Table of Content

Table of figures

Table of Table

1. Introduction

This report provides a comprehensive analysis of booking data for a city hotel and a resort hotel. The dataset includes booking information such as booking date, length of stay, number of guests, parking availability, and other relevant details. This analysis aims to uncover insights and trends that can guide strategic decisions for the hospitality industry.

2. Data Cleaning

In the realm of data analytics, the initial step often involves rectifying missing values in a dataset, a crucial task for data analysts and scientists. The presence of missing values can potentially impact the performance of machine learning models. To delve into the process of addressing missing values,

The framework for this data analysis comprises the following components:

- Visual Studio Code (VS Code) for code editing
- Python 3.10
- Jupyter Notebook
- Anaconda for streamlined package management.

The Dataset is in CSV format, necessitating the use of the 'pandas' package. The 'read_csv' function provided by pandas facilitates the transformation of CSV data into a DataFrame. The code snippet below accomplishes this:

```
import pandas as pd
hotel_raw = pd.read_csv('hotel_bookings.csv', header=0)
print(hotel_raw.head())
```

Figure 1

2.1 Identifying Columns with Missing Values

A function, 'missing_value', was defined to discern columns containing missing values along with the number and percentage of such values. The following is the code snippet for this function:

```python
def missing_value(df):
    col_na = list(df.columns[df.isna().any()])
    missing_numbers = list(df[col_na].isna().sum())
    ratio_na = list(map(lambda num: (num / len(df)) * 100, missing_numbers))
    final_na = pd.DataFrame({'Column_name': col_na, 'Missing_num': missing_numbers, 'Ratio': ratio_na})
    print(final_na)       You, 6 hours ago • init
```

Figure 2

Invoking 'missing_value(hotel_raw)' produced the subsequent output:

	Column_name	Missing_num	Ratio
0	children	4	0.003350
1	country	488	0.408744
2	agent	16340	13.686238
3	company	112593	94.306893

Figure 3

The output underscores that the 'company' column registers the highest frequency of missing values, while the 'children' column has the fewest. This information prompts us to address the 'company' column in more detail.

2.2 Handling Company and Agent Columns

Upon thorough examination of the dataset's description provided on the website, it became apparent that the 'company' and 'agent' columns contained numerical identifiers for the company and agent associated with each hotel reservation. Notably, a significant number of missing values were observed in these columns. This observation stemmed from the fact that numerous reservations were made without specifying a particular company or agent. The use of the 'np.nan' value in these columns signifies that no company or agent was utilized for the booking. To enhance the integrity of these columns, two essential actions were undertaken

```python

# Replace missing values in 'company' column with "Not_using_company"
hotel_raw['company'].fillna("Not_using_company", inplace=True)

# Replace missing values in 'agent' column with "Not_using_agent"
hotel_raw['agent'].fillna("Not_using_agent", inplace=True)

```

Figure 4

2.3 Handling the Children Column

Upon careful examination of the dataset, it was observed that the "children" column contains only 4 missing values. Further analysis revealed that these missing values correspond to cases where there were no children accompanying the individuals who

made the room reservations. Considering that most entries in this column are already indicated by the value 0, it is appropriate to replace the missing values (denoted as np.nan) with the value 0. This adjustment enhances the accuracy and clarity of the data representation. The replacement process is accomplished using the following code snippet:.

```
# Replace missing values in 'children' column with 0
hotel_raw.loc[hotel_raw['children'].isna(), 'children'] = 0
```

Figure 5

2.4 Handling the Country Column

Continuing with our data analysis process, we've reached the point of addressing missing values in the 'country' column. To make an informed decision, it's crucial to assess the impact of missing values on our dataset. Upon a thorough examination, we observed that most entries in the 'country' column are complete, with only 488 instances being incomplete. Considering the relatively small proportion of missing values and the potential influence on our analysis, we've opted to adopt a pragmatic approach. Instead of attempting to impute the missing data, we've decided to remove the corresponding rows from our dataset. This choice helps maintain the integrity of our analysis by ensuring that the remaining data is representative and accurate.

To proceed with this strategy by employing the drop method to eliminate the rows with missing values in the 'country' column. This step will allow me to continue this analysis. with a more refined and reliable dataset

```
# Analyze 'country' column and drop rows with missing values
print("Before dropping rows:", len(hotel_raw))
hotel_raw = hotel_raw.drop(hotel_raw[hotel_raw['country'].isna()].index)
print("After dropping rows:", len(hotel_raw))
```

Figure 6

2.5 Conclusion of Missing Data and Data Wrangling.

Removing missing values from a dataset constitutes an integral phase in data analysis. Addressing this issue ensures the reliability of subsequent analyses and machine learning model outcomes. Through the implementation of data preprocessing techniques, such as data type conversion, value replacement, and data dropping, we have effectively managed missing values in the Hotel Booking dataset.

3. Classification Performance Analysis

I evaluated the performance of a classification model on predicting booking cancellations. The confusion matrix and classification report provide insights into the model's effectiveness:

Table 1

Confusion Matrix:	**Classification Report:**
True Negative (TN): 12957	Precision: 0.76
False Positive (FP): 1950	Recall: 0.70
True Positive (TP): 6278	F1-Score: 0.73
False Negative (FN): 2693	Accuracy: 0.81

The model shows good precision and recall, achieving an accuracy of 81%. This means that the model effectively predicts both cancellations and successful bookings.

3.1 Special Requests Analysis

The dataset reveals insights into special requests made by guests:

- Total Special Requests: 68215
- Average Special Requests per Booking: 0.57

This data indicates that guests commonly make special requests, with an average of 0.57 requests per booking.

```
34  # Calculate total special requests and average special requests
35  special_requests_info = hotel_raw['total_of_special_requests'].agg(['sum', 'mean'])
36
37  # Print the special requests information
38  print("\nTotal Special Requests and Average Special Requests:")
39  print(special_requests_info)
40
```

Figure 7

3.2 Average Daily Rate for Last 3 Months

The analysis of the average daily rate for the last three months indicates that the average daily rate was approximately $143.47. This insight can help the hotel management in setting competitive pricing strategies.

```
# Analyze average daily rate for the last 3 months
hotel_raw['arrival_date'] = pd.to_datetime(hotel_raw['arrival_date_year'].astype(str) + '-' + hotel_raw['arrival_date_month'] + '-01')
last_3_months = (hotel_raw['arrival_date'] >= '2017-06-01') & (hotel_raw['arrival_date'] <= '2017-08-31')
average_daily_rate_last_3_months = hotel_raw.loc[last_3_months, 'adr'].mean()

# Print the average daily rate for the last 3 months
print("\nAverage Daily Rate for Last 3 Months:", average_daily_rate_last_3_months)
```

Figure 8

3.3 Total Revenue Lost due to Cancellations.

Based on the analysis, the total revenue lost due to cancellations is approximately $4,503,379.52. This information emphasizes the financial impact of cancellations and underscores the need for strategies to minimize them.

```
# Calculate total revenue lost due to cancellations
total_cancellations = hotel_raw['is_canceled'].sum()
average_daily_rate = hotel_raw['adr'].mean()
total_revenue_lost = total_cancellations * average_daily_rate

# Print the total revenue lost
print("Total Revenue Lost due to Cancellations:", total_revenue_lost)
```

Figure 9

3.4 Cancellation Patterns by Customer Type

Cancellation patterns were analysed based on customer types:

- Contract: Total Cancellations: 1262, Average Cancellations: 0.31
- Group: Total Cancellations: 59, Average Cancellations: 0.10
- Transient: Total Cancellations: 36514, Average Cancellations: 0.41
- Transient-Party: Total Cancellations: 6389, Average Cancellations: 0.25

The analysis highlights varying cancellation rates among different customer types, enabling targeted marketing and service improvement strategies.

```
# Calculate total cancellations and average cancellations for each customer type
cancellation_info = hotel_raw.groupby('customer_type')['is_canceled'].agg(['sum', 'mean'])

# Print the cancellation information
print("\nTotal Cancellations and Average Cancellations by Customer Type:")
print(cancellation_info)
```

Figure 10

3.5 Conclusions of Analysis

This analysis provides valuable insights for decision-making in the hospitality industry. The classification model effectively predicts booking cancellations, and the analysis of special requests, average daily rates, and cancellation patterns offers actionable information for improving operational efficiency and enhancing customer experience.