Hotel Bookings Data Analysis

Rohit Ramichetty rohithra@buffalo.edu
Vrashi Shrivastava vrashish@buffalo.edu

1. Problem Statement

For this Project, we are going to take a Hotel Booking dataset, clean it by handling missing values and managing the format of the data, perform EDA on it to understand it better, and run it through multiple models to draw intelligent information from it regarding the profits revenue, the possible bookings along with cancellations, and customer satisfaction of the hotel to ultimately predict the demand of the hotel and how can it be benefitted through data-driven decision-making.

2. Usability and Target User

The problem of hotel booking demand is significant because it directly affects the revenue and profitability of the hospitality industry. Hotel managers need to accurately predict the demand for their rooms and make decisions regarding pricing, staffing, and inventory management to optimize their profitability. However, predicting demand can be challenging due to factors such as seasonal variations, economic conditions, and unpredictable events such as pandemics or natural disasters.

In addition to the financial impact, hotel booking demand also has implications for customer satisfaction and experience. Overbooking, cancellations, or long wait times can result in negative reviews and loss of future business.

Therefore, understanding the factors influencing hotel booking demand and developing accurate prediction models can provide a competitive advantage for hotels and enhance the overall customer experience. The dataset available on Kaggle provides an opportunity to explore these factors and develop predictive models to address this problem. By analyzing patterns in the data, we can identify the key factors that impact hotel booking demand, develop strategies to optimize inventory management, and ultimately increase profitability and customer satisfaction.

3. Contribution of the Project

The project has the potential to contribute significantly to the problem domain of hotel booking demand by providing insights into the factors that influence the market and developing accurate prediction models. By analyzing the dataset available on Kaggle, we can identify patterns and trends in customer behavior, seasonality, and other external factors that impact hotel booking demand. This analysis can help hotel managers make informed decisions regarding pricing, inventory management, and staffing to optimize their revenue and profitability.

Furthermore, accurate prediction models can help hotels avoid overbooking, reduce cancellations, and improve the overall customer experience. By forecasting demand with greater accuracy, hotels can ensure they have the right number of rooms available, reduce wait times for guests, and minimize the likelihood of disruptions or customer complaints.

The contribution of this project is crucial because hotel booking demand is a complex and dynamic problem. Predicting demand accurately is challenging, but critical for hotel managers to make informed decisions and optimize their profitability. With the increasing availability of data and the development of advanced machine learning algorithms, there is an opportunity to develop more accurate and sophisticated prediction models that can significantly enhance the performance of the hospitality industry.

4. Dataset

We picked the dataset from Kaggle([Hotel booking demand | Kaggle]())
The dataset is called "Hotel Booking Demand". It contains information about hotel bookings, including the type of hotel, the number of guests, the duration of stay, and whether the booking was canceled or not. The dataset is meant to be used for predictive modeling and analysis and is available for download in several formats, including CSV and SQL.
There are 32 columns in the dataset Majority of which is in integer or string format. The data is originally from the article [Hotel booking demand datasets - ScienceDirect](), written by Nuno Antonio, Ana Almeida, and Luis Nunes for Data in Brief, Volume 22, February 2019.

```
[ ] df
```

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	stays_in_week_ni
0	Resort Hotel	0	342	2015	July	27	1	0	
1	Resort Hotel	0	737	2015	July	27	1	0	
2	Resort Hotel	0	7	2015	July	27	1	0	
3	Resort Hotel	0	13	2015	July	27	1	0	
4	Resort Hotel	0	14	2015	July	27	1	0	

Fig 1. Sample of the data

5. Data Cleaning

To Clean the data, we performed the following steps:

A. Removing duplicate rows - In this step, we removed all those rows that contained duplicate values. This takes the redundancy away from the data and that is important because at the time of analysis, the duplicated values won't be calculated twice.

```
# Remove duplicate rows
df.drop_duplicates(inplace=True)
```

There were no duplicated rows left in the dataset after this operation:

```
df.duplicated()
```

```
0         False
1         False
2         False
3         False
4         False
          ...
119385    False
119386    False
119387    False
119388    False
119389    False
Length: 87396, dtype: bool
```

B. Handling missing values - In this step, we managed the missing values from the data to prevent any errors or miscalculation at the time of analysis and data visualization. We are replacing all the missing values with a default value ‘0’.

```
# Replace missing values with a default value (e.g. 0)
df.fillna(0, inplace=True)
```

There were no more null values in the dataset after this operation:

```
print(df.isnull().sum())
```

```
hotel                          0
is_canceled                    0
lead_time                      0
arrival_date_year              0
arrival_date_month             0
arrival_date_week_number       0
arrival_date_day_of_month      0
stays_in_weekend_nights        0
stays_in_week_nights           0
adults                         0
num_children                   0
babies                         0
meal                           0
```

C. Removing useless columns - In this step, we took out the columns that we did not need for the further analysis according to our end goal of the predictor. For our dataset, two of the columns: Company and Agent were not required for us, so we dropped those columns.

```
# Remove irrelevant columns
df.drop(['company', 'agent'], axis=1, inplace=True)
```

The columns were removed after this operation, so if we try to find a company column in the dataset, we could not find it and hit an error:

```
print(df[['company']].to_string(index=False))

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-9-a2aa3564abf9> in <module>
----> 1 print(df[['company']].to_string(index=False))

                              2 frames
/usr/local/lib/python3.8/dist-packages/pandas/core/indexing.py in _validate_read_indexer(self, key, indexer, axis)
   1372                 if use_interval_msg:
   1373                     key = list(key)
-> 1374                 raise KeyError(f"None of [{key}] are in the [{axis_name}]")
   1375 
   1376             not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())

KeyError: "None of [Index(['company'], dtype='object')] are in the [columns]"
```

D. Data type management for a column - In this step, we converted the datatype of the 'reservation_status_date' column to datetime. It was important to better reflect the data in that column so that at the time of EDA, we can perform operations based on the data type.

```
# Convert data type of a column
df['reservation_status_date'] = pd.to_datetime(df['reservation_status_date'])
```

The datatype of the column was changed post this operation:

```
0s
df['reservation_status_date'].dtypes

dtype('<M8[ns]')
```

E. Renaming column names - In this step, we renamed the columns to better reflect the values they hold. It is better for the readability and interpretability of the dataset. We renamed the column "Children" to "num_children" which better reflected the data in that column.

```
# Standardize column names
df.rename(columns={'children': 'num_children'}, inplace=True)
```

The name of the column was changed post this operation:

```
[17] df['num_children']

0         0.0
1         0.0
2         0.0
3         0.0
4         0.0
         ...
119385    0.0
119386    0.0
119387    0.0
119388    0.0
119389    0.0
Name: num_children, Length:
```

F. Finding outliers and removing them - Outliers can be a distraction when it comes to analysis of the data. So, to improve the quality of EDA outcome, we found the outliers in this step.

```
# Detect outliers in a column
Q1 = df['adr'].quantile(0.25)
Q3 = df['adr'].quantile(0.75)
IQR = Q3 - Q1
outliers = (df['adr'] < Q1 - 1.5 * IQR) | (df['adr'] > Q3 + 1.5 * IQR)

print(df.loc[outliers,'adr'])
```

```
396       230.67
523       249.00
526       241.50
584       240.64
641       233.00
           ...
119152    233.00
119247    235.00
119289    236.33
119339    229.00
119365    266.75
Name: adr, Length: 2490, dtype: float64
```

G. Removing the outliers - In the last step, we found some outliers. Because the outliers distract the data analysis, we replaced them in this step with a default value (median in this case).

```
# Replace outliers with a default value (e.g. median)
df.loc[outliers, 'adr'] = df['adr'].median()
```

After this step, the output had consistent values:

```
print(df.loc[outliers,'adr'])
```

```
396       98.1
523       98.1
526       98.1
584       98.1
641       98.1
          ...
119152    98.1
119247    98.1
119289    98.1
119339    98.1
119365    98.1
Name: adr, Length: 2490, dtype: float64
```

H. Stripping trailing spaces - Trailing spaces don't add any value to the data and they occupy resources and space. So, we removed the trailing spaces in this step to make the data clean and consistent.

```
# Remove leading/trailing spaces from string columns
df['country'] = df['country'].str.strip()
```

After this operation, the trailing spaces were removed and length of all the values became consistent:

```
df['country']

0         PRT
1         PRT
2         GBR
3         GBR
4         GBR
         ...
119385    BEL
119386    FRA
119387    DEU
119388    GBR
119389    DEU
Name: country, Length:
```

```
[28] df['country'].str.len()

0         3.0
1         3.0
2         3.0
3         3.0
4         3.0
         ...
119385    3.0
119386    3.0
119387    3.0
119388    3.0
119389    3.0
Name: country, Length: 8
```

I. Replacing inconsistent values - In this step we replaced the inconsistent values to more understandable ones. This step is important because inconsistent values hinder the EDA process.

Before running the operation:

```
df.loc[df['market_segment'] == 'Undefined']
```

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	sta
40600	City Hotel	1	2	2015	August	32	3	
40679	City Hotel	1	1	2015	August	32	5	

2 rows × 30 columns

Operation on dataset:

```
# Replace inconsistent values in a column
df['market_segment'] = df['market_segment'].replace('Undefined', 'Other')
```

After running the operation:

```
df.loc[df['market_segment'] == 'Undefined']
```

hotel	is_canceled	lead_time	arrival_date_year	arri

0 rows × 30 columns

```
df.loc[df['market_segment'] == 'Other']
```

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	sta
40600	City Hotel	1	2	2015	August	32	3	
40679	City Hotel	1	1	2015	August	32	5	

2 rows × 30 columns

J. Removing special characters - In this step we removed the special characters from the data in order to make it ready for analysis. This was an important step because we can not draw any intelligence from the special characters. We found some special characters in the column Reserved_room_type and replaced them with empty strings.w

```
# Remove special characters from string columns
df['reserved_room_type'] = df['reserved_room_type'].str.replace('/', '')
```

There were no values left with the special characters after running this step in the Reserved_room_type column:

```
[37] df.loc[df['reserved_room_type'].str.contains('/')]

      hotel  is_canceled  lead_time  arrival_date_year  arrival_date_
     ------------------------------------------------------------------
     0 rows × 30 columns
```

After these steps, our data was clean enough to proceed further to the EDA steps

6. Exploratory Data Analysis

To understand the data better, we performed the following steps:

A. The head of the dataset(first 5 rows)

```
print(df.head())

          hotel  is_canceled  lead_time  arrival_date_year arrival_date_month  \
0  Resort Hotel            0        342               2015               July
1  Resort Hotel            0        737               2015               July
2  Resort Hotel            0          7               2015               July
3  Resort Hotel            0         13               2015               July
4  Resort Hotel            0         14               2015               July

   arrival_date_week_number  arrival_date_day_of_month  \
0                        27                          1
1                        27                          1
2                        27                          1
3                        27                          1
4                        27                          1

   stays_in_weekend_nights  stays_in_week_nights  adults  ...  \
0                        0                     0       2  ...
1                        0                     0       2  ...
2                        0                     1       1  ...
3                        0                     1       1  ...
4                        0                     2       2  ...
```

B. Information about the dataset

```
print(df.info())

<class 'pandas.core.frame.DataFrame'>
Int64Index: 87396 entries, 0 to 119389
Data columns (total 30 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   hotel                           87396 non-null  object
 1   is_canceled                     87396 non-null  int64
 2   lead_time                       87396 non-null  int64
 3   arrival_date_year               87396 non-null  int64
 4   arrival_date_month              87396 non-null  object
 5   arrival_date_week_number        87396 non-null  int64
 6   arrival_date_day_of_month       87396 non-null  int64
 7   stays_in_weekend_nights         87396 non-null  int64
 8   stays_in_week_nights            87396 non-null  int64
 9   adults                          87396 non-null  int64
 10  num_children                    87396 non-null  float64
 11  babies                          87396 non-null  int64
 12  meal                            87396 non-null  object
 13  country                         86944 non-null  object
```

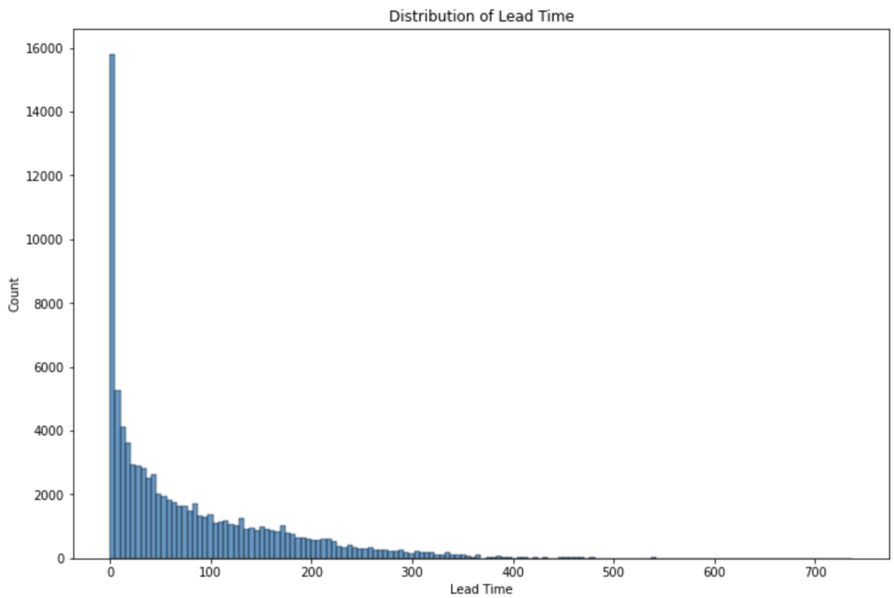
C. Checking for missing values

```
print(df.isnull().sum())

hotel                             0
is_canceled                       0
lead_time                         0
arrival_date_year                 0
arrival_date_month                0
arrival_date_week_number          0
arrival_date_day_of_month         0
stays_in_weekend_nights           0
stays_in_week_nights              0
adults                            0
num_children                      0
babies                            0
meal                              0
```

D. Checking statistics of the numerical values

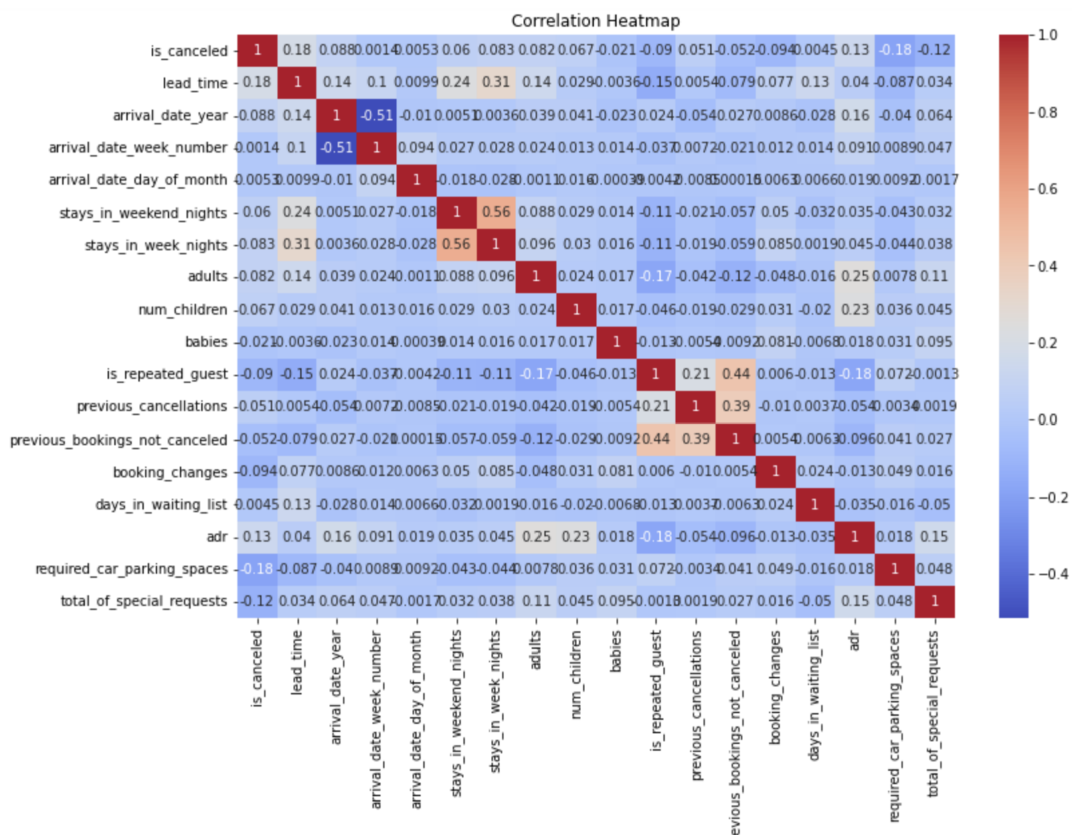
```
print(df.describe())

        is_canceled     lead_time  arrival_date_year  \
count  87396.000000  87396.000000       87396.000000
mean       0.274898     79.891368        2016.210296
std        0.446466     86.052325           0.686102
min        0.000000      0.000000        2015.000000
25%        0.000000     11.000000        2016.000000
50%        0.000000     49.000000        2016.000000
75%        1.000000    125.000000        2017.000000
max        1.000000    737.000000        2017.000000

       arrival_date_week_number  arrival_date_day_of_month  \
count              87396.000000               87396.000000
mean                  26.838334                  15.815541
std                   13.674572                   8.835146
min                    1.000000                   1.000000
25%                   16.000000                   8.000000
50%                   27.000000                  16.000000
75%                   37.000000                  23.000000
max                   53.000000                  31.000000
```

E. Plotting the distribution of lead_time to see the trend in the data

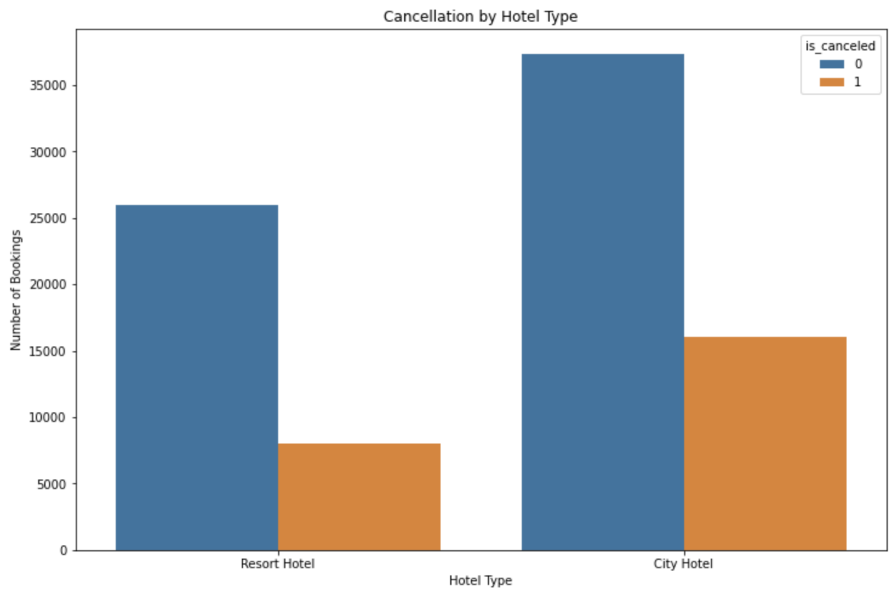


On X-axis: lead_time
On Y-axis: count

F. Visualizing the correlation- The next step we did was checking any correlation exists in the data, and we observed that few exist. We found strong correlation between stays_in_week_nights and stays_in_weekend_nights and also previous_cancellations and is_repeated_guest.

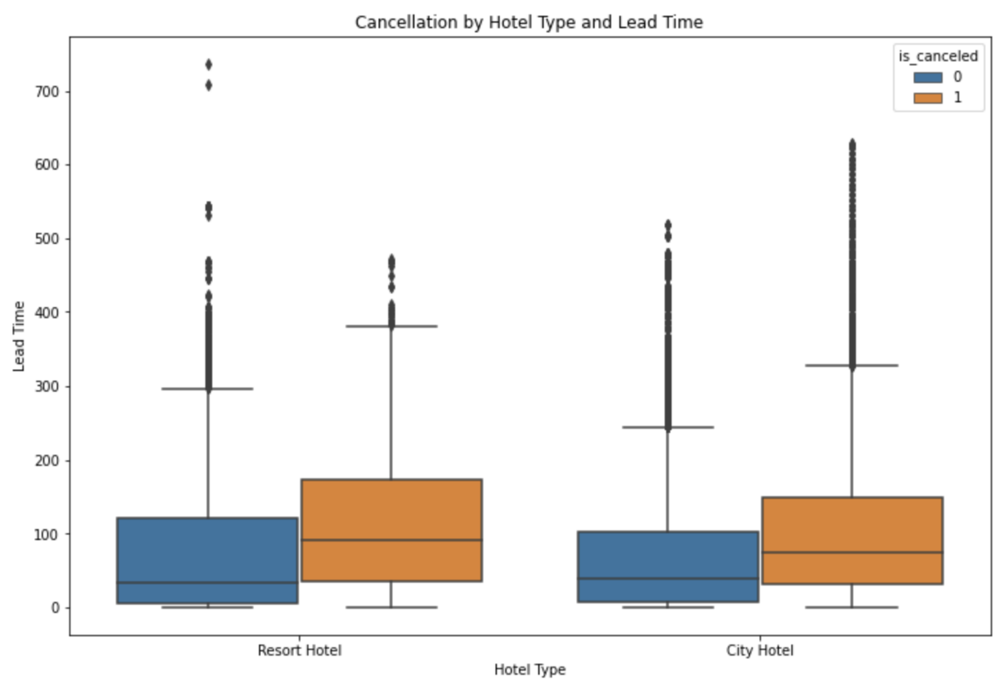


G. Categorizing data based on cancellation and hotel type and plotting a bar graph to represent the number of bookings in different categories.



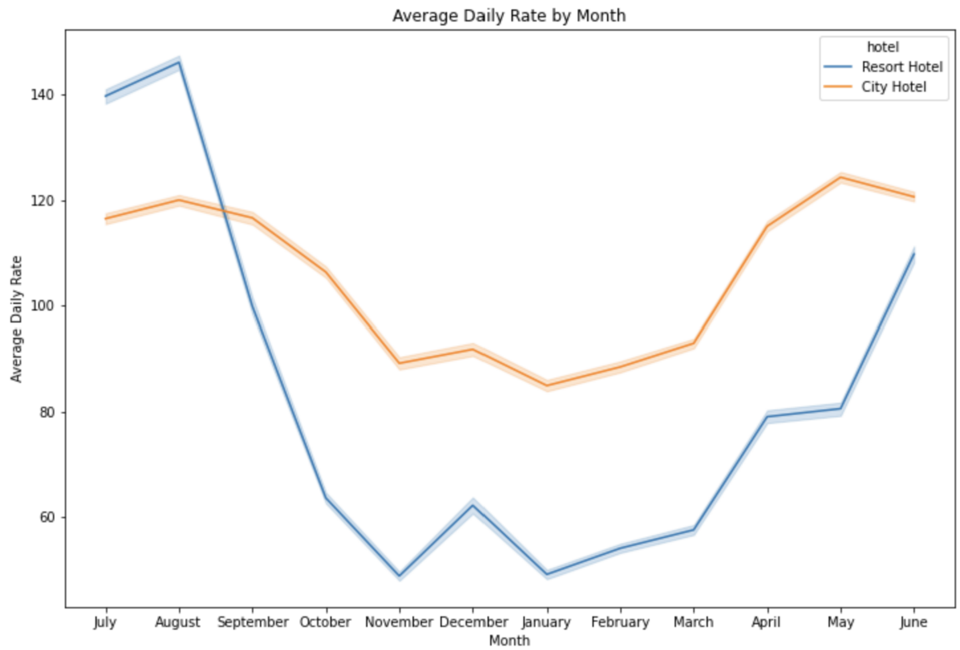
On X-axis: hotel_type
On Y-axis: number of bookings

H. Visualizing the data based on the relationship between categorical data(Hotel type) and numerical values(lead_time).



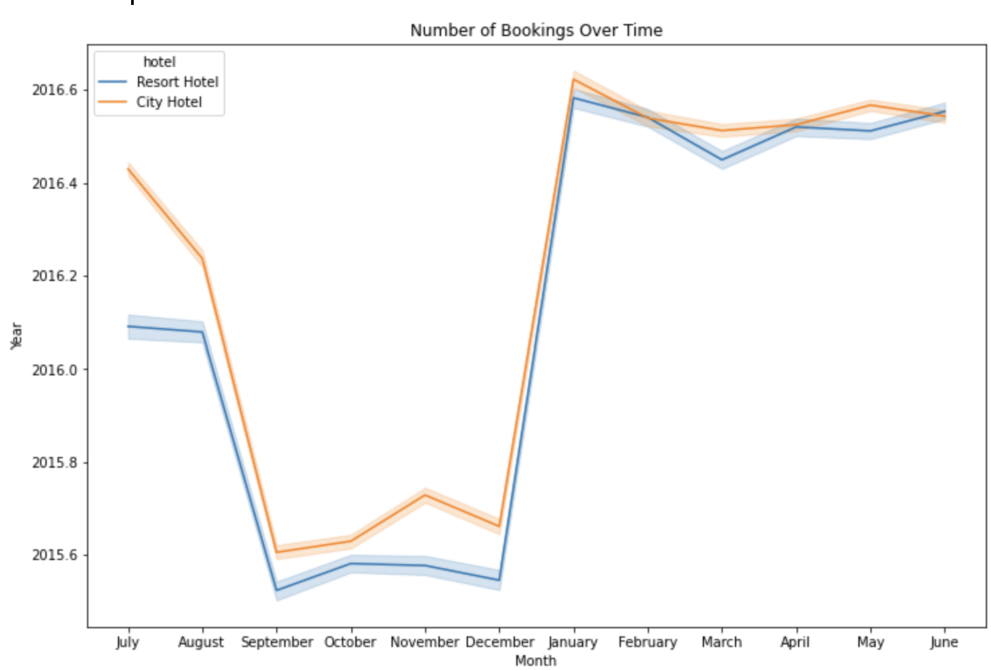
On X-axis: hotel_type
On Y-axis: lean_time

I. The average daily rate of booking - It can be observed that city hotel average bookings are higher than the average resort type hotel bookings.



On X-axis: all the months of the year
On Y-axis: Average daily rate of booking

J. Booking overtime - It can be observed that city type hotels and resort type hotels follow the same pattern.



On X-axis: all the months of the year
On Y-axis: year

7. Conclusion

After performing data cleaning and EDA, we can conclude that the number of cancellations in city hotels is more than in resort hotels but so are the number of bookings that are not canceled. The average daily rate of booking goes down in the months of November, December, and January and then picks back up for the rest of the year.
A similar pattern is shown in the overtime bookings with just a slight shift in the time of the year in that it starts going down from the month of September till December and then picks back up. This trend remains the same for the resort type and city type with city-type hotels having higher rates of booking compared to resort type.

8. References

- Kaggle dataset: https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand?resource=download&select=hotel_bookings.csv
- Article on data cleaning: Exploratory Data Analysis (EDA), Feature Selection, and machine learning prediction on time series data. | by oluyede Segun (jr) | Analytics Vidhya | Medium
- Kaggle reference on further details: Heart Attack - EDA + Prediction (90% accuracy) | Kaggle