UTILIZING BIG DATA TOOLS FOR AGODA'S CUSTOMER BOOKING BEHAVIOR ANALYSIS AND DYNAMIC PRICING OPTIMIZATION

Nur Hidayah binti Ahmad Shafii, Choon Yue Hua, Then Dao Qing, Loh Bi Jia, Boaz Yi Heng Chung, Diva Alifta Chandra, Yong Ting Kang, and Zeng Yanting

Faculty of Computer Science and Information Technology, Universiti Malaya
50603 Kuala Lumpur, Federal Territory of Kuala Lumpur, Malaysia

22120931@siswa.um.edu.my*, 17152027@siswa365.um.edu.my, 23057608@siswa365.um.edu.my, 23078886@siswa365.um.edu.my, 23059592@siswa.um.edu.my, 23069683@siswa.um.edu.my, 23083416@siswa.um.edu.my and s2168467@siswa.um.edu.my

*Corresponding Author
Received: day month 202x, Revised: day month 202x, Accepted: day month 202x
Published online: day month 202x

Abstract:

In the competitive travel and hospitality industry, understanding customer booking behavior and optimizing pricing strategies are critical for enhancing business performance. This project leverages big data tools to analyze Agoda's customer booking data, uncovering behavioral patterns and optimizing pricing strategies dynamically. The primary objectives are to analyze booking behaviors based on demographic and booking history, and to identify patterns that can inform dynamic pricing strategies. The process includes preprocessing data with Pig, querying with Hive and MapReduce, performing customer segmentation through clustering with Apache Spark, and developing a regression model for price optimization using Apache Spark. Data storage is managed through HDFS, and visualization is achieved using HBase and PowerBI. Additional tools like Python were employed to enhance preprocessing, while PowerBI supported advanced visualization. A comparison between Hive and MapReduce revealed that Hive outperforms MapReduce in execution speed and reduces complexity for analysis tasks. Clustering analysis identified four distinct customer segments characterized by travel frequency and average travel distance. Additionally, the Random Forest model exhibited the best performance for predicting hotel prices with the lowest RMSE of 20.31 and an R-squared value of 0.931. This project demonstrates the power of big data tools in analyzing customer behavior and optimizing pricing, highlighting the Hadoop ecosystem's contribution to data-driven decision-making within the travel industry.

Keywords: Agoda; Hadoop; pricing optimization; customer behavior; big data

1 INTRODUCTION

With the acceleration of digital transformation, organizations constantly generate massive amounts of both structured and unstructured data. These datasets are considered too large, unstructured, and fast-moving to be processed or analyzed by traditional methods. However, with the advancement of technology, tools like machine learning, distributed storage (Hadoop), and cloud computing have made big data analytics feasible. As a result, organizations can harness the insights gained from big data analytics to support their decision-making process. The tourism and hospitality industry is a clear example of the big data phenomenon. Given the widespread use of social media, user-generated content (UGC) data in the form of photos, videos, and texts is shared by tourists on the Internet. Meanwhile, global positioning system (GPS) data and Bluetooth data are collected through IoT sensor devices, creating a rich source of spatial-temporal tourist data. Aside from that, tourist activities and operations almost always involve online purchases, bookings, and web searches, all of which generate transaction data. From the examples above, big data sources in the tourism industry can be classified into three main categories: users, devices, and operations [1]. These data sources allow companies to understand tourist sentiments, movements, and behavior. By leveraging big data analytics, companies can gain a competitive advantage through dynamic pricing strategies, personalized marketing, and operational efficiency optimization [2]. Hence, travel companies offering booking services, such as Expedia, Booking.com, and Agoda, benefit greatly from big data.

In 2024, the global tourism market is projected to reach approximately USD 11.45 trillion, showing the recovery and growth of the tourism industry following the COVID-19 pandemic. Looking ahead to 2025, the market is expected to expand further to USD 11.94 trillion, which is a 1.04% growth in market size [3]. This growth is fueled by the emergence of new travel trends, such as adventures and medical tourism, and a shift in tourist preferences towards destinations in Asia like China, Malaysia, and Indonesia. As demand for online travel booking services rises, online travel agency platforms are utilizing big data analytics to capture the future market share by offering competitive pricing.

The online travel agency market is highly competitive in providing services to potential customers. Each company holds a significant share of the market by leveraging its unique strategies to attract different customer segments. Expedia and Booking.com dominate the market with their extensive global reach, offering plenty of bundled travel packages and different ranges of accommodation. They have an advantage in strong branding, investor support, and customer loyalty. Meanwhile, Agoda focuses on the Asian market as it provides competitive pricing and user-friendly platforms tailored to local preferences. Its innovative loyalty rewards and “Best Price Guarantee” program attract more budget-conscious travellers . Hence, each of

these companies uses big data to gain a competitive advantage in the tourism industry and achieve its business goals. Through the application of big data analytics, these companies optimize their strategies in real-time to meet user expectations.

Agoda is a leading company in the tourism and hospitality industry that provides online travel booking services. Its reservation services are mainly for accommodations, flights, activities, airport transport, and vacation rentals. The coverage of its business extends to more than 200 countries and is available in 39 languages, engaging different users from all over the world [4]. Agoda has become one of the most popular and fastest-growing platforms for users seeking affordable and convenient booking options.

Agoda was co-founded in 2005 by Michael Kenny and Robert Rosenstein in Phuket, Thailand. Before establishing Agoda, Kenny launched two travel websites, PlanetHoliday.com and PrecisionReservations.com. Both of them were eventually merged into the Agoda platform. This integration helped set the foundation for Agoda's expansive global reach. In 2007, Agoda was acquired by Booking Holdings, strengthening its market position and expanding its technology infrastructure [5].

Agoda's platform allows users to search for accommodations, compare hotel pricing, and plan trips based on personal preferences. Customers can filter their search by travel dates, destination, budget, property type, room amenities, and hotel ratings. The core strength of the company is its robust recommendation system, which offers personalized suggestions based on customers' preferences through its vast database [6]. This data-driven method helps Agoda improve user satisfaction and increase customer engagement by tailoring the travel booking experience to individual needs.

Agoda leverages a large and diverse database that includes search history, transaction records, customer reviews, and property information. By employing these advanced big data analytics to analyze its massive datasets, Agoda can discover trends and patterns in customer behavior on its platform. The outcome of the analysis can provide insight for Agoda to refine its pricing strategies, promotions, and personalized recommendations to ensure customers receive competitive deals. The efficient use of big data by Agoda enables its business to remain competitive in the industry.

2 PROBLEM STATEMENTS

The rapid advancement of digital technology has resulted in a transformative era for the online travel and hospitality industry. Although significant progress has been made in technological innovation, understanding customer purchasing behavior and determining accurate pricing on online booking platforms remain a challenge. These challenges arise due to the wide range of customer preferences, changing market trends, and the dynamic nature of pricing models.

The ability to understand customer behavior and optimize pricing strategies has become a critical determinant of success in this industry. Businesses may lose their competitive advantages, miss revenue opportunities, and fail to meet customer needs without effective tools to address these challenges. With the growing awareness among consumers regarding modern pricing strategies, it is essential to assess potential changes in consumer behavior under dynamic pricing scenarios.

The adoption of Hadoop technologies has enabled organizations to process large-scale data and extract valuable insights to inform data-driven decisions. While Hadoop has proven effective in handling complex datasets, its application in analyzing customer behavior and pricing strategies remains underexplored. The research questions to guide this study are as such:

i. How can Hadoop tools be used to address Agoda's challenges in analyzing booking behavior and pricing strategies?
ii. What patterns in booking behavior support dynamic pricing strategies?
iii. How can these patterns improve the pricing strategies of Agoda?

This study aims to evaluate the effectiveness of Hadoop in improving Agoda's ability to achieve the following objectives:

i. To analyze booking behaviors based on demographic and booking history
ii. To identify patterns that can inform dynamic pricing strategies

3 TOOLS AND TECHNOLOGIES

In this project, we used a wide range of tools and technologies to analyze customer booking behavior and improve pricing strategies. Apache Pig was used for preprocessing data. Next, Apache Hive was utilized for querying and exploratory data analysis (EDA) to uncover patterns in customer behavior and pricing trends. MapReduce was applied to evaluate price optimization and customer behavior, with its results compared against those from Hive for more comprehensive insights. Subsequently, Apache Spark was utilized for model training and performance evaluation. Data was stored and managed using HDFS, with HBase providing additional storage for specific needs. Aside from tools from the Hadoop ecosystem, additional tools like Power BI and Python were used respectively for visualization and to enhance results analysis via trend identification and actionable insights integration. The rationale for the choice of tools and how each tool was employed in this project are detailed in the methodology section.

4 METHODOLOGY

The methodology employed in this project is described below:

4.1 Data Preprocessing using Apache Pig and Python

Data preprocessing is a crucial step in data analysis. This process involves cleaning, transforming, and preparing data to ensure its quality and usability. In this project, Apache Pig, a high-level platform for processing large data sets, is used to preprocess and transform datasets. Additionally, Python is employed as a supplementary tool to enrich the dataset with day-of-the-week information.

The raw datasets are involved 3 datasets, which are 'users.csv' (1340 rows X 5 columns), 'flights.csv' (271888 rows X 10 columns) and 'hotels.csv' (40552 rows X 8 columns).

These 3 datasets provide detailed information about travel, combining data on flights, hotels, and users. The Flights dataset has 271,888 records with details like where flights start and end, their type, cost, distance, and travel dates. The Hotels dataset includes 40,552 entries about hotel stays, covering hotel names, locations, how long people stayed, and how much they spent. The Users dataset contains 1,340 profiles with basic information about travellers , such as their age, gender, and the companies they are linked to. Overall, these datasets provide information on understanding travel patterns, spending habits, and traveler demographics.

4.1.1 Apache Pig

Apache Pig is a high-level data flow platform designed to process large-scale data sets in Hadoop. It simplifies complex data transformations and analyses through a scripting language called Pig Latin. Pig excels in handling semi-structured and structured data and provides an abstraction over Hadoop MapReduce, making data processing tasks more intuitive and efficient.

Figure 4.1 shows the steps involved in Apache Pig for data preprocessing:

Figure 4.1 Workflow of Apache Pig

The data preprocessing begins by loading datasets using the PigStorage function to parse CSV files and defining their schema with the **AS** clause. The datasets are then joined using common keys: **userCode** for linking users and flights, and **travelCode** for linking flights and hotels, combining relevant information into a single dataset. Data cleaning is performed by removing duplicate entries with the **DISTINCT** operator and filtering out rows with missing values. Next, feature selection is applied to retain only the required attributes, ensuring a concise dataset. The **STRSPLIT** function is used to split date columns into day, month, and year components for further analysis. Finally, the processed data is stored in the file **combined_dataset.csv** for subsequent use.

Apache Pig script is shown as below:

```
1. Pig > nano combine_files.pig
2. Pig >
 -- Load the users.csv file
 users = LOAD '/home/<user>/users.csv' USING PigStorage(',')
 AS (code:CHARARRAY, company:CHARARRAY, name:CHARARRAY, gender:CHARARRAY, age:INT);

 -- Load the flights.csv file
 flights = LOAD '/home/<user>/flights.csv' USING PigStorage(',')
 AS (travelCode:CHARARRAY, userCode:CHARARRAY, from:CHARARRAY, to:CHARARRAY,
 flightType:CHARARRAY, price:DOUBLE, time:DOUBLE, distance:DOUBLE, agency:CHARARRAY,
 date:CHARARRAY);

 -- Load the hotels.csv file
 hotels = LOAD '/home/<user>/hotels.csv' USING PigStorage(',')
 AS (travelCode:CHARARRAY, userCode:CHARARRAY, name:CHARARRAY, place:CHARARRAY, days:INT,
 price:DOUBLE, total:DOUBLE, date:CHARARRAY);

 -- Join the datasets on userCode and travelCode
 combined = JOIN users BY code, flights BY userCode;
 combined = JOIN combined BY flights::travelCode, hotels BY travelCode;
```

```
-- Remove duplicates and clean missing values
cleaned_data = DISTINCT combined;
cleaned_data = FILTER cleaned_data BY (users::code IS NOT NULL) AND (hotels::travelCode IS NOT NULL);

-- Remove unnecessary columns
final_data = FOREACH cleaned_data GENERATE
    users::name AS user_name, users::gender AS gender, users::age AS age,
    flights::from AS flight_from, flights::to AS flight_to, flights::flightType AS flight_type,
    flights::price AS flight_price, flights::agency AS agency,
    hotels::name AS hotel_name, hotels::place AS hotel_place,
    hotels::days AS hotel_days, hotels::total AS hotel_total,
    flights::date AS date_flights, hotels::date AS date_hotels;

-- Split date_flights into day, month, and year
split_flights_date = FOREACH final_data GENERATE
        user_name, gender, age, flight_from, flight_to, flight_type, flight_price, agency,
        hotel_name, hotel_place, hotel_days, hotel_total,
        date_flights, date_hotels,
        FLATTEN(STRSPLIT(date_flights, '/')) AS (month_flights:CHARARRAY, day_flights:CHARARRAY,
year_flights:CHARARRAY);

-- Split date_hotels into day, month, and year
split_all_dates = FOREACH split_flights_date GENERATE
        user_name, gender, age, flight_from, flight_to, flight_type, flight_price, agency,
        hotel_name, hotel_place, hotel_days, hotel_total,
        date_flights, date_hotels,
        month_flights, day_flights, year_flights,
        FLATTEN(STRSPLIT(date_hotels, '/')) AS (month_hotels:CHARARRAY, day_hotels:CHARARRAY,
year_hotels:CHARARRAY);

-- Store the final result into updated_dataset_V2.csv
STORE split_all_dates INTO '/home/<user>/pigdata/combined_datatset.csv' USING PigStorage(',');
```

3. Pig > pig –x local combine_files.pig

4.1.2 Python

Python is used to enrich the dataset. The day of the week for the flight and hotel dates is done through Python. This is achieved using the pandas library. The final dataset size is 81104 rows X 28 columns.

Python script is shown as below:

```
import pandas as pd

# Load the dataset
input_file_path = '/home/<user>/pigdata/combined_dataset.csv'
output_file_path = '/home/ <user>/pigdata/updated_dataset.csv'

# Load the CSV file without headers
df = pd.read_csv(input_file_path, header=None)

# Identify columns M and N (13th and 14th from the left)
column_M_index = 12  # 13th from the left
column_N_index = 13  # 14th from the left

# Ensure columns M and N are treated as dates, with errors handled
df['M_day_of_week'] = pd.to_datetime(df.iloc[:, column_M_index], errors='coerce').dt.dayofweek
df['N_day_of_week'] = pd.to_datetime(df.iloc[:, column_N_index], errors='coerce').dt.dayofweek

# Save the updated dataset
df.to_csv(output_file_path, index=False, header=False)

print(f"File updated successfully. New file saved at: {output_file_path}")
```

4.2 Data Query with Apache Hive

Apache Hive is an open-source data warehouse system built atop Apache Hadoop, designed to facilitate the querying and analysis of large datasets using a SQL-like language known as HiveQL [7]. The vast amounts of data stored in distributed storage systems seamlessly. In 2024, the Apache Software Foundation announced the release of Apache Hive 4.0, which includes over 5,000 commits featuring new functionalities, bug fixes, and performance enhancements, underscoring Hive's ongoing relevance in the big data ecosystem [8]. Apache Hive enables efficient querying and analysis of massive datasets through a familiar SQL-like interface which leverages Hadoop's distributed storage and processing power.

First of all, the headers in the csv files are removed to avoid any header appearing in the query results. The next step involves moving the CSV file containing booking data (booking.csv) from the local file system to the Hadoop Distributed File System (HDFS). This is done using the 'hdfs dfs –put' command. The command copies the file from its local directory (/home/<user>/Downloads/booking.csv) to a designated path in HDFS, such as /user/hive/warehouse/.

```
1. user@user-VirtualBox:~$ hdfs dfs –put /home/<user>/Downloads/booking.csv /user/hive/warehouse/
```

Next, the table structure is defined in Hive. A CREATE TABLE statement is executed to create the "booking" table as shown in below code schema. The schema includes fields such as 'company', 'user_name', 'gender', 'age', 'origin', 'destination', 'flightType', and others. Each field is assigned an appropriate data type like STRING, INT, or DOUBLE. The ROW FORMAT DELIMITED clause specifies that the data fields in the file are separated by commas, and the STORED AS TEXTFILE clause indicates that the file format is plain text. This step sets up the Hive table to match the structure of the CSV file.

```
1. Hive> CREATE TABLE booking (
        company STRING,
        user_name STRING,
        gender STRING,
        age INT,
        origin STRING,
        destination STRING,
        flightType STRING,
        flight_price DOUBLE,
        time STRING,
        distance DOUBLE,
        agency STRING,
        hotel_name STRING,
        place STRING,
        days INT,
        hotel_price DOUBLE,
        total DOUBLE,
        flight_day INT,
        flight_month INT,
        flight_year INT,
        hotel_day INT,
        hotel_month INT,
        hotel_year INT,
        flight_day_of_week STRING,
        hotel_day_of_week STRING
        )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

Finally, the data from the CSV file is loaded into the 'booking' table using the LOAD DATA INPATH command. The path specified (/user/hive/warehouse/booking.csv) corresponds to the location of the file in HDFS. This command transfers the data from the CSV file into the Hive table, making it accessible for querying and analysis.

```
1. Hive> LOAD DATA INPATH '/user/hive/warehouse/booking.csv' INTO TABLE booking;
```

4.3 Data Query with MapReduce

MapReduce is a programming model introduced by Google to efficiently process large-scale datasets in distributed environments. It simplifies parallel computing through two key operations which are Map and Reduce. The Map function will process data into intermediate key-value pairs. On the other hand, the Reduce function will aggregate the key-value pairs to produce the final result. The model is highly scalable and fault tolerant. It is suitable for handling extensive data workloads [9], [10].

According to [NO_PRINTED_FORM] [9], the intermediate keys and values share the same domain as the output keys and values. The MapReduce framework processes data by first splitting input files into M chunks, each typically ranging between 16 and 64 MB. During the reduce phase, the intermediate data is sorted, and the Reducer iterates through each unique intermediate key. For every key, it passes the key and its associated set of intermediate values to the user-defined reduce function. The output result from the reduce function is then added to a final output file corresponding to that reduce partition.

The fault-tolerance of MapReduce allows for automatic task re-execution in case of system failures. It ensures reliability in distributed environments. Additionally, its scalability and ability to manage complex computations efficiently make it a valuable tool for big data processing [9], [11].

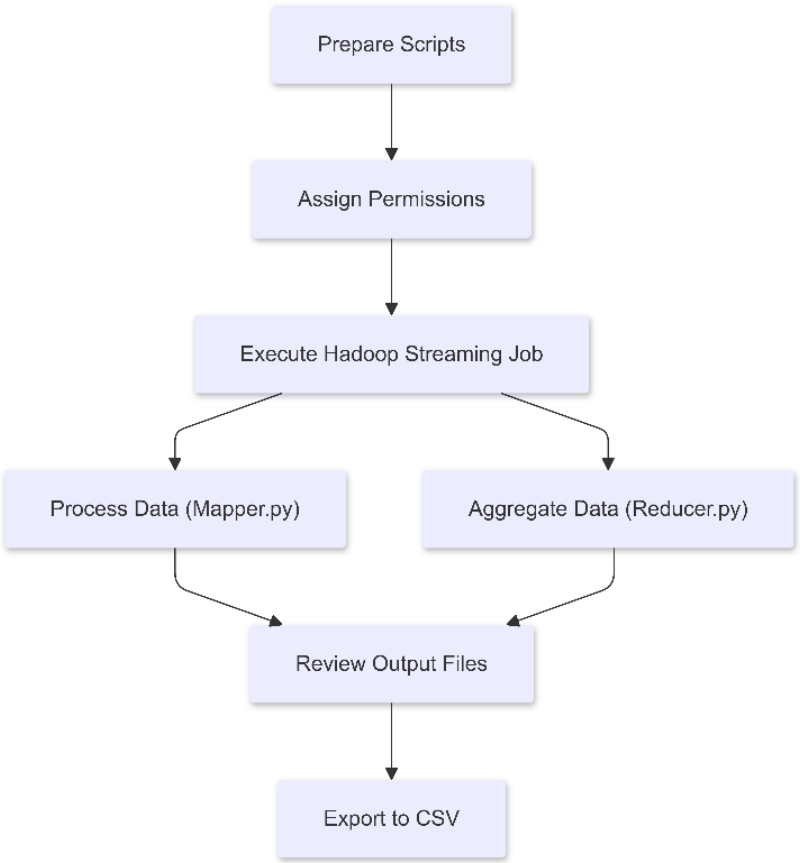

Figure 4.2 MapReduce Workflow

In this project, MapReduce was used to study customer booking behavior and evaluate pricing strategies as shown in Figure 4.2 Mapper.py was used to process the data by extracting relevant variables and converting them into key-value pairs. For instance, variables such as flight type, destination, and distance were processed to identify patterns and trends in the price analysis. Once the mapping phase was complete, intermediate key-value pairs were sorted and grouped by key. This phase ensured that all values associated with a particular key were ready for aggregation in the reducing phase. Reducer.py was used to calculate the values which enables the identification of meaningful patterns and insights. For example, it computed the average prices of flights for different destinations or categorized data by flight type (e.g., economic, premium, first-class). Then, the executable permissions were assigned using the following commands below.

```
1. chmod +x /home/<user>/mapper.py
2. chmod +x /home/<user>/reducer.py
```

The Hadoop streaming job was executed with the following template:

```
1. user@user-VirtualBox:~$ hadoop jar /home/<user>/hadoop/share/hadoop/tools/lib/hadoop-streaming-*.jar \
2.     -input /<user>/hadoop/input/<dataset_name>\
3.     -output /<user>/hadoop/<analysis_name> \
4.     -mapper /home/<user>/mapper.py \
5.     -reducer /home/<user>/reducer.py \
6.     -file /home/<user>/mapper.py \
7.     -file /home/<user>/reducer.py
```

After the job execution, the files in the output directory were listed, and the results were reviewed. The output was subsequently exported and saved as a CSV file for further analysis. Metrics such as average prices, demand trends, and seasonal variations were extracted to draw insights.

4.4 Data Storage using HADOOP Distributed File System (HDFS)

In this project, the Hadoop Distributed File System (HDFS) plays a vital role as the primary storage layer. It enables the efficient handling of large datasets throughout the entire workflow [12]. Serving as the backbone for data management, it supports preprocessing, querying, analysis and model training tasks that are performed using a variety of tools and technologies. Its architecture is distributed, scalable and fault-tolerant; this ensures that large datasets are not only stored securely but also remain accessible for all stages of the data analysis pipeline. This includes preprocessing, model training and evaluation [13].

Key Uses of HDFS

i. Centralized Data Storage

HDFS functions as a centralized repository for the storage of all raw, intermediate and processed datasets.[[14]] This ensures that data is readily accessible to all components of the ecosystem, including Apache Pig, Hive, MapReduce and Spark. The distributed architecture of HDFS enables it to manage large datasets by partitioning them into smaller blocks; these blocks are then disseminated across multiple nodes.

- Raw data is ingested into HDFS for preprocessing with Apache Pig. The commands provided below illustrate the steps necessary to establish a directory in HDFS and upload data from the local file system for centralized storage:

```
1. hdfs dfs -mkdir /user/vboxuser/booking_data
2. hdfs dfs -put /home/vboxuser/data/booking.csv /user/vboxuser/booking_data
3. pig -x mapreduce -f preprocessing.pig
```

- Processed data is stored back in HDFS for further querying and analysis using Hive.

```
1. hdfs dfs -mkdir /user/vboxuser/processed_data
2. hdfs dfs -put /home/vboxuser/data/clean_booking.csv /user/vboxuser/processed_data
```

ii. Data Partitioning and Replication

To ensure fault tolerance and high availability, HDFS automatically divides data into blocks and replicates them across various nodes in the cluster. This design makes the system resilient to hardware failures, guaranteeing continuous access to data even if some nodes go offline.

iii. Data Accessibility for Processing Tools

HDFS seamlessly integrates with tools like Pig, Hive, MapReduce, and Spark, enabling them to directly access the data stored in its directories. This integration eliminates the need for data transfers between storage systems, which reduces latency and enhances performance.

- Pig: Reads raw data from HDFS (`/user/vboxuser/booking_data`), preprocesses it, and writes the processed (cleaned) data back to HDFS in a new directory.
- Hive: Creates an external table that references the processed data in /user/vboxuser/processed_data. This setup allows for querying and exploratory data analysis (EDA) on the cleaned data, such as examining booking trends or pricing behavior.
- MapReduce: Analyzes the cleaned data for price optimization and customer behavior insights. It reads from /user/vboxuser/processed_data and saves the results in a new directory, /user/vboxuser/mapreduce_results.
- Spark: Loads the cleaned data from /user/vboxuser/processed_data, utilizes it for modeling, and saves the trained models in /user/vboxuser/models.

iv. Support for Structured and Semi-Structured Data

HDFS can store a variety of data formats, including CSV, JSON, ORC, and Parquet, making it well-suited for both structured and semi-structured data. This flexibility is essential for tasks like EDA in Hive and model training in Spark. It also ensures compatibility with different tools, simplifying the preprocessing and analysis of data for customer behavior studies and pricing strategy assessments.

v. Integration with HBase for Specific Needs

While HDFS serves as the main storage layer, HBase complements it by providing random read or write access for specific use cases, such as managing frequently accessed or dynamically updated datasets. This hybrid approach ensures efficient storage and retrieval based on the project's requirements.

vi. Scalable and Cost-Effective Data Management

As the volume of data grows, HDFS scales by adding more nodes to the cluster. [[15]] This feature ensures that the project can handle increasing amounts of customer booking and pricing data without compromising performance or reliability. Its open-source nature and compatibility with commodity hardware make it a cost-effective solution for managing large volumes of data.

4.5 Model Development using Spark

Apache Spark is an open-source distributed big data processing framework [16]. Originally created by Mater Zaharia from the University of California, Berkeley as a faster alternative to MapReduce in 2009, the codebase was later donated to the Apache Software Foundation [17]. According to the Apache Software Foundation, 80% of the Fortune 500 companies use Apache Spark, and there are over 2,000 contributors to the open-source project from industry and academia [8].

There are several characteristics of Apache Spark that contribute to its widespread popularity:

1. **Efficiency and speed:**

Apache Spark's distributed computing capabilities enable it to scale linearly. This is crucial for Big Data applications such as this project given the extensiveness of historical customer booking data. By leveraging its cluster resources, computations performed during model development can be completed quickly. Furthermore, Spark processes data in memory, which benefits machine learning workflows as they often involve iterative, repetitive tasks, such are hyperparameter tuning [18].

2. **Ease of Integration with Hadoop Ecosystem**

Spark integrates seamlessly with the Hadoop ecosystem, allowing it to leverage existing Hadoop infrastructure and tools. This compatibility ensures that Spark can handle both batch and streaming data, providing flexibility in how it processes and analyzes data [19]

3. **Language integration and machine learning support**

Spark supports multiple programming languages, including Python, Scala, Java, and R, making it accessible to a wide range of developers. Additionally, Spark's robust machine learning library, MLlib, provides built-in algorithms and tools for scaling machine learning tasks, such as clustering, regression, and classification, which enhances its suitability for data-driven projects like this one [20].

4. **Real-Time Data Processing:**

Although this project focuses on historical data, Spark's ability to process real-time streaming data is a key strength, as evidenced by its use in time-sensitive applications like heart arrhythmia detection [19]. In the future, this capability can be leveraged to analyze and model real-time booking patterns, providing dynamic insights into customer behavior and enabling more responsive decision-making.

In summary, Apache Spark is an ideal tool for machine learning tasks in big data analytics due to its flexible, fast, and easy-to-use design. These characteristics of Apache Spark form the rationale for using it for model development in this project.

The steps for installing Spark on an Ubuntu machine are outlined below:

a. Download, extract, and move Apache Spark to the /opt/spark directory on Ubuntu by running the following script:

```
1. wget https://downloads.apache.org/spark/spark-3.5.3/spark-3.5.3-bin-hadoop3.tgz.sha512
2. tar xvf spark-*.tgz
3. sudo mv spark-3.5.3-bin-hadoop3 /opt/spark
```

b. Set up the environment variables for Apache on the Ubuntu machine by adding the following to .bashrc before running *source .bashrc*:

```
1. export SPARK_HOME=/opt/spark
2. export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
3. export PYSPARK_PYTHON=/usr/bin/python3
```

c. Run *pyspark* in the terminal to run Apache Spark via the Python shell. Alternatively, .py scripts can be directly submitted to Apache Spark via the terminal by running the following command in the directory containing the .py script:

```
1. spark-submit example_script.py
```

d. Install the required dependencies to perform K-Means clustering and regression:

```
1. sudo apt-get install python3-matplotlib python3-pandas python3-seaborn
```

As described in the overview of tools, two machine learning tasks were undertaken in this project using Spark, namely:

- K-Means clustering: To perform customer segmentation for a better understanding of customer booking behavior and demographics
- Regression models: To build a dynamic pricing model

4.5.1 K-Means Clustering

The goal of K-Means clustering in this project is customer segmentation, grouping Agoda customers based on shared booking behaviors. This segmentation enables more targeted marketing campaigns, improved customer service, and the development of tailored products and services.

Customer segmentation can be based on four key dimensions [21]

- Geographic: Region, population density, or climate.
- Demographic: Age and family size,
- Psychographic: Lifestyle variables like interests and attitudes.
- Behavioral: Customer behavior toward products, such as loyalty or purchase readiness.

This project focuses on behavioral segmentation due to the dataset's emphasis on booking-related features. While some demographic and geographic data are present, K-Means clustering is less effective with categorical data because it relies on distance metrics like Euclidean or Manhattan, which are better suited for numerical variables [22], [23]. Numerical encoding of categorical variables often fails to capture their true relationships, leading to suboptimal clustering. Therefore, this project focuses on analyzing Agoda customers' booking behavior to achieve meaningful segmentation.

After installing Apache Spark on the Ubuntu machine, K-Means clustering was performed following the steps detailed below:

Step 1: Data Preprocessing

Although the data was already cleaned in the initial data preprocessing step, further processing was required to represent the data at a user level and ensure the data was in a suitable format for K-means clustering. The key steps involved are:

- Data aggregation: The flights and hotels datasets were summarized at user-level to obtain *age*, *avg_flight_price*, *total_mileage*, *total_flight_price*, *total_flights*, *avg_flight_distance*, *avg_flight_time*, *total_days_hotel*, *total_hotel_price*, *avg_hotel_price_daily*, and *total_hotels* using SQL on Apache Hive
- Data joins: The aggregated users, flights, and hotels tables were joined using SQL on Apache Hive, and a column was created for *total_price*.
- Encoding categorical features: *gender* and *company* were encoded using *StringIndexer* and *OneHotEncoder* using PySpark
- Feature selection: Irrelevant, intermediate, or redundant columns were dropped after transformation using PySpark.
- Feature engineering: Feature vectors were created using *VectorAssembler* and normalized using *StandardScaler* using PySpark.

Table 4.1 SQL commands for data preprocessing

Data Aggregation for Flights	Data Aggregation for Hotels	Data Joins
`CREATE TABLE agg_flights AS SELECT userCode, ROUND(AVG(price), 2) AS avg_flight_price, ROUND(SUM(distance),2) AS total_mileage, ROUND(SUM(price),2) AS total_flight_price, COUNT(*) AS total_flights, ROUND(AVG(distance), 2) AS avg_flight_distance, ROUND(AVG(time), 2) AS avg_flight_time FROM flights GROUP BY userCode;`	`CREATE TABLE agg_hotels AS SELECT userCode, SUM(days) AS total_days_hotel, ROUND(SUM(total), 2) AS total_hotel_price, ROUND(SUM(total) / SUM(days), 2) AS avg_hotel_price_daily, COUNT(*) AS total_hotels FROM hotels GROUP BY userCode;`	`CREATE TABLE agoda_customers AS SELECT a.code, a.gender, a.age, a.company, b.avg_flight_price, b.total_mileage, b.total_flight_price, b.total_flights, b.avg_flight_distance, b.avg_flight_time, c.total_days_hotel, c.total_hotel_price, c.avg_hotel_price_daily, c.total_hotels, b.total_flight_price+c.total_hotel_price AS total_price FROM users a JOIN agg_flights b ON (a.code = b.userCode) JOIN agg_hotels c ON (a.code = c.userCode);`

Table 4.2 Python Script submitted to Apache Spark for data preprocessing

Encoding Categorical Features	Data Joins

`# Count occurrences of each unique value in gender dataset.groupBy("gender").count().show()`
 	
`# Count occurrences of each unique value in company dataset.groupBy("company").count().show()`
 	
`# Index the gender column indexer = StringIndexer(inputCol="gender", outputCol="genderIndex") dataset = indexer.fit(dataset).transform(dataset)`
 	
`# One-hot encode the indexed column encoder = OneHotEncoder(inputCol="genderIndex", outputCol="genderVec") dataset = encoder.fit(dataset).transform(dataset)`
 	
`# Drop the original 'gender' column and index column dataset = dataset.drop("gender", "genderIndex")`
	`vec_assembler = VectorAssembler(inputCols = dataset.columns, outputCol='features')`
	
	`final_data = vec_assembler.transform(dataset)`
	
	`final_data.printSchema()`
	
	`scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures", withStd=True, withMean=False)`
	
	`# Compute summary statistics by fitting the StandardScaler scalerModel = scaler.fit(final_data)`

Step 2: Cluster Initialization and Model Selection

After preprocessing, the Elbow Method and the Silhouette Score were used to determine the optimal number of clusters (k) to be used for K-Means clustering. The Elbow Method obtains the optimal k value by identifying the point where adding more clusters only results in a marginal decrease in the Within-Cluster Sum of Squares (WCSS). WCSS indicates how spread out the data points are in each cluster [24]. Meanwhile, the Silhouette Score measures the quality of clustering for different k values [25].

Table 4.3 Python Script submitted to Apache Spark for cluster initialization

Elbow Method	Silhouette Score

`# Elbow Method for Optimal K`
 	
`# Calculate cost and plot`
`cost = np.zeros(10)`
 	
`for k in range(2,10):`
`  kmeans = KMeans().setK(k).setSeed(1).setFeaturesCol('features')`
`  model = kmeans.fit(final_data)`
`  cost[k] = model.summary.trainingCost`
 	
`# Plot the cost`
`df_cost = pd.DataFrame(cost[2:])`
`df_cost.columns = ["cost"]`
`new_col = [2,3,4,5,6,7,8, 9]`
`df_cost.insert(0, 'cluster', new_col)`
 	
`# Saving the values of WCSS`
`df_cost.to_csv("elbow_method_cost.csv", index=False)`
	`# Silhouette Score`
	
	`silhouette_scores = []  # List to store silhouette scores and k values`
	
	`evaluator = ClusteringEvaluator(predictionCol='prediction', featuresCol='scaledFeatures', metricName='silhouette', distanceMeasure='squaredEuclidean')`
	
	`# Loop to calculate silhouette scores for different k values`
	`for i in range(2, 10):`
	`  kmeans = KMeans(featuresCol='scaledFeatures', k=i, seed=123)`
	`  model = kmeans.fit(final_data)`
	`  predictions = model.transform(final_data)`
	`  score = evaluator.evaluate(predictions)`
	`  silhouette_scores.append({'k': i, 'silhouette_score': score})`
	`  print('Silhouette Score for k =', i, 'is', score)`
	
	`# Create a pandas DataFrame from the silhouette_scores list`
	`silhouette_df = pd.DataFrame(silhouette_scores)`

Step 3: Analysis of Clusters from K-Means

Upon determining the best k values, the clusters were analyzed by visualizing the clusters via cluster center location plots and pair plots using Python libraries like seaborn and matplotlib, which are supported by Spark. These analyses and visualizations provide an understanding of each customer cluster and what they represent in the context of Agoda's business.

4.5.2 Regression Modeling

Dynamic pricing is a strategy that adjusts the price of a product or service in real time based on market demand, competitor prices, customer behavior and other factors . By constructing a regression model, we can predict how customers will respond to different prices and thus maximize profits.

This project aims to optimize dynamic pricing strategies by analyzing customer booking behavior. Using powerful machine learning algorithms, pricing strategies on e-commerce platforms can be optimized [26]. This project uses regression models to predict a continuous dependent variable (i.e. price) in order to directly optimize pricing, which can more directly quantify the relationship between customer behavior and price.

This project also focuses on behavioral characteristics of customers, as the dataset mainly contains numerical features related to booking, and regression models perform well in dealing with numerical data. Although the dataset may contain some categorical features (such as gender, origin, destination, flight type, and association), we will use appropriate encoding methods (such as StringIndexer and OneHotEncoder) to convert them into numerical types for use in the model.

Step 1: Data Preprocessing

The purpose of this step is to clean and prepare the data for subsequent feature engineering and model training. It mainly includes the following steps. First, we use Spark to load data, and then further carry out missing value filling for features and handle duplicate values. Ensure the subsequent normal operation of the data.

Step 2: Feature Engineering

Feature engineering is an important step in machine learning to improve model performance. By processing and optimizing the raw data, it makes it more suitable for model learning. Feature engineering includes a variety of methods, such as feature derivation, feature scaling, feature vector, feature transformation and standardization. In the feature derivation process of this project, it was observed that many time-type features cannot be used directly but can be effectively used through feature derivation. This time, the difference between dates is used to derive a time feature. Then, considering the inconsistent scale of features, normalization is performed to eliminate the scale difference between features.

Table 4.3 Feature Derivation in PySpark

```
# Feature Derivation :Calculate date difference and extract date features
data_spark = data_spark.withColumn('days_in_advance', datediff(col('flight_date'), col('hotel_date'))) \
.withColumn('flight_weekday', dayofweek(col('flight_date'))) \
.withColumn('hotel_weekday', dayofweek(col('hotel_date'))) \
.withColumn('flight_weekofyear', weekofyear(col('flight_date'))) \
.withColumn('flight_month_of_year', month(col('flight_date'))) \
.withColumn('flight_year_of_data', year(col('flight_date')))

# Categorical feature encoding
categorical_features = ['gender', 'from', 'to', 'flightType', 'agency']
indexers = [StringIndexer(inputCol=col, outputCol=col+"_index", handleInvalid="keep")
for col in categorical_features]

# Numerical features
numerical_features = ['days_in_advance', 'flight_weekday', 'hotel_weekday',
'days', 'distance', 'time', 'flight_price']

#Combine all features
feature_cols = [col + "_index" for col in categorical_features] + numerical_features

# Create feature vector
assembler = VectorAssembler(
inputCols=feature_cols,
outputCol="features",
handleInvalid="keep"
)

# Standardization
scaler = StandardScaler(inputCol="features", outputCol="scaled_features",
withStd=True, withMean=True)
```

Step 3: Model selection and training

This step mainly involves selecting an appropriate regression model and training it using the training data, and ultimately obtaining a model that can effectively predict hotel prices. Three commonly used regression models were tried: decision tree regression, gradient boosting regression, and random forest regression. First, the original data set needs to be divided into a training set and a test set in an 8:2 ratio.

In decision tree regression, the maximum tree depth is adjusted to limit the complexity of the model and avoid overfitting. The default parameter configuration is used in gradient boosting regression. Similarly, in random forest regression, the maximum tree depth is also adjusted to control the complexity of the model. Similar to decision trees, limiting the depth of each tree helps improve the generalization ability of the model.

A pipeline was used to simplify the model training and prediction process and ensure consistency in the feature processing steps. The pipeline connects all the data preprocessing steps (such as feature scaling and categorical feature encoding) to the model itself. This allows all the steps to be completed in one operation, avoiding the hassle and potential for error of manually managing intermediate steps.

Table 4.4 Regression Modeling in PySpark

```
# 1. Split training and test sets
train_data, test_data = data_spark.randomSplit([0.8, 0.2], seed=42)
# 2. Define models
dt = DecisionTreeRegressor(labelCol="hotel_price",
featuresCol="scaled_features",
```

```
maxDepth=17)
gbt = GBTRegressor(labelCol="hotel_price",
featuresCol="scaled_features"
)
rf = RandomForestRegressor(labelCol="hotel_price",
featuresCol="scaled_features",
maxDepth=12,
)
def create_pipeline(model):
return Pipeline(stages=indexers + [assembler, scaler, model])
# 4. Train and evaluate models
models = {
"Decision Tree Regression": dt,
"Gradient Boosting Regression": gbt,
"Random Forest Regression": rf
}
```

Step 4: Model Evaluation

To evaluate the predictive performance of the models, we use three common metrics: root mean square error (RMSE), mean absolute error (MAE) and R-squared. These metrics provide a comprehensive picture of the accuracy and completeness of the model predictions and provide a measure for model selection. The model evaluation phase presupposes the performance of the models on the test data set. By calculating RMSE, MAE and R-squared, we can compare the predictive performance of different models and select the best model.

Step 5: Dynamic Price Optimization

- **Objective and background:**

After completing the model training and evaluation, we entered the dynamic price optimization stage. Demand estimation plays an important role in dynamic pricing because the optimal price can be obtained by maximizing returns based on the demand curve [27]. The goal of this stage is to adjust the hotel_price value based on the model's prediction results to maximize revenue. This study aims to explore how to use the price information predicted by the machine learning model to achieve dynamic pricing, thereby simulating revenue improvements on the test dataset.

- **Optimization strategy:**

We adopt a simple optimization strategy based on a price multiplier. First, a price fluctuation range is defined, and the impact of different price fluctuations on demand is simulated by combining the demand elasticity coefficient. For each original price in the test set, the code predicts a benchmark price, then calculates the expected revenue at different prices based on the demand elasticity and selects the price that maximizes revenue as the optimal price.

4.6 Visualization using HBase and PowerBI

Step 1: Understanding the Dataset and Visualization Requirements

Before beginning the conversion process, it is crucial to thoroughly understand the flight and hotel booking dataset stored in HBase. This involves identifying key attributes. Equally important is defining the visualization goals to determine which metrics and trends need to be highlighted in Power BI. Care must be taken to ensure the dataset does not contain sensitive information, like customer names or contact details, to maintain privacy and adhere to data protection regulations.

Step 2: Extracting Data from HBase

The next step is to extract data from HBase into a format suitable for transformation. This can be achieved using HBase APIs, HiveQL for querying data, or tools like Apache Sqoop for exporting data into a relational database. These tools make it possible to efficiently retrieve large amounts of data from HBase's columnar structure. While performing the extraction, it is essential to monitor the process closely to avoid issues such as data truncation or missing fields. Ensuring all necessary data columns and rows are included will save time during later stages.

Step 3: Loading Data into Power BI

The transformed data can now be loaded into Power BI using its built-in connectors. If the data is hosted on a relational database, connecting Power BI directly to the database server can streamline the workflow. Before loading, ensure the data source is accessible and that permissions or network settings are configured correctly. Additionally, consider optimizing the data size to

avoid performance lags during the creation of visualizations.

Step 4: Modeling Data in Power BI

In Power BI, the imported data must be organized and modeled to create meaningful relationships between tables. The Power Query Editor can be used to further clean and shape the data, such as merging columns, filtering out irrelevant rows, or creating calculated columns. It is essential to avoid creating ambiguous relationships between tables, as this can lead to errors in the visualization. A well-designed data model ensures that the dashboard will display accurate and reliable insights.

Step 5: Designing Visualizations

With the data modeled, the next step is to design the dashboards and reports. Use appropriate visualization types, such as line charts for trends, heatmaps for geographic analysis, and bar charts for categorical comparisons. Validate metrics, such as the total number of bookings or revenue, to ensure accuracy in the final visualizations. Thoughtful and well-structured dashboards will effectively communicate insights to stakeholders.

Step 6: Implementing Data Refresh

To keep the dashboards updated, configure a scheduled data refresh in Power BI. This involves syncing the HBase data source or its exported equivalent with Power BI at regular intervals. Monitor the refresh process to catch errors such as failed updates or incomplete data loads. Set an appropriate frequency that balances the need for up-to-date information with system performance to maintain a seamless user experience.

Step 7: Testing and Validation

Before finalizing the dashboards, thoroughly test and validate the data and visualizations. Cross-check aggregated metrics against the original HBase dataset to ensure consistency. Ensure filters and slicers in the dashboard function as expected. Stress testing the dashboards with large datasets can also reveal potential performance issues. Validating the entire workflow ensures that the visualizations provide accurate and actionable insights.

5 RESULTS AND DISCUSSIONS

5.1 Customer Demographics

i. Gender Distribution

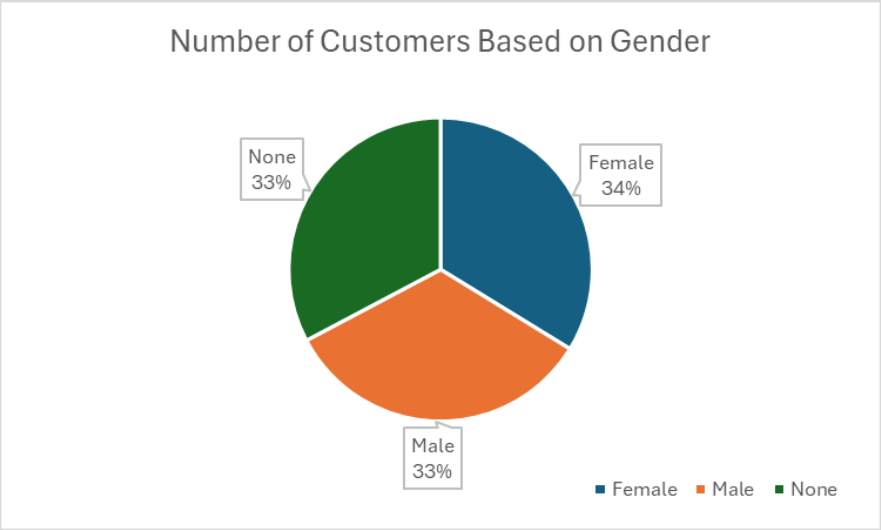

Figure 5.1 Pie Chart of Number of Customers Based on Gender.

In Figure 5.1 the gender distribution analysis reveals a relatively balanced proportion among the three categories which are female, male, and none. Female customers have the largest share with 27,380 bookings, closely followed by male customers with 27,170 bookings. Interestingly, the "none" category, which may include unspecified or non-binary gender identities also comprise 26,554 bookings. This near parity among the categories indicates inclusivity in the dataset and suggests that Agoda should cater to a diverse range of customers.

```
OK
female  27380
male    27170
none    26554
Time taken: 47.238 seconds, Fetched: 3 row(s)
```

Figure 5.2 Hive Output of Gender Distribution.

ii. Customer Age Distribution

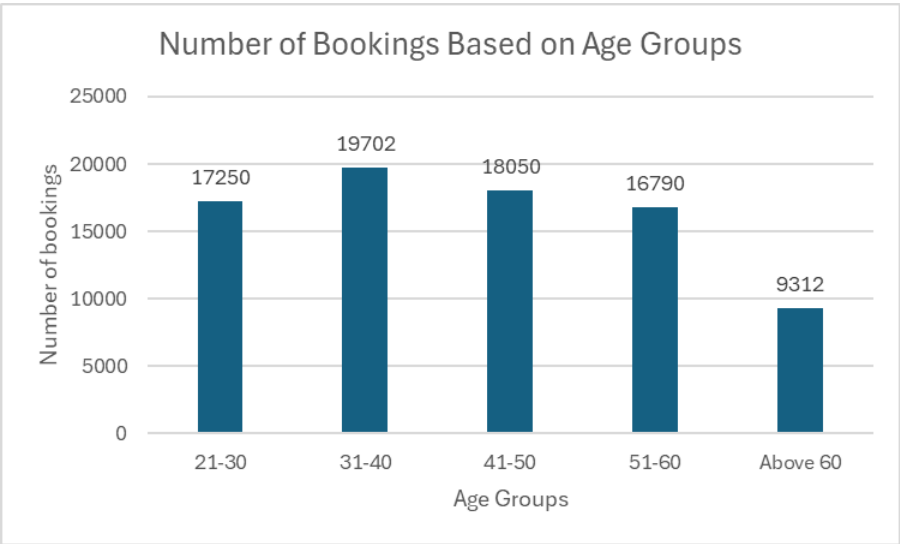

Figure 5.3 Bar Chart of Number of Bookings Based on Age Groups.

The total number of bookings across different age groups is shown in Figure 5.3. The customer age distribution indicates that the majority of bookings come from the age group of 31-40 (19,702 bookings) making it the most active demographic. This is closely followed by the 41-50 age group (18,050 bookings) suggesting that middle-aged individuals dominate the travel segment. This is potentially due to their financial stability and career-related travel needs. The 21-30 age group (17,250 bookings) reflects

a significant presence, likely driven by leisure or educational travel among younger adults. The 51-60 age group (16,790 bookings) also shows notable activity, possibly from individuals nearing retirement or engaging in lifestyle travel. Meanwhile, the above-60 age group (9,312 bookings) represents the smallest segment indicating lower travel activity among older individuals. This distribution highlights the importance of tailoring travel packages and services to meet the preferences of middle-aged and young adults, who have the largest share of customers.

```
OK
21-30   17250
31-40   19702
41-50   18050
51-60   16790
Above 60        9312
Time taken: 58.415 seconds, Fetched: 5 row(s)
```

Figure 5.4 Hive output of Customer Age Distribution

iii. Average Spending Based on Customer Age Distribution

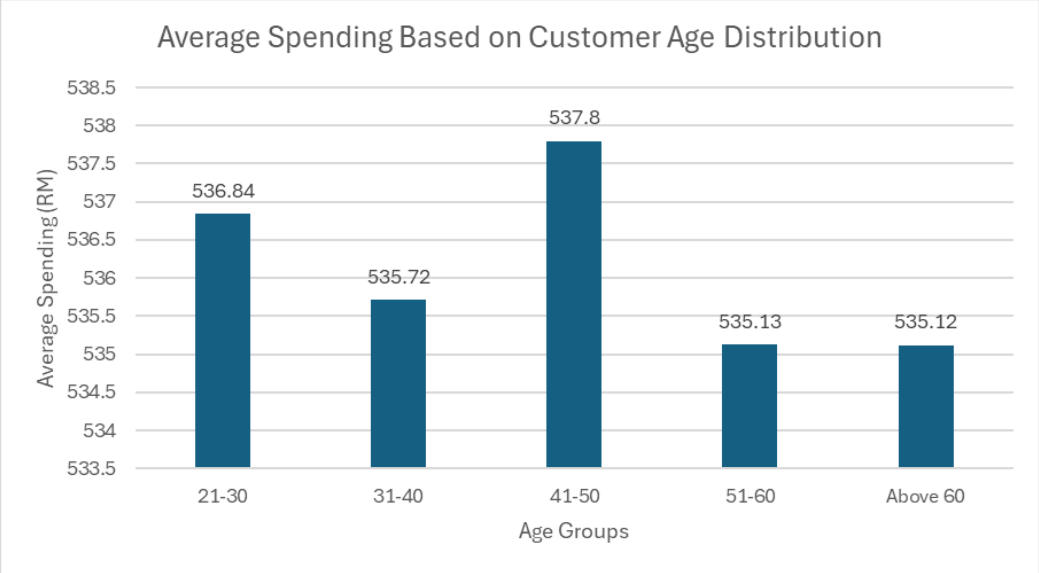

Figure 5.5 Bar Chart of Average Spending Based on Customer Age Distribution.

The average spending analysis across different age groups in Figure 5.5 reveals a consistent pattern, with all groups spending in the range of RM 535–RM 538 on average. The spending includes the flight ticket and hotel price. The highest average spending is observed in the 41-50 age group at RM 537.80, which is slightly higher than the others. This potentially reflects their financial stability and willingness to invest in comfort or premium services. The 21-30 age group follows closely with an average of RM 536.84, indicating younger adults are also significant contributors to travel expenditure. The 31-40, 51-60, and above 60 age groups show almost identical average spending which RM 535.72, RM 535.13 and RM 535.12 respectively. This pattern suggests a uniform pricing strategy or similar purchasing behavior among these demographics.

```
OK
41-50   537.8
21-30   536.84
31-40   535.72
51-60   535.13
Above 60        535.12
Time taken: 109.533 seconds, Fetched: 5 row(s)
```

Figure 5.6 Hive Output of Average Spending Based on Customer Age Distribution

iv. Customer Preferences by Flight Type

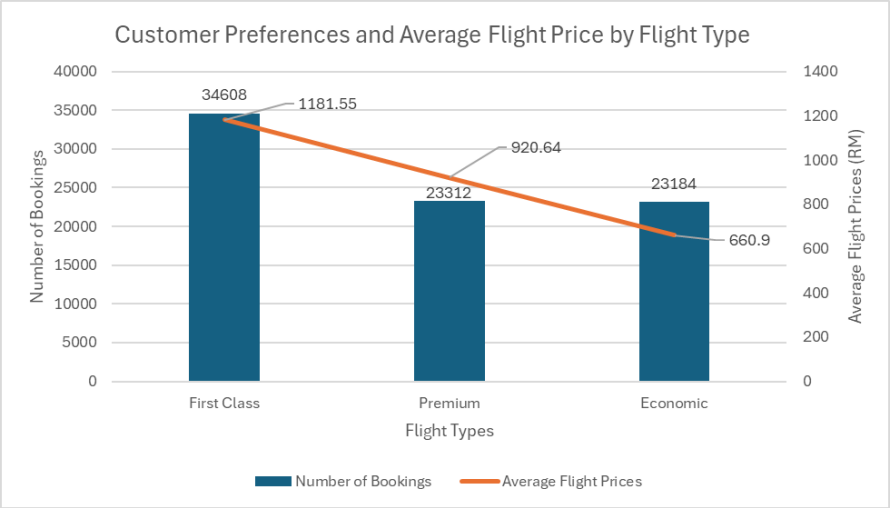

Figure 5.7 Based on flight types, bar chart shows number of bookings and line graph shows average flight prices.

The analysis Figure 5.7 highlights the significant differences between flight pricing across various flight types. Economic class showed the most cost-effective option with average price of RM 660.90. The average price for premium class is RM 920.64. It offers a balance between luxury and affordability, which is an attractive choice for business travellers or individuals that are looking forward to improving comfort at a reasonable cost. The first class showed the highest average flight price of RM 1181.95. It reflects its premium status with exclusive services that are suitable for traveler that's looking for high-end travel experience. However, the total bookings for first class were 34,608, premium 23,312 and economic 23,184, which indicates that first class remains the most popular choice despite its higher cost, possibly due to the value placed on exclusivity and superior service.

```
OK
firstClass      34608   1181.55
premium 23312   920.64
economic        23184   660.9
Time taken: 55.24 seconds, Fetched: 3 row(s)
```

Figure 5.8 Hive Output of Number of Bookings and Average Flight Prices by Class Type

5.2 Customer Booking Behavior

The goal of customer booking behavior analysis is to understand booking behavior based on gender and age group through the comparison of MapReduce and Hive outputs. The analysis will provide actionable insights such as understanding the relationship between distance and flight to support pricing mode and profitable ticket prices, enabling business to design loyalty programs for specific customers [28]. It can help target specific customers with marketing campaigns and supports better planning for peak seasons and high demand routes. Some key insights from the MapReduce and Hive Query include:

i. Age Gender Analysis

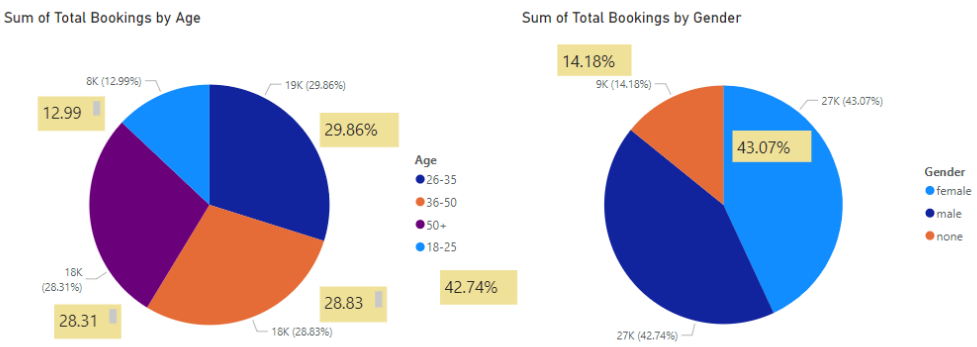

Figure 5.9 Age Gender Analysis

Gender analysis and age groups are different in terms of loyalty. As a result, this can optimize the market approach by recognizing the preferences and needs of different gender and age groups. and Analysis of age gender can provide businesses to understand the target audience. [29] with tailored marketing campaigns. In Figure 5.9 displays visual summary of a pie chart in total bookings

by age that reveals the largest age group, contributing 29.86% of bookings, two other age groups follow closely, contributing 28.83% and 28.31% of bookings. These three groups together account for 87.01% of total bookings and this indicates business effectively to a broad range of most active customers. And the smallest group, contributing only 12.99%, represents a segment less to make bookings. In terms of gender, the analysis shows that females contribute the largest share of bookings at 43.07%, while males closely at 42.74%. The remaining 14.18% of bookings are from customers with unspecified gender that highlights a gap in data collection which could be addressed to understand this segment. Figure 5.10 shows the raw data processed by MapReduce and Apache Hive to summarized insights in pie chart, the MapReduce output is raw and unstructured but provides the same output as Hive, the execution time is shown as 108.956 seconds, it reflects the overhead of using a higher level of abstraction compared to MapReduce.

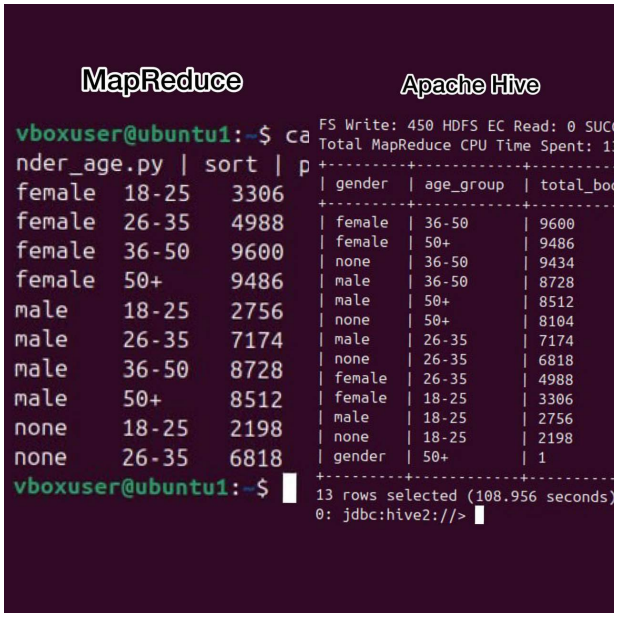

Figure 5.10 MapReduce vs Hive Output for Age-Gender Analysis

ii. Monthly Trend Revenue

Figure 5.11 Monthly Trend Revenue

Monthly revenue trends highlight revenue fluctuations and seasonal variations, with certain months, such as holiday seasons, spikes in holiday seasons [30]. In Figure 5.11 visualizes the monthly revenue in bar chart derived from Hive and MapReduce, it shows a significant spike in revenue during October (Month 10) and November (Month 11), this indicates seasonal activities urge higher sales during these months. The blue line represents average monthly revenue, to compare monthly performance to the yearly average. In Figure 5.12 shows the monthly revenue data from MapReduce and Apache Hive. MapReduce provides each month's total revenue in a list format, such as for January (Month 1) total revenue is RM 4,005,450.34 while in October (Month 10) has increased at $4,616,222.19. From these insights' businesses can use these trends to plan marketing campaigns and promotions that boost sales during seasonal months and optimize pricing strategies to maximize revenue.

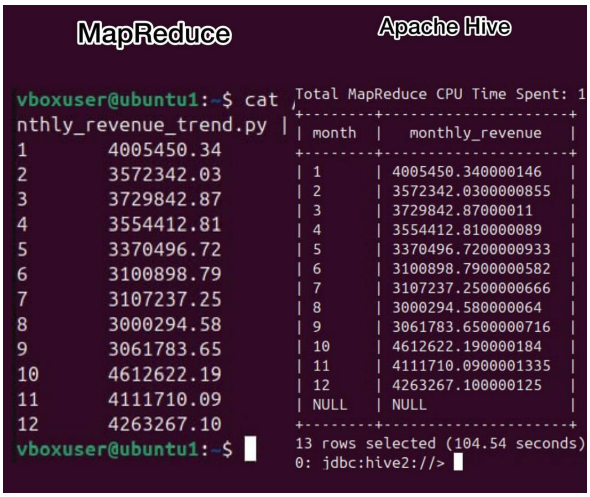

Figure 5.12 MapReduce vs Hive Output for Monthly Trend Revenue

iii. Peak Times

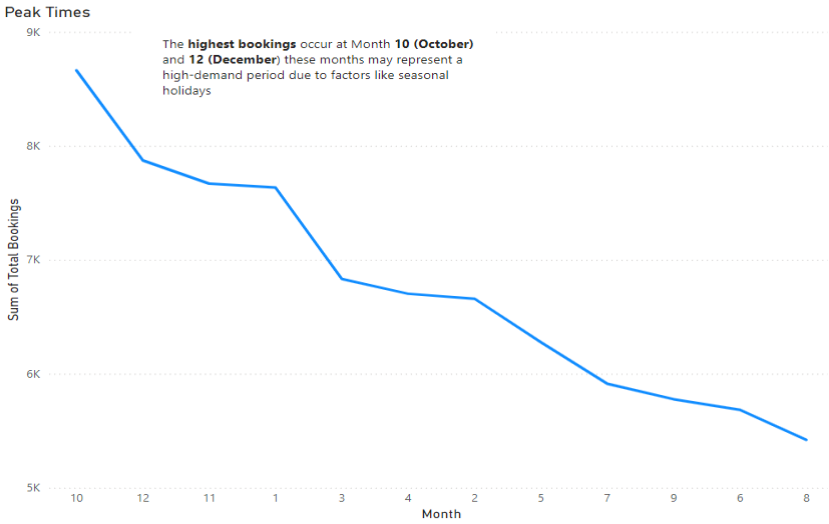

Figure 5.13 Line Graph of Peak Times

Understanding peak times enables better resource allocation and pricing adjustments. Peak booking times align with holidays, specific travel seasons, while drop-offs in bookings occur after major holidays [31], during mid-week periods, it shows Figure 5.13 visualizes line chart of the total monthly bookings during peak times October (Month 10) as the month with the highest bookings (8,663) followed by December (Month 12) is 7,873 and November (7,669), it shows gradual decline in bookings after the peak months and the lowest number of bookings occurring in August with the total number of bookings is 5,420. In Figure 5.14 displays the total number of bookings of each month in a raw data and unsorted format in MapReduce. In Hive Output the results are easy to analyze since the it is a structured output. By having these insights, businesses can adjust staffing and resources during peak times to enhance customer experience and use dynamic pricing to maximize profits during high-demand periods

Figure 5.14 MapReduce vs Hive Output of Peak Times Analysis

iv. Popular Destination by Gender

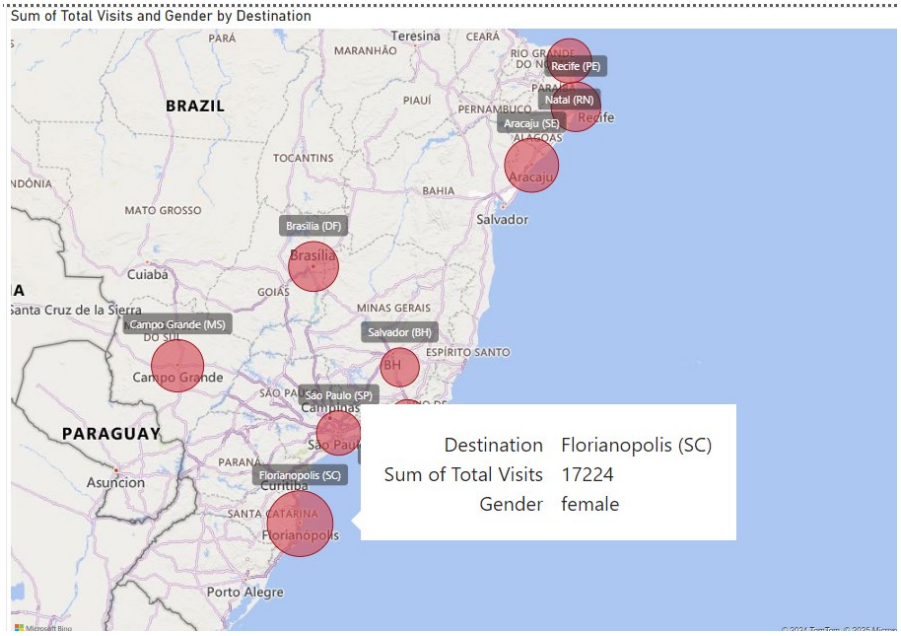

Figure 5.15 Geo map of Popular Destination

Popular destinations by gender reveal that different genders may have varying preferences for travel destinations. For instance, men may prefer business-focused destinations, while women may gravitate toward leisure-focused locations. In Figure 5.15 displays a geo map integrated the processed data from MapReduce and Hive. It shows Florianopolis (SC) is the most popular destination with the total number of visits is 17,224 across all genders with majority of female. In Figure 5.16 reveals the total visits by gender and destination in raw format, showing Florianopolis (SC) has 5,816 visits by female as the highest category. In similar, male visiting Florianopolis with the number of visits 6,037. This indicates the popularity among both genders. By having these insights, businesses can tailor marketing materials to highlight destinations based on the target gender and age group [32] and develop partnerships with local businesses at popular destinations to enhance offerings [33].

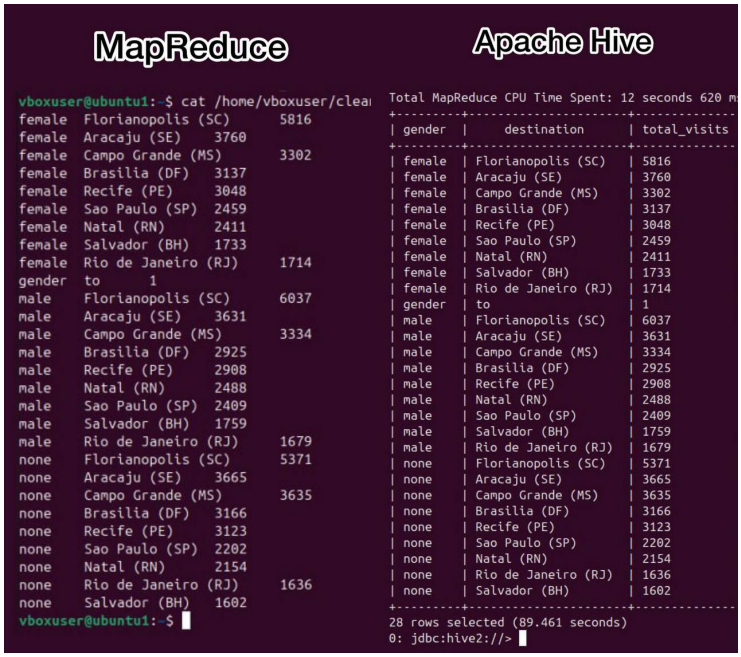

Figure 5.16 MapReduce vs Hive Output of Popular Destination Analysis

v. Identify the Highest Revenue

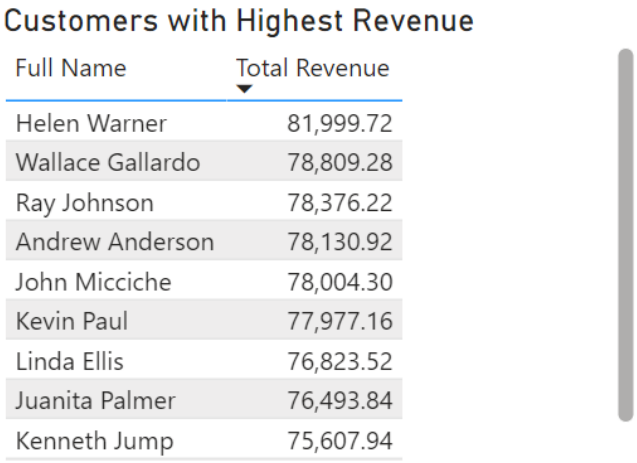

Figure 5.17 Customer of Highest Revenue

In Figure 5.17 represents the table visualization of customers with the highest total revenue. The structured format stands out with Helen Warner at the top with the total revenue of RM 81,999.72 followed by Wallace Gallardo ($78,809.28) and Ray Johnson (RM 78, 376.22). In Figure 5.18 displays the top customers in raw format, the Apache Hive shows the same data as MapReduce but in tabular format, and the query execution took 93.339 seconds. Identifying the highest revenue sources helps pinpoint the routes and customer segments that generate the most revenue, it contributes significantly to revenue through add-ons and upgrades [34]. Businesses can focus marketing and operational efforts on high-revenue routes and develop premium services to cater to business travellers [35].

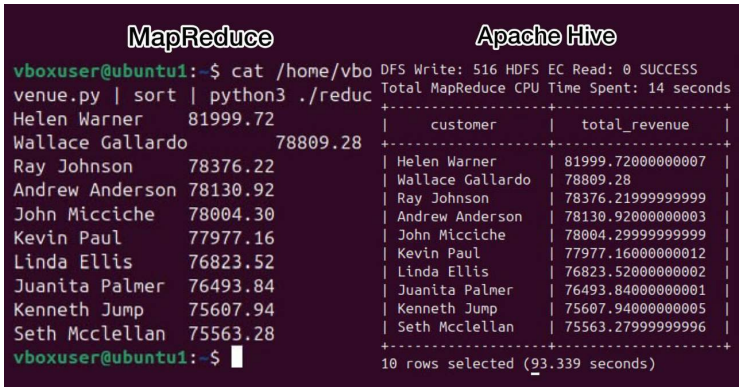

Figure 5.18 MapReduce vs Hive Output of Revenue Analysis

vi. Identify the Relationship between Distance and Price

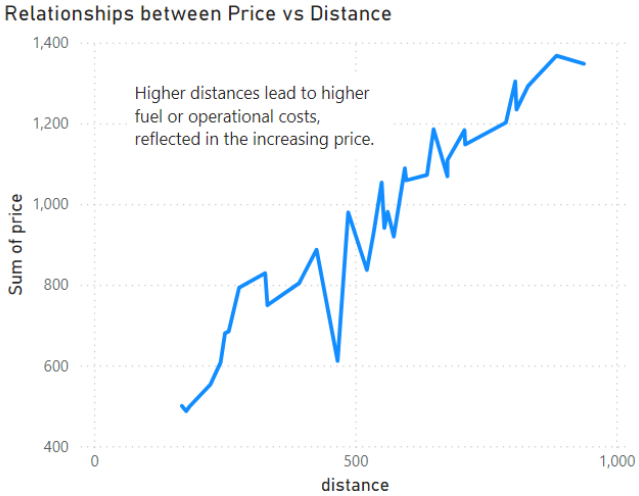

Figure 5.19 Line Graph of Price and Distance

The line chart shown in Figure 5.19 shows the relationship between distance and price. It highlights as the distance increases, the price rises significantly caused by factors such as higher fuel and operational costs, In Figure 5.20 shows the MapReduce and Apache Hive outputs between travel distance and average price, for instance, at 168.22 units, the average price is 499.63,

while at distance of 392.76 units the average price increase to 804.20. By understanding the relationship between distance and price, it helps optimize pricing strategies.

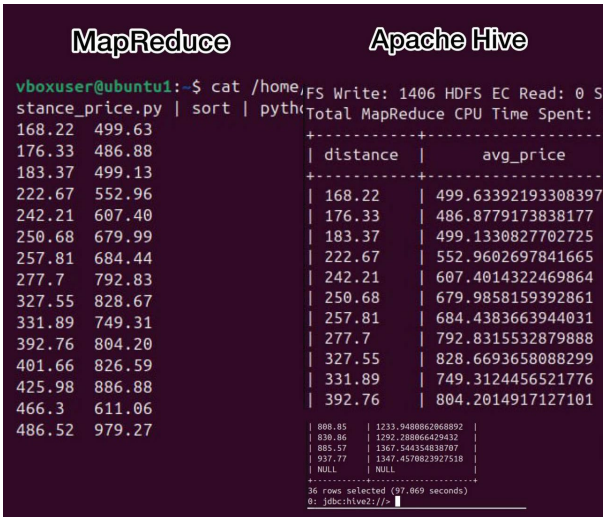

Figure 5.20 MapReduce vs Hive Output of Distance and Price Analysis

vii. Evaluate Peak Season vs Off Peak Season

Figure 5.21 MapReduce vs Hive Output Peak Season and Off-Peak Season

In Figure 5.21 shows the data of Peak Season and Off-Peak Season by comparing the average price and total bookings. From the outputs it shows while the peak season generates higher price in RM 960.2586 the number of customers bookings is lower than during the Off-Peak season. This indicates that customers prefer traveling to more affordable options and lower crowd levels, by providing these insights businesses can target promotions during peak seasons or commercialize off-peak popularity to retain steady revenue.

5.3 Price Optimization

By identifying how prices vary based on these attributes using Hive and MapReduce, we can uncover opportunities to maximize revenue while offering value to different customer segments. Key areas of focus include:

i. Average Flight and Hotel Price Across Destinations

Figure 5.22 Average Flight and Hotel Price Across Destinations

Figure 5.22 showed significant variations in flight and hotel prices across destinations. It showed that Salvador (BH) emerged as the most expensive destination for both flight and hotel prices with an average flight cost of RM 1179.23 and average hotel

prices of RM 263.41. This indicates its appeal as a premium destination, potentially due to high demand or exclusive offers. On the other hand, Natal and Salvador showed a high average hotel price of RM 242.88 and RM 263.41 respectively, while their flight costs remain relatively moderate at RM 866.97 and RM 1179.23. This showed a balanced yet premium travel experience. Conversely, cities like Sao Paulo are highly regarded as budget-friendly options with the lowest average flight cost of RM 826.55 and average hotel prices of RM 139.10. Figure 5.23 showed both MapReduce and Hive output in a similar manner.

From this analysis, Agoda should create a promotional activity based on destination pricing trends. For premium destinations like Salvador and Natal, Agoda should design luxury travel packages that combine flights and high-end hotels to attract travellers seeking high-quality experience. For budget-friendly destinations like Sao Paulo, Agoda can focus on affordable travel deals by low-cost flight promotions and hotel bundles to attract price-sensitive customers. Additionally, Agoda can highlight the balanced travel experience in Natal by marketing it as a premium yet accessible option. Offering dynamic pricing and destination-specific promotions will help Agoda attract a wide range of customers and increase bookings.

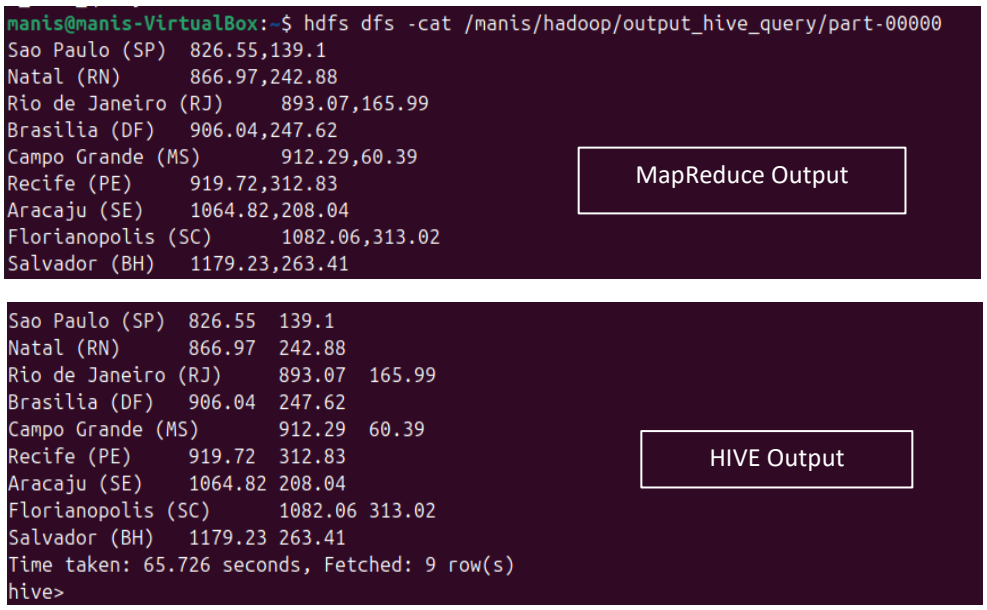

Figure 5.23 MapReduce and Hive Output of Average Flight and Hotel Price Across Destinations

ii. Seasonal Price Trends

Figure 5.24 Seasonal Price Trends

Figure 5.24 highlighted the peak travel months and corresponding average flight prices. Overall, the average flight price showed consistent pattern. On the other hand, there was a gradual decline in flight numbers from January (7635 flights) to June (5684 flights). The lowest flight numbers occurred in August with 5420 flights followed by a sharp increase in flight volume until October. October showed out as the busiest month with 8663 flights and an average price of RM 962.56, followed by December (7873 flights, RM 957.36) and November (7669 flights, RM 956.63). It revealed an increase in travel during the holiday season.

Hence, Agoda should focus on maximizing revenue from October to December with competitive pricing and promotions. For example, they can highlight holiday travel deals on the platform to attract travellers. Next, they need to offer more discounts and

promotional packages to encourage travel during the lowest-demand month such as August. Bundle deals with hotels and activities will make travel more appealing. Although the data in the MapReduce output is not sorted as shown in Figure 5.25, it still produced the same results as the Hive output. This demonstrated consistency and reliability in data processing across both tools.

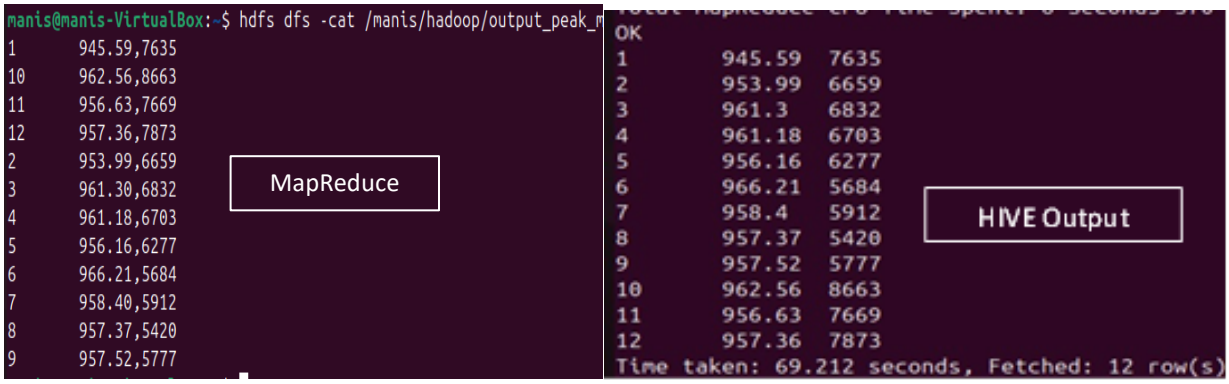

Figure 5.25 MapReduce and Hive Output for Seasonal Price Trends

iii. Distance-Based Price Analysis

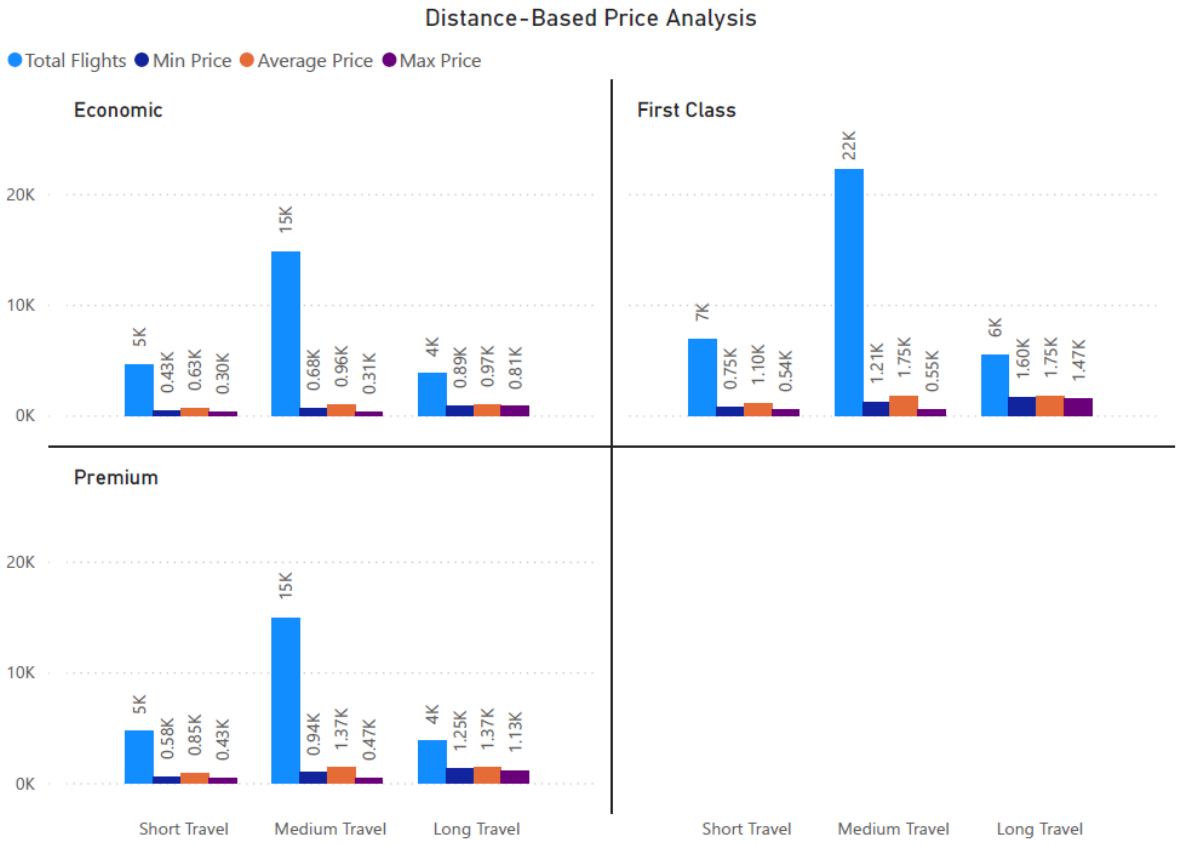

Figure 5.26 Distance-Based Price Analysis

The analysis in Figure 5.26 reveals distinct trends in pricing and flight volumes across Economic, Premium, and First-Class categories based on travel distances and Figure 5.27 showed the MapReduce output. For short travel (≤300 km), flight volumes are relatively steady across all classes with First Class showing a little increase. Economic flights showed a total of 4,628 flights and an average price of RM 629.54. Premium flights have similar flight volumes at 4,646 flights, but the average price increases to RM 851.15. It highlighted a 35.2% increase compared to the Economic. First-Class flights, with 6,854 flights and priced at an average of RM 1,096.58 showed a 74.2% increase compared to Premium. These trends highlight Economic class as the most cost-effective option for short distances, while Premium and First-Class cater to customers seeking additional comfort and luxury.

Medium travel (301–800 km) demonstrated as the most in-demand segment, with First Class showed a total of 22,250 flights and priced at an average of RM 1,747.31. This marks an 82% increase over the Premium class average price of RM 1,370.17. Economic flights remain the cheapest option in this category, with an average price of RM 959.79, and a total of 14,800 flights. The significant increase in flight volumes for medium-range distances especially in First-Class reflects strong demand and pricing power in this segment.

For Long Travel (>800 km), flight volumes decreased across all classes. Economic flights dropped to 3,756 flights, with an average price of RM 972.12. Premium flights at 3,782 flights with an average RM 1,367.88. In addition, First-Class flights with 5,504 flights and average price of RM 1,754.17 remained the most expensive. Despite the reduced demand, First-Class maintains its premium standard with consistently high pricing across all distance ranges. However, the gap between Premium and First-Class pricing narrows slightly compared to shorter distances.

Therefore, Agoda should use this information to improve its promotions and marketing strategy. For example, Agoda should offer discounts and packages for medium-distance flights, especially in Premium and First-Class categories, where demand is highest. Travel bundles offering medium-distance flights with hotels, car rentals and activities will definitely boost profits. Also, Agoda needs to collaborate with airlines to implement demand-based pricing models on their platform and promote exclusive deals to maximize sales during peak travel periods.

```
manis@manis-VirtualBox:~$ hdfs dfs -cat /manis/hadoop/output_flighttype_distance_analys
is/part-00000
economic,Long Travel (>800 km)  886.30,811.73,972.12,3756
economic,Medium Travel (301-800 km)     676.79,313.62,959.79,14800
economic,Short Travel (<=300 km)        427.16,301.51,629.54,4628
firstClass,Long Travel (>800 km)        1595.31,1468.03,1754.17,5504
firstClass,Medium Travel (301-800 km)   1210.97,549.55,1747.31,22250
firstClass,Short Travel (<=300 km)      753.77,539.39,1096.58,6854
premium,Long Travel (>800 km)   1253.30,1132.81,1367.88,3782
premium,Medium Travel (301-800 km)      942.25,469.92,1370.17,14884
premium,Short Travel (<=300 km) 580.64,427.25,851.15,4646
```

MapReduce Output

```
OK
economic        Long Travel (>800 km)   886.3   811.73  972.12  3756
economic        Medium Travel (301-800 km)      676.79  313.62  959.79  14800
economic        Short Travel (<=300 km) 427.16  301.51  629.54  4628
firstClass      Long Travel (>800 km)   1595.31 1468.03 1754.17 5504
firstClass      Medium Travel (301-800 km)      1210.97 1016.89 980.83  22250
firstClass      Short Travel (<=300 km) 753.77  1009.71 992.17  6854
flightType      Long Travel (>800 km)   NULL    flight_price    flight_price    1
premium Long Travel (>800 km)   1253.3  1132.81 1367.88 3782
premium Medium Travel (301-800 km)      942.25  1001.42 998.18  14884
premium Short Travel (<=300 km) 580.64  427.25  851.15  4646
Time taken: 61.176 seconds, Fetched: 10 row(s)
```

HIVE Output

Figure 5.27 MapReduce and Hive Output of Distance-Based Price Analysis

iv. Price Optimization Based on Flight Demand

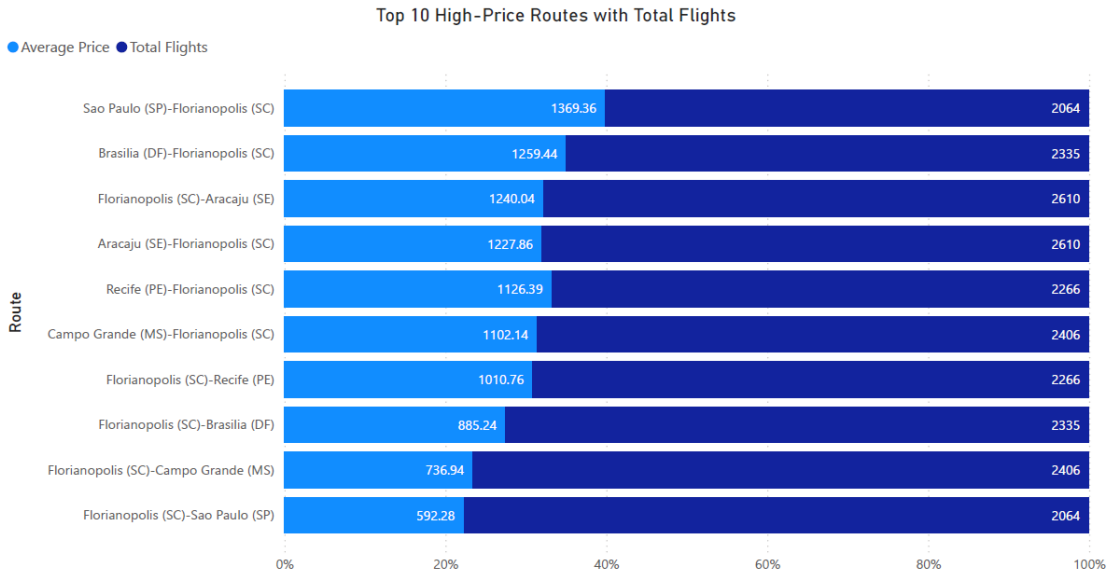

Figure 5.28 Top 10 High-Price Routes with Total Flights

Figure 5.28 provides valuable insights into the top 10 routes with the highest average prices alongside their total flight volumes and Figure 5.29 shows the MapReduce output. The route Sao Paulo (SP) to Florianopolis (SC) showed the highest average price at RM 1,369.36, accompanied by 2,064 flights. Brasilia (DF) to Florianopolis (SC) followed with an average price of RM 1,259.44 and 2,335 flights. Similarly, routes like Florianopolis (SC) to Aracaju (SE) and Aracaju (SE) to Florianopolis (SC) demonstrate high average prices of RM 1,240.04 and RM 1,227.86, respectively, each with 2,610 flights, showcasing significant demand and premium pricing.

These analyses indicate that ticket prices are influenced by both route distance and demand levels. Routes with high average flight prices show a strong preference for premium destinations. Agoda should promote luxury travel packages for these routes to maximize revenue. Since demand varies across routes, Agoda can optimize its pricing strategies and marketing efforts by creating destination-specific bundles for routes with high average flight prices.

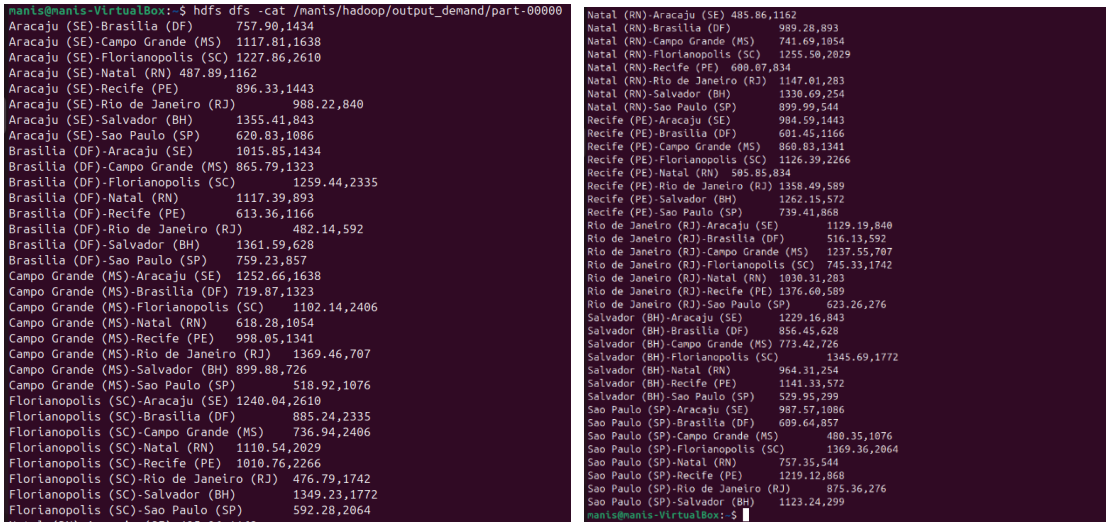

Figure 5.29 MapReduce Output of Price Optimization Based on Flight Demand

v. Price Variability by Booking Agency

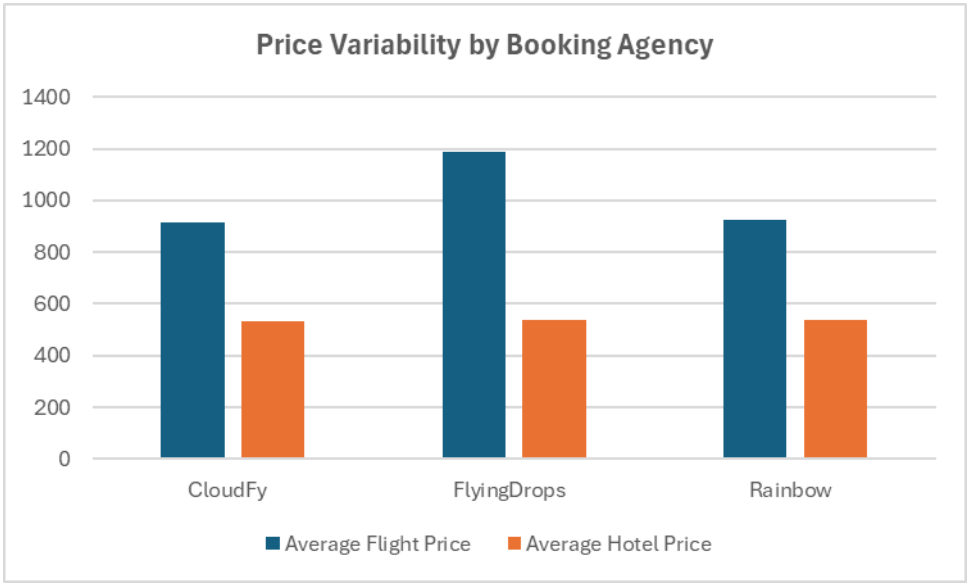

Figure 5.30 Price Variability by Booking Agency

Figure 5.31 showed MapReduce output and analysis of price differences among booking agencies shows clear variations in average flight and hotel prices illustrated in Figure 5.30. FlyingDrops demonstrated the highest average flight price at RM 1,186.13, followed by Rainbow at RM 922.96, and CloudFy at RM 917.02. In contrast, hotel price differences across agencies are minimal, with FlyingDrops averaging RM 539.72, Rainbow at RM 536.32, and CloudFy at RM 534.99. This highlighted that while flight prices vary considerably, hotel prices remain relatively stable across agencies. FlyingDrops appears to focus on premium flight pricing and target higher-end customers. In addition, CloudFy and Rainbow offer more competitive flight pricing.

Therefore, Agoda should use the consistent hotel pricing to create competitive travel bundles by pairing more affordable flights from CloudFy or Rainbow with hotels to target budget-conscious travellers . Besides, they can highlight price comparisons to showcase value for money and attract price-sensitive customers. Next, Agoda can market premium travel packages that feature FlyingDrops for high-end travellers while negotiating exclusive deals with CloudFy and Rainbow to secure discounts on flight and hotel bundles. By strategically positioning these offerings, Agoda can appeal to diverse customer segments and drive bookings.

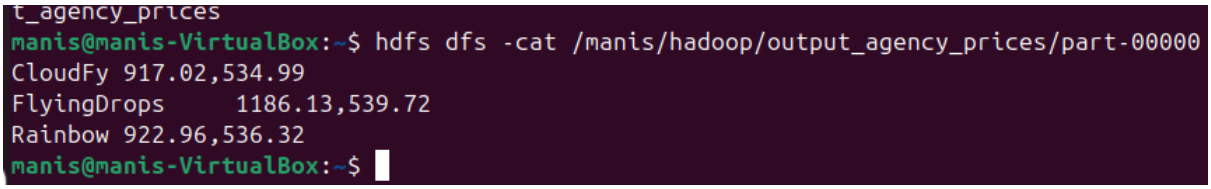

Figure 5.31 MapReduce Output for Price Variability by Booking Agency

5.4 Comparison Between Hive and MapReduce

Table 5.1 showed the comparison performance between Hive and MapReduce for analyzing average flight and hotel prices across destinations. We assume similar performance for other analyses. Hive uses a SQL-like interface that simplifies query execution by dividing tasks into stages. This makes Hive user-friendly and ideal for analytical tasks requiring quick insights. On the other hand, MapReduce works at a lower level of abstraction and offers detailed control over job configurations. This makes MapReduce a good option when handling large datasets or customized processing logic implementation.

In terms of performance, Hive demonstrated lower cumulative CPU time (~14.9 seconds across two stages) and faster execution compared to MapReduce, which took longer (~27 seconds and ~33 seconds for its two jobs). Next, Hive also managed intermediate data more efficiently, with less data written to disk during processing. In contrast, MapReduce had higher data transfer and handled large volumes of data aggregation well. Also, Hive simplified memory management, making it more accessible for users without extensive expertise in resource configuration. On the other hand, MapReduce required manual adjustments for its memory use (~291 MB for map tasks and ~182 MB for reduce tasks). Hive processed queries in a straightforward manner, while MapReduce required specific job settings. We need more setup time when using MapReduce.

In conclusion, Hive is a better option than MapReduce for this project since both produce the same results. Hive completes the task faster and with less complexity. While MapReduce is ideal for tasks that require detailed control and scalability, Hive is more efficient for analysis-focused projects with smaller or medium-sized datasets. As the dataset grows, it may be necessary to consider MapReduce to ensure the system can handle the increased workload. For now, Hive is the recommended tool to achieve the project objectives efficiently due to its simplicity, speed and it is more practical.

Table 5.1 Comparison Performance between Hive and MapReduce Tools

Tools	Hive	MapReduce
Stages/Jobs	2 Stages	2 Jobs
Mappers	1 mapper per stage	Job 1: 2 mappers; Job 2: 2 mappers
Reducers	1 reducer per stage	1 reducer per job
Cumulative CPU Time	Stage-1: 9.13 sec; Stage-2: 5.77 sec	Job 1: 7.06 sec; Job 2: 14.90 sec
HDFS Read	Stage-1: 13,467,419 bytes; Stage-2: 8,296 bytes	13,451,526 bytes
HDFS Write	Stage-1: 529 bytes; Stage-2: 458 bytes	263 bytes
Input Records	Not explicitly mentioned	Job 1: 81,105
Output Records	Not explicitly mentioned	Job 1: 81,104; Job 2: 9
Memory Usage (Peak)	Not explicitly mentioned	Map: ~291 MB; Reduce: ~182 MB
Execution Time	Stage-1: ~18 sec; Stage-2: ~16 sec	Job 1: ~27 sec; Job 2: ~33 sec

5.5 K-means Clustering

As stated previously, the aim of K-Means clustering in this project is to create meaningful customer segmentation based on their shared characteristics in support of Agoda's marketing campaigns, customer service, and new product development. The results of K-Means Clustering are detailed below:

5.5.1 Model Selection

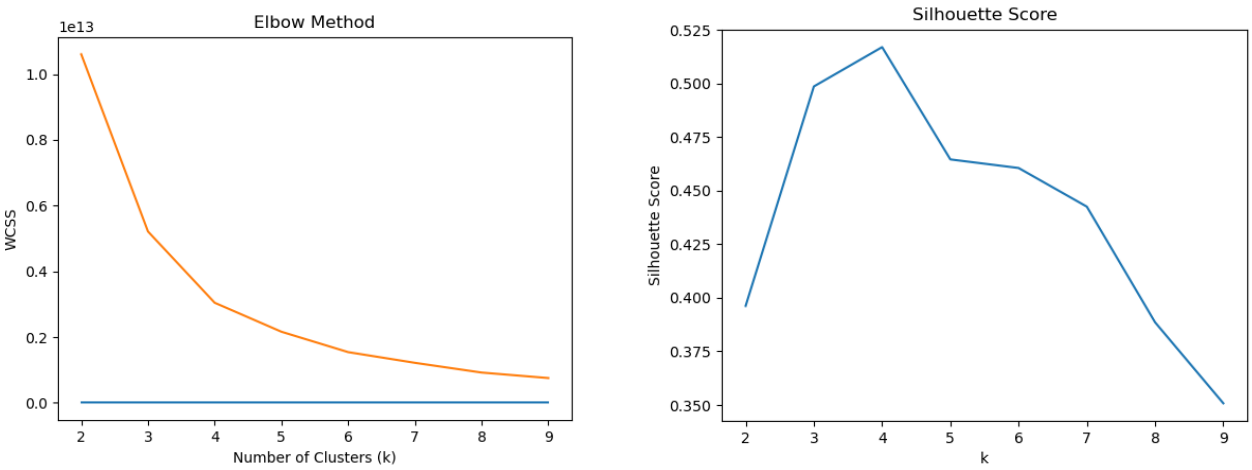

Figure 5.32. (a) Elbow Method plot (b) Silhouette Score plot

From Figure 5.32(a), the Elbow Method locates the elbow at k=4, indicating that the reduction in WCSS beyond k=4 is marginal. On the other hand, the Silhouette Score plot in Figure 5.32(b) shows that the Silhouette Score is the highest at k=4, which means that the cluster quality is the best at a k value of 4. Hence, k=4 was selected as the optimal value for the number of clusters.

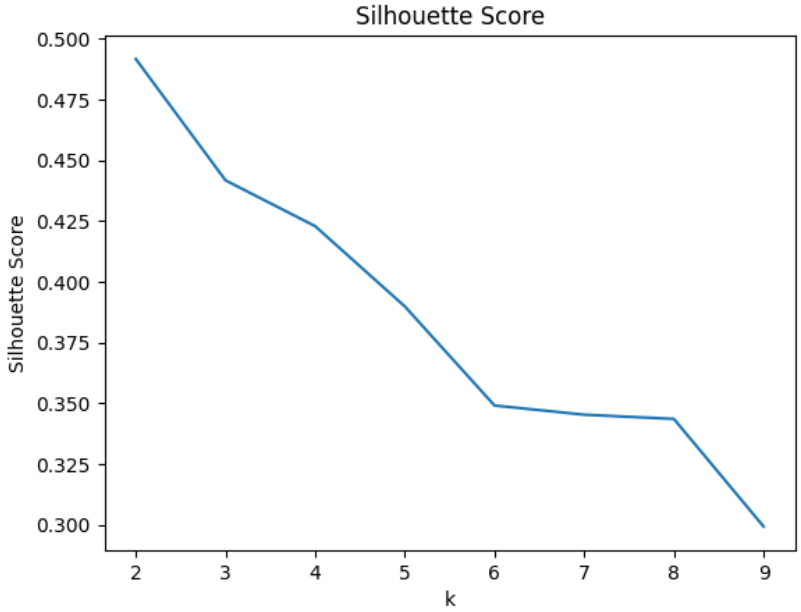

Figure 5.33. Silhouette Score plot upon inclusion of gender as a feature

In the preprocessing step for K-Means clustering, categorical features like gender and company were encoded to numerical values. However, closer examination revealed that the inclusion of these features resulted in poorer clustering performance, as demonstrated by the lower Silhouette Scores in Figure 5.33. This suggests that these categorical features are not contributing meaningful information to clustering or potentially introducing noise. Hence, all categorical columns (including the encoded features derived from them) were dropped from the column. Additionally, *userCode*, which represents the unique identifier for each customer was dropped as well to ensure that the clustering is based on relevant features only.

The features included in K-Means Clustering are listed and described in Table 5.2.

Table 5.2. List of features included in K-Means Clustering

Feature	Description
age	The age of the customer
avg_flight_price	The average price of flight tickets purchased by the customer.
total_mileage	The total flight distance traveled by the customer.
total_flight_price	The total expenses paid by the customer for all flight ticket purchases.
total_flights	The total number of flights taken by the customer.
avg_flight_distance	The average distance traveled by the customer per flight.
avg_flight_time	The average travel time per flight for the customer.
total_days_hotel	The total number of days the customer stayed in all hotels.
total_hotel_price	The total expenses paid by the customer for all hotel bookings.
avg_hotel_price_daily	The average daily cost of hotel bookings paid by the customer.
total_hotels	The total number of hotel bookings made by the customer.
total_price	The total expenses paid by the customer for all flight tickets and hotels.

5.5.2 Cluster Center Analysis

The coordinates of the centers of the four clusters formed are recorded in Table 5.3, while a visual representation of the coordinates of the cluster centers is illustrated in Figure 5.34. The analysis of these cluster center coordinates provides meaningful insight into the contribution of each feature to the clustering of data points and the characteristics of each cluster.

To analyze the cluster centers effectively, the percentage differences of the cluster centers from the average across all clusters were calculated for each feature, as recorded in Table 5.4. These percentages indicate how much the cluster centers differ from the baseline average for each feature. A positive percentage (highlighted in green) indicates that the cluster center has a higher value for that feature compared to the baseline, while a negative percentage (highlighted in red) indicates a lower value compared to the baseline. The intensity of the color represents the magnitude of the percentage difference.

The percentage difference values of the cluster centers yielded several key observations for the following features:

- total_mileage, total_price, total_flights, total_days_hotel, total_hotel_price, total_hotels:
    - Higher percentages for clusters 0 and 2: Customers in these clusters have traveled more frequently, covered longer total flight distances, and made more hotel bookings, resulting in higher flight and hotel expenses.
    - Lower percentages for clusters 1 and 3: Customers in these clusters have traveled less frequently, covered shorter total flight distances, and made fewer hotel bookings, leading to lower flight and hotel expenses.
- avg_flight_price, avg_flight_price, avg_flight_time:
    - Higher percentages for clusters 2 and 3: Customers in these clusters purchase more expensive flights and typically travel longer distances.
    - Lower percentages for clusters 0 and 1: Customers in these clusters typically purchase cheaper flights and travel shorter distances.
- avg_hotel_price_daily:
    - Slightly higher percentages for clusters 0 and 1: Customers in these clusters may spend slightly more per night during hotel stays.
    - Slightly lower percentages for clusters 2 and 3: Customers in these clusters may spend slightly less per night on hotel stays.
- age: The cluster centers do not appear to show significant differences from each other for this feature.

Based on the observations, the four clusters or segments of customers can be characterized as such:

- Cluster 0: Frequent travellers who typically travel shorter distances.
- Cluster 1: Infrequent travellers who typically travel shorter distances.
- Cluster 2: Frequent travellers who typically travel longer distances.
- Cluster 3: Infrequent travellers who typically travel longer distances.

Table 5.3. Coordinates of the cluster centers

cluster	age	avg_ flight_ price	total_ mileag e	total_ flight_ price	total_ flights	avg_ flight_ distanc e	avg_ flight_ time	total_ days_ hotel	total_ hotel_ price	avg_ hotel_ price_ daily	total_ hotels	total_ price
0	3.37	10.39	2.25	2.52	2.70	5.03	5.03	2.58	2.62	8.58	2.60	2.54
1	3.33	10.27	0.81	0.91	0.99	4.94	4.95	0.93	0.93	8.51	0.93	0.92
2	3.25	12.05	3.13	2.94	2.72	6.97	6.97	2.63	2.48	7.97	2.66	2.92
3	3.28	12.05	1.06	1.00	0.93	6.94	6.94	0.85	0.80	7.86	0.86	0.99

Table 5.4. Percentage difference of each cluster center from the average baseline across clusters for all features

cluster	age	avg_ flight_ price	total_ mileage	total_ flight_ price	total_ flights	avg_ flight_ distance	avg_ flight_ time	total_ days_ hotel	total_ hotel_ price	avg_ hotel_ price_ daily	total_ hotels	total_ price
0	1.9%	-7.1%	23.9%	36.8%	47.4%	-15.8%	-15.8%	47.6%	53.2%	4.3%	47.6%	38.0%
1	0.6%	-8.3%	-55.3%	-50.4%	-46.0%	-17.2%	-17.2%	-47.1%	-45.7%	3.4%	-47.2%	-50.1%
2	-1.8%	7.7%	72.7%	59.4%	48.1%	16.8%	16.7%	50.7%	45.4%	-3.2%	50.7%	58.4%
3	-0.7%	7.7%	-41.3%	-45.8%	-49.4%	16.2%	16.2%	-51.2%	-53.0%	-4.5%	-51.1%	-46.3%

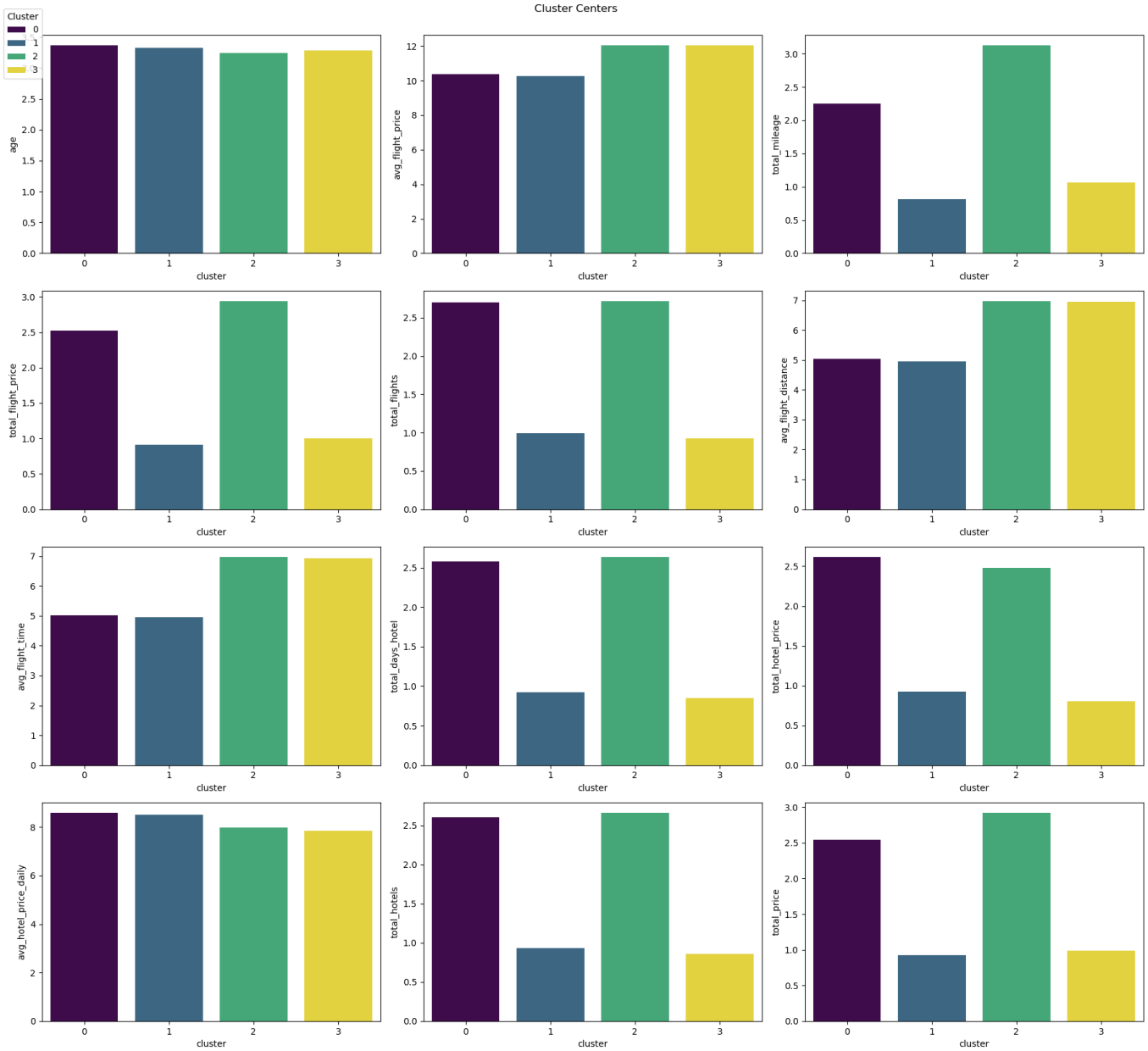

Figure 5.34. Bar plot of cluster center value for each feature.

5.5.3 2-Dimensional (2D) and 3-Dimensional (3D) Scatter plots

The features selected for 2D and 3D scatter plots in Figure 5.35 are representative of the key characteristics and observations identified via the cluster center analysis. The 2D scatter plot illustrates the strong discriminatory power avg_flight_distance holds while distinguishing between the clusters. This is evident as clusters with higher or lower average flight distances exhibit clear separations along this axis. While total_price does not visually separate some clusters in the scatter plots, its contribution to clustering is evident in the center analysis. It provides context when combined with other features like avg_flight_distance and total_mileage, providing a holistic view of a customer's travel and spending patterns. Meanwhile, the 3D scatter plot further enriches the analysis by integrating total_price and avg_hotel_price_daily into the visualization.

5.5.4 Pair Plots

The pair plots in Figure 5.36 visualize pairwise relationships between features and clusters, allowing for a visual assessment of clustering quality. In line with the previous cluster center analysis, features like age and avg_hotel_price_daily show limited discriminatory power. The overlapping data points and superimposed distribution curves for these features suggest they do not meaningfully contribute to distinguishing the clusters. While clusters remain visible, significant overlaps make it challenging to characterize them based solely on these features.

- Cluster 0: Frequent travellers who typically travel shorter distances.
- Cluster 1: Infrequent travellers who typically travel shorter distances.
- Cluster 2: Frequent travellers who typically travel longer distances.
- Cluster 3: Infrequent travellers who typically travel longer distances.

5.5.5 Recommendations

Based on the interpretation of the results from K-Means clustering, Agoda can elevate its business by implementing the following:

1. Tailored Marketing Campaigns

Different marketing campaigns can be created for different clusters of customers.

- Cluster 0 (Frequent travellers who travel shorter distances): Agoda can target these customers with promotions for frequent bookings, such as loyalty programs or discounts for shorter trips [36]. Personalized marketing can focus on offering attractive deals for nearby destinations or weekend getaways
- Cluster 1 (Less frequent travellers, shorter distances): This group may be more sensitive to price. Agoda can target them with promotions on budget-friendly hotels and short-distance flights. Highlighting affordable options, last-minute deals, and local experiences could be more appealing to these customers [37].
- Cluster 2 (Frequent travellers who travel longer distances): Agoda could offer these customers premium deals on international flights or longer-term hotel bookings. Since they tend to travel more frequently and cover long distances, they may appreciate tailored offers for frequent flyer programs, long-haul flight packages, or luxury hotels [36], [37].
- Cluster 3 (Less frequent travellers, longer distances): For these customers, Agoda could introduce tailored offers like seasonal promotions for vacations to faraway destinations. Long-term stay offers or package deals that include both flights and hotels might be more effective.

2. Customer Personalization

Agoda can personalize the user experience based on the clusters of frequent and infrequent travellers to improve their experience:

- Frequent travellers: Agoda can provide personalized recommendations for this segment based on their preferences, such as faster check-in, curated lists of hotels suited for frequent stays, or exclusive services [37].
- Infrequent travellers: For customers who travel less frequently, Agoda can provide helpful travel guides, discounts for first-time bookings, or special travel packages for special occasions (e.g., vacations, anniversaries).

3. Pricing Strategy Optimization

Agoda can leverage its understanding of customer spending behaviour to optimize its pricing strategy based on different clusters:

- Cluster 2 and 3 (Higher expenses on average): For these customers who tend to spend more per flight, Agoda can offer upselling opportunities such as premium accommodations, flights with additional services, or exclusive add-ons (e.g., lounge access, airport transfers) [37].
- Cluster 0 and 1 (Lower expenses on average): Agoda can introduce budget-friendly options and targeted promotions that align with the customer's preferences for lower-cost hotels and flights. Providing them with cost-effective packages or offering more affordable options could increase engagement [37].

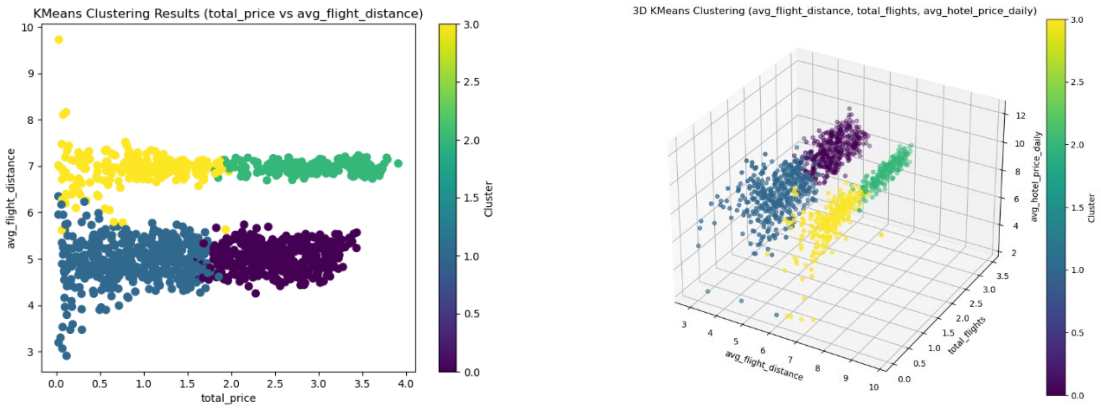

Figure 5.35. 2D and 3D scatter plots of the cluster based on avg_flight_distance, total_price, and avg_hotel_price_daily

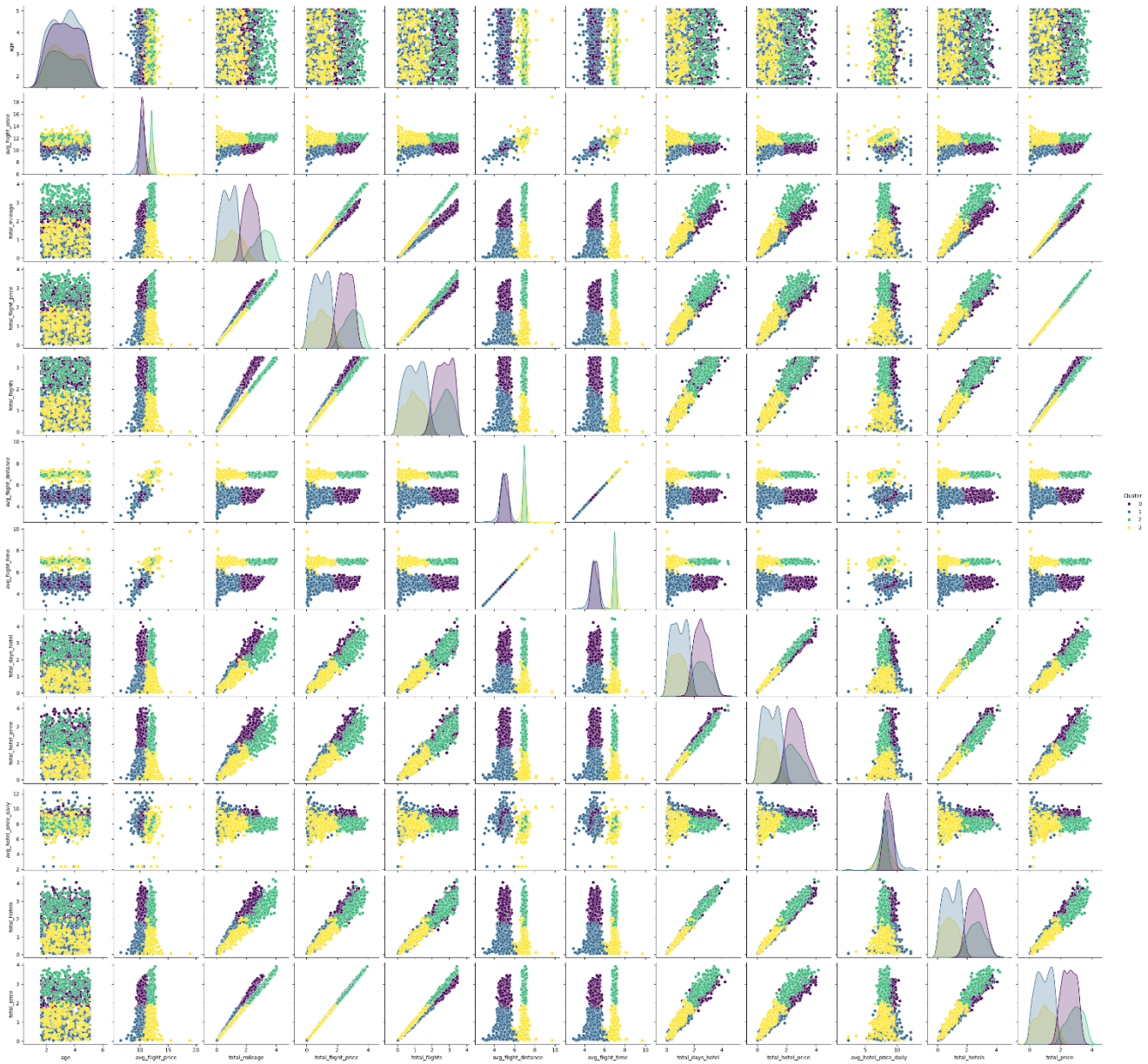

Figure 5.36. Cluster predictions for all pairs of features

5.6 Regression Modeling

5.6.1 Model Evaluation

Model evaluation is a critical step in ensuring the success of machine learning projects. It helps us build reliable and efficient models and ultimately achieve business goals. Overall, this experiment evaluated three models: random forest regression, gradient boosted regression and decision tree regression. The evaluation results clearly show that the random forest regression model performed best in all metrics as shown in Table 5.5.

Table 5.5. Comparison of model evaluation

Model Name	RMSE	MAE	R-squared
Random Forest Regression	20.2	12.31	0.931
Gradient Boosting Regression	20.49	14.02	0.929
Decision Tree Regression	28.3	8.15	0.865

Specifically, the mean square error (RMSE) of the random forest model was 20.20, which was better than the 28.30 of the decision tree models and the 20.49 of the gradient boosting model. At the same time, its mean absolute error (MAE) was 12.31, which was also lower than the 14.02 of the gradient boosting models and the 8.15 of the decision tree, indicating that the average prediction error is relatively small. Although the MAE is not the lowest, the significant decrease in RMSE indicates that the overall prediction effect of the model is better than the other two models. In addition, the random forest model has the highest

R-squared value, reaching 0.931, which is higher than the gradient boosting model's 0.929 and the decision tree model's 0.865, indicating that it best fits the data and can better explain the variance of the target variable.

5.6.2 The Feature Importance of the Optimal Model

The Figure 5.36 shows the importance of each feature in the best model, a random forest regression model. As can be seen, distance(importance: 0.3002) and time(importance: 0.2696) are the two most important factors affecting the prediction result, which is consistent with the actual situation, because distance (the spatial distance between two places) and flight time (time, the flight time required between two places) are usually important factors determining user booking behavior and hotel pricing strategies.

For Agoda users, the distance between the hotel and the user's location or their travel destination is an important consideration. A shorter distance usually means more convenient transportation and lower travel costs, so users are more likely to choose a hotel that is closer. Flight time directly affects users' travel costs and schedules, and a shorter flight time may mean that users are more willing to choose a higher-priced hotel. In contrast, features such as hotel_weekday are close to zero in importance, indicating that they have little effect on the prediction results, so they can be considered for removal in subsequent model optimization.

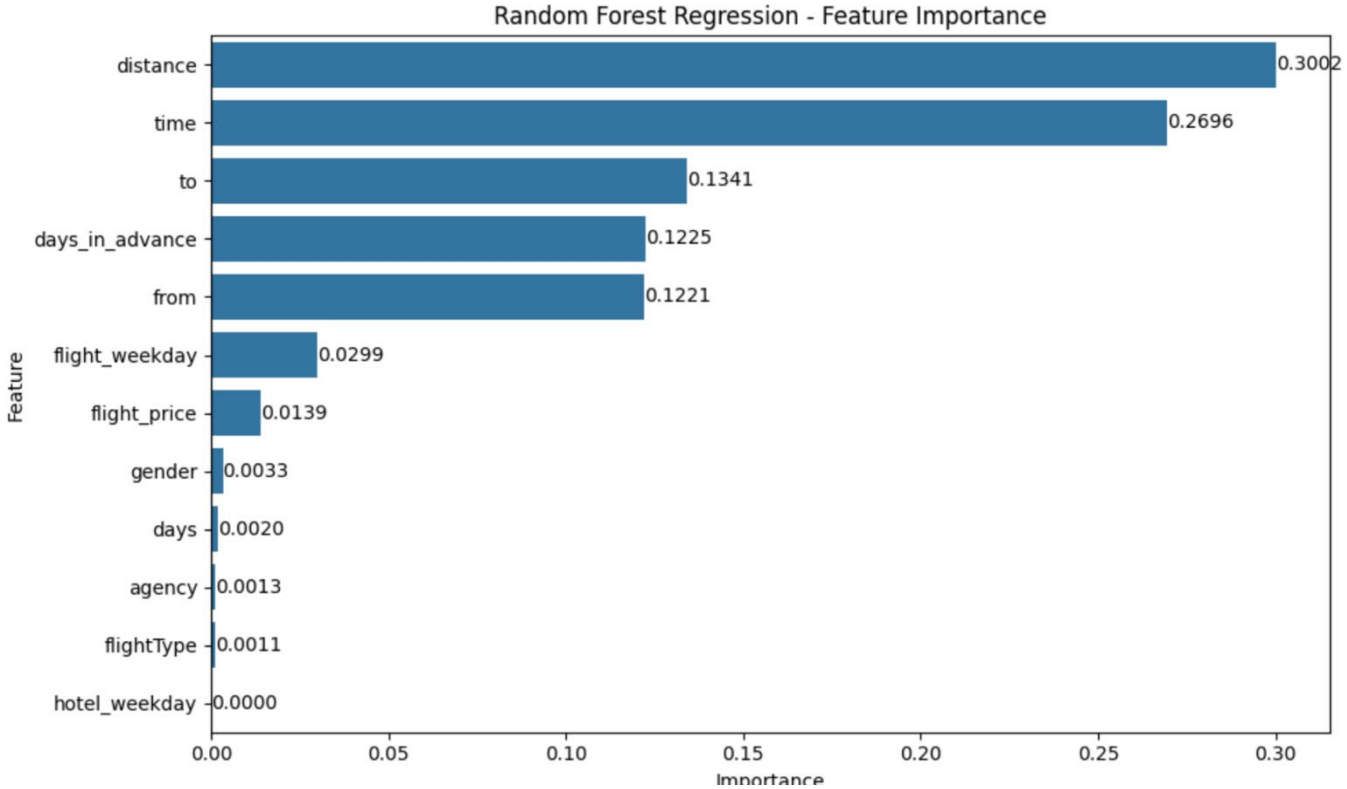

Figure 5.37 Feature Importance

5.6.3 Dynamic Pricing Optimization

This study designs and implements a dynamic pricing optimization strategy based on regression model-predicted hotel prices. The core idea of the strategy is to find the maximum strategy hotel_price that can maximize revenue by adjusting the original price, taking into account the elasticity of demand. We conducted a simulation experiment on the test dataset and conducted an in-depth analysis of the visualized optimization results to evaluate the effectiveness of the strategy.

First, a price fluctuation range (0.8 to 1.2 times the original price) was defined to limit the price adjustment range and avoid excessive price fluctuations. Then, for each original price in the test set, the model predicts a benchmark price. A demand_elasticity value (the default value is -1) is set to indicate the sensitivity of demand to price changes. A negative value indicates that an increase in price leads to a decrease in demand. Assuming that, we traverse the defined price range, calculate the expected revenue for each adjusted price (adjusted price × estimated demand), and select the adjusted price with the highest revenue as the optimal price.

The scatter plot on the left Figure 5.38 clearly shows that the dynamic pricing strategy mainly involves small adjustments around the original price. The optimal price is highly correlated with the original price, and the adjustment strategy is relatively simple and straightforward, without obvious restrictive pricing behaviour or extreme price fluctuations. This suggests that the model may consider the current market price to be relatively reasonable, or that there is no significant difference in demand elasticity between different price segments, so there is no need for significant price adjustments.

The line chart on the right Figure 5.38 clearly shows that the price adjustments of the dynamic pricing strategy are relatively concentrated, mainly between -3 and 2.5. The distribution of price changes shows that the box model as a whole tends to slightly reduce prices, with a central adjustment of about -1.5. This suggests that the model may consider the current market price to be slightly high, and that a small price reduction can more effectively stimulate demand while also maintaining a certain degree of stability.

Overall, the main method of dynamic pricing strategy is to make small adjustments to the original price, and the overall trend is to slightly reduce the price. Although the adjustment range is not large, the direction of adjustment is clear – stimulating demand by slightly reducing the price, which may increase the total price. The model maintains a relatively stable price when adjusting the price, avoiding extreme price fluctuations and obvious price fixing behaviour.

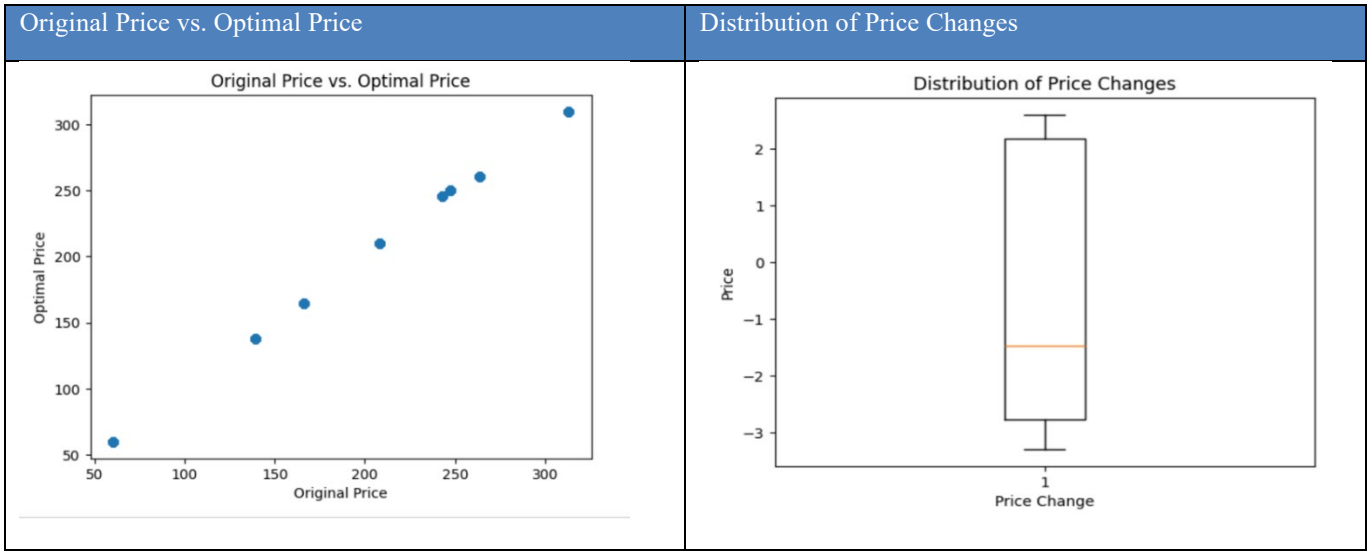

Figure 5.38 Feature Importance of Dynamic Pricing Analysis

5.7 Visualization using PowerBI

Figure 5.39 shows the visualization of flight data and the dashboard designed to provide actionable insights to stakeholders.

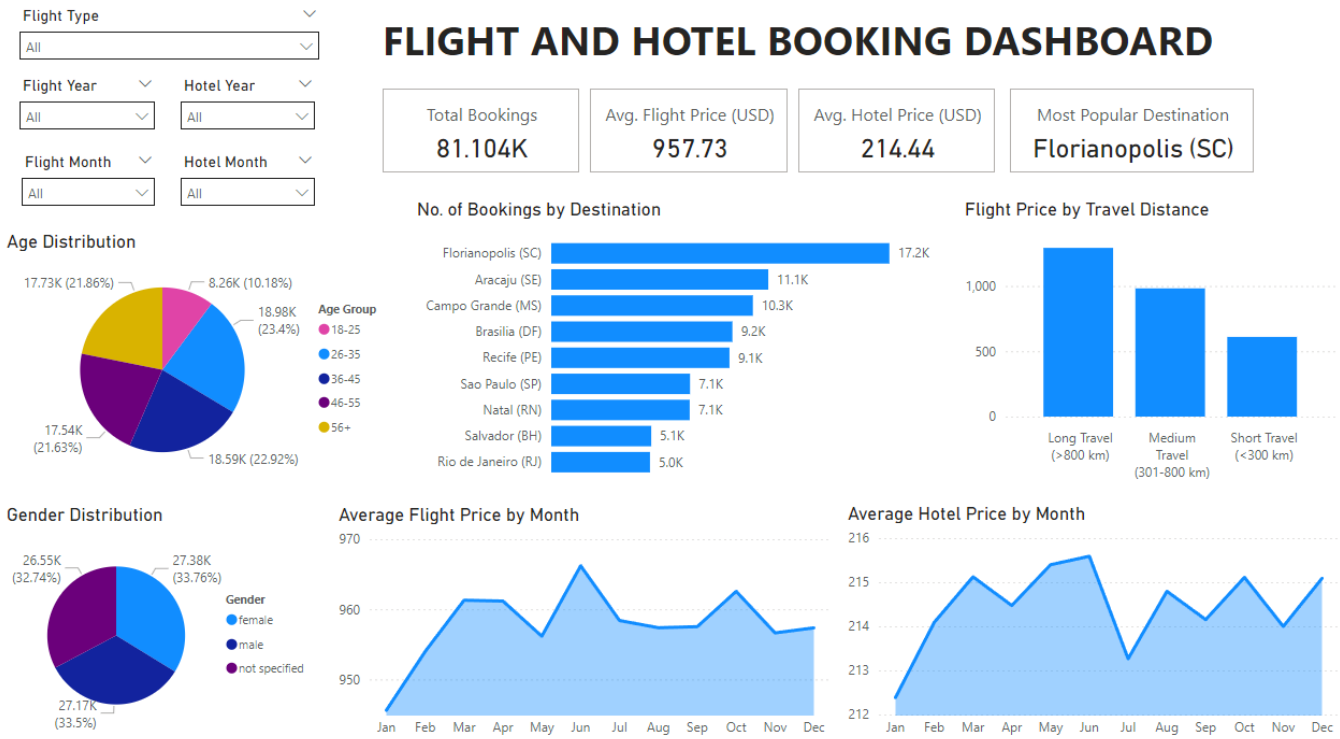

Figure 5.39 PowerBI Visualization

6 CONCLUSION

This project demonstrates the effective use of big data tools like Hadoop, Hive, and Spark to analyze customer booking behavior and optimize pricing strategies. The analysis revealed that middle-aged individuals (31–50 years) dominate bookings, with consistent spending across age groups, and economic flight classes are most preferred for short distances. Seasonal trends highlight peak travel during October and December, with off-peak opportunities in August. Destinations like Salvador and Natal have higher prices, reflecting their premium appeal, while Sao Paulo offers budget-friendly options. Hive proved more efficient than MapReduce for data analysis, delivering faster execution and simplicity. Additionally, K-Means clustering identified four distinct customer segments based on travel frequency and distance, enabling for more personalized pricing. Regression models were used to predict optimal hotel pricing, improving pricing accuracy and responsiveness. The Random Forest model delivered the best performance in predicting hotel prices, achieving the lowest RMSE of 20.31 and an R-squared value of 0.931. These insights enable targeted marketing, optimized pricing, and improved customer satisfaction, such as bundling promotions during peak seasons or offering discounts off-peak. Future work could integrate real-time data and advanced machine learning for deeper behavioral insights and dynamic pricing.

7 REFERENCES

[1] J. Li, L. Xu, L. Tang, S. Wang, and L. Li, "Big data in tourism research: A literature review," *Tour Manag*, vol. 68, pp. 301–323, 2018, doi: https://doi.org/10.1016/j.tourman.2018.03.009.

[2] L. Ardito, R. Cerchione, P. Del Vecchio, and E. Raguseo, "Big data in smart tourism: challenges, issues and opportunities," Sep. 14, 2019, *Routledge*. doi: 10.1080/13683500.2019.1612860.

[3] P. Atri, "Global Market Size, Forecast, and Trend Highlights Over 2025 - 2037," Research Nester . Accessed: Jan. 03, 2025. [Online]. Available: https://www.researchnester.com/reports/tourism-industry-market/109

[4] "About Agoda," Agoda. Accessed: Jan. 03, 2025. [Online]. Available: https://www.agoda.com/about-agoda/

[5] N. Kaveevivitchai, "Boom at the inn," Bangkok Post. Accessed: Jan. 03, 2025. [Online]. Available: https://www.bangkokpost.com/business/general/363111/boom-at-the-inn

[6] Q. Shambour, A. Adel, and M. Abualhaj, "A Hotel Recommender System Based on Multi-Criteria Collaborative Filtering," *Information Technology and Control*, vol. 51, pp. 390–402, Jan. 2022, doi: 10.5755/j01.itc.51.2.30701.

[7] P. Latane, P. Dhakate, V. S. Ubale, and D. Mantri, "Real-time Data Visualization for the IOT Application using Tableau Platform," in *2024 International Conference on Emerging Smart Computing and Informatics (ESCI)*, RisingWave, Jan. 2024, pp. 1–5. doi: 10.1109/ESCI59607.2024.10497376.

[8] The Apache Software Foundation, "Apache Software Foundation Announces Apache® Hive 4.0," The Apache Software Foundation Blog. Accessed: Jan. 03, 2025. [Online]. Available: https://news.apache.org/foundation/entry/apache-software-foundation-announces-apache-hive-4-0?utm_source=chatgpt.com

[9] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communication of the ACM*, vol. 5.1, pp. 107–113, Jan. 2008.

[10] I. A. T. Hashem, N. B. Anuar, A. Gani, I. Yaqoob, F. Xia, and S. U. Khan, "MapReduce: Review and open challenges," *Scientometrics*, vol. 109, no. 1, pp. 389–422, Oct. 2016, doi: 10.1007/s11192-016-1945-y.

[11] S. N. Khezr and N. J. Navimipour, "MapReduce and Its Applications, Challenges, and Architecture: a Comprehensive Review and Directions for Future Research," Sep. 01, 2017, *Springer Netherlands*. doi: 10.1007/s10723-017-9408-0.

[12] J. Holdsworth, "What is Hadoop Distributed File System (HDFS)? | IBM," IBM Thinks. Accessed: Jan. 05, 2025. [Online]. Available: https://www.ibm.com/think/topics/hdfs

[13] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1–10, 2010, [Online]. Available: https://api.semanticscholar.org/CorpusID:13925042

[14] "Apache Hadoop," Apache Software Foundation. Accessed: Jan. 05, 2025. [Online]. Available: https://hadoop.apache.org/

[15] "Hadoop Distributed File System (HDFS)," Databricks Glossary. Accessed: Jan. 05, 2025. [Online]. Available: https://www.databricks.com/glossary/hadoop-distributed-file-system-hdfs

[16] G. Cheng, S. Ying, B. Wang, and Y. Li, "Efficient Performance Prediction for Apache Spark," *J Parallel Distrib Comput*, vol. 149, pp. 40–51, 2021, doi: https://doi.org/10.1016/j.jpdc.2020.10.010.

[17] A. Woodie, "A Decade Later, Apache Spark Still Going Strong," BigDATAwire. Accessed: Jan. 03, 2025. [Online]. Available: https://www.bigdatawire.com/2019/03/08/a-decade-later-apache-spark-still-going-strong/

[18] A. K. Ratnala, K. Inampudi, and T. Pichaimani, "Evaluating Time Complexity in Distributed Big Data Systems: A Case Study on the Performance of Hadoop and Apache Spark in Large-Scale Data Processing," in *Journal of Artificial Intelligence Research and Applications*, London: Scientific Research Center, Mar. 2024, pp. 732–773.

[19] M. A. Salamkar, K. Allam, and J. Immaneni, "The Big Data Ecosystem: An overview of critical technologies like Hadoop, Spark, and their roles in data processing landscapes," *Journal of AI-Assisted Scientific Discovery*, vol. 1, no. 2, pp. 355–377, Sep. 2021, [Online]. Available: https://scienceacadpress.com/index.php/jaasd/article/view/218

[20] A. Rajpurohit, P. Kumar, R. Kumar, and R. Kumar, "A Review on Apache Spark," *SSRN Electronic Journal*, Jan. 2024, doi: 10.2139/ssrn.4492445.

[21] T. Kansal, S. Bahuguna, V. Singh, and T. Choudhury, "Customer Segmentation using K-means Clustering," in *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, Belgaum, India: IEEE, Dec. 2018, pp. 135–139. doi: 10.1109/CTEMS.2018.8769171.

[22] D.-T. Dinh, V.-N. Huynh, and S. Sriboonchitta, "Clustering mixed numerical and categorical data with missing values," *Inf Sci (N Y)*, vol. 571, pp. 418–442, Sep. 2021, doi: 10.1016/j.ins.2021.04.076.

[23] T. M. Ghazal *et al.*, "Performances of K-Means Clustering Algorithm with Different Distance Metrics," *Intelligent Automation & Soft Computing*, vol. 29, no. 3, pp. 735–742, 2021, doi: 10.32604/iasc.2021.019067.
[24] M. Gul and M. A. Rehman, "Big data: an optimized approach for cluster initialization," *J Big Data*, vol. 10, no. 1, p. 120, Jul. 2023, doi: 10.1186/s40537-023-00798-1.
[25] M. Kossakov, A. Mukasheva, G. Balbayev, S. Seidazimov, D. Mukammejanova, and M. Sydybayeva, "Quantitative Comparison of Machine Learning Clustering Methods for Tuberculosis Data Analysis," in *CIEES 2023*, MDPI, Jan. 2024, p. 20. doi: 10.3390/engproc2024060020.
[26] C. Sarkar, D. Gupta, U. Gupta, and B. B. Hazarika, "Leaf disease detection using machine learning and deep learning: Review and challenges," *Appl Soft Comput*, vol. 145, p. 110534, 2023, doi: https://doi.org/10.1016/j.asoc.2023.110534.
[27] L. Wen, K. Zhou, W. Feng, and S. Yang, "Demand Side Management in Smart Grid: A Dynamic-Price-Based Demand Response Model," *IEEE Trans Eng Manag*, vol. 71, pp. 1439–1451, 2024, doi: 10.1109/TEM.2022.3158390.
[28] M. Gunay, M. N. Ince, and A. Cetinkaya, "Apache Hive Performance Improvement Techniques for Relational Data," in *2019 International Artificial Intelligence and Data Processing Symposium (IDAP)*, 2019, pp. 1–6. doi: 10.1109/IDAP.2019.8875898.
[29] X. Wang, Y. Liu, S. Li, and H. Wang, "Peak-Season Price Adjustments in Shared Accommodation: The Role of Platform-Certified Signals and User-Generated Signals," *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 19, no. 2, pp. 1164–1184, 2024, doi: 10.3390/jtaer19020060.
[30] A. Haleem, M. Javaid, M. Asim Qadri, R. Pratap Singh, and R. Suman, "Artificial intelligence (AI) applications for marketing: A literature-based study," *International Journal of Intelligent Networks*, vol. 3, pp. 119–132, 2022, doi: https://doi.org/10.1016/j.ijin.2022.08.005.
[31] S. H. Krishna, K. Kaur, B. Rajalakshmi, S. Lakhanpal, B. Sule, and I. Sumalatha, "Competitive Edge Using Big Data Analytics to Improve Customer Relationship Management," in *2024 International Conference on Communication, Computer Sciences and Engineering (IC3SE)*, Semantic Scholar, 2024, pp. 1553–1557.
[32] K. S. Htet and M. Myint Sein, "Effective Marketing Analysis on Gender and Age Classification with Hyperparameter Tuning," in *2020 IEEE 2nd Global Conference on Life Sciences and Technologies (LifeTech)*, 2020, pp. 247–248. doi: 10.1109/LifeTech48969.2020.1570616797.
[33] S. Lee and B. G. Kim, "Attribute of Big Data Analytics Quality Affecting Business Performance," *Journal of Social Computing*, vol. 4, no. 4, pp. 357–381, 2023, doi: 10.23919/JSC.2023.0028.
[34] Y. Huo and J. Qiao, "Research on the decision of pricing and booking of strategic shippers under the advance booking mode," in *2021 International Conference on Big Data and Intelligent Decision Making (BDIDM)*, 2021, pp. 222–225. doi: 10.1109/BDIDM53834.2021.00053.
[35] L. Pande and S. Sengupta, "Digital Commerce and Big Data revolutionizing the tourism industry: A review article," in *2024 IEEE 9th International Conference for Convergence in Technology (I2CT)*, 2024, pp. 1–5. doi: 10.1109/I2CT61223.2024.10544348.
[36] I. Kisliakova, "Strategy to increase domestic and international direct bookings for a hotel," Karelia University of Applied Sciences, 2022. [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/780989/Kisliakova_Iuliia.pdf?sequence=5
[37] X. Tran, "Strategies of Revenue Management," Sep. 2024, [Online]. Available: https://pressbooks.uwf.edu/revenuemanagementillustrated/chapter/chapter-5/