Big Data

Prepared by : Dania Ramsi Almubiden

Table of content

Introduction

In this project, I explored and analyzed hotel data from three different sources: a JSON file, a CSV file, and a merged dataset created by combining the two. The primary objective was to gain insights into various hotel-related metrics and use this information to predict prices based on various features. I utilized big data tools and machine learning models to process, analyze, and visualize the data effectively.

Data Collection and Processing

JSON File

The JSON file provided information on various hotels, including their ratings, prices, and locations. To begin, I loaded the data, which took approximately 0.04 seconds. The JSON data contained three key collections: airbnbHotels, bookingHotels, and hotelsComHotels.

```
JSON Data Load Time: 0.04 seconds
dict_keys(['airbnbHotels', 'bookingHotels', 'hotelsComHotels'])
```

I ensured that there were no missing values in essential columns like rating and price.value before proceeding. The initial statistical description revealed that some columns had missing values, which were later handled through imputation or deletion, depending on the context.

Summary statistics for numerical columns:

	stars	distanceFromCenter	price.value	price.taxesAndCharges
count	450.000000	500.000000	500.000000	42.000000
mean	3.333333	91.976000	178.030000	68.809524
std	0.934372	159.061889	141.123694	69.751209
min	1.000000	0.500000	40.000000	5.000000
25%	3.000000	1.200000	98.000000	28.500000
50%	3.000000	4.450000	148.000000	49.000000
75%	4.000000	150.000000	205.250000	82.250000
max	5.000000	500.000000	1509.000000	438.000000

CSV File

The CSV file provided extensive data on hotel bookings, including information on lead_time, total_stay, price_per_person, and is_canceled. I started by cleaning and processing the data, creating new columns such as total_stay and booking_window to assist in further analysis.

```
# Create new columns using withColumn
df = df.withColumn('total_stay', col('stays_in_weekend_nights') + col('stays_in_week_nights'))
df = df.withColumn('booking_window', col('lead_time'))
```

Merged File

The merged file was created by joining the JSON and CSV files on the common price column. This was done to analyze the relationship between ratings and prices, among other variables, across different data sources. The data loading time for the merged dataset was approximately 1.84 seconds.

The JSON data was flattened, and the CSV data had missing values handled appropriately before the merge. The final dataset allowed me to conduct in-depth analyses, such as computing the average rating for each hotel type:

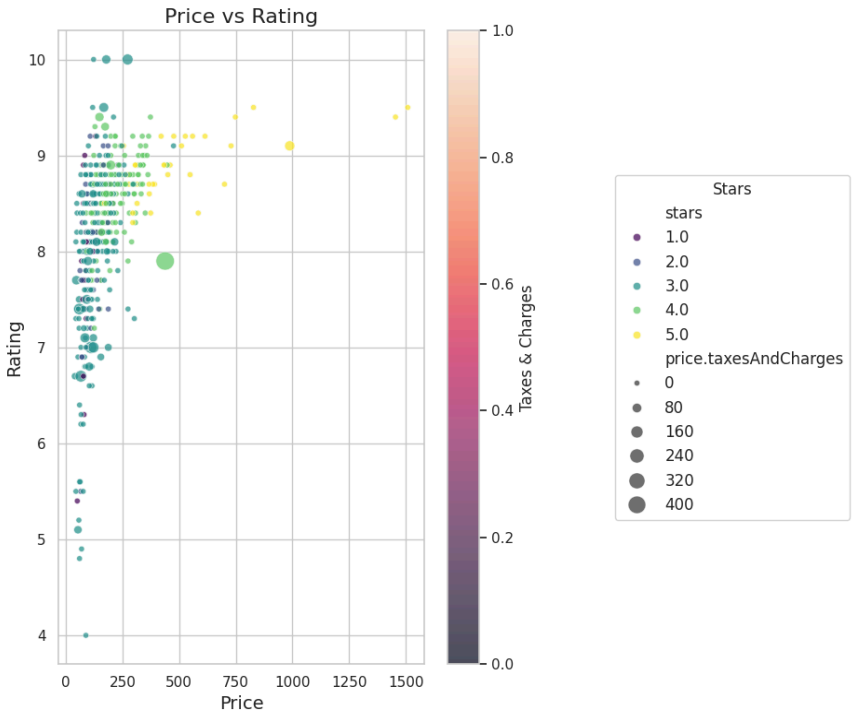
```
+------------+-----------------+
|       hotel|       avg_rating|
+------------+-----------------+
|  City Hotel|4.671754415195507|
|Resort Hotel|4.694823675800993|
+------------+-----------------+
```

Data Manipulation and Aggregation

Once the data was cleaned and prepared, I performed various data manipulations to gain meaningful insights. For example, the CSV file allowed the aggregation of data by hotel type, leading to insights like the average adr per hotel type.

```
df = df.withColumn('price_per_person', col('adr') / (col('adults') + col('children') + 1))
```

Furthermore, I measured the computing and storage resources during data processing. For instance, memory usage was around 16.7%, and CPU usage was 12.6%, indicating efficient resource management during the analysis.



Measuring Loading Time, Computing Resources, and Storage

Understanding the performance metrics during data loading and processing is crucial in any big data project. This section outlines the loading times, CPU and RAM utilization, and storage capacity used when handling the JSON, CSV, and merged datasets.

1. Loading Time

Efficient loading of large datasets is essential in data processing workflows. For this project, loading time was measured for the JSON, CSV, and merged datasets, with results showing the following:

- **JSON File**: Loading Time – 421.97 seconds
- **CSV File**: Loading Time – 549.11 seconds
- **Merged File**: Loading Time – 1382.32 seconds

The increase in loading time for the merged file was expected, given the larger size and complexity of combining two datasets. Merging introduces additional computational overhead, leading to longer ingestion times.

2. Computing Resources

Computing resources, especially CPU and RAM, are key to ensuring efficient data processing. The following metrics were recorded during the project:

- **JSON File**:
    - CPU Usage: 33.6%
    - RAM Usage: 14.8%
- **CSV File**:
    - CPU Usage: 35.0%
    - RAM Usage: 27.1%
- **Merged File**:
    - CPU Usage: 50.4%
    - RAM Usage: 26.1%

The merged file, due to its larger size and more complex operations, required significantly more CPU power. However, the RAM usage remained stable, indicating efficient memory management despite the larger data size.

3. Storage Resources

Storage availability is a critical factor in big data processing, particularly when handling large datasets. The following disk usage metrics were observed:

- **JSON File**:
    - Disk Space Usage: 31% (Used: 33GB, Available: 75GB)
- **CSV File**:
    - Disk Space Usage: 32% (Used: 34GB, Available: 74GB)
- **Merged File**:
    - Disk Space Usage: 32% (Used: 34GB, Available: 74GB)

4. Summary and Interpretation

The JSON and CSV files had manageable loading times and resource usage. However, the merged dataset, with its added complexity, required significantly more CPU power and time. Efficient resource management ensured smooth processing of even the most complex data without exceeding system limits.

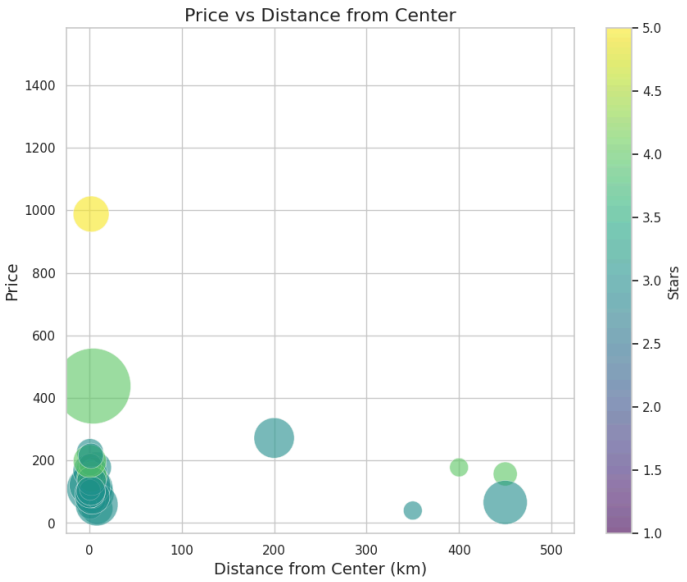
Statistical Analysis and Visualization

JSON File Analysis

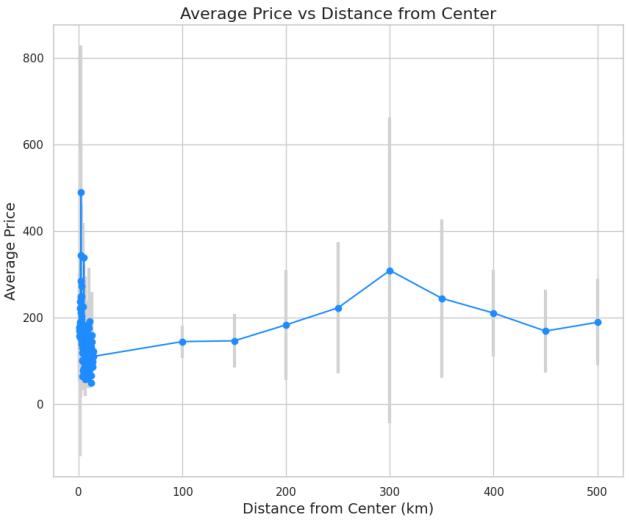
I performed several exploratory data analysis (EDA) steps on the JSON data. This included plotting the distribution of prices and examining the relationship between price and rating.

Below are some of the visualizations:

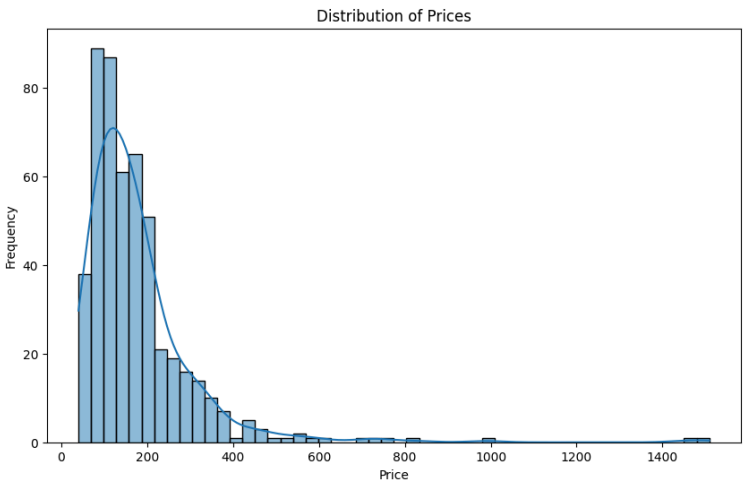
- **Price vs. Distance:**



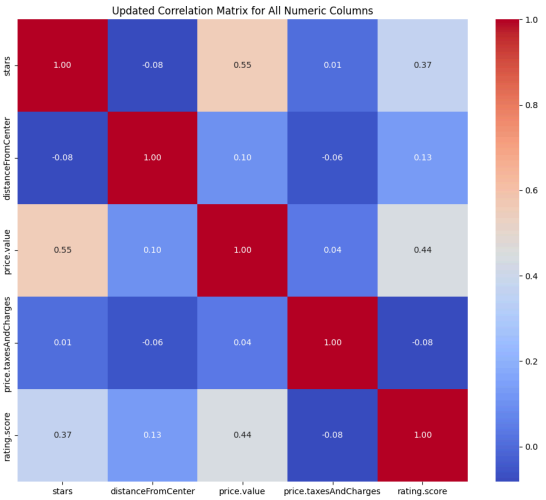
- **Average Price vs. Distance:**



- **Distribution of Prices:**



- **Correlation Matrix:**

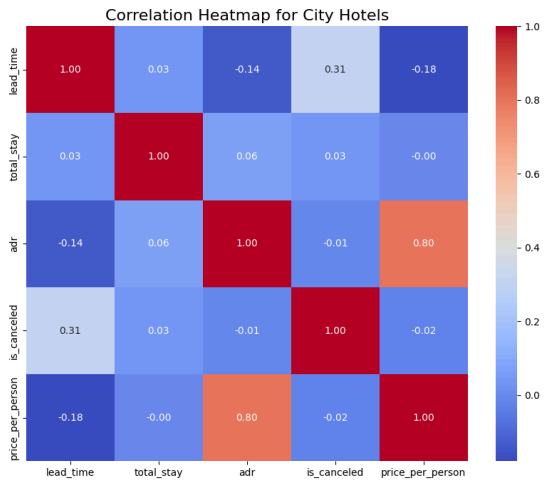


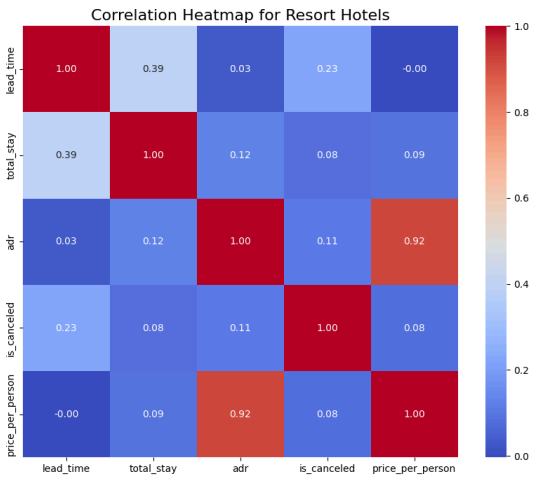
CSV File Analysis

In the CSV file, I performed correlation analyses to understand how different features influenced `adr` and booking cancellations.

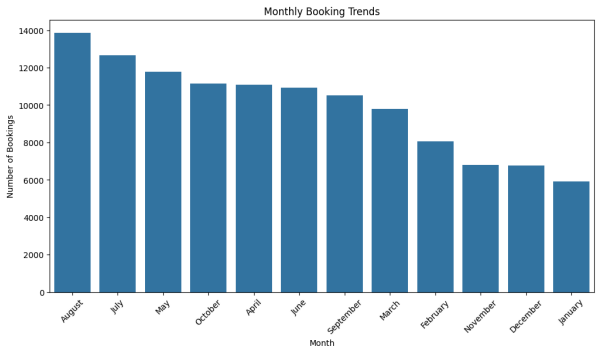
Some key visualizations include:

- **Correlation Heatmap for City Hotels:**

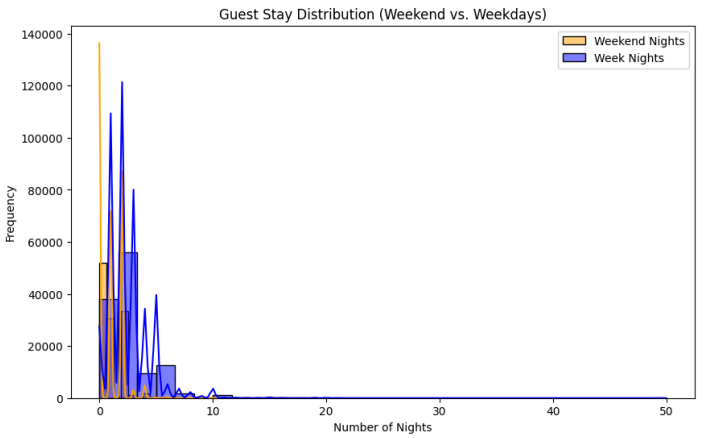




- **Monthly Booking Trends:**

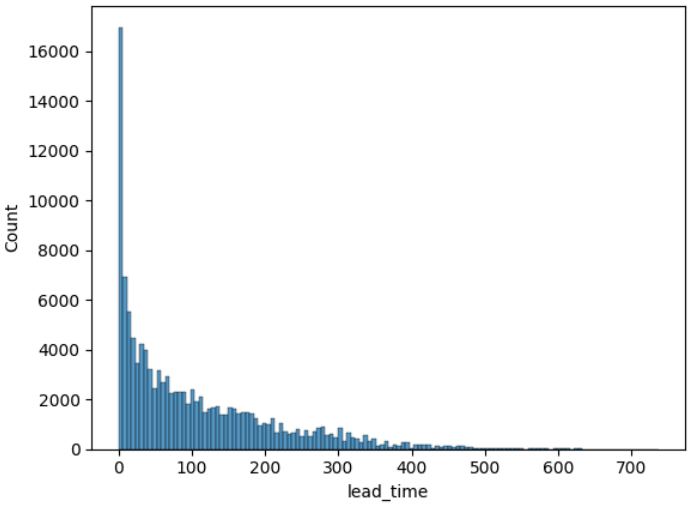


- **Gusset Stay Distribution:**



- **Booking Cancellation by Hotel Type**

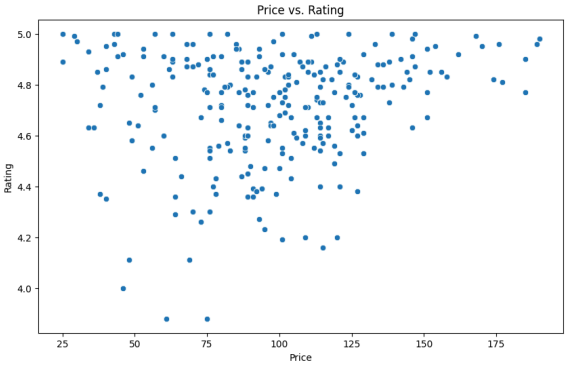
- **Booking Cancellation by Hotel Type**



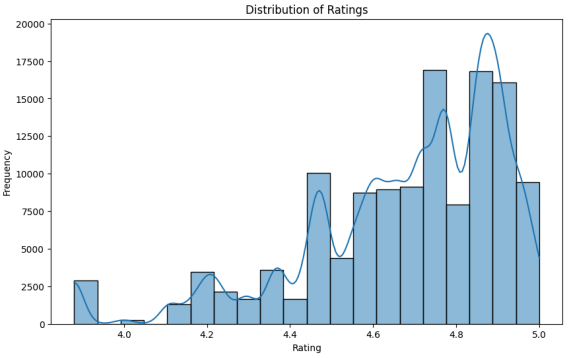
Merged File Analysis

The merged file analysis was crucial in understanding the relationship between price and rating across different hotel types. A scatter plot and a histogram were created to visualize these relationships:

- ***Price vs. Rating Scatter Plot:***



- ***Distribution of Ratings:***



Machine Learning Model Application

In this project, I applied machine learning models to each of the three datasets—JSON, CSV, and the merged dataset— to predict hotel prices and understand the factors influencing these prices. Below, I detail the approach taken for each dataset, including the models used, their performance, and the insights gained.

1. JSON File Model

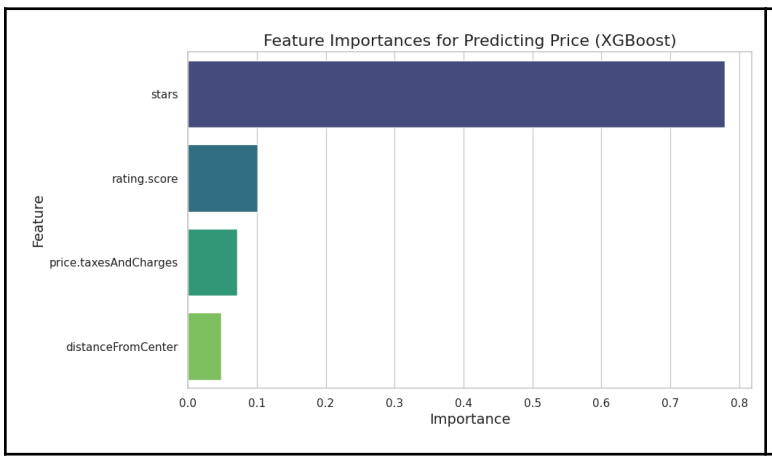
Data Overview: The JSON file contained hotel data, including features such as the distance from the center, star ratings, taxes, and overall rating scores. After performing data cleaning and EDA (Exploratory Data Analysis), I proceeded to build a model to predict the price of a hotel based on these features.

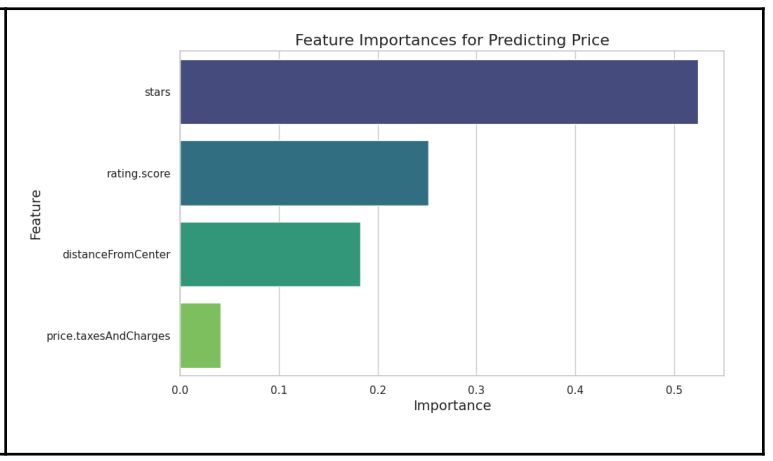
Model Applied:

- **Random Forest Regressor**
  - RMSE: 53.08
  - R^2: 0.63
- **XGBoost Regressor**
  - RMSE: 63.22
  - R^2: 0.52

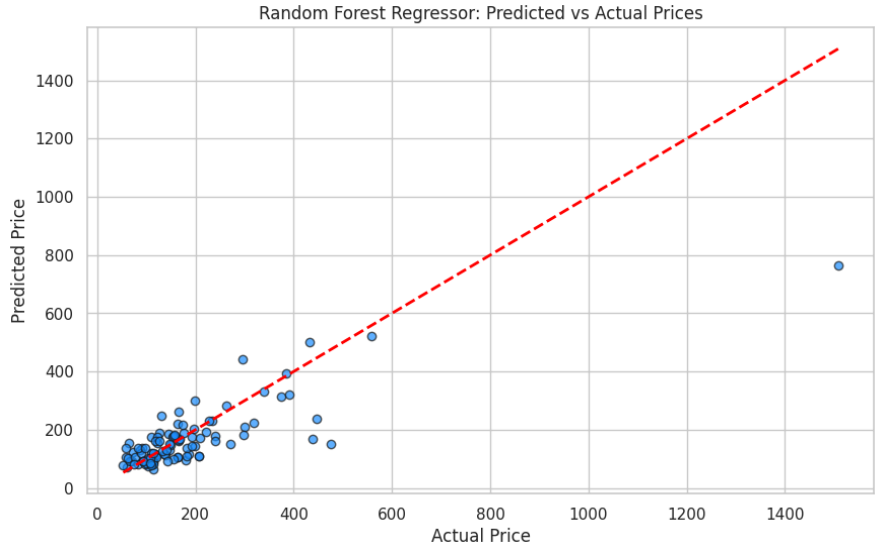
Feature Importances: The most important features for predicting the price in the JSON dataset were:

- **Stars** (52.45% importance)
- **Rating Score** (25.18% importance)
- **Distance From Center** (18.24% importance)





Model Performance: Among the models tested, the Random Forest Regressor performed better, with a lower RMSE and a higher R^2 value, indicating that it could more accurately predict the hotel prices based on the provided features. The XGBoost model, although powerful, was slightly less effective in this scenario.

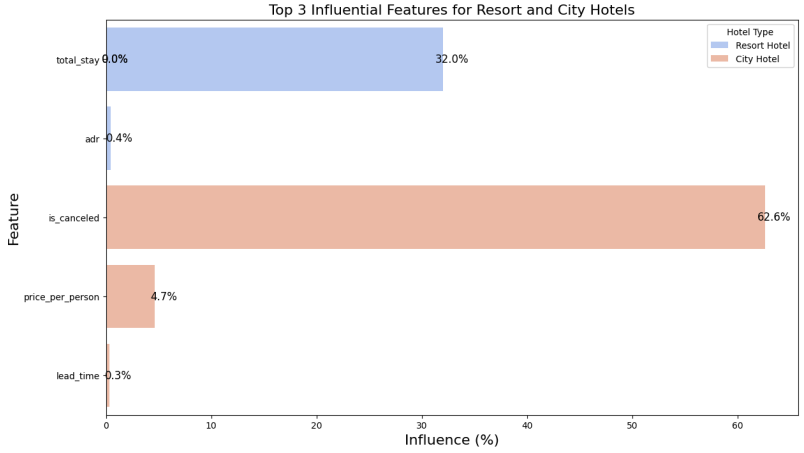


2. CSV File Model

Data Overview: The CSV file contained detailed booking information, including the lead time, the total stay, ADR (Average Daily Rate), and whether the booking was canceled. This data was used to determine whether a booking was for a city hotel or a resort hotel.

Model Applied:

- **Logistic Regression**
  - Area Under ROC: 0.7097
  - Accuracy: 70.97%
  - Precision: 69.81%
  - Recall: 70.97%
  - F1 Score: 67.09%



Key Findings: The logistic regression model provided good performance in distinguishing between city hotels and resort hotels. The features that contributed most to this prediction included:

- **Lead Time**
- **Total Stay**
- **ADR**
- **Price Per Person**

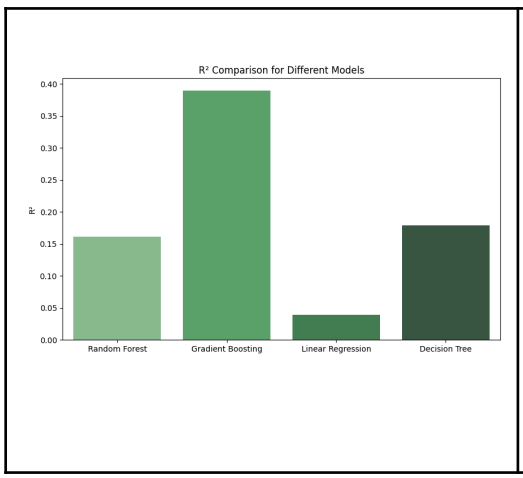
These features influenced the likelihood of a booking being for a city hotel versus a resort hotel, with the model achieving an overall accuracy of 70.97%.

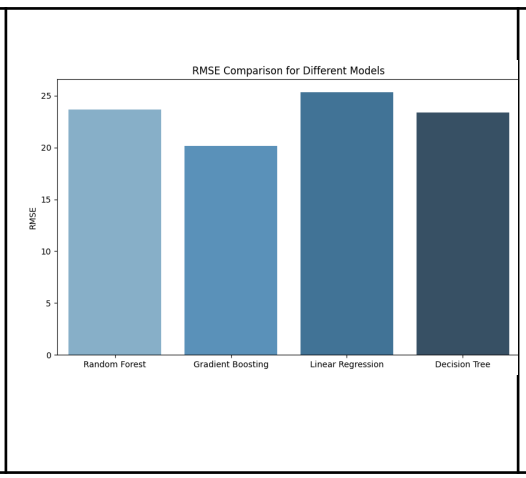
3. Merged Dataset Model

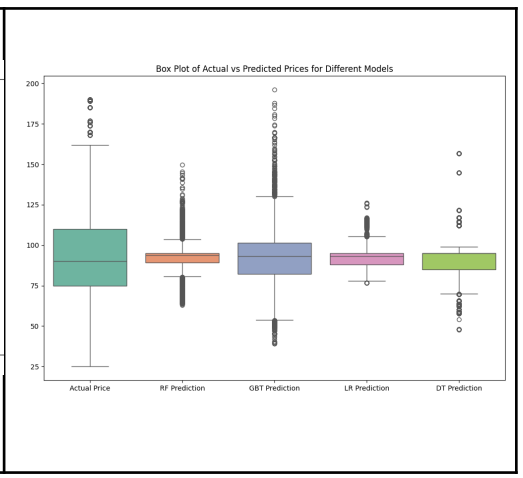
Data Overview: The merged dataset combined features from both the JSON and CSV files, with the common column being the price. This dataset was particularly useful for understanding how features from both booking behavior and hotel characteristics affect pricing.

Model Applied: To find the best model for predicting hotel prices based on the comprehensive set of features from both the JSON and CSV files, I tested several models, including:

- **Random Forest Regressor**
  - RMSE: 23.6535
  - R^2: 0.1608
- **Gradient Boosting Regressor**
  - RMSE: 20.1794
  - R^2: 0.3892
- **Linear Regression**
  - RMSE: 25.3125
  - R^2: 0.0390
- **Decision Tree Regressor**
  - RMSE: 23.3966
  - R^2: 0.1789

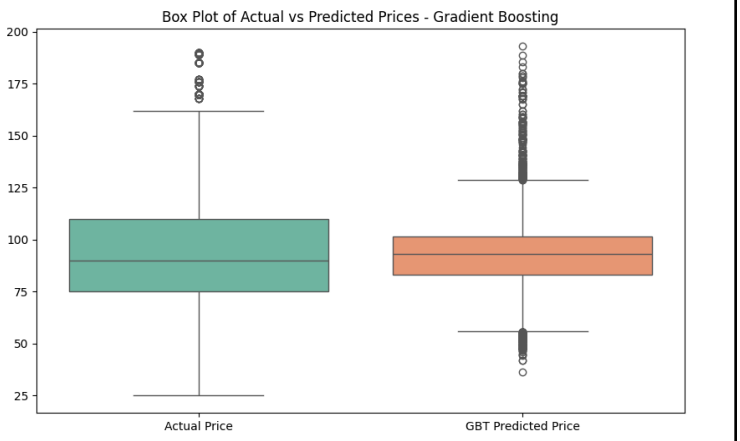


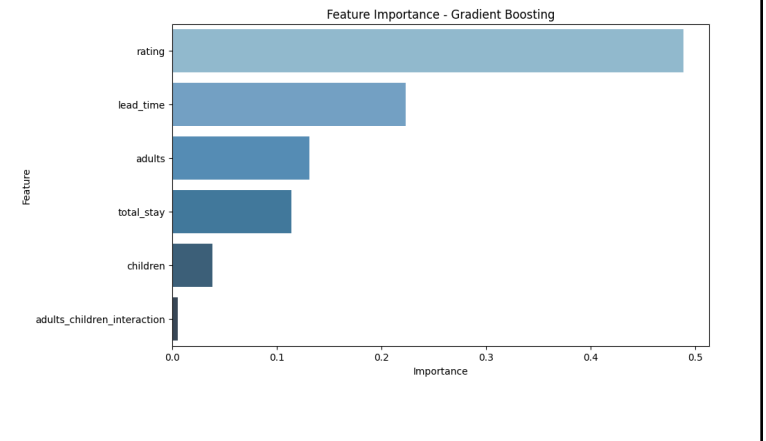




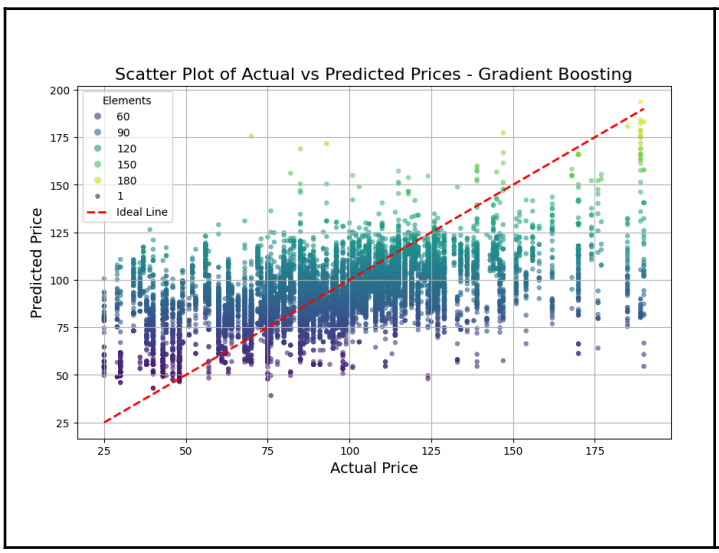
Model Evaluation Metrics: The performance of these models was evaluated using metrics such as RMSE and R^2. Based on the results, the Gradient Boosting Regressor emerged as the best model for this dataset:

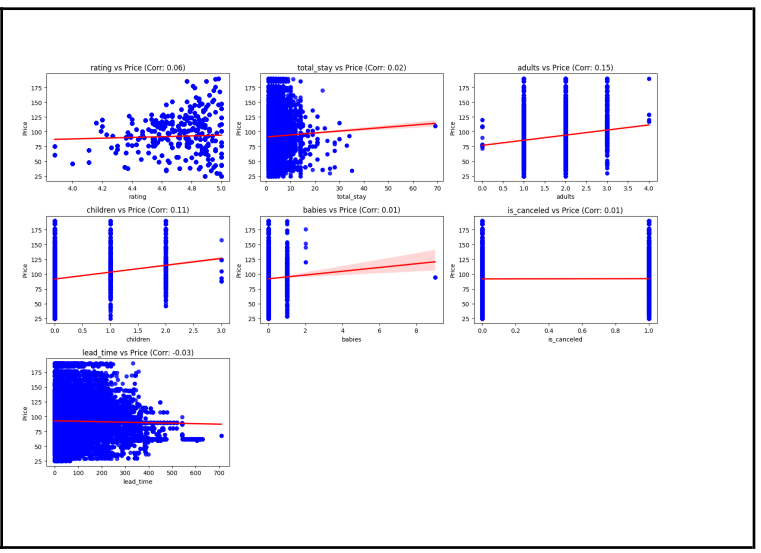
- **Gradient Boosting Regressor**
  - RMSE: 20.3671
  - R^2: 0.3735





Further Refinements: Despite applying further refinements, such as filtering and additional feature engineering, the Gradient Boosting model remained the best performer without significant improvement.





Model Performance: The Gradient Boosting model showed strong performance, with an RMSE of 20.1794 and an R^2 value of 0.3892. This model outperformed the others, providing the best predictions for hotel prices based on the comprehensive set of features from both the JSON and CSV files.

Key Interpretation and Findings

In this section, I will outline the key findings and interpretations from analyzing the three datasets—JSON, CSV, and the merged dataset—using various plotting and visualization approaches. I will also justify the selection of tools and the data preprocessing steps undertaken, and evaluate the effectiveness of the different models applied.

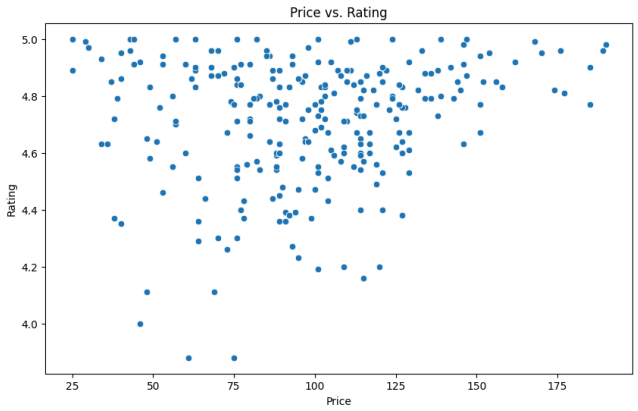
1. JSON File: Hotel Characteristics and Pricing

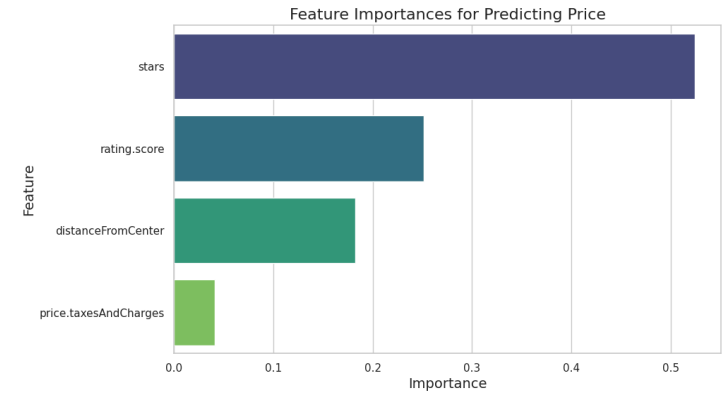
Findings:

- **Feature Importance:** The most influential factors in predicting hotel prices were the star rating of the hotel, the overall rating score, and the distance from the city center. These three features accounted for over 95% of the model's predictive power, with star ratings being the most significant.
- **Price Distribution:** Higher star ratings and better overall ratings generally correlated with higher prices. Hotels closer to the city center also tended to have higher prices, though this relationship was less strong than ratings.

Visualization Approaches:

- **Bar Plot for Feature Importance:** This plot clearly showed the impact of each feature on hotel pricing. By visually representing the importance percentages, it was easy to identify which factors were most critical in pricing decisions.
- **Scatter Plot of Price vs. Rating:** This plot helped visualize the relationship between the hotel ratings and their prices, showing that higher-rated hotels typically charged more.





Justification for Tools and Preprocessing:

- **Random Forest Regressor:** Chosen for its ability to handle feature importance calculation effectively, making it easier to interpret which features were most significant.
- **Data Preprocessing:** Handling missing values in features like price.taxesAndCharges and normalizing the rating.score helped improve the model's performance and accuracy.

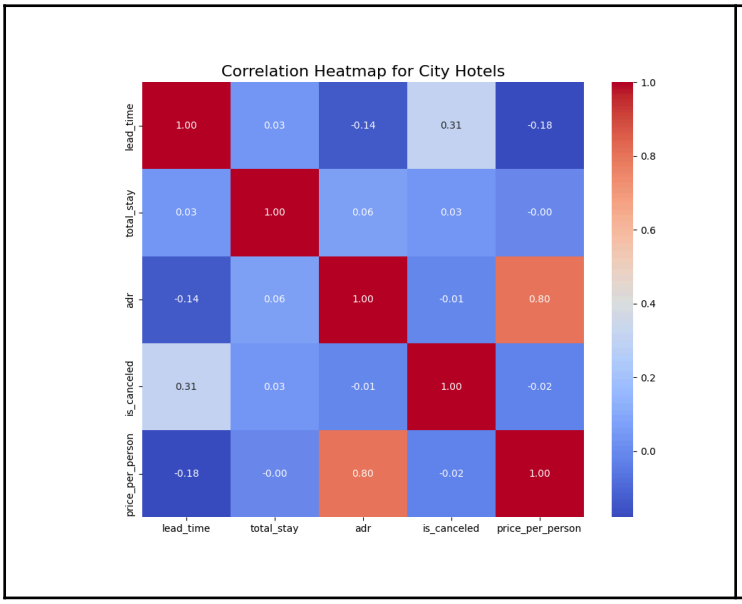
2. CSV File: Booking Behavior and Hotel Type Prediction

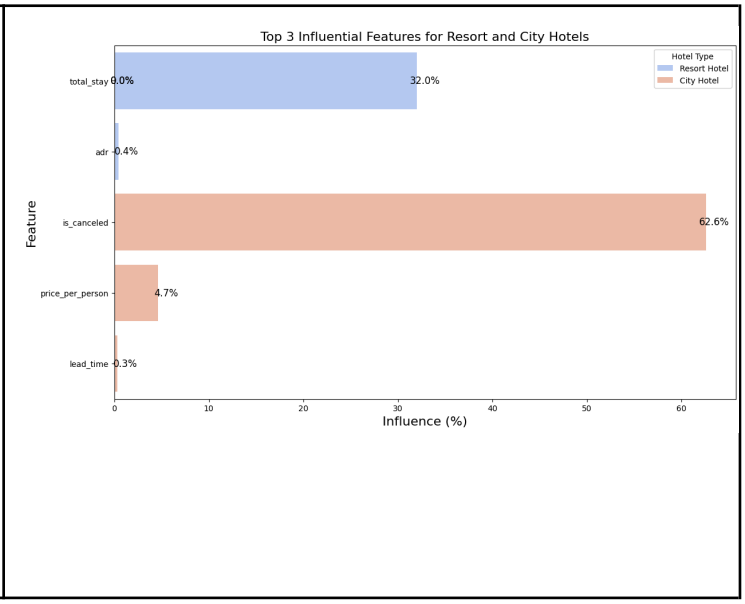
Findings:

- **Correlation Analysis:** The lead_time had a slight negative correlation with the ADR, particularly for city hotels, suggesting that longer lead times might lead to lower prices.
- **Prediction of Hotel Type:** Using features such as lead time, total stay, ADR, and price per person, the logistic regression model was able to predict whether a booking was for a city hotel or a resort hotel with reasonable accuracy (around 70%).

Visualization Approaches:

- **Heatmap for Correlation:** The heatmap provided a clear visual of the relationships between different features, highlighting weak and strong correlations.
- **Bar Plot for Feature Influence on Hotels:** The bar plot was utilized to assess the impact of various features on city and resort hotels. For city hotels, the most influential factors were *is_canceled* and *price_per_person*, indicating that these aspects heavily determine the nature of bookings. On the other hand, for resort hotels, *total_stay* and *adr* (average daily rate) emerged as the key features. This visualization clearly showcased how different features played a significant role in the classification between city and resort hotels, enabling a direct comparison between the two.





Justification for Tools and Preprocessing:

- **Logistic Regression:** Selected due to its suitability for binary classification tasks and its interpretability. It allowed for a straightforward analysis of how different features influenced the probability of a booking being for a city or resort hotel.
- **Data Preprocessing:** Filling missing values and creating new features such as price_per_person were critical steps in ensuring the model performed well, as they allowed the model to capture the nuances of booking behavior.

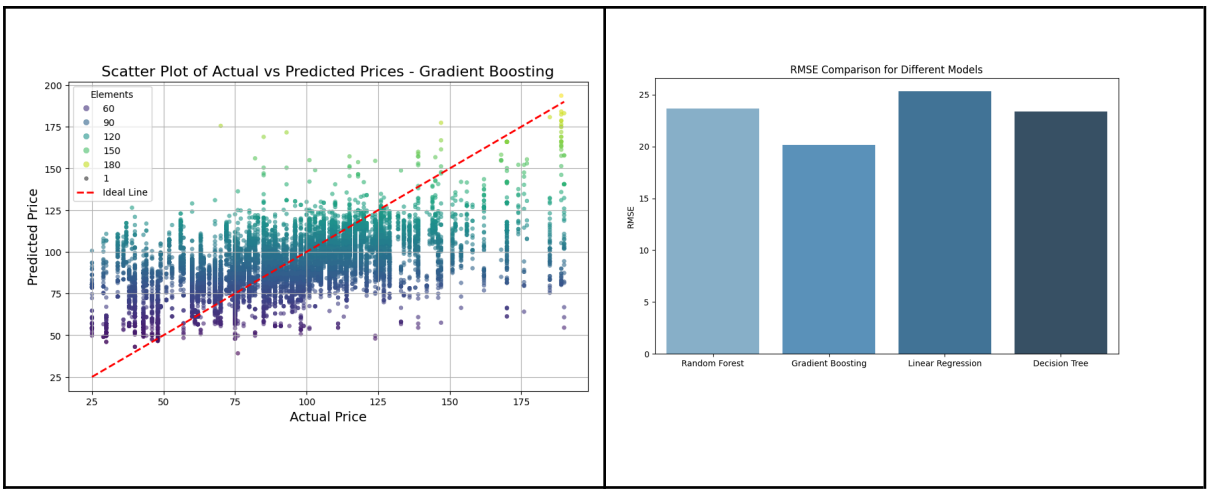
3. Merged Dataset: Comprehensive Price Prediction

Findings:

- **Feature Impact:** In the merged dataset, combining booking behavior with hotel characteristics provided a richer dataset for price prediction. Features such as rating score, total stay, and lead time were found to be the most influential in determining hotel prices.
- **Model Performance:** Gradient Boosting emerged as the best model, with a significantly lower RMSE compared to other models, making it the most effective for predicting prices.

Visualization Approaches:

- **Scatter Plot of Actual vs. Predicted Prices:** This plot demonstrated how closely the model's predictions matched the actual prices, with Gradient Boosting showing the closest fit.
- **Box Plot for Model Comparisons:** By comparing the RMSE and R^2 values across different models, the box plot provided a visual assessment of which model performed best.



Justification for Tools and Preprocessing:

- **Gradient Boosting Regressor:** Chosen for its ability to handle complex, non-linear relationships between features, which is essential in a dataset combining diverse characteristics and booking behavior.
- **Data Preprocessing:** Merging the datasets on the price column required careful handling of missing values and ensuring consistent data types. Feature engineering, such as creating interaction terms between adults and children, further enhanced the model's predictive power.

Evaluation of Model Effectiveness

Model Selection:

- Across the three datasets, the selection of models was driven by the nature of the prediction task—regression for price prediction and classification for hotel type. Random Forest and Gradient Boosting were particularly effective for regression tasks due to their ability to handle complex interactions and non-linearities in the data.

Model Performance:

- The Gradient Boosting model consistently outperformed other models, especially in the merged dataset, where it provided the lowest RMSE and the highest R^2, indicating a strong ability to generalize to new data.
- Logistic Regression was effective for the classification task, providing a good balance between interpretability and performance.

Consideration of Hadoop for the Given Scenario

In this project, where PySpark was used to analyze and model data from JSON and CSV files.

Hadoop vs. PySpark

1. **Data Storage and Management:**
    - **Hadoop (HDFS):** Stores data across multiple nodes, offering high scalability and fault tolerance. However, managing HDFS can be complex.
    - **PySpark:** Provides more flexibility, allowing data processing from various sources, including HDFS, with less complexity.
2. **Data Processing:**
    - **Hadoop (MapReduce):** Effective for batch processing but can be slower due to frequent disk I/O.
    - **PySpark:** Uses in-memory processing with RDDs and DataFrames, making it faster, especially for iterative tasks like machine learning.
3. **Ease of Use:**
    - **Hadoop:** Requires a steeper learning curve and more complex setup.
    - **PySpark:** Offers a user-friendly API similar to pandas, making it easier to perform complex data operations.
4. **Performance:**
    - **Hadoop:** May struggle with iterative algorithms due to disk I/O bottlenecks.
    - **PySpark:** Optimized for in-memory processing, leading to better performance in this scenario.

Conclusion

Given the size and complexity of the data in this project, PySpark was the more suitable choice. It provided efficient data processing, ease of use, and better performance for machine learning tasks. While Hadoop excels in handling massive datasets and distributed processing, PySpark offered the right balance for this use case.

Advantages and Challenges of Data Preparation and Sourcing

Advantages

1. **Comprehensive Data Integration:**
    - The combination of JSON and CSV files allowed for a richer dataset, incorporating various aspects of hotel information, including pricing, ratings, and booking details. This integration provided a more holistic view of the data, which was crucial for accurate analysis and modeling.
2. **Flexibility in Data Processing:**
    - Using tools like PySpark, the data from both JSON and CSV formats was processed seamlessly. PySpark's ability to handle large datasets efficiently made it easier to perform complex transformations and aggregations, leading to better insights.
3. **Effective Data Cleansing:**
    - The data preparation phase involved handling missing values, normalizing data, and creating new features. These steps were crucial in ensuring the quality and reliability of the data, which directly impacted the effectiveness of the machine learning models.
4. **Enhanced Analytical Capabilities:**
    - By preparing the data meticulously, the models could be trained more effectively. The creation of new features like price_per_person and total_stay improved the predictive power of the models, leading to better performance metrics.

Challenges

1. **Data Inconsistencies:**
    - Aligning and merging data from different sources required careful handling of inconsistencies, particularly in common fields like price.
2. **Handling Missing Values:**
    - Both datasets had missing values in critical columns. For example, the JSON data had missing values in the stars and price.taxesAndCharges columns. Deciding how to handle these gaps (e.g., filling with mean values, dropping rows) was a challenge that impacted the final model's performance.
3. **Complex Feature Engineering:**
    - The process of creating new features, such as price_per_person, required careful consideration. This step was not just about creating new columns but ensuring that these features would add value to the models without introducing bias or overfitting.
4. **Data Sourcing Complexity:**
    - Gathering data from different formats (JSON and CSV) and ensuring they were compatible for merging was challenging. The differences in structure and content required additional preprocessing steps to harmonize the data before analysis.

Stages of the Big Data Lifecycle and Its Impact on Decision-Making

1. Data Collection and Ingestion

- **Process:**
  - Data was collected from two sources: a CSV file with hotel bookings and a JSON file with hotel attributes. These datasets were ingested into the processing environment, ensuring accuracy and completeness.
- **Impact:**
  - Reliable data collection is crucial for accurate analysis, forming the basis for informed decision-making within organizations.

2. Data Preparation and Cleansing

- **Process:**
  - Data was cleaned by handling missing values, removing irrelevant columns, and creating new features like `price_per_person`.
- **Impact:**
  - Proper data preparation enhances the quality of analysis, leading to more reliable and actionable insights for decision-making.

3. Data Storage and Management

- **Process:**
  - The cleaned data was stored efficiently, enabling quick access and processing, with careful management of storage resources.
- **Impact:**
  - Efficient data management supports timely decision-making by ensuring data availability and performance.

4. Data Analysis

- **Process:**
  - Statistical and machine learning analyses were conducted, including models like Logistic Regression and Gradient Boosting, to uncover patterns and predict outcomes.
- **Impact:**
  - Analytical insights drive strategic decisions by revealing trends and predicting future outcomes.

5. Data Modeling and Prediction

- **Process:**
  - Machine learning models were trained and evaluated to predict hotel prices, with the Gradient Boosting model selected for its superior performance.
- **Impact:**
  - Predictive modeling provides organizations with foresight, aiding in strategic planning and optimization.

6. Data Visualization and Reporting

- **Process:**
    - Insights were visualized using charts and graphs, making complex data more understandable and actionable.
- **Impact:**
    - Effective visualization facilitates better communication of insights, supporting informed decision-making.

7. Decision-Making and Action

- **Process:**
    - Insights from the analysis were used to guide decisions, such as pricing strategies based on key influencing factors.
- **Impact:**
    - Data-driven decisions improve efficiency and competitive advantage by basing strategies on solid data insights.

Discussion on Hive and Apache Spark: Components, Architecture, and Decision-Making Impact

Apache Hive:

Apache Hive is built on Hadoop and is designed for **data warehousing and batch processing** with a SQL-like interface (HiveQL). It uses:

- **Metastore:** Stores metadata.
- **Driver:** Manages query execution.
- **Execution Engine:** Runs tasks via MapReduce.

Strengths: Ideal for **large-scale data warehousing,** scalable, and familiar for those with SQL experience.

Challenges: High **latency due to MapReduce**, limited for real-time processing.

Apache Spark:

Apache Spark is a fast, in-memory data processing engine. It supports diverse workloads, from batch processing to real-time analytics. Key components include:

- **RDDs:** Immutable distributed datasets enabling fast computations.
- **Execution Engine:** Handles tasks in-memory, making it faster than MapReduce.

Strengths: Speed, versatility, supports real-time analytics, and is easier to use with APIs in multiple languages.

Challenges: High memory usage, complex configuration for optimal performance.

Comparison in Decision-Making:

- **Hive:** Best for batch processing and large-scale SQL queries, but not suitable for real-time analytics.
- **Spark:** Excels in real-time processing, machine learning, and complex data transformations but requires more resources and careful tuning.

Advantages and Challenges:

- **Advantages:** Both tools offer scalability and integration with Hadoop, making them powerful for big data processing.
- **Challenges:** Managing these tools can be complex and costly, with Hive introducing latency and Spark requiring high memory.

Conclusion:

I this scenario, Spark's speed, and flexibility would be more effective for real-time analytics, while Hive remains valuable for large-scale batch processing. The choice depends on the specific needs of your data processing tasks.