завдання або процесу, а також різні атрибути: пріоритет, термін виконання. Залежно від типу Дії, прямокутник може змінюватись (рис. 1.4).

Рис. 1.4 – Основні типи Дій в нотації BPMN

Шлюз (Gateway) відображається у вигляді ромбу (рис. 1.5).

Рис. 1.5 – Типи Шлюзів в нотації BPMN

Потоки (Flow): Потік Послідовності (Sequence Flow), Потік Повідомлень (Message Flow),Асоціації (Association), Асоціація даних (Data Association) позначаються у вигляді різного типу направлених стрілок в залежності від типу потоку (рис. 1.6). Кожен з них має свої підвиди аналогічно до попередніх пунктів.

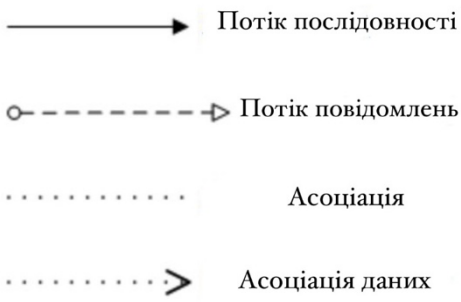


Рис. 1.6 – Основні типи Потоків в нотації BPMN

Категорія Зон відповідальності (Swimlanes) має наступне відображення (рис. 1.7).

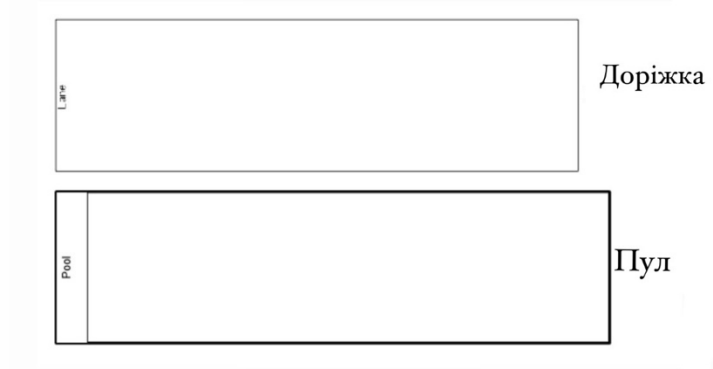


Рис. 1.7 – Типи Зон відповідальності в нотації BPMN

Позначення Артефактів та Даних зображені на рис. 1.8 та 1.9 відповідно.

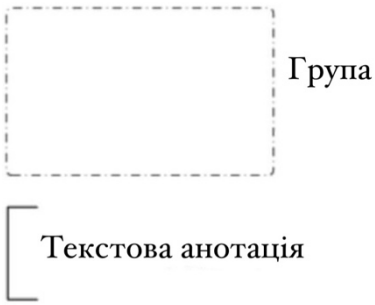


Рис. 1.8 – Типи Артефактів в нотації BPMN

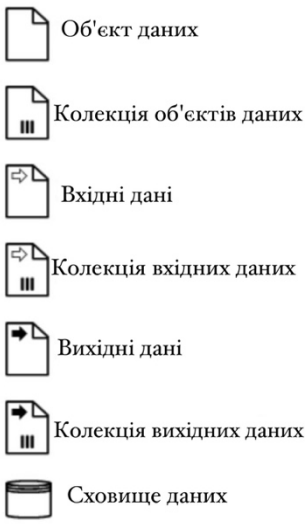


Рис. 1.9 – Типи Даних в нотації BPMN

Розглянемо один найпростіший приклад діаграми (рис. 1.10) [7]:

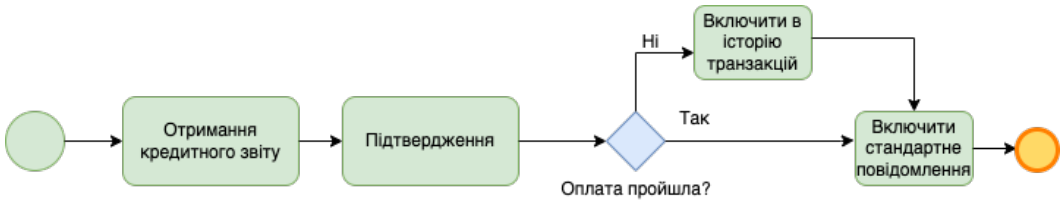


Рис. 1.10 – Діаграма бізнес-процесу в нотації BPMN

Читаючи зліва направо, бачимо кружок, який відповідає за початкову подію. Далі ідуть два завданні, з‘єднані елементом Потоку Послідовності. Перше завдання - отримання кредитного звіту, друге - підтвердження. Шлюз, зображений у вигляді ромбу перевіряє чи пройшла оплата. Якщо так - відправляємо стандартне повідомлення, якщо ні - включаємо в історію транзакцій та відправляємо стандартне повідомлення. Після цього натикаємось на завершальну подію, яка свідчить про завершення змодельованого процесу. Цей БП може описувати процес перевірки кредитної історії клієнта перед наданням йому кредиту.

UML (Unified Modeling Language) - уніфікована, неформалізована мова моделювання, яка використовується для розробки систем і бізнес процесів. Її можна віднести до процесного моделювання так як її інструментарій дає змогу розробляти діаграми, які ілюструють послідовність дій та операцій.

Створення UML групою дослідників на чолі з Граді Бучем, Іваром Якобсоном та Джеймсом Рамбо датується 1994 роком. Кожен з них розробив власний варіант мови моделювання. Граді Буч створим мову моделювання об‘єктів (Object Modeling Technique) OMT. Івар Якобс - мову моделювання класів (Object-Oriented Software Engineering) OOSE, а Джеймс Рамбо - метод моделювання IEEE 1074-1991. У 1995 ця група об‘єдналась, щоб створити єдину стандартну мову моделювання. Так і з‘явився UML.

UML має широке застовування.

1. Комунікація з клієнтом. Він може не розуміти всіх технічних тонкостей та термінології, але дана мова моделювання зможе йому допомогти зрозуміти процес за допомогою інтуїтивно зрозумілих зображень.
2. Тестування сценаріїв. UML надає можливість проводити тестування можливих сценаріїв та окремих функцій всередині БП.
3. Проєктування. UML діаграми придатні для моделювання архітектури як малих так і великих проєктів, на базі якої потім пишеться код.
4. Оцінка та аналіз програмних проєктів. UML моделювання використовується для оцінки та аналізу певних недоліків проєкту та можливості їх виправлення, а також для розуміння, яку частину проєкту доцільно вдосконалити.
5. Реверс - інжиніринг. UML моделі можуть бути створені на базі вже існуючого коду програми.
6. Розробка документації. UML дає змогу створення документації, що описує функції системи та їхню взаємодію.

В контексті UML існує велика кількість типів діаграм. У версії 2.5 існує 14 типів діаграм: діаграми що представляють статичну структуру додатку, поведінкові аспекти системи, фізичні аспекти функціонування системи. [8] Варто зазначити, що не всі діаграми є однаково корисними для використання, тому необхідно залучати певні діаграми, опираючись на контекст проєкту. Найбільш вживаними є:

- діаграма прецедентів (Use-case diagram),
- діаграма класів (Class diagram),
- діаграма активностей (Activity diagram),
- діаграма послідовності (Sequence diagram),
- діаграма розгортання (Deployment diagram),
- діаграма співробітництва (Collaboration diagram),
- діаграма об‘єктів (Object diagram),
- діаграма станів (Statechart diagram).

Розглянемо детальніше декотрі з них.

Діаграма прецедентів описує функціональну взаємодію між системою та її користувачами. Вона складається з наступних елементів: актор (учасник) - зовнішня сутність, що взаємодіє із системою та виконує певні дії; прецедент (Use Case) - функція, яку виконує система для користувачів; система - об‘єкт, що виконує прецеденти для задоволення потреб користувачів. Прецеденти зображуються у вигляді овалів, а актори позначаються людськими фігурами.

Рис. 1.11 – Приклад діаграми прецедентів

На рис. 1.11 зображений простий приклад діаграми, де у якості системи виступає ресторан, актори: клієнт, офіціант, менеджер та кухар, прецеденти: оформлення замовлення, приготування страви, обслуговування клієнта, управління замовленнями. Побудована діаграма за допомогою сервісу diagrams.net

Діаграма класів є однією з основних діаграм UML. Вона використовується для візуалізації класів, інтерфейсів, спадковості та асоціацій між класами. Класи виступають в ролі будівельних блоків систем і є

шаблонами для створення об‘єктів. Як зазвичай, клас містить атрибути (поля) та методи (операції) для роботи з атрибутами. Інтерфейс - спосіб взаємодії з класом через його методи. Спадковість визначає відношення у разі коли один клас успадковує атрибути та методи іншого класу. Асоціація визначає відношення у разі коли один клас використовує атрибути або методи іншого класу.

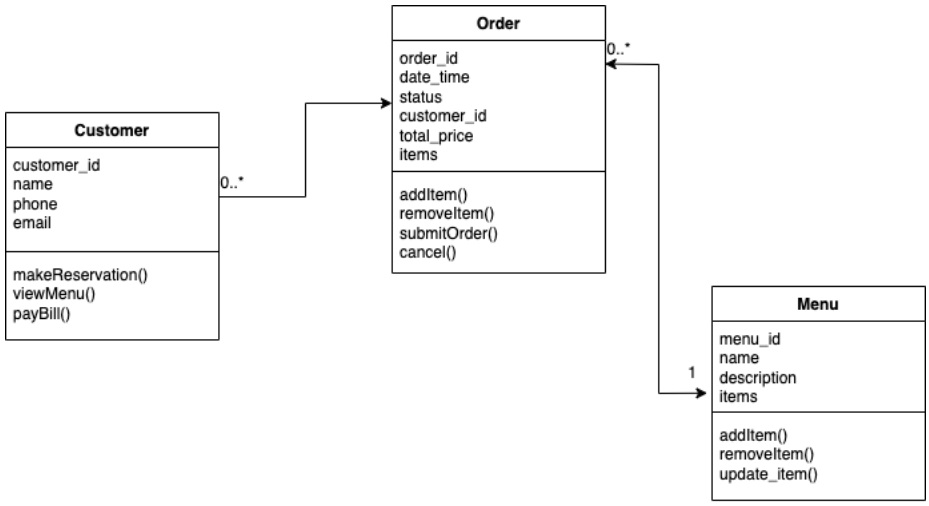


Рис. 1.12 – Приклад діаграми класів

На рис. 1.12 зображена проста діаграма класів. Кожен з класів Клієнта, Замовлення та Меню має свої атрибути та методи та певні зв‘язки. Наприклад зв‘язок від Клієнта до Замовлення з позначкою «0..*» означає, що клієнт може мати 0 або більше Замовлень.

Діаграма розгортання використовується в UML для побудови та моделювання фізичної архітектурної системи та інфраструктури. Така діаграма складається з вузлів (Nodes), компонентів, інтерфейсів та зв‘язків між компонентами та вузлами. Компонентів у кожному вузлі може бути декілька та вони взаємодіють між собою за допомогою різних інтерфейсів.

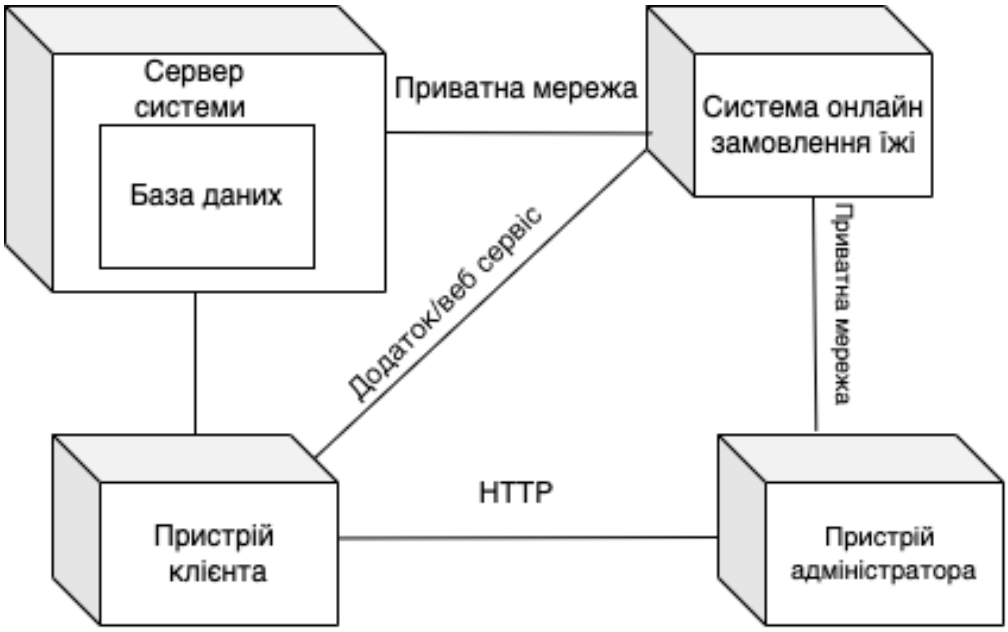


Рис. 1.13 – Приклад діаграми розгортання

На рис. 1.13 зображена проста діаграма розгортання для додатку/веб сервісу онлайн замовлень їжі. Побудована діаграма за допомогою сервісу diagrams.net.

Таким чином було розглянуто декілька основних типів UML діаграм, які не є універсальними, а застосовується відповідно до контексту задачі або проєкту.

Як і у кожного методу, у UML є свої переваги та недоліки. Серед недоліків можна виділити наступні:

1) велика кількість діаграм, які дуже рідко використовуються;
2) відсутність стандартів використання - кожен фахівець реалізує діаграму по своєму, що може призвести до нерозуміння її іншими фахівцями;
3) надмірна деталізація, що може завадити розуміти повну картину проєкту.

Якщо говорити про переваги, на думку спадають:

1) UML ілюструє об‘єктно-орієнтований підхід, тому використання цієї мови моделювання з легкістю дозволить переходити від проєктування до реалізації програмного забезпечення;

2) UML може використовуватись для моделювання різних систем у різних галузях;
3) UML діаграми можуть слугувати навчальним матеріалом, завдяки простому та інтуїтивному сприйняттю, для ознайомлення працівників з внутрішніми процесами та структурою підприємства.
4) можливість описувати систему та її аспекти з будь-якого ракурсу.

В цілому, мова UML є потужним та широко розповсюдженим інструментом для моделювання БП.

1.3.2 Метод Монте Карло

Цей метод дістав назву від однойменного міста в князівстві Монако. Монте Карло широко славиться своїми азартними іграми, а саме казино, в той час як метод Монте Карло має дещо спільне з гральною рулеткою, основним елементом якої є кулька. Але математики дуже вдосконалили її - тепер це не просто кулька, що обертається, а комп‘ютерна програма, яка називається генератором випадкових чисел. Тим не менш за допомогою методу Монте Карло не можна виграти в рулетку☺

Історія виникнення цього методу сягає в часи другої світової війни, а саме в 1940 роки, коли американські вчені розробляли перший ядерний реактор. Питання, яке постало перед ними - як обчислити поведінку нейтронів у реакторі? До речі, причетним до виникнення цього методу є польський математик та фізик, який народився у Львові у 1909, - Станіслав Улам. А також Джон фон Нейман.

При цьому теоретичне підґрунтя методу Монте Карло було відоме ще раніше. Однак відсутність на той час електронних обчислювальних машин загальмовувала популяризацію цього методу.

Переходячи до суті методу, варто зазначити, що він полягає в описі певного процесу не на базі аналітичного апарату, а за допомогою випадковості. Точніше, випадковим чином генеруються набори даних, що застосовуються для моделювання різноманітних сценаріїв. Також він базується на теорії імовірностей та статистиці: використовуються випадкові числа з відповідним розподілом, з метою оцінити або наблизити значення математичних функцій, оптимізаційних задач і тд. Таким чином, методом Монте Карло можна назвати будь-який математичний метод, який в своїй основі використовує генератор випадкових чисел.

Базовий принцип [9]

Принцип базується на оцінці середніх значень, математичних сподівань, і деяких компонентів системи.

При застосуванні методу Монте Карло до детермінованих задач, для початку необхідно побудувати імовірнісний простір *(Ω, σ, p, X)*,
поняття сформоване Колмагоровим в 30-х роках 20-го століття.

Ω - простір елементарних подій *ω*,

σ -сигма алгебра, сімейство підмножин Ω.,

p - функція імовірності, яка призначає імовірність в інтервалі [0:1] кожній підмножині Ω,

X - випадкова величина, яка залежить від результату експерименту.

Таким чином математичне сподівання E(X) цієї системи буде збігатися з шуканим розв‘язком I нашої детермінованої задачі.

Математичне сподівання та дисперсія випадкової величини X знаходяться як перший та другий центральні моменти:

$$E(X) \coloneqq \int_{\Omega} dp\, X \qquad\qquad \sigma^2 \coloneqq \int_{\Omega} dp \left(X - E(X)\right)^2$$

Якщо ж задача не детермінована, визначена в імовірнісних термінах, то це можна взяти за основу для «аналогового» моделювання та оцінки. Тим не менш, I - одна числова величина, про яку можна думати як про інтеграл.

Стохастичне наближення до I отримується шляхом створення незалежної послідовності випадкових подій $\omega = [\omega_i | i = 1, \dots N]$

$$E(X_N) = I_N(\omega) = \frac{1}{N}\sum_{i=1}^{N} X(\omega_i)$$

I_N - середнє арифметичне багатьох (N) оцінок $X(\omega_i)$ повторюваного експерименту.

Стає інтуїтивно зрозуміло, що I_N сходитиметься, в деякому сенсі, до *E(X),* тобто і до *I*, так як число зразків *N* збільшується.

Зокрема, закони великих чисел та центральні граничні теореми (ЦГТ) гарантують точність та збіжність методу Монте Карло. ЦГТ підтверджує, що розподіл випадкової величини $I_N(\omega)$, для досить великих *N,* сходиться до Гаусівського розподілу, з математичним сподіванням $I = E(X)$ та дисперсією $\sigma^2(I_N) = \frac{\sigma^2(X)}{N}$. Тому застосовуються типові результати аналізу статистичної помилки за законами Гаусівського розподілу.

В застосуванні методу Монте Карло поширеною практикою є наводити результати як:

$I \approx I_N \pm \sigma(I_N)$, або $I \approx I_N \pm 2 * \sigma(I_N)$

Зазвичай, на практиці дисперсію $\sigma^2(X)$ більш важко обчислити ніж математичне сподівання $E(X)$. Тому її замінюють на емпіричну дисперсію:

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}[X(\omega_i) - E(X_N)]^2$$

$$= \frac{1}{N-1}\left(\sum_{i=1}^{N} x^2(\omega_i)\right) - \frac{1}{N}\left[\sum_{i=1}^{N} x(\omega_i)\right]^2)$$

За умови великого розміру вибірки *N,* емпірична дисперсія є досить точною оцінкою теоретичної дисперсії $\sigma^2(X)$:

$s^2 \to \sigma^2,\ \sigma^2(I_N) \approx s_N^2 = \frac{1}{N}\ s^2$

Якщо вибірка мала: $N < 100$, для аналізу помилок слід застосовувати t-розподіл Стьюдента.

Для генерації вибірки використовують псевдовипадкові генератори (ПВГ) чисел. Вони є базовим елементом у методі Монте Карло. ПВГ - алгоритм, який генерує «випадкову» числову послідовність, яка насправді є детермінованою. Популярним методом генерації ПВГ є лінійний конгруентний метод з використанням лінійної рекурсії.

Спектр та сфера застосування методу Монте Карло є досить широкими:

1) фінанси та оцінка фінансових інструментів;
2) наука про матеріали, фізика: моделювання поведінки мікрочастинок, рідин та газів;
3) біологія та медицина: моделювання біологічних систем, таких як молекули ДНК, білки і тд;
4) візуалізація та комп‘ютерна графіка: генерація зображень та відео;
5) бізнес-процеси: прийняття рішень в умовах невизначеності, прогнозування фінансових показників.

Якщо детальніше розглядати використання цього методу в БП, то він може бути корисний для оцінок ризиків, наприклад в інвестиційних проєктах або в страхуванні. Він дозволить змоделювати можливі сценарії та ризики для прийняття рішень.

Також доречне застосування метод Монте Карло може мати в прогнозування фінансових показників, таких як прибуток, витрати, продажі і тд, в моделюванні різноманітних процесів, наприклад, транспортних мереж, виробничих ліній.

Безперечно, метод Монте Карло має ряд переваг, таких як висока гнучкість в використанні, висока точність, можливість застосування для

складних систем. Але тим не менш він потребує великих обчислювальних ресурсів, може мати низьку швидкість виконання та вимагати великої кількості ітерацій.

1.3.3 Метод функціонального моделювання SADT

SADT (Structures Analysis and Design Technique) - методологія проєктування призначеня для аналізу та моделювання складних систем. Цей метод виник в 60 - 70х роках XX століття в рамках проєкту NASA в США. Засновником цього методу вважається Дуглас Росс, який очолював групу науковців Массачусетського технічного інституту. SADT вперше вийшла на ринок в 1975 році, після чого, до 1981 року нею вже користувалося більше 50 компаній, які займалися різними за своєю структурою галузями: від телефонних мереж до обліку матеріально-технічних ресурсів.

SADT - це комплекс методів, правил та процедур, на основі яких здійснюється моделювання складних ієрархічних систем. Ця методологія побудована на описі систем, розгляді її функцій або об‘єктів. Також варто зазначити, що існують два типи моделей в SADT: функціональне моделювання - моделі, що опираються на опис функцій системи, та моделі даних або інформаційні моделі - націлені на об‘єкти, що існують у системі та їхню взаємодію.

Функціональні моделі слугують для аналізу потреб користувачів, розробки вимог до системи та проєктування її архітектури. Моделі даних мають застосування в описі даних та розробки баз даних.

Методологія SADT є базовою в комплексі методологій IDEF (Integrated DEFinition).

В результаті використання методу SADT отримується модель, яка містить діаграми та елементи текстів. Діаграми, які є головними компонентами моделі, будується за допомогою SA-блоків - графічних елементів, які описують певну функцію, процес або підсистему в системі, та дуг. Місце з‘єднання дуги з блоком визначає тип інтерфейсу (тип взаємодії між функціями або підсистемами) [10]. Тип інтерфейсу може бути реалізований через передачу даних, передачу сигналу і тд. Блоки розташовуються в ієрархічній структурі та можуть взаємодіяти між собою за допомогою потоків даних.

Варто зазначити, що головною особливістю SADT є послідовне впровадження все глибших рівнів деталізації при побудові діаграм. Дана методологія використовує декомпозицію для подання системи у вигляді діаграм, що мають ієрархічну структуру. Кожен рівень ієрархії описує все більш деталізовані процеси та функції системи. Водночас зв‘язки між рівнями зберігаються, що дозволяє відслідковувати взаємодію між усіма елементами системи.

Тож першим етапом побудови SADT моделі є зображення системи у вигляді простої сутності - блоку та дуг. Зазвичай цей блок розміщується в верхній частині моделі. На блоці вказується загальне ім‘я, так як цей єдиний блок описує цілу систему. В основному діаграма містить 3-6 блоків, щоб переходи на більш деталізовані рівні відбувались поступово.

Наведемо приклад опису процесу замовлення їжі, що зображений на рис. 1.14, на найпростішому рівні ієрархії:

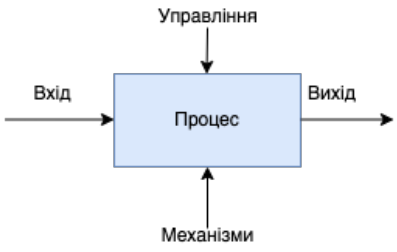


Рис. 1.14 – Приклад процесу у вигляді простої сутності

Вхід: замовлення, меню;

Вихід: готова їжа, рахунок, звіт про продажі;

Управління: правила приготування їжі, правила обліку продажів;

Механізми: кухня, обладнання, працівники;

Після побудови основного блоку використовується декомпозиція блоку на менші компоненти. Таким чином модель може бути деталізована до необхідного рівня. Наступним етапом є ітеративне рецензування моделі з метою її вдосконалення та оптимізації. Це процес виявлення помилок в першій версії моделі, який може повторюватися декілька разів перш ніж буде досягнуто оптимального результату.

Також варто зазначити, що на базі SADT засновані інші методології сімейства IDEF: IDEF0, IDEF1, IDEF2.

Як і кожен інструмент моделювання, метод SADT має свої переваги та недоліки. Серед переваг можна виділити наступні:

1. Структурований підхід моделювання, що добре ілюструє взаємодію між процесами та їхніми компонентами;
2. Ієрархічність дозволяє розглядати процеси на різних рівнях деталізації;
3. Зручна візуалізація;

Недоліки включають наступні пункти:

1. Обмежена використовуваність: важко зобразити складні процеси, які мають велику кількість залежностей та взаємодій між компонентами;
2. Обмежена наочність: хоча цей метод має під собою графічне представлення, деякі складні системи можуть бути складними для візуалізації;

1.4 Аналіз типових бізнес процесів у сфері ресторанного бізнесу

Одним із основних факторів розвитку сучасного суспільства є виробництво та споживання усіляких благ, таких як матеріальні, соціальні і тд., які задовольняють потреби суспільства.

Ресторанний бізнес є великим організмом та структурою, яка має важливі взаємопов‘язані елементи. Вони охоплюють широке коло аспектів, до яких входять приготування страв, психологія та культура споживання і обслуговування, управління продажами, а також цінова і рекламна політика.

З метою підтримання конкурентоспроможності та стійкості ресторанного бізнесу, необхідно щоб заклад мав формалізовані та налагоджені основні бізнес-процеси, включаючи приготування страв, обслуговування гостей тощо.

В цій сфері існує ряд типових БП, які мають великий вплив на розвиток, ефективність та успішність бізнесу. Ключовими з них є наступні:

- процес приготування страв,
- процес обслуговування гостей,
- управління продажами,
- управління запасами,
- управління персоналом.

Розглянемо детальніше процес управління продажами. Він включає в себе розробку стратегій продажу, ціноутворення, маркетингову політику та просування продукції. Вагомим фактором, який дозволяє спростити процес управління продажами є прогнозування кількості клієнтів. Обсяг та потік клієнтів впливають на швидкість обробки замовлень, запаси продуктів, залучення персоналу. На підставі прогнозу можна планувати кількість необхідного персоналу, обсяги закупівель продуктів та розподіл робочого часу і ресурсів.

На мою думку, процес управління продажами доречно моделювати за допомогою методу функціонального моделювання SADT. Бо саме цей метод базується на розгляді функцій процесу та взаємозв‘язків між ними.

Якщо деталізовувати процес управління запасами, то можна виділити наступні функції: збір та аналіз замовлень, управління запасами, ціноутворення та промоції, управління зв‘язками з клієнтами, прогнозування попиту.

Процес обслуговування гостей є досить складним для аналізу, так як потрібно враховувати багато факторів. Прибуття клієнтів до закладу може відбуватися самостійно або за допомогою резервації столиків. При прийнятті замовлення важливими аспектами є швидкість та точність обробки замовлень. Поки гості чекають свої страви, персонал повинен забезпечувати подачу посуду, напоїв, інших додаткових предметів. Важливо забезпечити якісне обслуговування та відповідати вимогам клієнтів. Процес розрахунку повинен бути швидким та зручним, охоплювати різні види оплати. Також варто пам‘ятати про збір відгуків та формування рейтингу від клієнтів для поліпшення якості обслуговування.

Для моделювання процесу обслуговування можна застосувати метод Монте-Карло через його здатність моделювання імовірнісних сценаріїв та оцінки ризиків. А також цей метод дозволить врахувати випадкові фактори, наприклад, час очікування, вибір страв, тривалість прийому їжі.

Першим етапом моделювання є визначення вхідних параметрів, які впливають на процес обслуговування. Такими параметрами можуть бути: середній час обслуговування клієнта, розподіл часу між прибуттям клієнтів, кількість доступних офіціантів.

Далі відбувається генерація випадкових чисел, для кожного вхідного параметра в межах встановлених імовірнісних розподілів. Після чого відбувається симуляція шляхом повторення процесу з випадковими

значеннями для вхідних параметрів. Наступним кроком збираються отримані дані та аналізуються для подальшої оптимізації та прийняття рішень.

1.5 Висновки до розділу 1

У даному розділі були розглянуті основні визначення понять, що стосуються бізнес-процесів, їхні основні елементи та структура. Також було проведено поверхневий огляд існуючих методів моделювання бізнес-процесів, які включають в себе графічні методи, метод Монте-Карло та метод функціонального моделювання SADT.

Окремим пунктом розділу був виділений аналіз типових бізнес-процесів у сфері ресторанного бізнесу, де було розглянуто процес управління продажами та процес обслуговування відвідувачів.

2 ВИБІР І ОПИС МЕТОДІВ МОДЕЛЮВАННЯ ТА ПРОГНОЗУВННЯ ПРОЦЕСІВ У СФЕРІ РЕСТОРАННОГО БІЗНЕСУ

З розширенням сфери використання data science та машинного навчання, відкриваються великі перспективи і для бізнесу. Багато галузей використовують ці технології для покращення ефективності та досягнення конкурентних переваг на ринку.

У цьому розділі будуть розглянуті різні підходи та моделі для подальшого їх застосування в прогнозуванні та моделюванні бізнес процесів.

2.1Лінійна регресія

Регресійний аналіз - метод математичної статистики, який встановлює взаємозв‘язок між залежною змінною (відкликом) та однією або декількома незалежними змінними (предикатами). Регресійний аналіз виконується для того, аби визначити кореляції між двома або більше змінними, що мають причинно-наслідкові зв‘язки, та виконати прогнози на основі цього зв‘язку [11].

Лінійний регресійний аналіз встановлює лінійну залежність та може поділятись на 2 основні типи: просту лінійну регресію та множинну лінійну регресію.

Проста лінійна регресія виражається формулою:

$$y = a_0 + a_1 x + e,$$

де у -залежна змінна, х - незалежна. a_0 − є точкою перетину лінії регресії та вертикальною віссю координат. a_1 − нахил лінії регресії, e − випадкова помилка, яка виражає вплив випадкових факторів на залежну змінну.

Множинна лінійна регресія має наступне математичне формулювання:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$$

Як і в попередньому випадку, y −залежна змінна, x −незалежна змінна, β −параметр, ε − похибка.

Для нашого випадку (прогнозування кількості клієнтів) можна побудувати наступну модель:

$$p = \nu_0 + \nu_1 x_1 + \nu_2 x_2 + \cdots + \nu_n x_n$$

p − прогнозована кількість клієнтів;

$x_0, x_1, \ldots x_n$ − характеристики клієнта;

$\nu_0, \nu_1, \ldots \nu_n$ − вагові коефіцієнти, що визначають вплив відповідних характеристик на кількість клієнтів.

2.2 Випадковий ліс (Random Forest)

Випадковий ліс - це класифікатор, що складається з набору дерев-класифікаторів $\{h(x, \theta_k), k = 1, \ldots\}$, де θ_k незалежні однаково розподілені випадкові вектори, і кожне дерево випадкового лісу після обробки вхідних даних х видає свою власну прогнозовану мітку класу. Фінальний результат - клас, який набирає найбільшу кількість голосів серед усіх дерев [12].

Кожне дерево у випадковому лісі навчається незалежно одне від одного на випадковій підвибірці вхідних даних.

2.3 Авторегресійна модель з інтегрованими ковзними середніми ARIMA

ARIMA (Autoregressive integrated moving average) - статистична модель для прогнозування часових рядів. Вона включає в себе 3 основні компоненти: авторегресійну складову (AR), складову інтегрованого ряду (I) та ковзне середнє (MA).

Ця модель позначається як ARIMA (p, d ,q):

- p - кількість членів авторегресії;
- d - кількість різниць;
- q - кількість ковзних середніх.

Параметр p визначає кількість попередніх значень залежної змінної, які враховуються в авторегресійній складовій моделі. Параметр d визначає кількість диференціювань, що застосовуються до початкового часового ряду.

Це допомагає усунути тренд або сезонність, для зменшення залежності від часу. Параметр q визначає кількість ковзних середніх, які згладжують випадкові шуми у ряді.

Авторегресійна складова [13]

В авторегресійних моделях припускається, що лінійна функція Y_t попередніх значень задається рівнянням:

$$Y_t = \alpha_1 Y_{t-1} + \varepsilon_t$$

Кожне спостереження складається з випадкової складової ε_t та лінійної комбінації коефіцієнта саморегресії і попереднього спостереження.

Інтегрований процес

До класу інтегрованих процесів належать часові ряди, які визначаються кумулятивним ефектом деякої діяльності. Різниці між послідовними

спостереженнями можуть бути незначними, навіть якщо поведінка ряду є непередбачуваною. Іншими словами, хоч ряд може бути мінливим, проте середнє значення різниць залишається стабільним протягом часу. Наступне рівняння визначає інтегрований процес:

$$Y_t = Y_{t-1} + \varepsilon_t ,$$

де ε_t - білий шум. Диференціювання першого порядку передбачає, що різниця між двома послідовними значеннями *Y* є постійною.

Ковзне середнє

Поточне значення процесу ковзного середнього є лінійною комбінацією поточного збурення та одного або декількох попередніх збурень. Порядок ковзного середнього вказує на кількість попередніх періодів, врахованих у поточному значенні.

Модель ковзного середнього першого порядку, яка позначається як MA(1), виглядає наступним чином:

$$x_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

$x_t -$ значення в поточний момент часу;

$\mu -$ середнє значення процесу;

$\varepsilon_t -$ шум в поточний момент часу;

$\theta -$ параметр, що визначає вагу попереднього випадкового шуму на поточне значення.

Модель ковзного середнього q-го порядку, аналогічно, має вигляд:

$$x_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

2.4 LSTM

LSTM (Long Short-Term Memory) - відноситься до рекурентного типу нейронних мереж RNN (Recurrent Neural Networks). RNN - клас нейронних

мереж, який призначений для роботи з послідовними даними. Вхідні дані мають залежність від попередніх часових кроків.

RNN мережі привабливі тим, що в теорії вони здатні пов‘язувти попереді дані з поточними для розв‘язання певної задачі. Але це працює лише у випадках, коли розрив між релевантною інформацією та моментом, де її доречно застосувати, невеликий. В такому разі RNN мережі для навчання можуть використовувати попередню інформацію.

LSTM мережі, запропоновані Хохрейтером та Шмідхубером в 1997 році, створені для уникнення проблеми довгострокової залежності. Вони здатні запам‘ятовувати інформацію протягом тривалого часу.

LSTM мережі мають наступну структуру [14], що зображена на рис. 2.1:

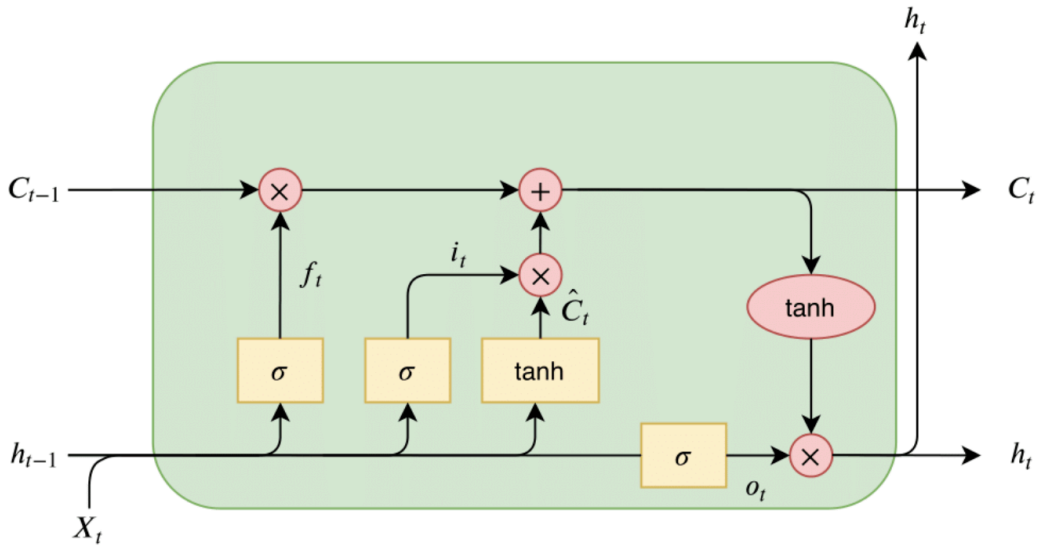


Рис. 2.1 – Структура комірки LSTM мережі

X_t – вхідний часовий крок;

h_t – вихід;

C_t – стан комірки;

f_t – забутий гейт;

i_t – вхідний гейт;

o_t – вихідний гейт;

$\hat{C}_t$ – внутрішній стан комірки.

Операції позначені світло-червоним кольором.

Якщо детальніше розглядати основні елементи структури LSTM мереж, то вони включають гейти (ворота): гейт забуття (forget gate), гейт оновлення (update gate) та вихідний гейт/ гейт виходу (output gate). Ці елементи регулюють потік інформації в мережі.

Гейти забуття визначають, яка інформація повинна бути забута або викинута з внутрішнього стану комірки. Гейти оновлення визначають, яка нова інформація повинна бути збережена в поточному стані пам‘яті. Вихідні гейти визначають, яку інформацію потрібно вивести з поточного стану пам‘яті. Обчислення гейтів та нового значення кандидата в LSTM мережі в момент часу t:

$$f_t = \sigma(W_f * [h_{t-1}, X_t] + b_f)$$
$$i_t = \sigma(W_i * [h_{t-1}, X_t] + b_i)$$
$$o_t = \sigma(W_o * [h_{t-1}, X_t] + b_o)$$
$$\hat{C}_t = tanh(W_C * [h_{t-1}, X_t] + b_C)$$

В цих формулах $[h_{t-1}, X_t]$ представляє конкатенацію попереднього прихованого стану h_{t-1} та поточного вхідного сигналу X_t.

$W_C,\ W_i, W_o, W_C -$ вагові матриці;

$b_f, b_i, b_o, b_C -$ зсуви;

$\sigma -$ сигмоїдна функція активації.

Обчислення оновленого внутрішнього стану комірки:

$$C_t = \ i_t * \hat{C}_t + f_t * C_{t-1}$$

Обчислення виходу:

$$h_t = o_t * \tanh(C_t)$$

Поєднання комірок LSTM мережі наведено на рис. 2.2.

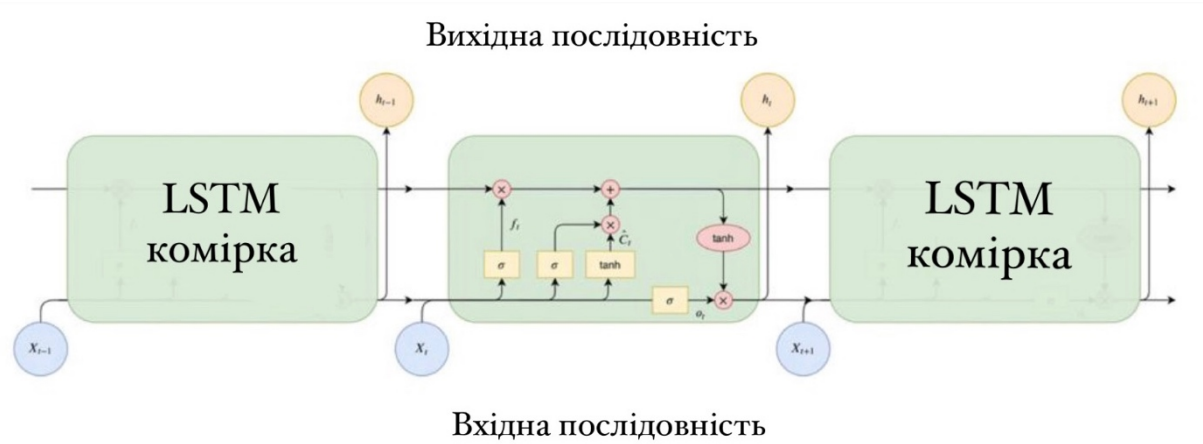


Рис. 2.2 – Поєднання комірок LSTM мережі

2.5 Методи оцінки якості моделі

Під час побудови моделей необхідні інструменти для оцінки їхньої ефективності з метою вдосконалення подальших результатів. Оцінка якості моделей є невід‘ємним процесом в машинному навчанні та дає нам об‘єктивну міру того, наскільки добре модель працює на нових даних. Використання таких метрик дозволить проводити порівняльний аналіз різних моделей та визначати їхні сильні і слабкі сторони.

В цьому розділі будуть розглянуті кілька ключових методів оцінки якості моделі, які широко застосовуються на практиці.

2.5.1 R-squared (коефіцієнт детермінації)

R-squared (R-квадрат) - метрика для оцінки якості моделі в машинному навчанні та статистиці, зазвичай в регресійному аналізі. Ця міра застосовується лише для простої лінійної регресії, у випадку ж множинної регресії - коефіцієнт детермінації необхідно коригувати.

Значення R-квадрату варіюється від 0 до 1. Значення 1 вказує на ідеальну відповідність моделі даним, коли прогнозоване значення не відрізняється від фактичного. Значення 0 вказує на повну відсутність залежності між відкликом та предикатом.

Формула для обчислення R-квадрату виглядає наступним чином:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

SS_{res} – сума квадратів залишків (різниця між спостережуваними значеннями та прогнозованими значеннями моделі)

SS_{tot} – загальна сума квадратів відхилень спостережуваних значень від їхнього середнього.

Якщо до моделі додавати нові змінні, без врахування їхньої значущості, то значення коефіцієнту детермінації збільшується або залишається на одному рівні. Пояснити це можна тим, що загальна сума квадратів SS_{tot} завжди залишається постійною і регресійна модель намагається зменшити значення суми квадратів залишків SS_{res}, знаходячи деяку кореляцію з новим атрибутом.

Таким чином, оцінювання моделі лише метрикою R-квадрат може бути оманливим, оскільки вона не відображає справжню значущість змінних у моделі. Краще також враховувати інші показники, такі як скоригований R-квадрат.

Скоригований R-квадрат - модифікація R-квадрату, яка враховує кількість незалежних змінних. Він дає більш об‘єктивну оцінку якості моделі, коли в неї включено багато змінних.

Формула для обчислення скоригованого R-квадрату:

$$AdjustedR^2 = 1 - \frac{(1-R^2)(n-1)}{n-k-1},$$

$n -$ загальна кількість спостережень;

$k -$кількість незалежних змінних.

2.5.2 MSE

Mean Squared Error (MSE) - одна з відомих метрик для оцінки якості моделі у прогнозуванні. Вона вимірює середнє квадратичне відхилення між прогнозованими значеннями та фактичними значеннями вибірки. Значення цієї метрики ніколи не буває негативним, адже помилка зводиться у квадрат.

Формула для обчислення MSE:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2,$$

$N -$кількість спостережень;

$y_i -$фактичне значення;

$\hat{y}_i -$ прогнозоване значення.

Чим більше значення MSE, тим більша помилка моделі. Також варто зазначити, що дана метрика залежить і від масштабу даних. Тож якщо змінні мають різну шкалу, то це може значно вплинути на MSE. В такому разі перед застосуванням моделі необхідно проводити нормалізацію або стандартизацію змінних.

2.5.3 MAE

Mean Absolute Error (MAE) - ще одна метрика для оцінки якості моделі. Формула для обчислення наступна:

$$MAE = \frac{1}{n}\sum_{j=1}^{n} |y_i - \hat{y}_i|,$$

n −кількість спостережень;

y_i −фактичне значення;

$\hat{y}_i$ − прогнозоване значення.

MAE широко використовувана метрика через легкість інтерпретації результату. Значення MAE вимірюється в тих же одиницях, що і вихідні дані. Основна перевага цієх метрики в тому, що вона не підноситься до квадрату, що дозволяє уникнути впливу великих відхилень на загальну оцінку ефективності моделі.

2.6 Висновки до розділу 2

У даному розділі було розглянуто методи машинного навчання, які можуть бути застосовані для моделювання бізнес-процесів, зокрема у сфері ресторанного бізнесу. Серед цих методів були представлені: лінійна регресія, Random Forest, ARIMA та LSTM мережі. В ролі метрик для оцінки якості моделі було запропоновано R-squared, MAE та MSE.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДЕЛЕЙ ПРОГНОЗУ ТА ЇХ ОЦІНКА

В цьому розділі буде представлено програмну реалізацію моделей описаних в попередньому розділі. Код буде реалізовано мовою програмування Python з використанням різноманітних бібліотек в веб-інтерактивному середовищі розробки Jupyter-lab.

3.1 Опис вхідних даних

Дані, які будуть використані в цій роботі, взяті з сайту Kaggle.com [15]. Це дані, подані у вигляді декількох таблиць, які містять інформацію про резервування та відвідування ресторанів в місті Токіо з різних джерел (AirREGI/ Restaurant Board та Hot Pepper Gourmet). Також є додаткова інформація про розташування та жанр ресторану.

Таблиця air_reserve, зображена на рисунку 3.1, містить інформацію про бронювання столиків у ресторанах. Кожен рядок представляє окреме бронювання і містить інформацію про ідентифікатор ресторану в системі AirREGI/ Restaurant Board (air_store_id), дату та час бронювання (reserve_datetime), дату та час візиту (visit_datetime), а також кількість відвідувачів, які забронювали столик (reserve_visitors)

	air_store_id	visit_datetime	reserve_datetime	reserve_visitors
0	air_877f79706adbfb06	2016-01-01 19:00:00	2016-01-01 16:00:00	1
1	air_db4b38ebe7a7ceff	2016-01-01 19:00:00	2016-01-01 19:00:00	3
2	air_db4b38ebe7a7ceff	2016-01-01 19:00:00	2016-01-01 19:00:00	6
3	air_877f79706adbfb06	2016-01-01 20:00:00	2016-01-01 16:00:00	2

Рис. 3.1 – Таблиця air_reserve

Таблиця air_store_info, зображена на рисунку 3.2, містить інформацію про ресторани, що зареєстровані в системі. Кожен рядок відповідає окремому ресторану і містить його ідентифікатор (air_store_id), жанр (air_genre_name), р назву району розташування (air_are_name), широту (latitude) та довготу (longtitude).

	air_store_id	air_genre_name	air_area_name	latitude	longitude
0	air_0f0cdeee6c9bf3d7	Italian/French	Hyōgo-ken Kōbe-shi Kumoidōri	34.695124	135.197853
1	air_7cc17a324ae5c7dc	Italian/French	Hyōgo-ken Kōbe-shi Kumoidōri	34.695124	135.197853
2	air_fee8dcf4d619598e	Italian/French	Hyōgo-ken Kōbe-shi Kumoidōri	34.695124	135.197853
3	air_a17f0778617c76e2	Italian/French	Hyōgo-ken Kōbe-shi Kumoidōri	34.695124	135.197853
4	air_83db5aff8f50478e	Italian/French	Tōkyō-to Minato-ku Shibakōen	35.658068	139.751599
5	air_99c3eae84130c1cb	Italian/French	Tōkyō-to Minato-ku Shibakōen	35.658068	139.751599

Рис. 3.2 – Таблиця air_store_info

Таблиця air_visit, зображена на рисунку 3.3, включає в себе інформацію про дату відвідування (visit_date) конкретного ресторану (air_store_id) та кільскість гостей (visitors). Змінна visitors є прогнозованою змінною в цій задачі.

	air_store_id	visit_date	visitors
0	air_ba937bf13d40fb24	2016-01-13	25
1	air_ba937bf13d40fb24	2016-01-14	32
2	air_ba937bf13d40fb24	2016-01-15	29
3	air_ba937bf13d40fb24	2016-01-16	22

Рис. 3.3 – Таблиця air_visit

Таблиця date_info, викладена на рисунку 3.4, містить інформацію про дати і календарні дані, такі як день тижня та позначення святкового дня.

	calendar_date	day_of_week	holiday_flg
0	2016-01-01	Friday	1
1	2016-01-02	Saturday	1
2	2016-01-03	Sunday	1
3	2016-01-04	Monday	0

Рис. 3.4 – Таблиця date_info

Таблиця hpg_reserve, зображена на рисунку 3.5, містить інформацію про резервування столиків у ресторанах за допомогою системи Hot Pepper Gourmet (hpg). В ній містяться ідентифікатор ресторану в системі Hot Pepper Gourmet (hpg_store_id), дата та час відвідування ресторану (visit_datetime), дата та час коли було зроблено резервування (reserve_datetime) і кількість відвідувачів, для яких було зроблено резервування (reserve_visitors).

	hpg_store_id	visit_datetime	reserve_datetime	reserve_visitors
0	hpg_c63f6f42e088e50f	2016-01-01 11:00:00	2016-01-01 09:00:00	1
1	hpg_dac72789163a3f47	2016-01-01 13:00:00	2016-01-01 06:00:00	3
2	hpg_c8e24dcf51ca1eb5	2016-01-01 16:00:00	2016-01-01 14:00:00	2
3	hpg_24bb207e5fd49d4a	2016-01-01 17:00:00	2016-01-01 11:00:00	5
4	hpg_25291c542ebb3bc2	2016-01-01 17:00:00	2016-01-01 03:00:00	13

Рис. 3.5 – Таблиця hpg_reserve

Таблиця hpg_store_info, зображена на малюнку 3.6, містить додаткову інформацію про ресторани, які обслуговуються системою hpg: жанр кухні та розташування (позначення полів аналогічне до таблиці air_store_info рисунок 3.2).

	hpg_store_id	hpg_genre_name	hpg_area_name	latitude	longitude
0	hpg_6622b62385aec8bf	Japanese style	Tōkyō-to Setagaya-ku Taishidō	35.643675	139.668221
1	hpg_e9e068dd49c5fa00	Japanese style	Tōkyō-to Setagaya-ku Taishidō	35.643675	139.668221
2	hpg_2976f7acb4b3a3bc	Japanese style	Tōkyō-to Setagaya-ku Taishidō	35.643675	139.668221
3	hpg_e51a522e098f024c	Japanese style	Tōkyō-to Setagaya-ku Taishidō	35.643675	139.668221

Рис. 3.6 – Таблиця hpg_store_info

Таблиця store_id_relation, що на рисунку 3.7, містить відношення між ідентифікаторами ресторанів з системи AirREGI/Restaurant Board (air) та системи Hot Pepper Gourmet (hpg).

	air_store_id	hpg_store_id
0	air_63b13c56b7201bd9	hpg_4bc649e72e2a239a
1	air_a24bf50c3e90d583	hpg_c34b496d0305a809
2	air_c7f78b4f3cba33ff	hpg_cd8ae0d9bbd58ff9

Рис. 3.7 – Таблиця store_id_relation

3.2 Попередня обробка даних

Для подальшого аналізу та моделювання дані потрібно підготувати належним чином. В цей процес включають: злиття таблиць, обробку пропущених значень, кодування категоріальних змінних, можливе видалення непотрібних змінних.

Після зчитування даних в окремі датафрейми, ми отримали 7 окремих таблиць. Першим кроком буде розбити дати (дати відвідування та резерву) на окремі складові: день, місяць, рік. Час ми використовувати не будемо, а просто згрупуємо дані таким чином, щоб отримувати інформацію повністю за один день.

Тож для датасетів air_reserve, air_visit, date_info, hpg_reserve створюємо окремі стовпці, які містять розбиту на складові дату та час відвідування та резервування. При цьому початкові стовпці з датами видаляються. Приклад коду зображений на рисунку 3.8

```
air_reserve['visit_year'] = air_reserve['visit_datetime'].dt.year
air_reserve['visit_month'] = air_reserve['visit_datetime'].dt.month
air_reserve['visit_day'] = air_reserve['visit_datetime'].dt.day
air_reserve['reserve_year'] = air_reserve['reserve_datetime'].dt.year
air_reserve['reserve_month'] = air_reserve['reserve_datetime'].dt.month
air_reserve['reserve_day'] = air_reserve['reserve_datetime'].dt.day
air_reserve.drop(['visit_datetime','reserve_datetime'], axis=1, inplace=True)
```

Рис. 3.8 – Розбиття дат на складові для таблиці air_reserve

Для таблиці hpg_reserve виконуємо групування за стовпцями «hpg_store_id»,
«visit_year», «visit_month», «visit_day», «reserve_year», «reserve_month», «reserve_month» та сумує числові стовпці. Цей крок дозволить отримати загальну кількість резервів для кожного ресторану за певну дату. Для застосування моделей машинного навчання необхідно звести всі таблиці, які у нас є в одну. Робиться це за допомогою фунцкції merge(): з‘єднуємо таблицю hpg_reserve та store_id_relation по стовпцю hpg_store_id, який надалі видаляємо з таблиці hpg_reserve. За допомогою функції concat() об‘єднуємо таблиці air_reserve та hpg_reserve. В таблиці air_reserve групуємо та сумуємо кількість людей, що забронювали столик, по стовпцях air_store_id, visit_year, visit_month, visit_day. Далі з‘єднуємо таблиці air_reserve та date_info для додаткової інформації про календарні дні. Також з‘єднуємо air_reserve та air_store_info по стовпцю air_store_id. І нарешті з‘єднуємо таблиці air_reserve з air_visit в кінцевий датасет resto.

Виконавши ці кроки, ми отримали датасет resto (рисунок 3.9), з яким надалі будемо працювати. Стовпці були перейменовані для спрощення розуміння та зручності.

	id	year	month	day	reserv	weekday	isholiday	genre	area	latitude	longitude	visitors
1478	air_08ef81d5b7a0d13f	2016	10	15	3	Saturday	0	Italian/French	Tōkyō-to Chūō-ku Tsukiji	35.670651	139.771861	14.0
4175	air_c88467d88b2c8ecd	2017	4	22	14	Saturday	0	Italian/French	Tōkyō-to Chiyoda-ku Kudanminami	35.694003	139.753595	21.0
5280	air_26f10355d9b4d82a	2016	11	23	19	Wednesday	1	Other	Ōsaka-fu Ōsaka-shi Kyōmachibori	34.687697	135.495413	36.0
3640	air_c6a164dd4060e960	2017	3	30	4	Thursday	0	Dining bar	Ōsaka-fu Ōsaka-shi Ōgimachi	34.705362	135.510025	6.0
4630	air_cbe867adcf44e14f	2016	3	22	2	Tuesday	0	Izakaya	Fukuoka-ken Kitakyūshū-shi None	33.866465	130.764923	NaN

Рис. 3.9 – Таблиця resto

Тепер необхідно провести аналіз датасету, виявити чи є пропущенні дані, перевірити унікальні значення, провести описову статистику та візуалізувати дані, щоб виявити корисну для нас інформацію. Поверхневий огляд даних буде реалізовано за допомогою профілювання Pandas.

```
Profile = ydata_profiling.ProfileReport(resto)
profile.to_file("reports_data/resto.html")
```

Загальна інформація зображена на рисунку 3.10.

Dataset statistics

Number of variables	12
Number of observations	42193
Missing cells	6495
Missing cells (%)	1.3%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	4.2 MiB
Average record size in memory	104.0 B

Variable types

Categorical	6
Numeric	6

Рис. 3.10 – Статистика датасету resto за допомогою профайлингу

Цей датасет містить пусті значення в колонці visitors. Тому з таких даних ми зробимо окремий датасет, який і буде використовуватись для прогнозування кількості відвідувачів. При цьому такі записи ми вилучимо з датасету resto.

Після вилучення пропущених значень з датасету візуалізуємо дані. Подивимось на зміну середньої кількості відвідувачів та резервів по кожному місяцю за всі роки (рисунок 3.11) та загальну кількість відвідувачів та резервів впродовж всіх років помісячно (рисунок 3.12).

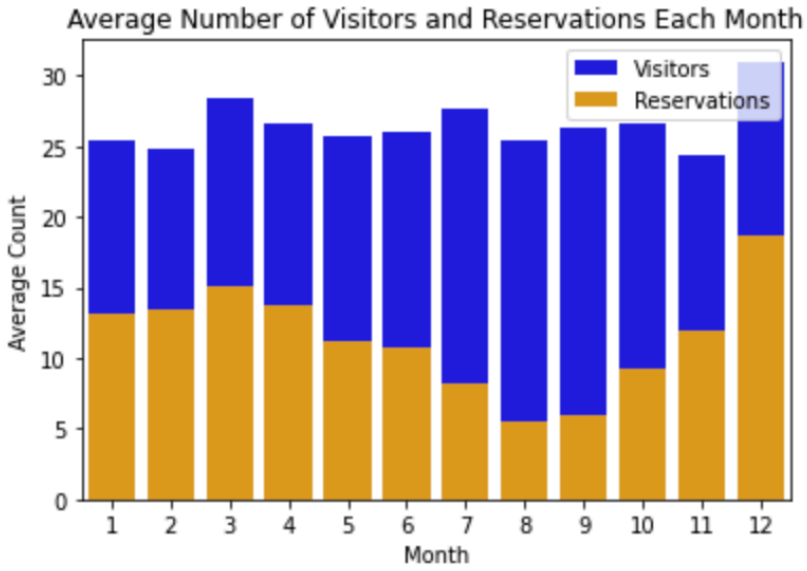


Рис. 3.11 – Середня кількість відвідувачів та резервів та кожен місяць

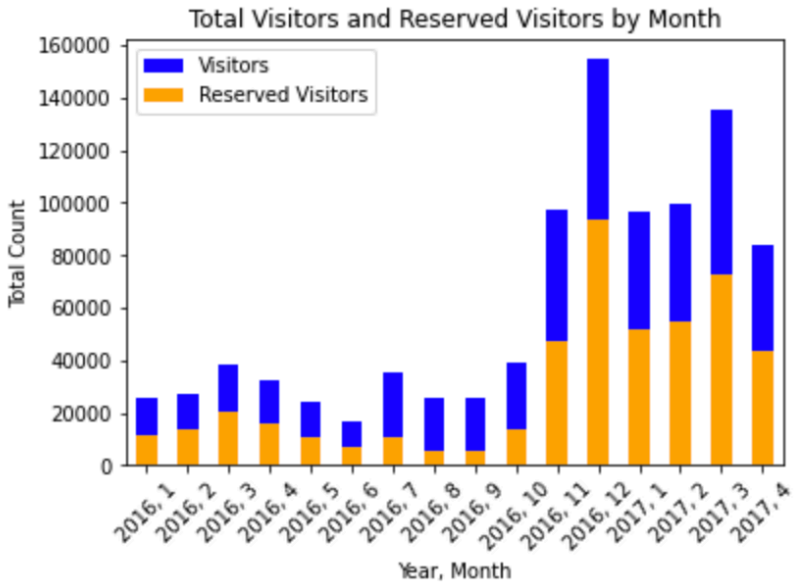


Рис. 3.12 – Загальна кількість відвідувачів та резервів помісячно

Можна помітити на рис. 3.11 незначне просідання кількості резервів для серпня та вересня, що не можна сказати про кількість відвідувачів. З рис. 3.12 видно різкий стрибок вгору по кількості в листопаді 2016 року.

Потрібно також дізнатись як впливає день тижня на середню кількість відвідувачів по ресторанах. Для цього згрупуємо інформацію по дням тижня та порахуємо середню кількість гостей. Результат представлено на малюнку 3.13

```
weekday
Monday       20.727648
Tuesday      21.655753
Thursday     22.820548
Wednesday    23.945107
Friday       30.095985
Sunday       30.216188
Saturday     33.595937
```

Рис. 3.13 – Середня кількість гостей залежно від дня тижня

Як бачимо, що п‘ятниця, неділя та субота найбільш відвідувані дні, що логічно. Ця інформація корисна для майбутнього кодування категоріальних змінних. Для лінійної регресії доречно буде закодувати дні тижня наступним чином: понеділок – 1, вівторок – 2, четвер – 3, середа – 4, п‘ятниця – 5, неділя – 6, субота – 7. Таким чином буде припущено, що субота має найбільшу вагу ніж усі інші дні тижня.

Подивимось на статистику по змінним reserve та visitors на рисунку 3.14

	reserv	visitors
count	35698.000000	35698.000000
mean	13.392991	26.848115
std	17.141707	18.334479
min	1.000000	1.000000
25%	4.000000	13.000000
50%	9.000000	23.000000
75%	18.000000	36.000000
max	1633.000000	244.000000

Рис. 3.14 – Статистика змінних reserve, visitors

Кожен запис в таблиці відповідає одному календарному дню, тобто максимальне значення зарезервованих столиків в якийсь день було 1663. З огляду на те, що така ситуація малоймовірна, було вирішено вважати це за помилку та вилучити занадто великі значення, які могли бути занесені в систему випадково. Для вилучення таких відхилень було застосовано статистичний метод z-score.

Feature Engineering – це процес створення нових ознак (фіч) в даних з метою поліпшення продуктивності моделі. Це один з ключових етапів у побудові моделей.

Задля виявлення сезонності було створено нову ознаку – кількість відвідувачів в минулий день тижня. Наприклад, якщо сьогодні вівторок, то для цього дня додається інформація по кількості відвідувачів минулого вівторка, у разі відсутності запису за минулий – інформація за позаминулий. Якщо є тенденція, що кількість відвідувачів в певний день тижня на минулому тижні має великий вплив на кількість відвідувачів в поточний день, то ця ознака може допомогти моделі зробити більш точний прогноз. Назва нової ознаки – visitors_last_dow. Цей тип змінної можна віднести до лаг функції. Один з поширених прикладі лаг-функцій – створення лаг-змінних для цільової змінної, де значення попередніх часових кроків стають предикаторами для прогнозування майбутніх значень. Лаг-змінні можуть відображати ступінь автокореляції в даних, дозволяють виявити шаблони повторюваності або циклічності а також виявити затримки ефектів або залежностей в часових рядах.

На базі широти та довготи можна створити ознаку віддаленості від центру. В датасеті існує 72 унікальні значення ознаки area. Всі ці області/райони розміщені в Японії. Тож було зібрано дані з координатами центрів міст, до яких прилягають ці області/райони, та пораховано в кілометрах відстань від ресторану до центру міста.

На рисунку 3.15 відстані розбиті на 7 груп: менше 5 кілометрів до центру, 5-10 км, 10-20 км, 20-70 км, 70-100 км, 100-125 км, більше 125 км.

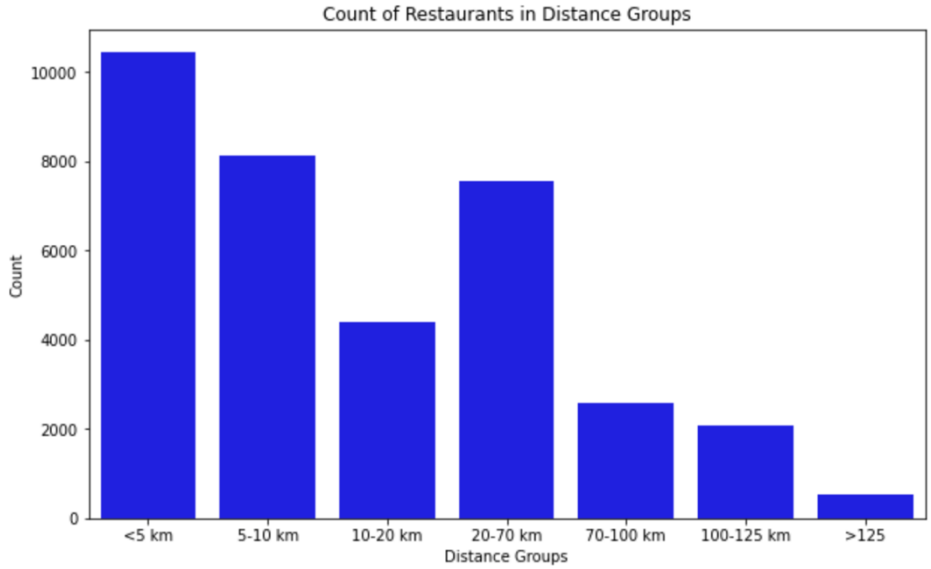


Рис. 3.15 – Кількість ресторанів в кожній групі

То чи впливає розташування безпосередньо на кількість відвідувачів? З графіку на малюнку 3.16 видно, що в середня прологарифмована кількість відвідувачів для цих груп відстаней не показує суттєвих відмінностей у межах своїх стандартних відхилень.

Рис. 3.16 – Середня логарифмована кількість відвідувачів в кожній групі

Розглянемо змінну кількість ресторанів в кожній області та як вона пов‘язана з кількістю відвідувачів. Щоб її обчислити ми просто підраховуємо

кількість ресторанів у відповідних областях. Оцінимо середню кількість відвідувачів, перетворену за допомогою логарифмування. На графіку (рис. 3.17) видно, що ніякого впливу кількість ресторанів в області немає.

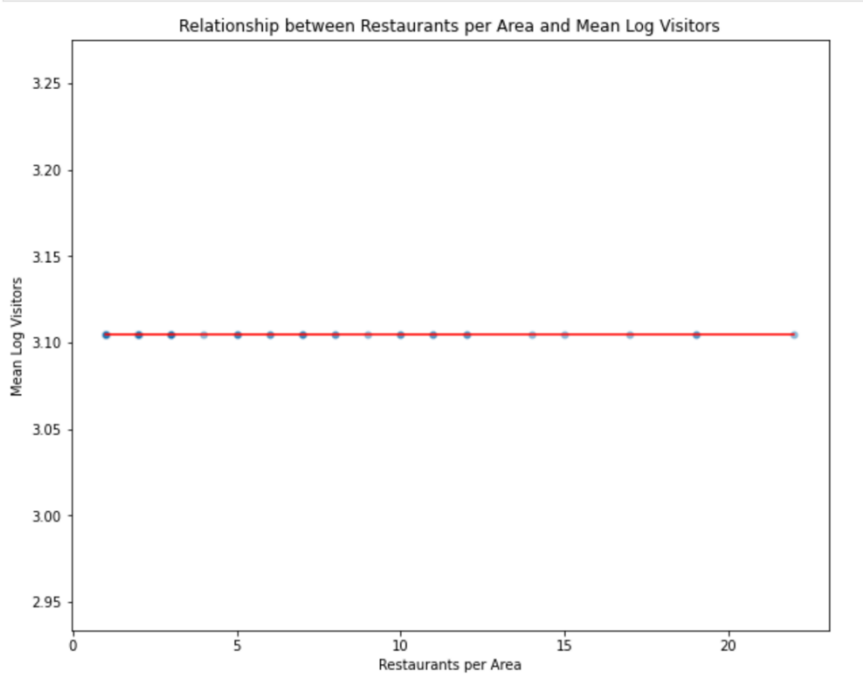


Рис. 3.17 – Зв‘язок між кількістю ресторанів в області та середньою логарифмічною кількістю відвідувачів

Розуміння того, чи наступний день вихідний або свято, може допомогти в прогнозуванні. Розглянемо змінні next_day_is_holiday, next_day_is_weekend та їхню кореляцію з цільовою змінною (рис. 3.18). Бачимо слабку кореляцію між next_day_is_weekend та visitors, що свідчить про наявність слабкого зв‘язку.

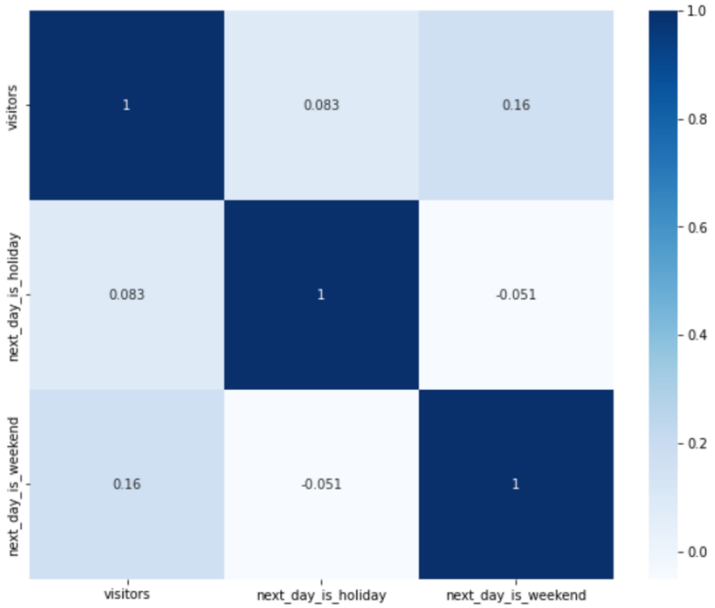


Рис. 3.18 – Кореляційна матриця

Для безпосередньої реалізації моделей необхідно закодовувати категоріальні змінні і для цього існують кілька способів, які мають свої переваги та недоліки.

1. One-Hot-Encoding (Одиничне кодування) – перетворює кожну категорію у новий бінарний стовпець. Основною перевагою є те, що цей метод не вводить неправильних взаємозв‘язків між категоріями. Проте він призводить до збільшення розмірності даних, якщо є багато унікальних категорій.
2. Label Encoding (Маркування категорій) – призначає числові значення категоріям. Він не збільшує розмірність даних, проте штучно вводить порядок між категоріями, що може неправильно вплинути на модель.
3. Frequency Encoding (Кодування частотою) – замінює категорії їх частотою в даних. Не збільшує розмірність даних, але може бути вразливим до рідкісних категорій з низькою частотою та надавати надмірну вагу частим категоріям.
4. Target Encoding – використовує інформацію про цільову змінну для перетворення категоріальних. Він добре працює з лінійними моделями, але виникає ризик перенавчання.

В моїй роботі буде виконано кодування для змінної genre за допомогою функції get_dummies, змінною area за допомогою CatBoostEncoder та змінної weekday за допомогою LabelEncoding (для всіх моделей крім лінійної регресії). Як показав профайлинг (рисунок 3.19) категоріальна змінна area має високу кардинальність, тому важливо правильно підібрати метод кодування, який збереже інформацію та зможе впоратись з перенавчанням.

Рис. 3.19 – Профайлинг для змінної area

3.3 Лінійна регресія

Для побудови лінійної регресії будемо застосовувати метод найменших квадратів, модель OLS з бібліотеки sklearn. Розіб‘ємо дані на тестовий та навчальний набори, закодуємо категоріальні змінні. Побудуємо модель та використаємо метрики MAE, MSE, R-squared для оцінки її якості.

Таким чином отримали наступні результати разом з кривими навчання на рисунку 3.18

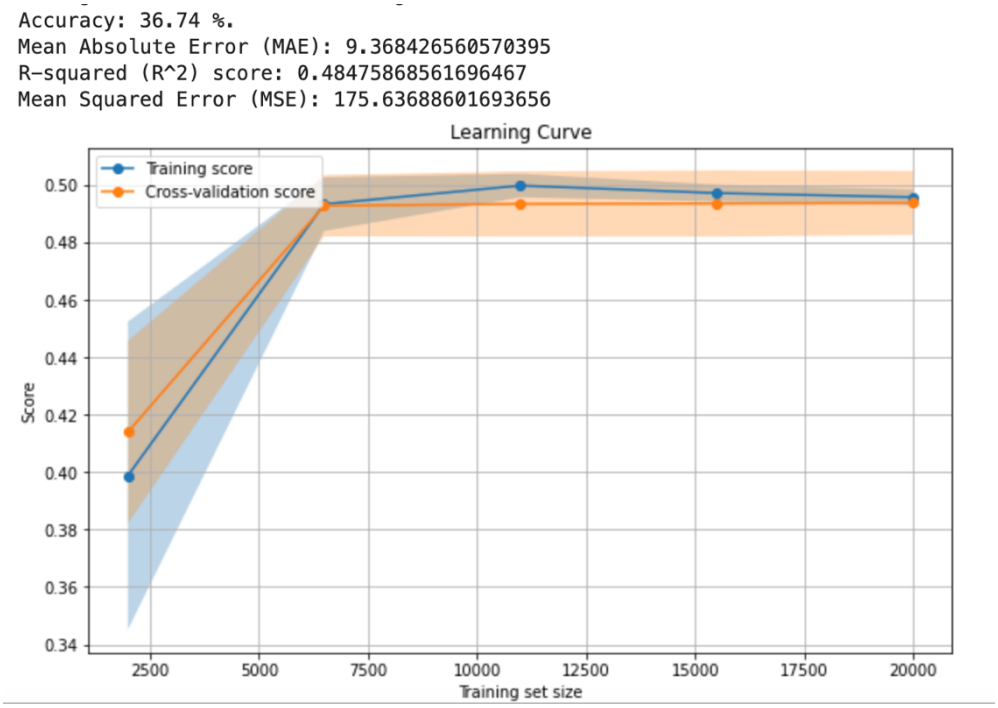


Рис. 3.18 – Оцінка якості моделі OLS для visitors

Застосуємо регуляризацію Lasso Regression для усунення мультиколінеарності в ознаках та усунення менш важливих ознак. Регуляризація накладає штраф на модель за використання менш важливих ознак. Шляхом додавання такого штрафного члена, Lasso стимулює модель встановлювати коефіцієнти менш важливих ознак близькими до нуля. Це призводить до стиснення набору змінних, де лише найважливіші залишаються з ненульовими коефіцієнтами. Таке стиснення допомагає боротись з перенавчанням та зменшує складність моделі. Результат застосування регуляризації зображено на малюнку 3.19

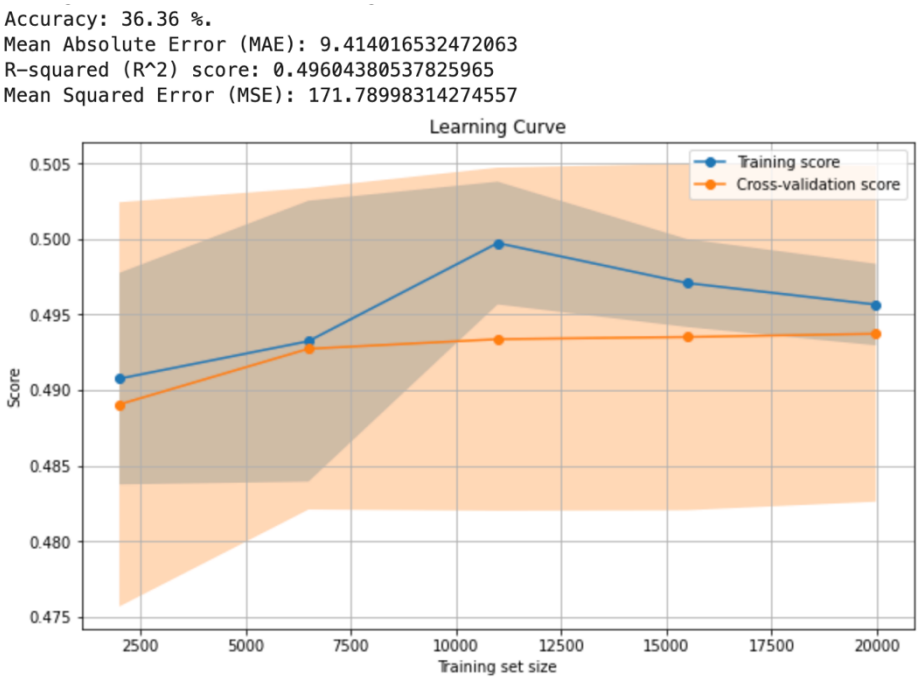


Рис. 3.19 – Результат застосування регуляризації

R-квадрат збільшився на 0.1, середня квадратична похибка зменшилась на 4.1, а середня абсолютна похибка збільшилась на 0.04. В цілому вдалося дещо покращити модель.

Спробуємо побудувати модель не для цілого датасету, а для окремого ресторану, вибравши відповідний id, у якого найбільша кількість записів – 374 При цьому ознаки, які характеризували місце ресторану необхідно одразу прибрати, бо вони не надають ніякої інформації. Побудувавши модель та обрахувавши метрики, отримали наступні результати (рисунок 3.20)

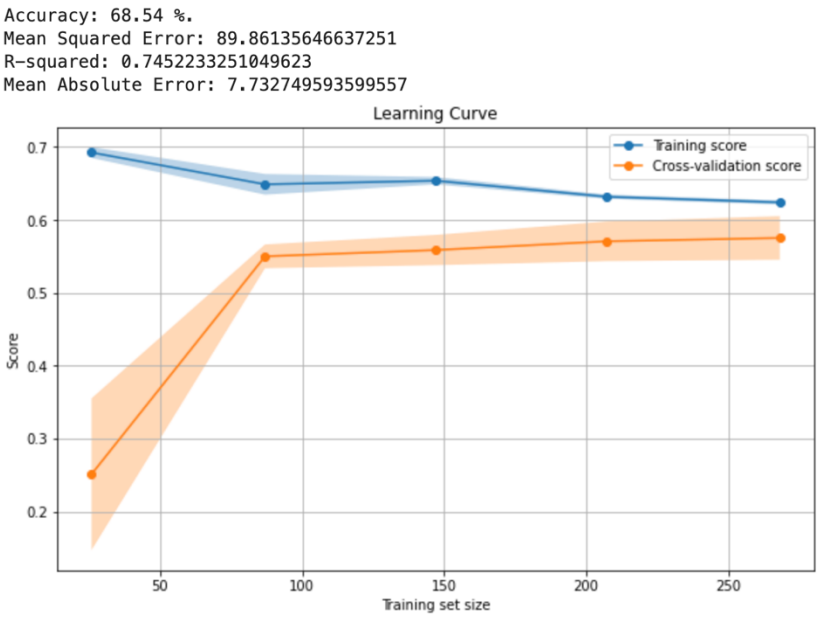


Рис. 3.20 – Оцінка моделі для конкретного ресторану

Як виявилось, набагато ефективніше прогнозувати кількість відвідувачів по-ресторанно ніж для набору ресторанів.

Заради експерименту, спробуємо спрогнозувати середнє значення кількості відвідувачів щоденно для всіх ресторанів. Алгоритм для цього випадку такий самий, отримали наступні результати на рисунку 3.21

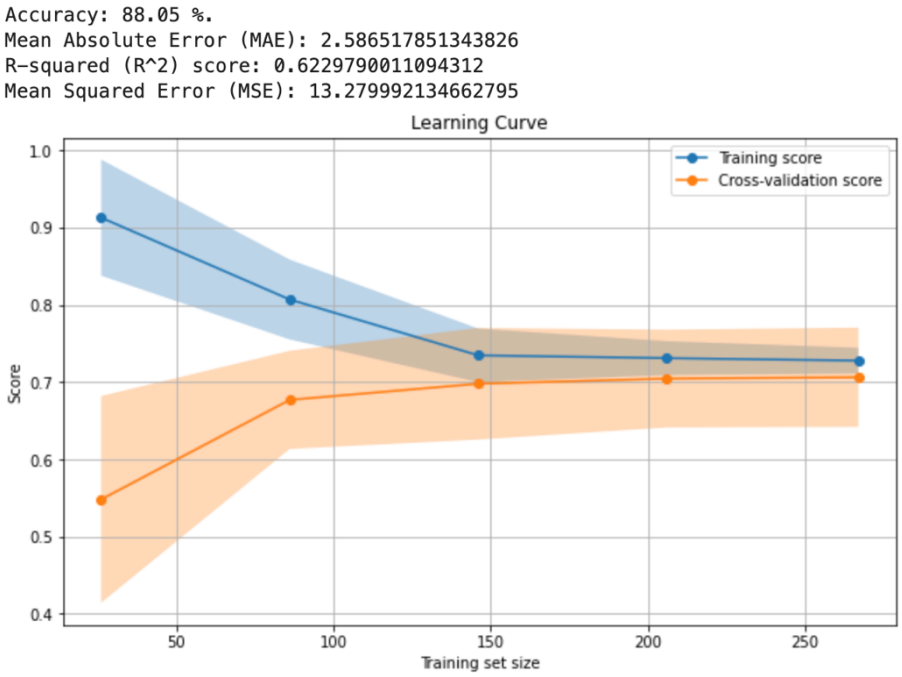


Рис. 3.21 – Оцінка моделі для прогнозування середнього значення

3.4 Випадковий ліс (Random Forest)

Для побудови випадкового лісу будемо застосовувати модель RandomForestRegressor з бібліотеки sklearn. Розіб‘ємо дані на тестовий та навчальний набори, закодуємо категоріальні змінні. Побудуємо модель та використаємо метрики MAE, MSE, R-squared для оцінки її якості. При цьому параметри випадкового лісу залишили усталеними. Результат кривих на малюнку 3.22

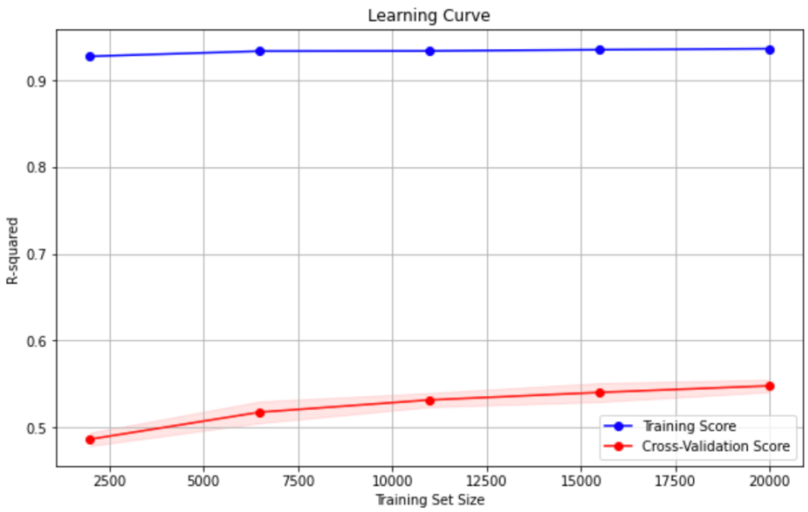


Рис. 3.22 – Криві навчання

З графіку чітко видно тенденцію до перенавчання – великий розрив між кривими. При цьому розрив постійний, додавання нових даних не допоможе. Необхідно зменшити складність моделі, використовуючи менше ознак, методи регуляризації та підбір гіперпараметрів випадкового лісу

Спробуємо прибрати деякі ознаки на свій розсуд, підібрати гіперпараметри та подивимось на результат на рисунку 3.23 та на метрики якості моделі на рисунку 3.23.

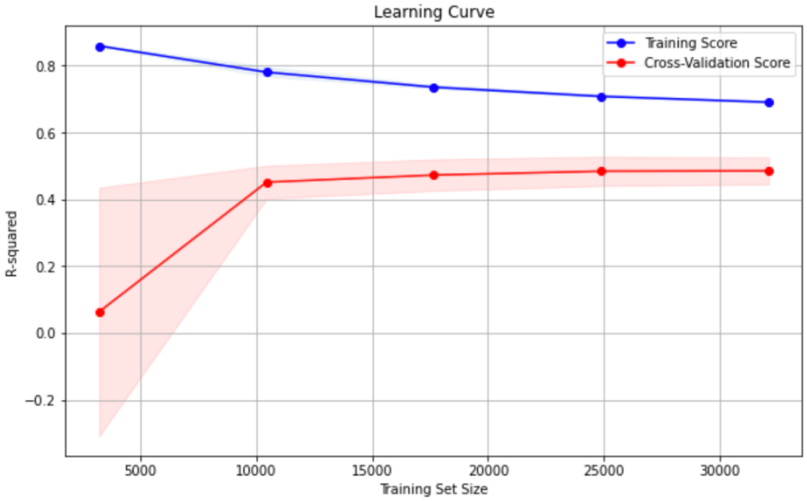


Рис. 3.23 – Криві навчання

При цьому ми маємо точність 41.05% , R-квадрат - 0.532, MAE - 8.733, MSE - 158.274. Якщо на даному етапі порівнювати випадковий ліс та лінійну регресію – випадковий ліс впорався дещо краще.

Перейдемо до задачі виявлення важливих ознак, та усунення неважливих. Для цього нам знадобиться метод feature_importances_ і ранжування по рівню важливості. Ось, що ми отримали (рис. 3.24)

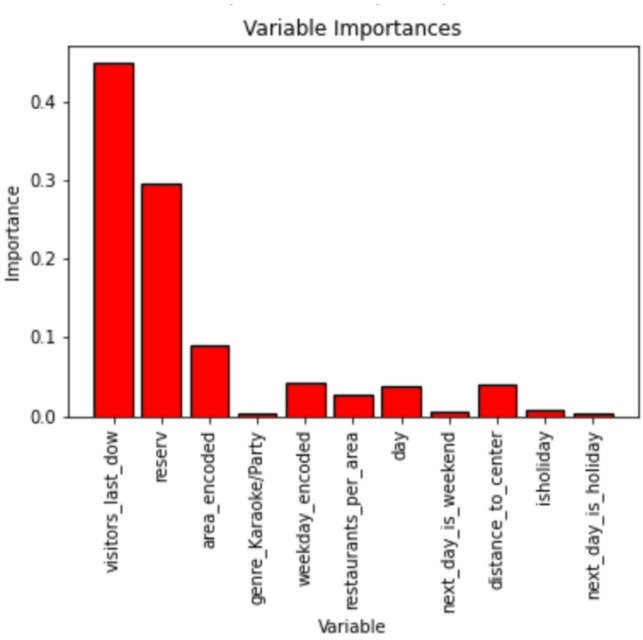


Рис. 3.24 – Ранжування ознак

Встановимо поріг важливості у 85% та створимо діаграму сукупної важливості, щоб візуалізувати кумулятивний внесок ознаки у загальну важливість (рис. 3.25).

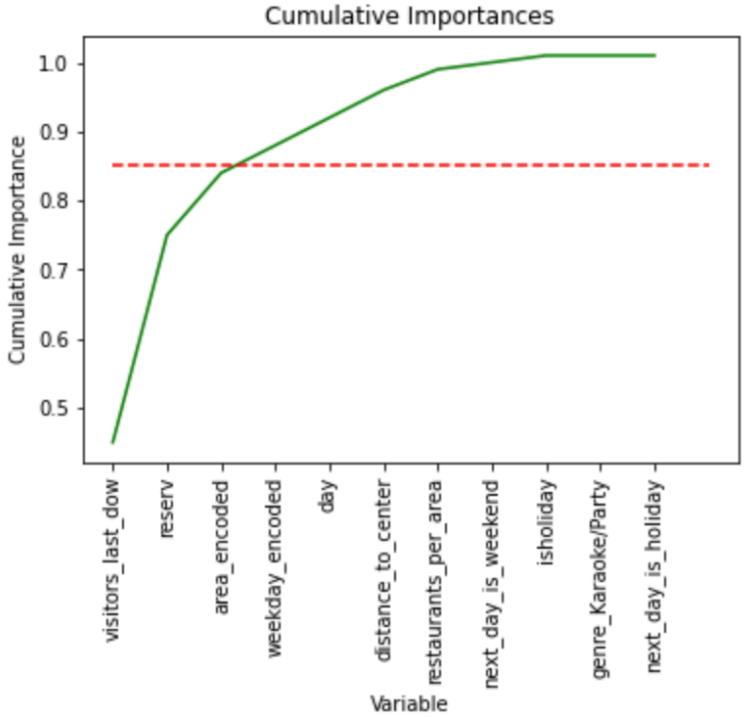


Рис. 3.25 – Кумулятивна важливість

Найважливішими 4-ма ознаками виявились visitors_last_dow, reserv, area_encoded та weekday_encoded. Побудуємо модель на основі цих ознак та побачимо, що отримаємо в результаті на рисунку 3.26

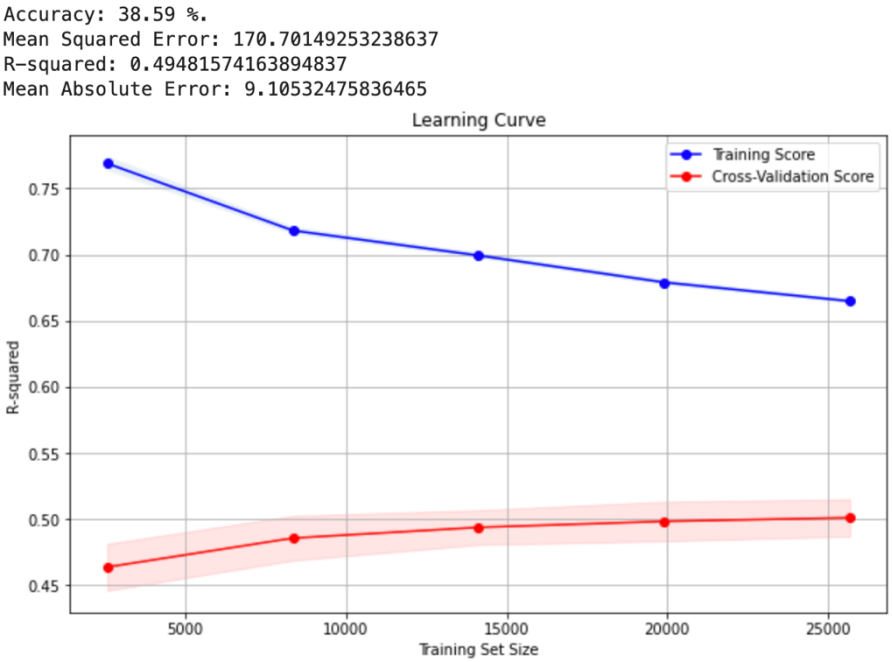


Рис. 3.26 – Криві навчання та метрики оцінки моделі

Нам вдалось ще трохи зменшити перенавчання, але, на жаль, втративши при цьому точність аж в 3%.

Тепер повторимо експеримент, як і в розділі з лінійною регресію, будемо будувати модель для конкретного ресторану. Приберемо ознаки, які стосуються розташування ресторану та не несуть ніякої додаткової інформації, та підберемо гіперпараметри за допомогою GridSearch (рисунок 3.27)

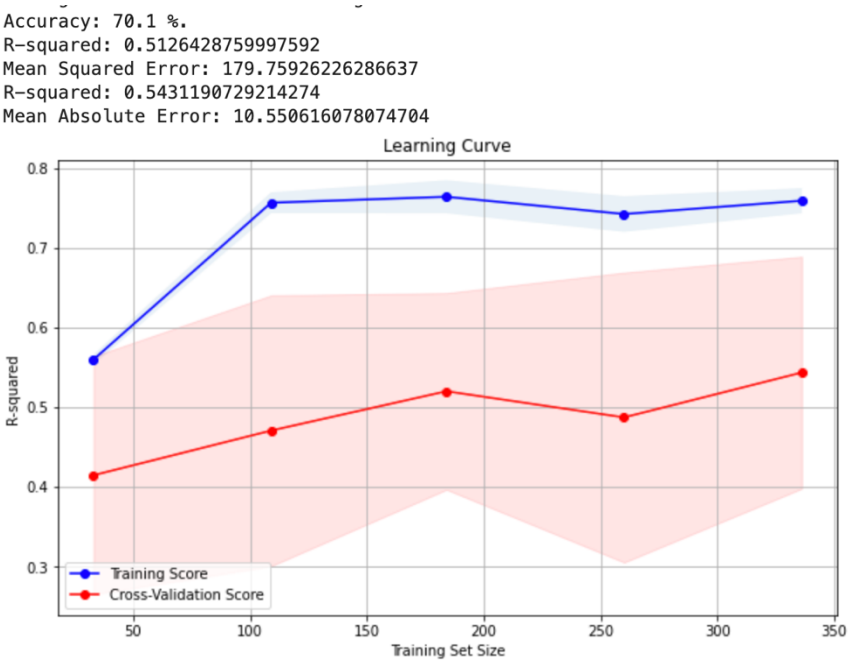


Рис. 3.27 –Криві навчання та метрики оцінки моделі

Отже, як і у випадку з лінійною регресією, модель для конкретного ресторану показала себе краще, ніж модель для повного початкового набору даних зі збільшенням точності на 31,5%.

Тепер побудуємо модель для прогнозу середньої кількості відвідувачів (рис. 3.28)

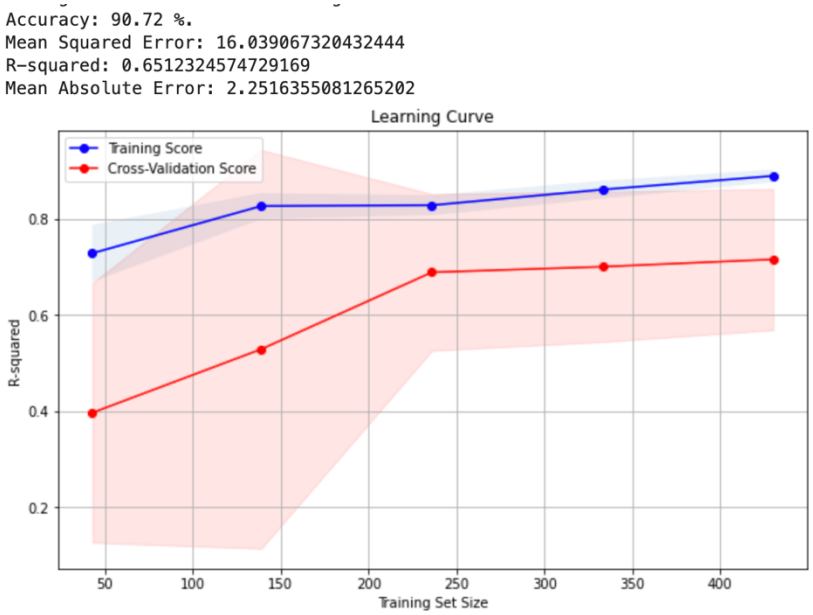
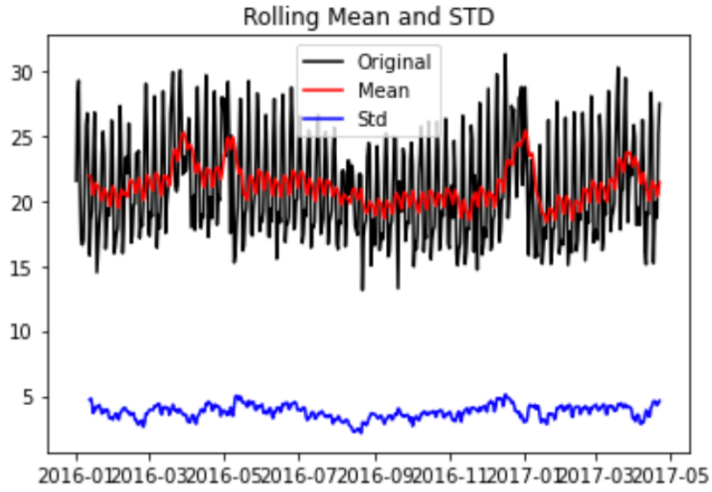


Рис. 3.28 – Криві навчання та метрики оцінки моделі

3.5 ARIMA

Для побудови моделі ARIMA будемо використовувати дані з таблиці air_visit. Прогнозувати будемо середню кількість відвідувань за день по всіх ресторанах.

Першим кроком є перевірка часового ряду на стаціонарнсіть, для цього нам знадобиться тест Дікі-Фуллера. Результати зображені на рисунку 3.29.




```
Results of Dickey-Fuller Test:
Test Statistic                  -3.796104
p-value                          0.002947
#Lags Used                      16.000000
Number of Observations Used    461.000000
Critical Value (1%)             -3.444615
Critical Value (5%)             -2.867830
Critical Value (10%)            -2.570120
```

Рис. 3.29 – Перевірка на стаціонарність часового ряду

Тестова статистика дорівнює -3.796104 вона вимірює наскільки сильно дані спростовують нульову гіпотезу про нестаціонарність. Тому ми відхиляємо нульову гіпотезу і робимо висновок про стаціонарність часового ряду.

Далі необхідно побудувати графіки ACF та PCF для аналізу автокореляції в часовому ряді і визначити оптимальні параметри моделей прогнозування. ACF зображує кореляцію між поточним і попередніми

значеннями часового ряду. PACF, в свою чергу, визначає прямий вплив попередніх значень на поточні. Він допомагає виявляти причинно-наслідкові зв‘язки між змінними.

Графіки для нашого часового ряду зображені на рисунку 3.30

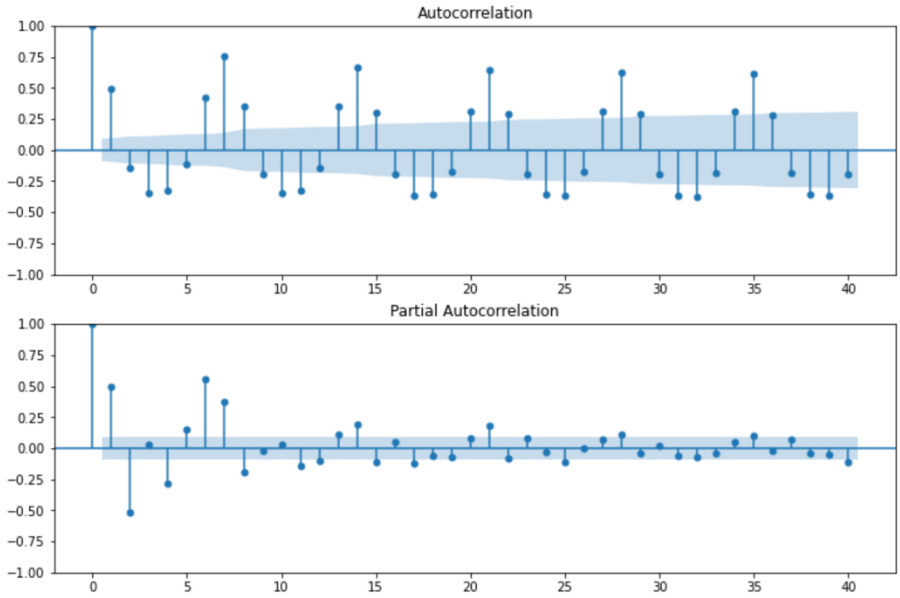


Рис. 3.30 – ACF та PACF

На графіку ACF бачимо патерн, який повторюється кожні 7 лагів. Це відповідає 7-денному сезонному терміну. Існують правила, за якими можна обирати порядок сезонної частини моделі. Через наявну сезонність вирішено застосовувати модель SARIMA, яка є розширенням моделі ARIMA.

Тож, якщо ряд має виразний та стабільний сезонний зразок, то потрібно використовувати порядок сезонного диференціювання (інакше модель передбачатиме, що сезонний зразок послаблюється з часом). Проте ніколи не потрібно використовувати більше ніж одне сезонне диференціювання або більше ніж 2 порядки загального диференціювання (сезонного та несезонного) [16].

Тож, хоч наш ряд і є стаціонарним, все одно існує необхідність продиференціювати часовий ряд. Це робиться шляхом включення сезонної різниці в модель SARIMA.

Отже, сезонний порядок має наступний вигляд (,1, , 7).

Для підбору параметрів було написано спеціальну функцію autoSARIMA. Найкращими параметрами виявились [1, 0, 2, 0, 1, 1, 7]. Побудувавши модель, отримали наступний результат (рисунок 3.31)

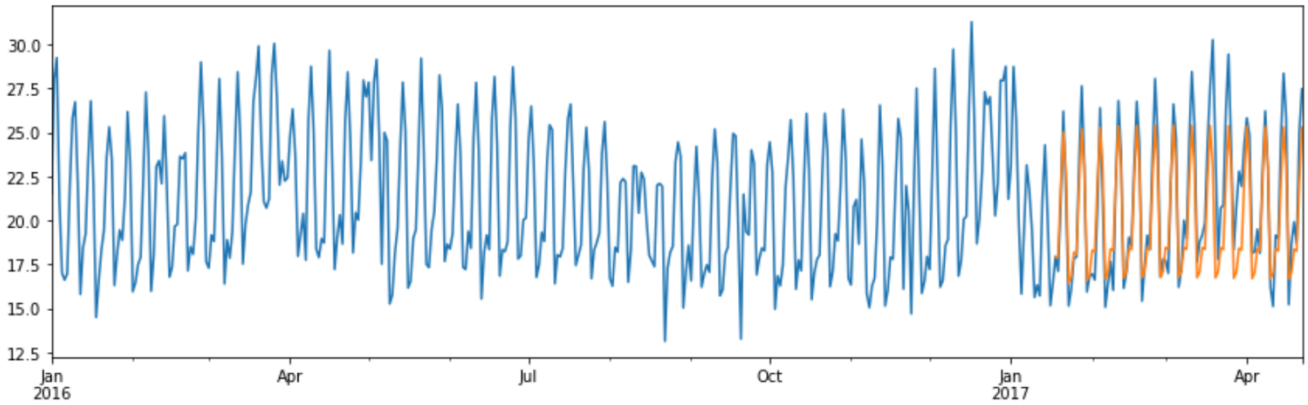


Рис. 3.31 – Прогноз моделі SARIMA(1, 0, 2)(0, 1, 1, 7)

Обраховані метрики якості моделі зображені на малюнку 3.31

```
R-squared: 0.7663008569246865
Mean Squared Error (MSE): 3.814843564438434
Mean Absolute Error (MAE): 1.4600647784869438
Mean Absolute Percentage Error (MAPE): 6.631100258547418
```

Рис. 3.31 – Метрики

SARIMA досить непогано впоралась з прогнозуванням.

3.6 LSTM

Для застосування нейронних мереж LSTM будемо також використовувати набір даних air_visit, та прогнозувати середню кількість відвідувачів за день для всіх ресторанів.

Спочатку необхідно підготувати дані для навчання: для LSTM вони підготовлюються у вигляді послідовностей: визначається розмірність вікна, тобто кількість попередніх часових кроків, які будуть використані для прогнозування. В нашому випадку це буде 7, це пов‘язано з сезонністю, яку ми виявили в пункті ARIMA. Далі створимо набори даних завдяки функції create_dataset. Ця функція бере часовий ряд даних та розділяє його на вікна фіксованого розміру та відповідні значення наступного кроку. За допомогою torch.tensor дані кодуються для подальшої обробки в моделі LSTM.

Для визначення архітектури створимо клас LSTMNetwork, який є підкласом nn.Module з бібліотеки PyTorch. LSTMNetwork визначає архітектуру моделі, яка включає в себе LSTM шар та лінійний шар. LSTM шар обробляє послідовні дані, такі як часові ряди, а лінійний шар використовується для прогнозування вихідного значення на основі останнього вихідного стану LSTM шару.

У конструкторі LSTMNetwork вказуємо розмірність вхідного шару input_size, розмірність прихованого шару hidden_size, розмірність вихідного шару output_size та кількість шарів num_layers. Ініціалізуємо оптимізатор і функцію втрат, створюємо завантажувача даних DataLoader, тренуємо модель, валідуємо модель та генеруємо прогнози (рисунок 3.32)

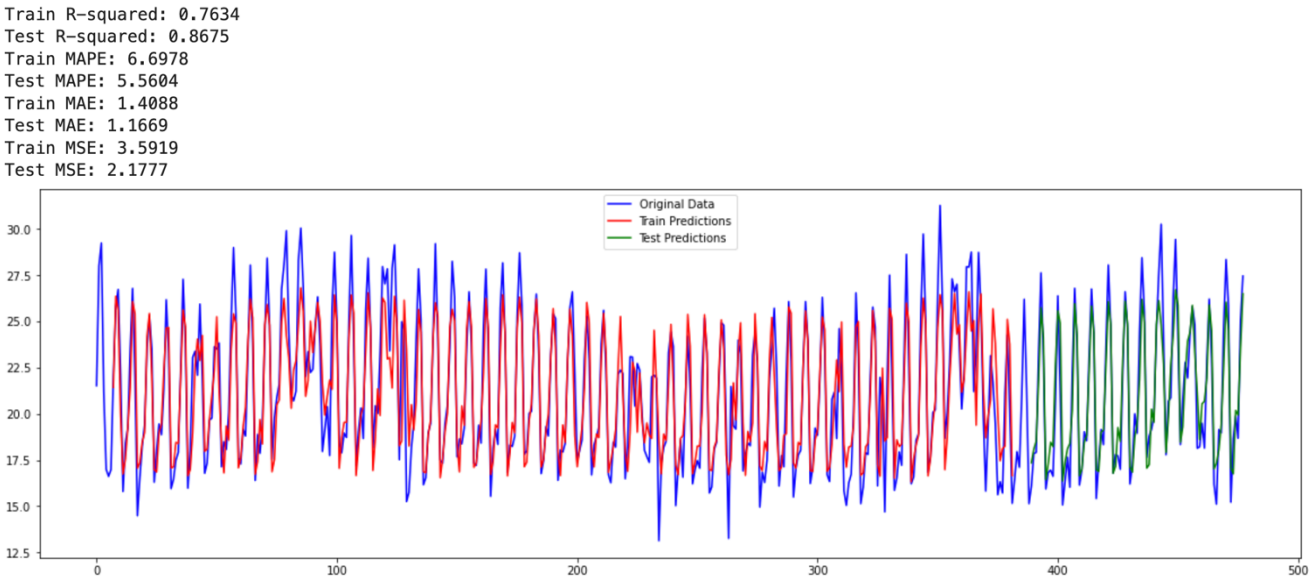


Рис. 3.32 – LSTM

Коефіцієнт детермінації 0.86 – найкращий з поміж відповідних коефіцієнтів інших моделей. Проте, ще можливо працювати над поліпшенням продуктивності.

3.7 Порівняльний аналіз побудованих моделей

Остаточні оцінки якості моделей зобразимо у вигляді таблиці 3.1

Таблиця 3.1 – Метрики оцінки якості для моделей

	R-squared	MAE	MSE
Лінійна регресія	0.65	2.47	12.4
Випадковий ліс	0.67	2.47	13.01
SARIMA	0.77	1.46	3.81
LSTM	0.87	1.16	2.17

В таблиці наведені порівняння продуктивності моделей для задачі прогнозування середньої кількості відвідувачів для закладів харчування, які наявні в системі.

З аналізу цих метрик видно, що LSTM мережі показують найвищу точність у прогнозуванні кількості відвідувачів у порівнянні з іншими моделями. Вона має найвище значення коефіцієнта детермінації (R-квадрат) на рівні 0.87, що свідчить про високу здатність моделі пояснювати варіацію в даних. MAE і MSE для LSTM також найнижчі, що свідчить про меншу середню похибку прогнозування.

SARIMA також показує гарні результати зі значеннями R-квадрат, MAE та MSE, що перевищують ті, що спостерігаються в лінійній регресії та

випадковому лісі. SARIMA може бути ефективним методом прогнозування в контексті кількості відвідувачів ресторанів.

Лінійна регресія та випадковий ліс показують меншу точність у порівнянні з SARIMA та LSTM. Хоча вони можуть бути простими та інтерпретованими моделями, вони можуть не враховувати складні залежності та контекстуальні зміни, що присутні у даних про відвідуваність ресторанів.

3.8 Висновки до розділу 3

В даному розділі було проведено прогнозування кількості відвідувачів в ресторанах за допомогою різних моделей. Були розглянуті різні випадки цільової змінної: прогнозування кількості відвідувачів для одного ресторану та прогнозування середньої кількості гостей щоденно для усіх закладів системи. Останній варіант може мати декілька корисних аспектів для бізнесу, а саме: стратегічне планування, управління ресурсами, маркетингові рішення, бенчмаркінг та аналіз.

По результатам найбільш точною виявились LSTM мережі з коефіцієнтом детермінації 0.87, найбільш неточною – лінійна регресія з відповідною метрикою в 0.62.

В якості подальших досліджень можна вдосконалювати побудовані моделі, збирати додаткові дані (погодні умови, події, які потенційно можуть впливати), аналіз та усунення аномалій при зборі даних.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

В заданому розділі буде проведено оцінювання основних характеристик для майбутнього програмного продукту, що спеціалізується на прогнозуванні кількості відвідувачів в ресторанному бізнесі.

Дана реалізація буде сприяти проведенню усіх необхідних досліджень, що дасть змогу якісно дослідити питання не лише в Україні, проте у всьому світі.

Також в даному дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього застосовувався апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу стійкості фінансових показників. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є наступні:

- функціонування на персональних комп'ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка можливого програмного продукту, яка дозволяє аналізувати різні характеристики, що безпосередньо впливають на стійкість підприємства. Беручи за основу цю функцію, можна виділити наступні:

F_1 – вибір мови програмування

F_2 – якісний аналіз даних.

F_3 – графічні показники.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1:

а) Python.

б) R.

Функція F_2.

а) Застосування вбудованих функцій.

б) Створення своїх обчислень значень.

Функція F_3:

а) Використання шаблонних графіків.

б) Створення своїх.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

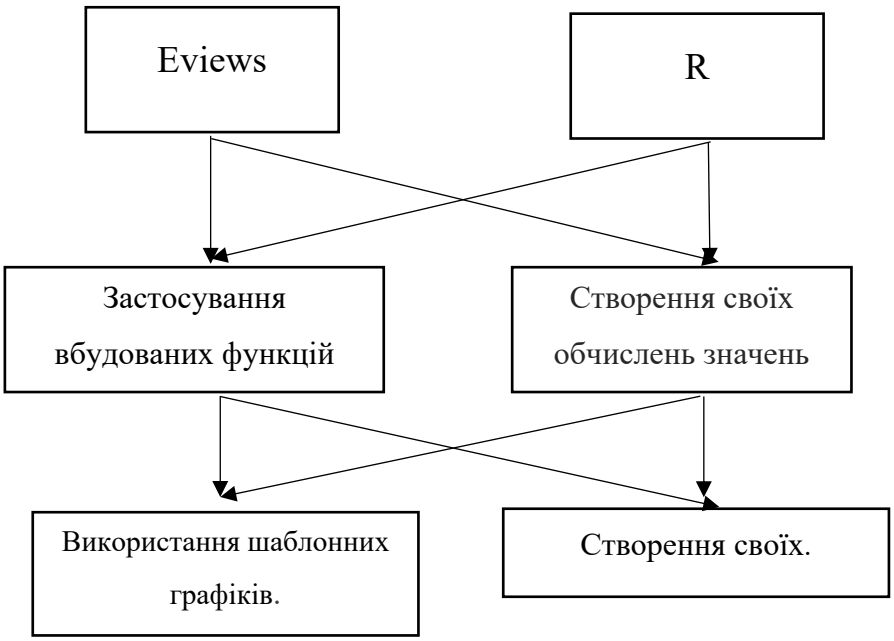


Рис 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантиів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

Таблиця 4.1 - Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	*А*	Має широкий спектр бібліотек для наукових обчислень, машинного навчання та обробки даних	Може бути повільнішим у порівнянні з деякими іншими мовами програмування
	Б	Має багатий набір пакетів для статистичного аналізу та візуалізації даних, добре підходить для академічних та статистичних досліджень	Менш підходить для загального програмування та розробки великих додатків, менший вибір загальних бібліотек
F_2	*А*	Забезпечують зручну та швидку реалізацію загальних операцій без необхідності додаткового програмування	Можуть мати обмежену функціональність або бути менш гнучкими,
	Б	Дозволяє здійснювати більш спеціалізовані та контрольовані операції	Може вимагати більше часу та зусиль, а також бути складнішою з точки зору кодування
F_3	*А*	Дозволяє швидко та зручно створювати професійно виглядаючі графіки	Може обмежувати гнучкість та індивідуальний настрій графіків
	Б	Надає повний контроль над виглядом та функціональністю графіків	Може вимагати більше часу, знань та навичок у програмуванні та візуалізації даних

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1*:*

Віддаємо перевагу варіануту А через його загальну доступність та простоту написання коду, відкидаючи варіант Б для спрощення роботи.

Функція F_2:

Програма допускає обрання обох варіантів. Можливо використати варіанти А чи Б.

Функція F_3*:*

Реалізація першого варіанту є сприйнятливою для програми. Це варіант А.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

F_1а – F_2а – F_3а

F_1а – F_2б – F_3а

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- *X1* – швидкодія мови програмування;
- *X2* – об’єм пам’яті для обчислень та збереження даних;
- *X3* – час навчання даних;

- *X4* – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.2.

Таблиця 4.2 - Основні параметри програмного продукту

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	оп/мс	60	90	120
Об’єм пам’яті	X2	Мб	70	60	40
Час попередньої обробки даних	X3	мс	90	80	70
Потенційний об’єм програмного коду	X4	кількість рядків коду	2500	1500	800

За даними таблиці 4.3 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

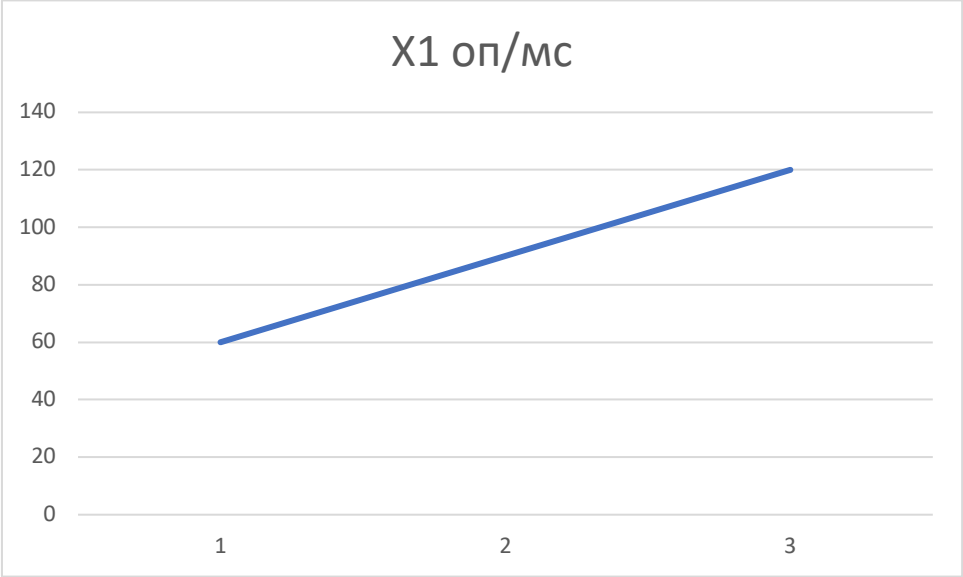


Рис 4.2 – X1, швидкодія мови програмування

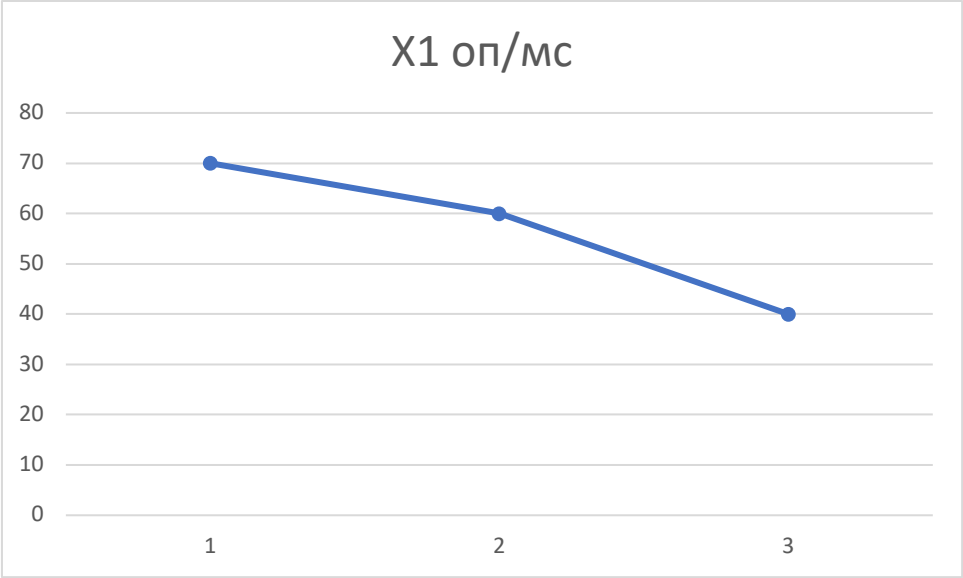


Рис 4.3 – X2, об’єм пам’яті

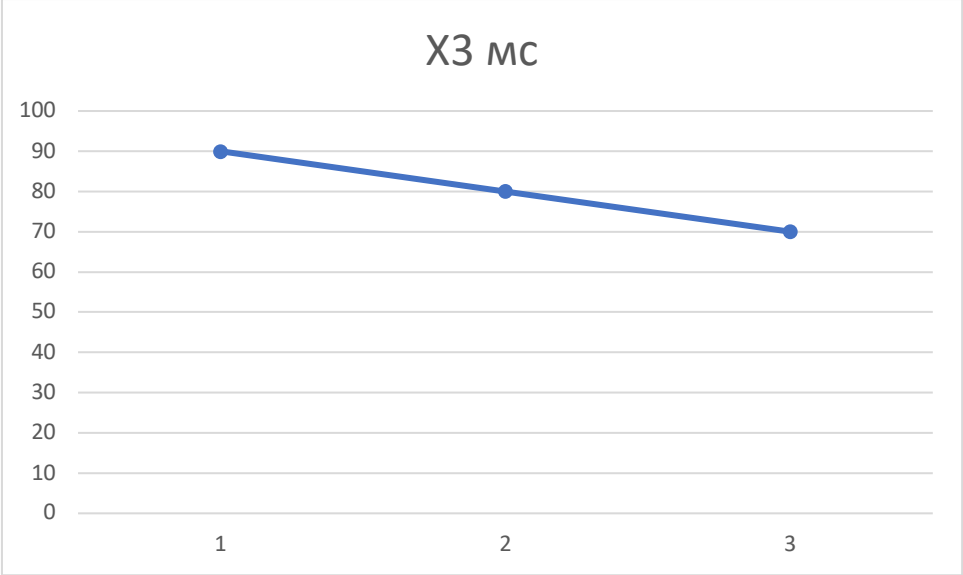


Рис 4.4 – X3, час попередньої обробки даних

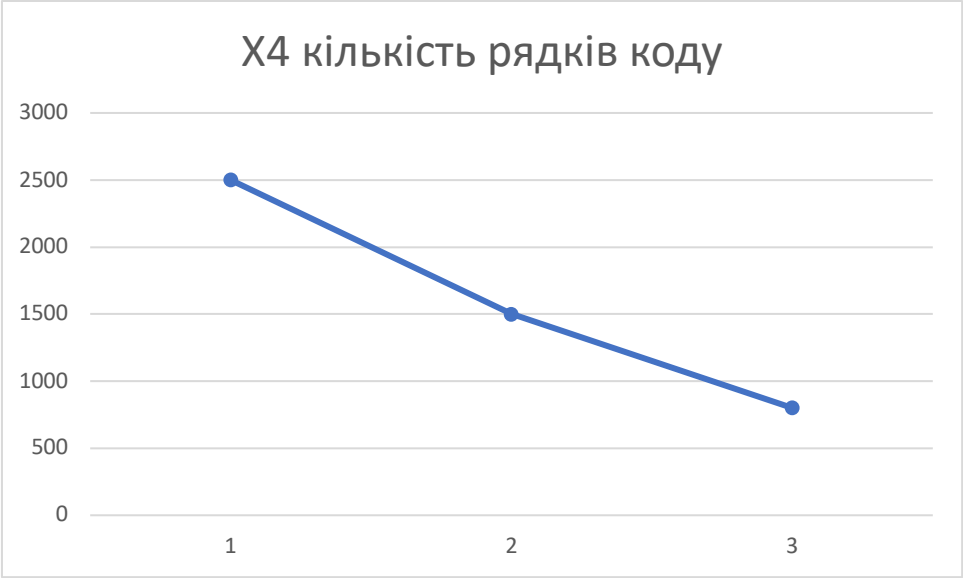


Рис 4.5 – X4, потенційний об’єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;
– перевірку придатності експертних оцінок для подальшого використання;
– визначення оцінки попарного пріоритету параметрів;
– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 - Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхи-лення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	2	1	2	1	1	2	1	10	-7,5	56,25
X2	Об'єм пам'яті	Mb	4	2	3	2	3	3	3	20	2,5	6,25
X3	Час попередньої обробки даних	мс	1	3	1	3	2	1	2	13	-4,5	20,25
X4	Потенційний об'єм програмного коду	Кількість рядків коду	3	4	4	4	4	4	4	27	9,5	90,25
	Разом		10	10	10	10	10	10	10	70	0	173

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \qquad (4.1)$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5 \qquad (4.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T. \qquad (4.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 173. \qquad (4.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 141}{7^2(4^3 - 4)} = 0{,}706 > W_k = 0{,}67. \qquad (4.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 - Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	<	<	<	<	<	<	<	0.5
X1 і X3	>	<	>	<	<	>	<	<	0.5
X1 і X4	<	<	<	<	<	<	<	<	0.5
X2 і X3	>	<	>	<	>	>	>	>	1.5
X2 і X4	>	<	<	<	<	<	<	<	0.5
X3 і X4	<	<	<	<	<	<	<	<	0.5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}. \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю А=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (4.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{вi}} = \frac{b'_i}{\sum_{i=1}^{n} b'_i}, \quad (4.9)$$

$$b'_i = \sum_{i=1}^{N} a_{ij} b_j \quad (4.10)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 - Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1	0,5	0,5	0,5	2,5	0,16	9,25	0,16	34,125	0,16
X2	1,5	1	1,5	0,5	4,5	0,28	16,25	0,28	59,125	0,28
X3	1,5	0,5	1	0,5	3,5	0,22	12,25	0,21	41,875	0,2
X4	1,5	1,5	1,5	1	5,5	0,34	21,25	0,35	77,875	0,36
Всього:					16	1	59	1	213	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2* (Об'єм пам'яті), *X3* (час попередньої обробки даних) та *X4* (потенційний об'єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X1* (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\,, \qquad (4.11)$$

де n – кількість параметрів;

$K_{вi}$– коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	110	28	0,16	4,48
F3	А	X2	50	30	0,28	8,4
	Б	X3	72	18	0,2	3,6
F4	А	X4	1000	23	0,36	8,28

За даними з таблиці 4.6 за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \qquad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 4{,}48 + 8{,}4 + 8{,}28 = 21{,}16 ;$$

$$K_{K2} = 4{,}48 + 3{,}6 + 8{,}28 = 16{,}26 .$$

Як видно з розрахунків, кращим є 1 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.13)$$

де T_P – трудомісткість розробки ПП;

$K_П$ – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

$K_М$ – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =37 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.8. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ}$ = 0.9. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 37 \cdot 1.8 \cdot 0.9 = 59{,}94 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_Р$ =29 людино-днів, $К_П$ =0.9, $К_{СК}$ = 1, $К_{СТ}$ =0.8:

$$T_2 = 29 \cdot 0.9 \cdot 0.8 = 20.88 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (59{,}94 + 20.88 + 4.8 + 20.88) \cdot 8 = 852 \text{ людино-годин.}$$
$$T_{II} = (59{,}94 + 20.88 + 6.91 + 20.88) \cdot 8 = 868{,}88 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 25000 грн., один аналітик в області даних з окладом 37000. Визначимо середню зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} \text{ грн.,} \qquad (4.14)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$С_Ч = \frac{25000 + 25000 + 37000}{3 \cdot 21 \cdot 8} = 148{,}81 \text{ грн.} \qquad (4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$С_{ЗП} = С_{ч} \cdot Т_{i} \cdot К_{Д}, \quad (4.16)$$

де $С_{ч}$– величина погодинної оплати праці програміста;

$Т_{i}$ – трудомісткість відповідного завдання;

$К_{Д}$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 148,81 \cdot 852 \cdot 1.2 = 152143,35$ грн.

II. $С_{ЗП} = 148,81 \cdot 868.88 \cdot 1.2 = 155157,64$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 152143,35 \cdot 0.22 = 33471,54$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 155157,64 \cdot 0.22 = 34134,68$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_{М}$)

Так як одна ЕОМ обслуговує одного програміста з окладом 25000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо

$$С_{Г} = 12 \cdot М \cdot К_{З} = 12 \cdot 25000 \cdot 0,2 = 60000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_{Г} \cdot (1 + К_{З}) = 60000 \cdot (1 + 0.2) = 72000 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 72000 \cdot 0,22 = 15840 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 10000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.4 \cdot 0.25 \cdot 10000 = 3500 \quad грн.,$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.4 \cdot 10000 \cdot 0.08 = 1120 \quad грн.,$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 16) \cdot 8 \cdot 0.35 =$$

$$= 627,2 \text{ години},$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 627,2 \cdot 0,4 \cdot 0,3 \cdot 4,87 = 366,53 \quad грн.,$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 10000 \cdot 0,67 = 6700 \quad грн.$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н, \qquad (4.17)$$

$С_{ЕКС} = 72000 + 15840 + 3500 + 1120 + 366,53 + 6700 = 99526,53$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 99526,53 / 627,2 = 158,7 \quad грн/год.$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М-Г} \cdot Т, \qquad (4.18)$$

I. $С_М = 158,7 \cdot 852 = 135212,4$ грн.

II. $С_М = 158,7 \cdot 868,88 = 137891,26$ грн.

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67, \qquad (4.19)$$

I. $С_Н = 152143,35 \cdot 0,67 = 101936$ грн.

II. $С_Н = 155157,64 \cdot 0,67 = 103995,6$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_{М} + С_{Н}, \quad (4.20)$$

I. $С_{ПП}$ = 152143,35+ 33471,54 + 135212,4 +101936 = 422763,29

II. $С_{ПП}$ = 155157,64 + 34134,68 + 137891,26+ 103995,6 = 431179,18 грн

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР j} = К_{Кj} / С_{Фj}, \quad (4.21)$$

$$К_{ТЕР1} = 21,16 / 422763,29 = 5,005 \cdot 10^{-5},$$

$$К_{ТЕР2} = 16,26 / 431179,18 = 3,771 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1}=5,005 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 5,005 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

– вибір мови програмування – Python,

– реалізація важливої постановки з допомогою вбудованих функцій,
– використання стандартного інтерфейсу для побудови значень.

Даний варіант виконання програмного комплексу є оптимальним для заданої задачі.

4.8 Висновки до розділу 4

В даній частині було проведено повний функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту.

В результаті виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, було визначено та проведено оцінку основних функцій програмного продукту, а також знайдено параметри, які його характеризують.

На основі аналізу вибрано варіант реалізації програмного продукту.

ВИСНОВКИ

У даній роботі було проведено дослідження бізнес-процесів у сфері ресторанного бізнесу та застосування методів машинного навчання для їх моделювання.

У першому розділі були розглянуті основні поняття та елементи бізнес-процесів, а також існуючі методи їх моделювання. Також проведений аналіз типових бізнес-процесів у ресторанному бізнесі.

У другому розділі були представлені методи машинного навчання, які можуть бути використані для моделювання бізнес-процесів. Для оцінки якості моделей запропоновано використовувати метрики R-squared, MAE та MSE.

У третьому розділі було здійснено прогнозування кількості відвідувачів ресторанів за допомогою різних моделей. Найточнішими виявились LSTM мережі, а найменш точною – лінійна регресія. Запропоновано покращення моделей шляхом збирання додаткових даних та аналізу та усунення аномалій.

У четвертому розділі проведено функціонально-вартісний аналіз програмного продукту та знайдено оцінку його основних функцій. Також обрано варіант реалізації програмного продукту.

Загальною метою роботи було виявлення способів моделювання бізнес-процесів у ресторанному бізнесі з використанням методів машинного навчання та функціонально-вартісного аналізу програмного продукту.

Результати дослідження можуть бути використані для покращення управління ресторанними бізнес-процесами, прогнозування кількості відвідувачів та стратегічного планування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. В.В.Нетепчук. Управління бізнес-процесами: Навч. посібник. – Рівне: НУВГП, 2014. – 158 с.

2. Г. Ситник. Класифікація бізнес-процесів підприємства торгівлі на основі процесного підходу // Схід. — 2012. — № 5 (119). — С. 54-61. — Бібліогр.: 14 назв. — укр.

3. Laguna, M., Marklund, J. Business Process Modeling, Simulation and Design (2nd ed.). Chapman and Hall/CRC, 2013. https://doi.org/10.1201/b14763

4. Данченко О.Б. Практичні аспекти реінжинірингу бізнес-процесів – К.: Університет економіки та права «КРОК», 2017. – 238 с.

5. Корзаченко О. В. Моделювання бізнес-процесів підприємств: методології, підходи та методи // Науковий вісник Херсонського державного університету. Сер. : Економічні науки. - 2015. - Вип. 11(1). - С. 171-175.

6. О. Г. Додонов, В. Р. Сенченко, О. В. Коваль, А. В. Бойченко. Моделювання сценаріїв аналітичної діяльності на основі нотації BPMN та OWL // Реєстрація, зберігання і обробка даних. - 2020. - Т. 22, № 1. - С. 31–48.

7. Stephen A. White and Derek Miers. BPMN Modeling and Reference Guide. Future Strategies Inc., 2008.

8. UML для бізнес-моделювання: для чого потрібні діаграми процесів [URL]: https://evergreens.com.ua/ua/articles/uml-diagrams.html

9. Reiter, Detlev. The Monte Carlo Method, an Introduction, 2007. 10.1007/978-3-540-74686-7_3.

10. Методологія функціонального моделювання SADT. Склад функціональної моделі, ієрархія діаграм, типи зв'язку між функціями [URL]: https://studfile.net/preview/14510912/

11. Gülden Kaya Uyanık, Neşe Güler - A Study on Multiple Linear Regression Analysis - Procedia - Social and Behavioral Sciences - Vol.106 - pp.234 - DOI : 10.1016/j.sbspro.2013.12.027 - december - 2013

12. Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001).

13. Jamal Fattap, Latifa Ezzine1, Zineb Aman2 , Haj El Moussami2, and Abdeslam Lachhab1, Forecasting of demand using ARIMA model, International Journal of Engineering Business Management Volume 10: 19 Âª The Author(s) 2018.

14. Insights into LSTM architecture [URL]: https://thorirmar.com/post/insight_into_lstm/

15. Recruit Restaurant Visitor Forecasting. Data [URL]: https://www.kaggle.com/competitions/recruit-restaurant-visitor-forecasting

16. Summary of rules for identifying ARIMA models [URL]: http://people.duke.edu/~rnau/arimrule.htm

ДОДАТОК А. Лістинг програмного модуля

```
import numpy as np
import pandas as pd
import seaborn as sns
from IPython.display import display
import matplotlib.pyplot as plt
import matplotlib
import scipy.stats as stats
from datetime import date
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
import ydata_profiling
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.ensemble import RandomForestRegressor
from statsmodels.tsa.arima.model import ARIMA
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import statsmodels.formula.api as smf
import statsmodels.api as sm
import warnings
warnings.filterwarnings('ignore')
air_reserve = pd.read_csv('recruit-restaurant-visitor-
forecasting/air_reserve.csv',parse_dates=['visit_datetime','reserve_datetime'])
air_store_info = pd.read_csv('recruit-restaurant-visitor-
forecasting/air_store_info.csv')
air_visit = pd.read_csv('recruit-restaurant-visitor-
forecasting/air_visit_data.csv',parse_dates=['visit_date'])
```

```
date_info = pd.read_csv('recruit-restaurant-visitor-forecasting/date_info.csv',
parse_dates=['calendar_date'])
hpg_reserve = pd.read_csv('recruit-restaurant-visitor-forecasting/hpg_reserve.csv',
parse_dates=['visit_datetime', 'reserve_datetime'])
hpg_store_info = pd.read_csv('recruit-restaurant-visitor-
forecasting/hpg_store_info.csv')
sample_submission = pd.read_csv('recruit-restaurant-visitor-
forecasting/sample_submission.csv')
store_id_relation = pd.read_csv('recruit-restaurant-visitor-
forecasting/store_id_relation.csv')
air_reserve['visit_year'] = air_reserve['visit_datetime'].dt.year
air_reserve['visit_month'] = air_reserve['visit_datetime'].dt.month
air_reserve['visit_day'] = air_reserve['visit_datetime'].dt.day
air_reserve['reserve_year'] = air_reserve['reserve_datetime'].dt.year
air_reserve['reserve_month'] = air_reserve['reserve_datetime'].dt.month
air_reserve['reserve_day'] = air_reserve['reserve_datetime'].dt.day
air_reserve.drop(['visit_datetime','reserve_datetime'], axis=1, inplace=True)
air_visit['visit_day'] = air_visit['visit_date'].dt.day
air_visit['visit_month'] = air_visit['visit_date'].dt.month
air_visit['visit_year'] = air_visit['visit_date'].dt.year
air_visit.drop(['visit_date'], axis=1, inplace=True)
date_info['calendar_day'] = date_info['calendar_date'].dt.day
date_info['calendar_month'] = date_info['calendar_date'].dt.month
date_info['calendar_year'] = date_info['calendar_date'].dt.year
date_info.drop(['calendar_date'], axis=1, inplace=True)
hpg_reserve['visit_year'] = hpg_reserve['visit_datetime'].dt.year
hpg_reserve['visit_month'] = hpg_reserve['visit_datetime'].dt.month
hpg_reserve['visit_day'] = hpg_reserve['visit_datetime'].dt.day
hpg_reserve['reserve_year'] = hpg_reserve['reserve_datetime'].dt.year
hpg_reserve['reserve_month'] = hpg_reserve['reserve_datetime'].dt.month
hpg_reserve['reserve_day'] = hpg_reserve['reserve_datetime'].dt.day
hpg_reserve.drop(['visit_datetime','reserve_datetime'], axis=1, inplace=True)
hpg_reserve = hpg_reserve.groupby(['hpg_store_id', 'visit_year', 'visit_month',\

'visit_day','reserve_year','reserve_month','reserve_day'], as_index=False).sum()
hpg_reserve = pd.merge(hpg_reserve, store_id_relation, on='hpg_store_id',
how='inner')
hpg_reserve.drop(['hpg_store_id'], axis=1, inplace=True)
air_reserve = pd.concat([air_reserve, hpg_reserve])
```

```
air_reserve = air_reserve.groupby(['air_store_id', 'visit_year', 'visit_month',
'visit_day'], as_index=False).sum()
air_reserve.drop(['reserve_day', 'reserve_month', 'reserve_year'], axis=1,
inplace=True)
air_reserve = pd.merge(air_reserve, date_info, left_on=['visit_year', 'visit_month',
'visit_day'],
                       right_on=['calendar_year', 'calendar_month', 'calendar_day'],
how='left')
air_reserve.drop(['calendar_year', 'calendar_month', 'calendar_day'], axis=1,
inplace=True)
air_reserve = pd.merge(air_reserve, air_store_info, on='air_store_id', how='left')
resto = pd.merge(air_reserve, air_visit, on=['air_store_id', 'visit_year',
'visit_month', 'visit_day'], how='left')
weekday_encoding = {
    'Monday': 1,
    'Tuesday': 2,
    'Wednesday': 4,
    'Thursday': 3,
    'Friday': 5,
    'Saturday': 7,
    'Sunday': 6
}
resto2['weekday_encoded'] = resto2['weekday'].map(weekday_encoding)
#data['weekday_encoded'] = data['weekday'].map(weekday_encoding)
import category_encoders as ce
encoder = ce.CatBoostEncoder(cols=['area'])
y = resto2['visitors']
resto2['area_encoded'] = encoder.fit_transform(resto2['area'], y)
resto2=resto2.drop('area', axis=1)
X=data.drop(columns=[ 'visitors'])
y = data['visitors']  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
X_train = sm.add_constant(X_train)
model = sm.OLS(y_train, X_train)
result = model.fit()
print(result.summary())
X_test = sm.add_constant(X_test)
y_pred = result.predict(X_test)
model = LinearRegression()
```

```
predictions = result.predict(X_test)
errors = abs(predictions - y_test)
print('Average absolute error:', round(np.mean(errors), 2), 'degrees.')
mape = 100 * (errors / y_test)
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error (MAE):", mae)
r2 = r2_score(y_test, y_pred)
print("R-squared (R^2) score:", r2)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)
train_sizes, train_scores, test_scores = learning_curve(model, X_train, y_train,
cv=5)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)
plt.figure(figsize=(10, 6))
plt.plot(train_sizes, train_scores_mean, 'o-', label='Training score')
plt.plot(train_sizes, test_scores_mean, 'o-', label='Cross-validation score')
plt.fill_between(train_sizes, train_scores_mean - train_scores_std, train_scores_mean
+ train_scores_std, alpha=0.3)
plt.fill_between(train_sizes, test_scores_mean - test_scores_std, test_scores_mean +
test_scores_std, alpha=0.3)
plt.xlabel('Training set size')
plt.ylabel('Score')
plt.title('Learning Curve')
plt.legend(loc='best')
plt.grid(True)
plt.show()
X = data.drop(['visitors'], axis=1)  # Features
y = data['visitors']  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=13)

rf_regressor =
RandomForestRegressor(min_samples_split=3,min_samples_leaf=3,max_depth=12,
max_features=8, n_estimators=150,random_state=13)
```

```
rf_regressor.fit(X_train, y_train)
y_pred = rf_regressor.predict(X_test)
predictions = rf_regressor.predict(X_test)
train_sizes, train_scores, test_scores = learning_curve(rf_regressor, X, y, cv=10)
errors = abs(predictions - y_test)
print('Average absolute error:', round(np.mean(errors), 2), 'degrees.')
mape = 100 * (errors / y_test)
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("R-squared:", r2)
print("Mean Absolute Error:", mae)
train_scores_mean = np.mean(train_scores, axis=1)
train_scores_std = np.std(train_scores, axis=1)
test_scores_mean = np.mean(test_scores, axis=1)
test_scores_std = np.std(test_scores, axis=1)
plt.figure(figsize=(10, 6))
plt.title("Learning Curve")
plt.xlabel("Training Set Size")
plt.ylabel("R-squared")
plt.grid()
plt.fill_between(
    train_sizes, train_scores_mean - train_scores_std, train_scores_mean +
train_scores_std, alpha=0.1
)
plt.fill_between(
    train_sizes, test_scores_mean - test_scores_std, test_scores_mean +
test_scores_std, alpha=0.1, color="r"
)
plt.plot(train_sizes, train_scores_mean, 'o-', color="b", label="Training Score")
plt.plot(train_sizes, test_scores_mean, 'o-', color="r", label="Cross-Validation
Score")
plt.legend(loc="best")
plt.show()
importances = list(rf_regressor.feature_importances_)
feature_importances = [(feature, round(importance, 2)) for feature, importance in
zip(X.columns, importances)]
```

```
feature_importances = sorted(feature_importances, key=lambda x: x[1], reverse=True)

[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in
feature_importances]

x_values = list(range(len(importances)))
plt.bar(x_values, importances, orientation='vertical', color='r', edgecolor='k',
linewidth=1.2)
plt.xticks(x_values, X.columns, rotation='vertical')
plt.ylabel('Importance')
plt.xlabel('Variable')
plt.title('Variable Importances')
plt.show()
sorted_importances = [importance[1] for importance in feature_importances]
sorted_features = [importance[0] for importance in feature_importances]

cumulative_importances = np.cumsum(sorted_importances)
plt.plot(x_values, cumulative_importances, 'g-')
plt.hlines(y=0.85, xmin=0, xmax=len(sorted_importances), color='r',
linestyles='dashed')
plt.xticks(x_values, sorted_features, rotation='vertical')
plt.xlabel('Variable')
plt.ylabel('Cumulative Importance')
plt.title('Cumulative Importances')
plt.show()
cumulative_importances = np.cumsum(sorted_importances)
threshold_index = np.where(cumulative_importances > 0.85)[0][0] + 1
print('Number of features for 85% importance:', threshold_index)
important_feature_names = [importance[0] for importance in
feature_importances[:threshold_index]]
important_indices = [X.columns.get_loc(feature) for feature in
important_feature_names]
important_train_features = X_train.iloc[:, important_indices]
important_test_features = X_test.iloc[:, important_indices]
print('Important train features shape:', important_train_features.shape)
print('Important test features shape:', important_test_features.shape)
print('Features:', important_test_features.head())
def test_stationarity(time_series):
    time_series = time_series['visitors']
    moving_avg = time_series.rolling(window=12).mean()
```

```
    moving_std = time_series.rolling(window=12).std()

    # Plot original time series, rolling mean, and rolling standard deviation
    orig = plt.plot(time_series, color='black', label='Original')
    mean = plt.plot(moving_avg, color='red', label='Mean')
    std = plt.plot(moving_std, color='blue', label='Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean and STD')
    plt.show(block=False)
    print("Results of Dickey-Fuller Test:")
    dftest = adfuller(time_series, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic', 'p-value', '#Lags
Used', 'Number of Observations Used'])
    for key, value in dftest[4].items():
        dfoutput['Critical Value (%s)' % key] = value
    print(dfoutput)

test_stationarity(air_visit)
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(air_visit.visitors_mean, lags=40, alpha=.05, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(air_visit.visitors_mean, lags=40, alpha=.05, ax=ax2)
print("ACF and PACF of the visit mean:")
def mean_squared_log_error(y_pred, y_true, **dict):
    '''Assume y_true starts earlier than y_pred, y_true is NaN free, and NaN in
y_pred are only in the beginning'''
    ind_after_NaN = y_pred.first_valid_index()
    if y_true.index[0] > y_pred.index[0]:
        return "Check indices of prediction and true value"
    ind_1st_common = y_true.index[y_true.index == y_pred.index[0]]
    ind_start = max(ind_after_NaN, ind_1st_common)
    ind_end = y_true.index[-1]
    return mean_squared_error(np.log(y_true[ind_start:ind_end] + 1),
np.log(y_pred[ind_start:ind_end] + 1)) ** 0.5

def plotSARIMA(labels, pred):
    fig = plt.figure(figsize=(12, 8))
    layout = (2, 2)
    ax1 = plt.subplot2grid(layout, (0, 0), colspan=2)
```

```
    ax3 = plt.subplot2grid(layout, (1, 0))
    ax4 = plt.subplot2grid(layout, (1, 1))
    labels.plot(ax=ax1)
    pred.plot(ax=ax1, title='MSLE: %.4f' % mean_squared_log_error(pred, labels))
    ax3 = sm.graphics.tsa.plot_acf(results.resid, lags=40, alpha=.05, ax=ax3,
title="ACF of residuals")
    ax4 = sm.graphics.tsa.plot_pacf(results.resid, lags=40, alpha=.05, ax=ax4,
title="PACF of residuals")
    plt.tight_layout()
    print("ACF and PACF of residuals")
def autoSARIMA(endog, exog=None, date_train_end=None, pred_days=[-12, 12],
verbose=True, ranges=(slice(1, 3), slice(0, 1), slice(1, 3), slice(0, 2), slice(1,
2), slice(1, 2), slice(7, 8))):
    # Instantiate my version of the grid with parameters and scores
    global grid
    grid = []
    # Get indices up to which you do train and prediction
    if date_train_end is None:
        ind_train = endog.index[-1]
    else:
        ind_train = np.where(endog.index == date_train_end)[0][0]
    # Brute optimization
    results_brute = brute(runSARIMA, ranges=ranges, args=(endog, exog, (ind_train,
pred_days),), full_output=True, finish=None)
    # First coefficients run two times for some reason or another
    del grid[0]
    # Print/Plot results
    if verbose:
        print("Best parameters: {}".format([int(p) for p in results_brute[0]]))
        print("Best score:          {}".format(results_brute[1]))
        gr = plotautoSARIMA(results_brute, verbose)
    return results_brute, gr

def plotautoSARIMA(results_brute, verbose=True):
    # Print/Plot results
    if not verbose:
        return None
    # Plot scores by parameter values
    gr = pd.DataFrame({'params': [''.join(str(n) for n in g[0]) for g in grid],
'score': [row[1] for row in grid], 'aic': [row[2] for row in grid]})
```

```
    print("All parameters and scores: \n")
    print(gr.head(1000).to_string())
    ax1 = gr.plot('params', 'score', rot=90, grid=True, figsize=(15, 4))
    ax2 = gr.plot('params', 'aic', rot=90, secondary_y=True, ax=ax1)
    ax1.set_ylabel('Score')
    ax2.set_ylabel('AIC')
    plt.xticks(range(len(gr)), gr.params, rotation=90)
    return gr

def runSARIMA(coeffs, *args):
    endog = args[0]
    exog = args[1]
    # Process the row indices for training and prediction
    ind_train = args[2][0]
    pred_days = args[2][1]
    ind_pred = [len(endog) + pred_days[0], len(endog) + pred_days[1]]
    if ind_pred[0] > ind_train:
        # ind_pred[0]=ind_train
        raise ValueError('Make sure prediction bounds begin at least at len(endog):
pred_days[0] must be <= %i ' % (ind_train - len(endog)))
    exog_train, exog_pred, start_params = None, None, list()
    if exog is not None:
        if ind_pred[1] > len(exog):
            raise ValueError('Make sure prediction bounds end  <= len(exog):
pred_days[1] must be <= %i ' % (len(exog) - len(endog)))
        exog_train = exog[:ind_train]
        exog_cols = 1 if len(exog.shape) == 1 else exog.shape[1]
        start_params.extend(0.1 * np.ones(exog_cols - 1))
        exog_pred = exog[ind_pred[0] - 1:ind_pred[1]]
        exog_pred = pd.DataFrame(exog_pred)
    # Get the hyperparameters
    order = coeffs[0:3].tolist()
    seasonal_order = coeffs[3:7].tolist()
    trend = 'c' if (order[1] == 0) else 'n'
    # Train SARIMA and fit it on data, predict to get scores
    try:
        mod = sm.tsa.statespace.SARIMAX(endog[:ind_train], exog_train, trend=trend,
order=order, seasonal_order=seasonal_order)
        start_params.extend(0.1 * np.ones(len(mod.params_complete)))
        fit = mod.fit(start_params=start_params)
```

```
        pred = fit.predict(start=ind_pred[0], end=ind_pred[1], exog=exog_pred)
        aic = fit.aic
        score = mean_squared_log_error(pred[:-pred_days[0]], endog[ind_pred[0]:])
        if np.isnan(aic):
            aic, score = np.inf, np.inf
    except:  # Tip: Try to set starting parameters in .fit()
        import sys
        print("Error:", sys.exc_info())
        print("{},{},'{}', len(start_params)={}\n".format(coeffs[0:3], coeffs[3:],
trend, len(start_params)))
        aic, score = np.inf, np.inf
    # Sorry, but I don't like the grid in the output of brute results_brute[2]
    global grid
    grid.append([coeffs, score, aic])
    return score
mod = sm.tsa.statespace.SARIMAX(resto_only.visitors[:383], trend='c', order=(1, 0,
2), seasonal_order=(0, 1, 1, 7))
results = mod.fit()
pred = results.predict(start=383, end=478)
actual = air_visit.visitors_mean[383:478]
pred = pred[:actual.shape[0]]  # Match the length of pred with actual
mse = mean_squared_error(actual, pred)
mae = np.mean(np.abs(actual - pred))
mape = np.mean(np.abs((actual - pred) / actual)) * 100
r2 = r2_score(actual, pred)
print("R-squared:", r2)
print("Mean Squared Error (MSE):", mse)
print("Mean Absolute Error (MAE):", mae)
print("Mean Absolute Percentage Error (MAPE):", mape)
plotSARIMAX(air_visit.visitors_mean, pred)
timeseries = air_visit[["visitors_mean"]].values.astype('float32')
# train-test split for time series
train_size = int(len(timeseries) * 0.8)
test_size = len(timeseries) - train_size
train, test = timeseries[:train_size], timeseries[train_size:]

def create_dataset(dataset, lookback):
    """Transform a time series into a prediction dataset

    Args:
```

```
        dataset: A numpy array of time series, first dimension is the time steps
        lookback: Size of window for prediction
    """
    X, y = [], []
    for i in range(len(dataset) - lookback):
        feature = dataset[i:i+lookback]
        target = dataset[i+lookback]
        X.append(feature)
        y.append(target)
    return torch.tensor(X), torch.tensor(y)

lookback = 7  # Initial lookback value
X_train, y_train = create_dataset(train, lookback=lookback)
X_test, y_test = create_dataset(test, lookback=lookback)

class LSTMNetwork(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, num_layers):
        super(LSTMNetwork, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers=num_layers,
batch_first=True)
        self.linear = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        x, _ = self.lstm(x)
        x = self.linear(x[:, -1, :])
        return x

input_size = 1
hidden_size = 50
output_size = 1
num_layers = 2

model = LSTMNetwork(input_size, hidden_size, output_size, num_layers)
optimizer = optim.Adam(model.parameters())
loss_fn = nn.MSELoss()
loader = data.DataLoader(data.TensorDataset(X_train, y_train), shuffle=True,
batch_size=8)

n_epochs = 100
for epoch in range(n_epochs):
```

```
    model.train()
    for X_batch, y_batch in loader:
        y_pred = model(X_batch)
        loss = loss_fn(y_pred, y_batch)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
    if epoch % 100 != 0:
        continue

    model.eval()
    with torch.no_grad():
        y_pred_train = model(X_train)
        train_rmse = torch.sqrt(loss_fn(y_pred_train, y_train))
        y_pred_test = model(X_test)
        test_rmse = torch.sqrt(loss_fn(y_pred_test, y_test))

    print("Epoch %d: train RMSE %.4f, test RMSE %.4f" % (epoch, train_rmse,
test_rmse))



with torch.no_grad():
    # Generate predictions for train and test data
    y_train_pred = model(X_train)
    y_test_pred = model(X_test)
y_train_true = y_train.view(-1).numpy()
y_train_pred = y_train_pred.view(-1).detach().numpy()
y_test_true = y_test.view(-1).numpy()
y_test_pred = y_test_pred.view(-1).detach().numpy()

def calculate_mape(y_true, y_pred):
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

train_mape = calculate_mape(y_train_true, y_train_pred)
test_mape = calculate_mape(y_test_true, y_test_pred)
train_mae = mean_absolute_error(y_train_true, y_train_pred)
test_mae = mean_absolute_error(y_test_true, y_test_pred)

# Calculate MSE
train_mse = mean_squared_error(y_train_true, y_train_pred)
```

```
test_mse = mean_squared_error(y_test_true, y_test_pred)
train_r2 = r2_score(y_train_true, y_train_pred)
test_r2 = r2_score(y_test_true, y_test_pred)

print("Train R-squared: %.4f" % train_r2)
print("Test R-squared: %.4f" % test_r2)
print("Train MAPE: %.4f" % train_mape)
print("Test MAPE: %.4f" % test_mape)
print("Train MAE: %.4f" % train_mae)
print("Test MAE: %.4f" % test_mae)
print("Train MSE: %.4f" % train_mse)
print("Test MSE: %.4f" % test_mse)
with torch.no_grad():
    # Shift train predictions for plotting
    train_plot = np.ones_like(timeseries) * np.nan
    train_plot[lookback:train_size] = y_train_pred.reshape(train_size - lookback, -1)
    test_plot = np.ones_like(timeseries) * np.nan
    test_plot[train_size + lookback:len(timeseries)] =
y_test_pred.reshape(len(timeseries) - train_size - lookback, -1)


plt.figure(figsize=(20, 6))
plt.plot(timeseries, c='b', label='Original Data')
plt.plot(train_plot, c='r', label='Train Predictions')
plt.plot(test_plot, c='g', label='Test Predictions')
plt.legend()
plt.show()
timeseries = air_visit[["visitors_mean"]].values.astype('float32')
# train-test split for time series
train_size = int(len(timeseries) * 0.8)
test_size = len(timeseries) - train_size
train, test = timeseries[:train_size], timeseries[train_size:]

def create_dataset(dataset, lookback):
    """Transform a time series into a prediction dataset

    Args:
        dataset: A numpy array of time series, first dimension is the time steps
        lookback: Size of window for prediction
    """
```

```
    X, y = [], []
    for i in range(len(dataset) - lookback):
        feature = dataset[i:i+lookback]
        target = dataset[i+lookback]
        X.append(feature)
        y.append(target)
    return torch.tensor(X), torch.tensor(y)

lookback = 7  # Initial lookback value
X_train, y_train = create_dataset(train, lookback=lookback)
X_test, y_test = create_dataset(test, lookback=lookback)

class LSTMNetwork(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, num_layers):
        super(LSTMNetwork, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers=num_layers,
batch_first=True)
        self.linear = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        x, _ = self.lstm(x)
        x = self.linear(x[:, -1, :])
        return x

input_size = 1
hidden_size = 50
output_size = 1
num_layers = 2
model = LSTMNetwork(input_size, hidden_size, output_size, num_layers)
optimizer = optim.Adam(model.parameters())
loss_fn = nn.MSELoss()
loader = data.DataLoader(data.TensorDataset(X_train, y_train), shuffle=True,
batch_size=8)
n_epochs = 500
train_losses = []
test_losses = []
for epoch in range(n_epochs):
    model.train()
    for X_batch, y_batch in loader:
        y_pred = model(X_batch)
```

```
            loss = loss_fn(y_pred, y_batch)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

        if epoch % 100 != 0:
            continue

        model.eval()
        with torch.no_grad():
            y_pred_train = model(X_train)
            train_loss = loss_fn(y_pred_train, y_train).item()
            y_pred_test = model(X_test)
            test_loss = loss_fn(y_pred_test, y_test).item()

        train_losses.append(train_loss)
        test_losses.append(test_loss)
        print("Epoch %d: train loss %.4f, test loss %.4f" % (epoch, train_loss,
test_loss))

# Plot the epoch vs. loss graph
plt.plot(range(0, n_epochs, 100), train_losses, label='Train Loss')
plt.plot(range(0, n_epochs, 100), test_losses, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

ДОДАТОК Б. Презентація

ФОРМАЛІЗАЦІЯ ТА МОДЕЛЮВАННЯ БІЗНЕС-ПРОЦЕСІВ В ЗАДАЧАХ УПРАВЛІННЯ СКЛАДНИМИ ОБ‘ЄКТАМИ

Виконала: Уманець Марія
Керівник: доцент, к.ф.м.н. Яковлева Алла Петрівна

АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ

Ефективне управління бізнес-процесами є критичним фактором для досягнення конкурентної переваги в сучасному світі, де швидкі зміни технологій, ринкових умов та клієнтських вимог вимагають гнучкості та швидкого реагування.

Впровадження методів машинного навчання у бізнес-сфері дозволяє покращити точність прогнозування, зрозуміти складні залежності та тренди, виявити невидимі закономірності в даних.

ОБ‘ЄКТ ДОСЛІДЖЕННЯ

Методи моделювання та формалізації бізнес-процесів

ПРЕДМЕТ ДОСЛІДЖЕННЯ

Моделі машинного навчання для прогнозування в сфері бізнесу

МЕТА ДОСЛІДЖЕННЯ

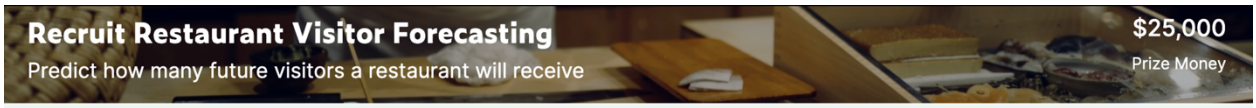
Реалізація методів машинного навчання для прогнозування кількості відвідувачів на основі даних з японських ресторанів

ПОСТАНОВКА ЗАДАЧІ

1. Огляд існуючих методів моделювання та формалізації бізнес-процесів
2. Реалізація моделей машинного навчання для прогнозування
3. Аналіз результатів та висновки

СТРУКТУРА ПРОГРАМИ

1. Опис вхідних даних
2. Попередня обробка та візуалізація даних
3. Побудова моделей
4. Порівняльний аналіз результатів та ідеї для подальших досліджень



ВХІДНІ ДАНІ

Доступ до ключових наборів даних, що може зробити можливим автоматизоване прогнозування майбутніх клієнтів. Зокрема, Recruit Holdings володіє Hot Pepper Gourmet (сервіс огляду ресторанів), AirREGI (сервіс ресторанних точок продажу) і Restaurant Board

- З двох окремих сайтів: Hot Pepper Gourmet (hpg), тут користувачі можуть шукати ресторани, а також робити бронювання онлайн, AirREGI / Restaurant Board
- Місце розташування : Японія

ПОПЕРЕДНЯ ОБРОБКА ДАНИХ

1. Злиття таблиць

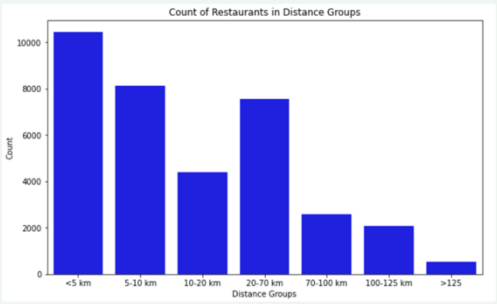
	id	year	month	day	reserv	weekday	isholiday	genre	area	latitude	longitude	visitors
1478	air_08ef81d5b7a0d13f	2016	10	15	3	Saturday	0	Italian/French	Tōkyō-to Chūō-ku Tsukiji	35.670651	139.771861	14.0
4175	air_c88467d88b2c8ecd	2017	4	22	14	Saturday	0	Italian/French	Tōkyō-to Chiyoda-ku Kudanminami	35.694003	139.753595	21.0
5280	air_26f10355d9b4d82a	2016	11	23	19	Wednesday	1	Other	Ōsaka-fu Ōsaka-shi Kyōmachibori	34.687697	135.495413	36.0
3640	air_c6a164dd4060e960	2017	3	30	4	Thursday	0	Dining bar	Ōsaka-fu Ōsaka-shi Ōgimachi	34.705362	135.510025	6.0
1630	air_cbe867adcf44e14f	2016	3	22	2	Tuesday	0	Izakaya	Fukuoka-ken Kitakyūshū-shi None	33.866465	130.764923	NaN

2. Обробка пропущених значень

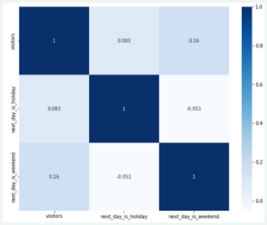
Dataset statistics	
Number of variables	12
Number of observations	42193
Missing cells	6495
Missing cells (%)	1.3%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	4.2 MiB
Average record size in memory	104.0 B

Variable types	
Categorical	6
Numeric	6

3. Додавання нових ознак




```
weekday
Monday      20.727648
Tuesday     21.655753
Thursday    22.820548
Wednesday   23.945107
Friday      30.095985
Sunday      30.216188
Saturday    33.595937
```



МЕТРИКИ ОЦІНКИ ЯКОСТІ МОДЕЛІ

Коефіцієнт детермінації R-квадрат

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

Середня квадратична помилка

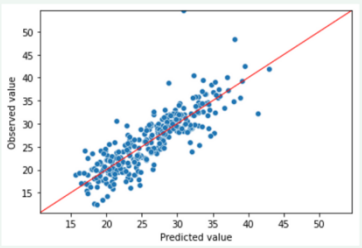
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

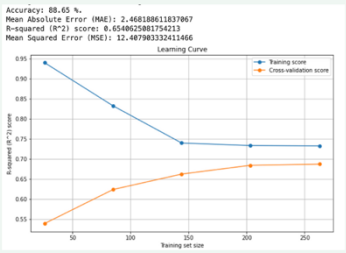
Середня абсолютна помилка

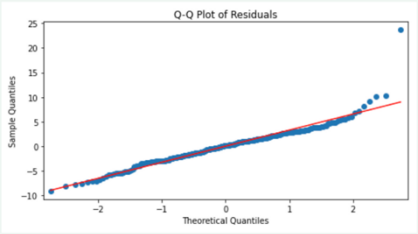
$$\mathrm{MAE} = \frac{1}{\mathrm{n}}\sum_{\mathrm{i}=1}^{\mathrm{n}}\left|y_i - \hat{y}_i\right|$$

ЛІНІЙНА РЕГРЕСІЯ

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$$



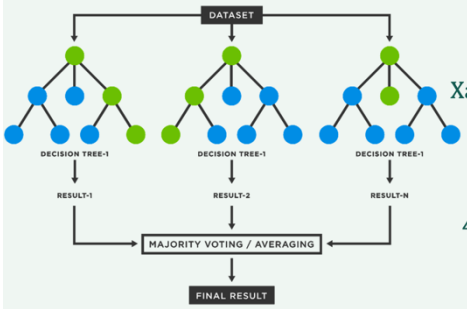




Основні компоненти лінійної регресії:

1. Залежна змінна
2. Незалежні змінні
3. Лінійна модель
4. Оцінка коефіцієнтів
5. Оцінка якості моделі
6. Аналіз помилок

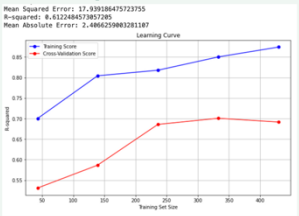
ВИПАДКОВИЙ ЛІС

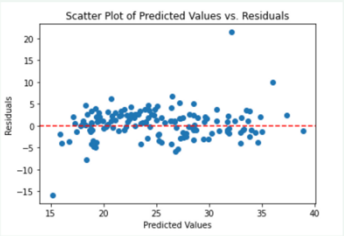


Характеристики випадкового лісу

1. Ансамбль дерев рішень
2. Випадковість
3. Важливість ознак
4. Обробка пропущених значень
5. Підбір параметрів

Криві навчання





Для запобігання перенавчанню можна використовувати наступні підходи:

- зменшення кількості дерев
- обмеження глибини дерев
- крос-валідація
- збільшення набору даних

ARIMA

$$SARIMA\underbrace{(p,d,q)}_{non-seasonal}\underbrace{(P,D,Q)_m}_{seasonal}$$

є розширенням моделі ARIMA, яка враховує сезонні закономірності в даних часових рядів

Кроки виконання
1. Перевірка на стаціонарність
2. Вибір параметрів
3. Побудова моделі
4. Діагностика моделі

AR

$$Y_t = \alpha_1 Y_{t-1} + \varepsilon_t$$

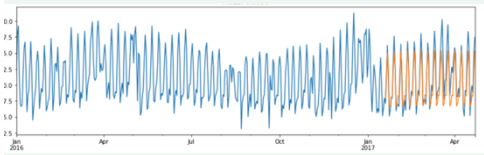
I

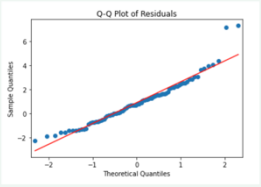
$$Y_t = Y_{t-1} + \varepsilon_t$$

MA

$$x_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

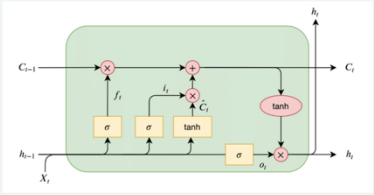
Отриманий результат





LSTM

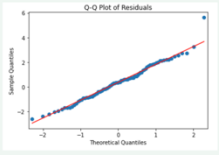
Структура комірки



Звичайний блок LSTM складається з комірки, вхідного вентиля, вихідного вентиля та пропускного вентиля.

Основні задачі які вирішує LSTM:
1. Прогнозування часових рядів
2. Машинний переклад
3. Розпізнання мови

Отриманий результат



МОДЕЛІ

ЛІНІЙНА РЕГРЕСІЯ

Застосування методу найменших квадратів та регуляризації L1
Отримано коефіцієнт детермінації 0.65

ВИПАДКОВИЙ ЛІС

Застосування GridSearch та Feature Selection
Отримано коефіцієнт детермінації 0.65

ARIMA

Застосування розширеної моделі SARIMA для виявлення сезонності
Отримано коефіцієнт детермінації 0.77

LSTM

Застосування двошарової LSTM мережі
Отримано коефіцієнт детермінації 0.87

ОТРИМАНІ РЕЗУЛЬТАТИ РОБОТИ МОДЕЛЕЙ

	R-squared	MAE	MSE
Лінійна регресія	0.65	2.47	12.4
Випадковий ліс	0.65	2.4	17.93
SARIMA	0.77	1.46	3.81
LSTM	0.87	1.16	2.17

ПОДАЛЬШІ ДОСЛІДЖЕННЯ

1. Вдосконалення побудованих моделей
2. Застосування інших підходів
3. Збирання додаткових даних (погодні умови, події)
4. Аналіз та усунення аномалій системи збору даних

ДЯКУЮ ЗА УВАГУ!