Hotel Reservations Cancellation Prediction

The aim of this project to predict the possible reservations that are going to cancelled by the customers by analyzing various features and variables associated with the reservation.

Context

The online hotel reservation channels have dramatically changed booking possibilities and customers' behavior. A significant number of hotel reservations are called-off due to cancellations or no-shows. The typical reasons for cancellations include change of plans, scheduling conflicts, etc. This is often made easier by the option to do so free of charge or preferably at a low cost which is beneficial to hotel guests but it is a less desirable and possibly revenue-diminishing factor for hotels to deal with.

Data Dictionary

Column Name	Description
Booking_ID	unique identifier of each booking
no_of_adults	number of adults
no_of_children	number of children
no_of_weekend_nights	number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay at the hotel
no_of_week_nights	number of week nights (Monday to Friday) the guest stayed or booked to stay at the hotel
meal_type	meal type booked by the customer
required_car_parking_spaces	Does the customer require a car parking space? (0 - No, 1- Yes)
lead_time	Number of days between the date of booking and the arrival date
arrival_year	Year of arrival
arrival_month	Month of arrival
arrival_date	Date of arrival
market_segment	Market segment designation
repeated_guest Is the customer a repeated guest? (0 - No, 1- Yes)	
no_previous_cancellations	Number of previous bookings that were canceled by the customer prior to the current

Column Name	Description
	booking
previous_bookings_not_canceled	Number of previous bookings not canceled by the customer prior to the current booking
avg_price_per_room	Average price per day of the reservation; prices of the rooms are dynamic. (in euros)
no_of_special_requests	Total number of special requests made by the customer (e.g. high floor, view from the room, etc)
booking_status	Flag indicating if the booking was canceled or not

In [ ]:
```
#Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

In [ ]:
```
#Loading the dataset
df = pd.read_csv('Hotel Reservations.csv')
df.head()
```

Out[ ]:

	Booking_ID	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights
0	INN00001	2	0	1	2
1	INN00002	2	0	2	3
2	INN00003	1	0	2	1
3	INN00004	2	0	0	2
4	INN00005	2	0	1	1

Data Preprocessing Part 1

In [ ]:
```
#Checking the shape of the dataset
df.shape
```

Out[ ]:
```
(36275, 19)
```

In [ ]:
```
#Dropping the identifier column
df.drop(['Booking_ID'], axis=1, inplace=True)
```

Combining the year, month and day columns into a single column for date of arrival (yyyy/mm/dd)

In [ ]:
```
df['date of arrival'] = df['arrival_year'].astype(str) + '/' + df['arrival_month

#type casting the date column
df['date of arrival'] = pd.to_datetime(df['date of arrival'],format='mixed', inf
#dropping the columns
df.drop(columns=['arrival_date', 'arrival_month', 'arrival_year'], inplace=True)
```

In [ ]:
```
#checking for null values
df.isnull().sum()
```

Out[ ]:
```
no_of_adults                            0
no_of_children                          0
no_of_weekend_nights                    0
no_of_week_nights                       0
type_of_meal_plan                       0
required_car_parking_space              0
room_type_reserved                      0
lead_time                               0
market_segment_type                     0
repeated_guest                          0
no_of_previous_cancellations            0
no_of_previous_bookings_not_canceled    0
avg_price_per_room                      0
no_of_special_requests                  0
booking_status                          0
date of arrival                        37
dtype: int64
```

In [ ]:
```
df.dropna(inplace=True)
df.reset_index()
```

Out[ ]:

	index	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights
0	0	2	0	1	2
1	1	2	0	2	3
2	2	1	0	2	1
3	3	2	0	0	2
4	4	2	0	1	1
...	...	...	...	...	...
36233	36270	3	0	2	6
36234	36271	2	0	1	3
36235	36272	2	0	2	6
36236	36273	2	0	0	3
36237	36274	2	0	1	2

36238 rows × 17 columns

In [ ]:
```
#checking data types
df.dtypes
```

Out[ ]:
```
no_of_adults                                     int64
no_of_children                                   int64
no_of_weekend_nights                             int64
no_of_week_nights                                int64
type_of_meal_plan                               object
required_car_parking_space                       int64
room_type_reserved                              object
lead_time                                        int64
market_segment_type                             object
repeated_guest                                   int64
no_of_previous_cancellations                     int64
no_of_previous_bookings_not_canceled             int64
avg_price_per_room                             float64
no_of_special_requests                           int64
booking_status                                  object
date of arrival                         datetime64[ns]
dtype: object
```

In [ ]:
```
# checking for unique values in each column
df.nunique()
```

Out[ ]:
```
no_of_adults                               5
no_of_children                             6
no_of_weekend_nights                       8
no_of_week_nights                         18
type_of_meal_plan                          4
required_car_parking_space                 2
room_type_reserved                         7
lead_time                                352
market_segment_type                        5
repeated_guest                             2
no_of_previous_cancellations               9
no_of_previous_bookings_not_canceled      59
avg_price_per_room                      3919
no_of_special_requests                     6
booking_status                             2
date of arrival                          549
dtype: int64
```

Descriptive Statistics

In [ ]:
```
df.describe()
```

Out[ ]:

	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	requirec
count	36238.000000	36238.000000	36238.000000	36238.000000	
mean	1.845301	0.105221	0.810475	2.204206	
min	0.000000	0.000000	0.000000	0.000000	
25%	2.000000	0.000000	0.000000	1.000000	
50%	2.000000	0.000000	1.000000	2.000000	
75%	2.000000	0.000000	2.000000	3.000000	
max	4.000000	10.000000	7.000000	17.000000	
std	0.518572	0.402540	0.870992	1.410784	

Here the minimum average price per room and number of adults is zero, which is not possible so, I will replace the price with with mean value and drop the rows with zero adults.

In [ ]:
```
df['avg_price_per_room'].replace(0,df['avg_price_per_room'].mean(), inplace=True
```

In [ ]:
```
#drop where adults are 0
df.drop(df[df['no_of_adults'] == 0].index, inplace = True)
```

In [ ]:
```
df.head()
```

Out[ ]:

	no_of_adults	no_of_children	no_of_weekend_nights	no_of_week_nights	type_of_meal
0	2	0	1	2	Meal
1	2	0	2	3	Not Se
2	1	0	2	1	Meal
3	2	0	0	2	Meal
4	2	0	1	1	Not Se

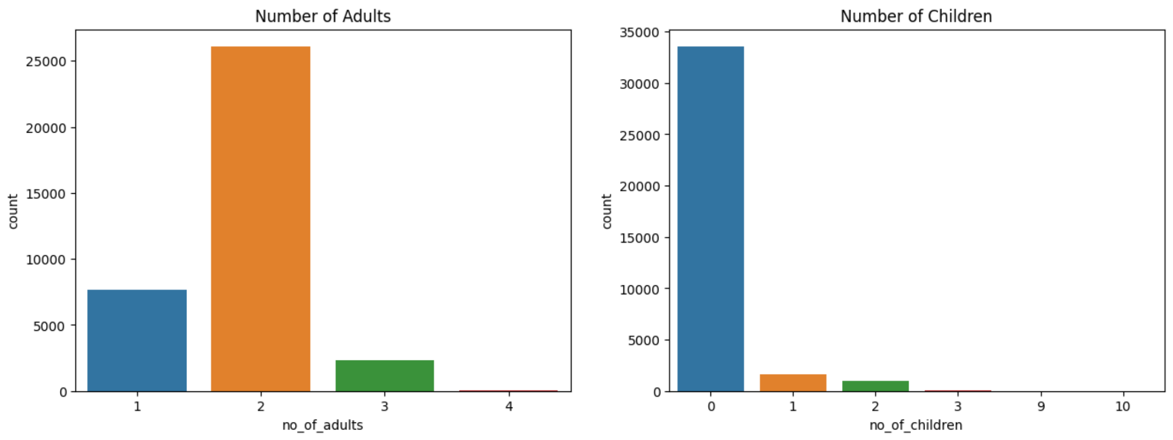
Exploratory Data Analysis

In the exploratory data analysis, I will be visualizing the data to get a better understanding of the data and to see if there are any trends or patterns in the data. First I will begin with looking at the distribution of the data and then I will look at the relationship between the independent variables and the target variable.

Guest Information

In [ ]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
sns.countplot( x = 'no_of_adults', data = df, ax=ax[0]).set_title('Number of Adu
sns.countplot( x = 'no_of_children', data = df, ax=ax[1]).set_title('Number of C
```

Out[ ]:
```
Text(0.5, 1.0, 'Number of Children')
```

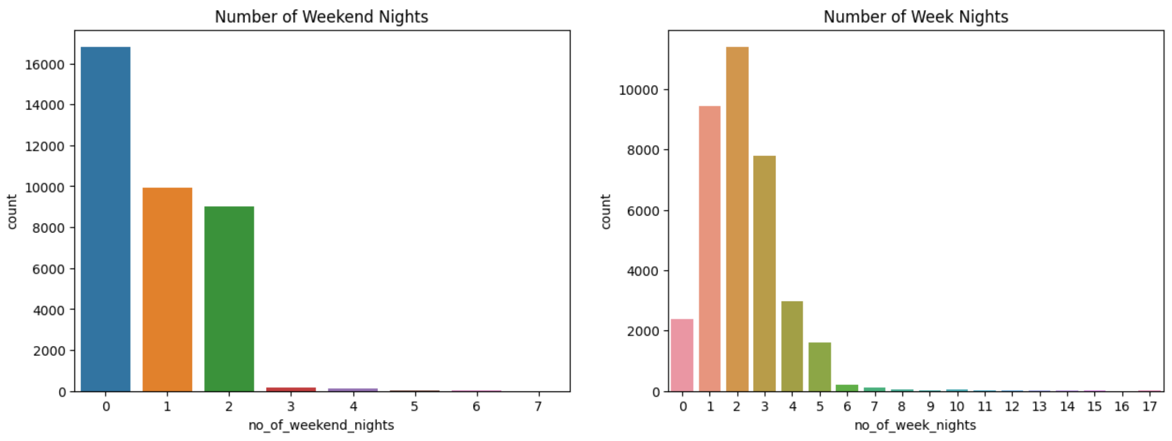


These graphs shows the distribution of the guest information which includes number of adults and children. The majority of bookings were made for 2 adults with no children which could mean that most of the bookings were made for couples. The second most common booking was for 1 adult with no children which could mean that most of the bookings were made for business trips. A few bookings were made with 1 or 2 children which could be by a family.

Time Spent at Hotel

In [ ]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
sns.countplot(x = 'no_of_weekend_nights', data = df, ax=ax[0]).set_title('Number
sns.countplot(x = 'no_of_week_nights', data = df, ax=ax[1]).set_title('Number of
```

Out[ ]:
```
Text(0.5, 1.0, 'Number of Week Nights')
```

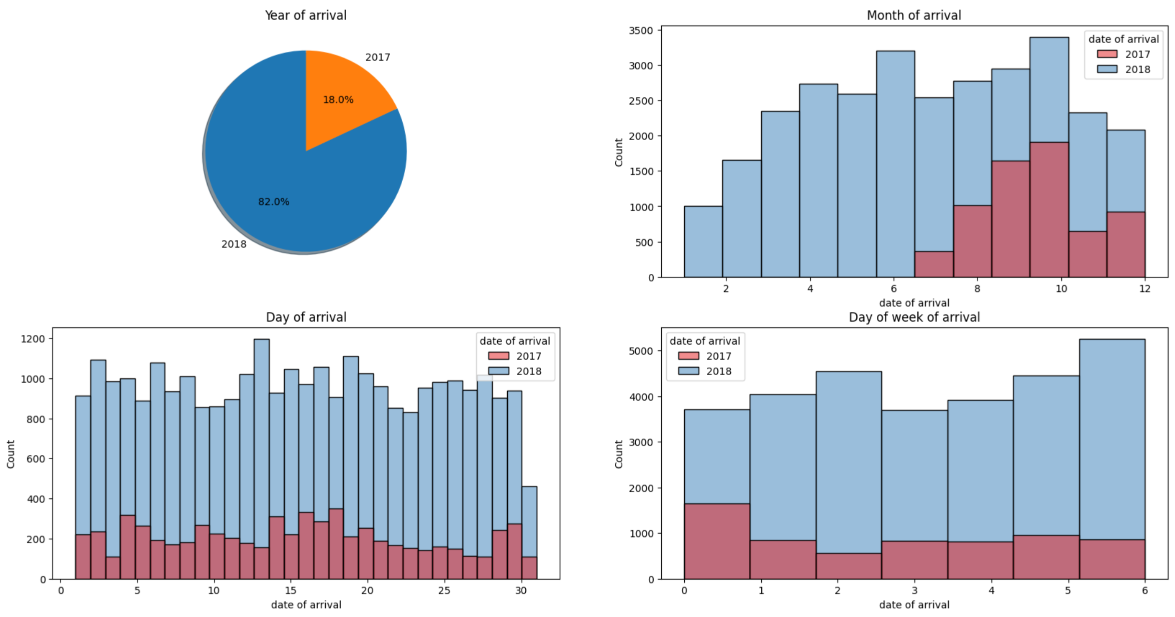


These graphs shows that most of the guest reserved to staye at the hotel on non weekend nights. The majority of the hotel bookings were for 1 or 2 nights. However, considerable number of bookings take place for the weekends. From this I assume that the bookings for the weekends were for vacation and the those for the weekdays were for business trips or for other reasons.

Date of Arrival

In [ ]:
```
fig, ax = plt.subplots(2,2,figsize=(20,10))

#year of arrival
ax[0,0].pie(df['date of arrival'].dt.year.value_counts(), labels = [2018,2017],
ax[0,0].set_title('Year of arrival')

#month of arrival
sns.histplot(x = df['date of arrival'].dt.month, ax=ax[0,1], bins=12, hue = df['

#day of arrival
sns.histplot(x = df['date of arrival'].dt.day, ax=ax[1,0], bins=31, hue = df['da

#day of week of arrival
sns.histplot(x = df['date of arrival'].dt.dayofweek, ax=ax[1,1], bins=7, hue = d
```

Out[ ]:
```
Text(0.5, 1.0, 'Day of week of arrival')
```



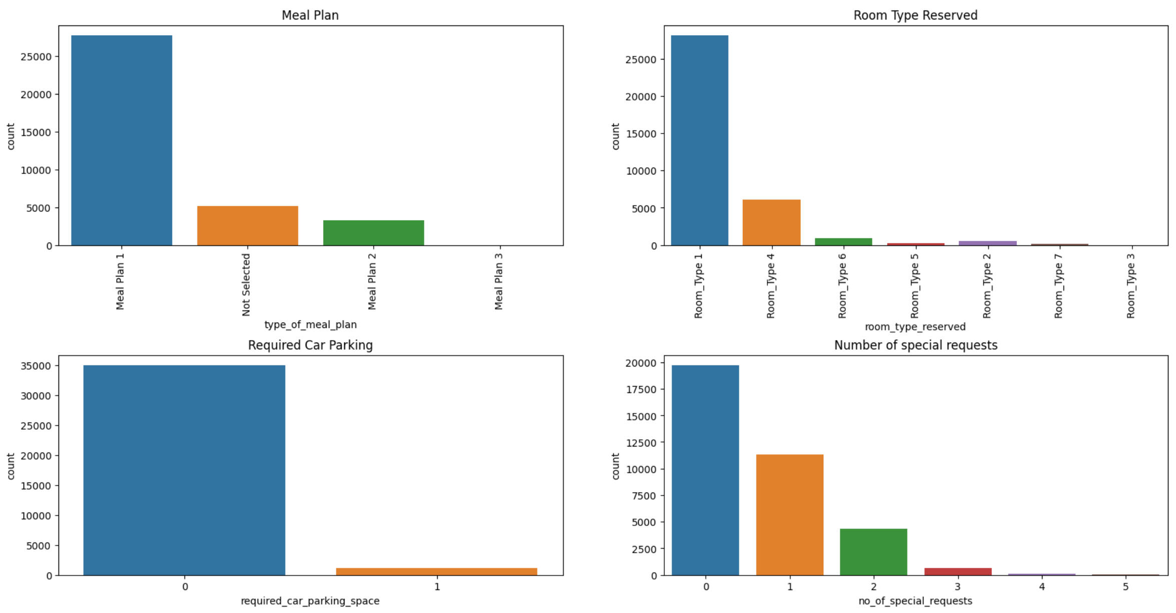
These graphs shows the number of bookings for the specific date, day, month and year. In the dataset, majority of the bookings were in 2018, i.e. 82%. In both the years the month of October had most booings as compared to other months. In 2017 nearly 2000 bookings in october and in 2018 nearly 3500. In addtion to that June had highest number of bookings after October. Coming to the days of the month, In 2017, 4,14,16,18 days from the month had the most reservations. In 2018, 2,7,14,19 days of the month had the most reservations. In the days of the week, Sundays had the highest number of reservations in 2017, whereas the Saturdays had the highest number of reservations in 2018.

From the above visualizations, I can conclude that more bookings were made in June and October particulary on the second and third weeks and during the weekends.

Services

In [ ]:
```
fig, ax = plt.subplots(2,2,figsize=(20,10))
fig.subplots_adjust(hspace=0.5)

sns.countplot(x = 'type_of_meal_plan', data = df, ax=ax[0,0]).set_title('Meal Pl
ax[0,0].xaxis.set_tick_params(rotation=90)

sns.countplot(x = 'room_type_reserved', data = df, ax=ax[0,1]).set_title('Room T
ax[0,1].xaxis.set_tick_params(rotation=90)

sns.countplot(x = 'required_car_parking_space', data = df, ax=ax[1,0]).set_title

sns.countplot(x = 'no_of_special_requests', data = df, ax=ax[1,1]).set_title('Nu
```

Out[ ]:
```
Text(0.5, 1.0, 'Number of special requests')
```

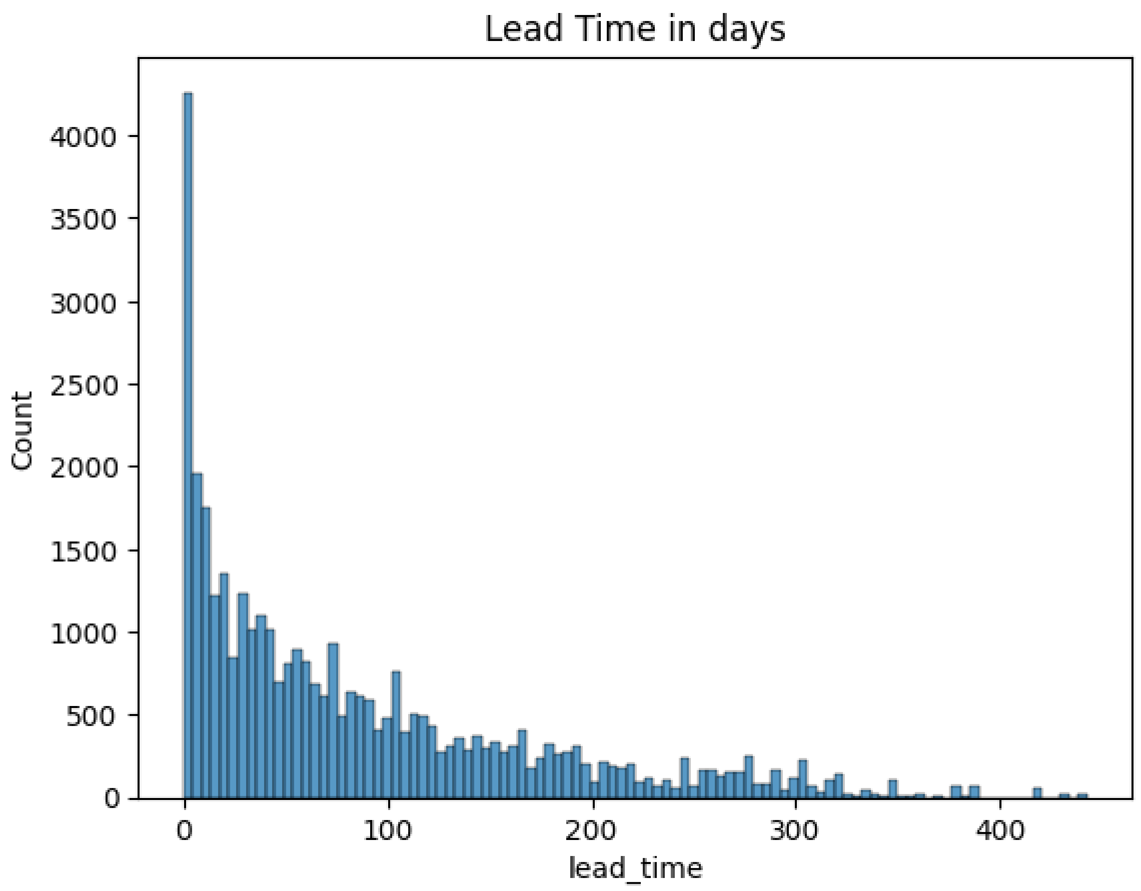


The above graphs shows the type of services of the hotel taken by the guests during reservations. Majority of the guests preferred Meal Plan1 and Room Type 1 and no special requests during reservations and most of them don't require parking space. Moreover a significant number of reservations were made without specifiying the type of meal plan, which could mean that the guests might have meal outside the hotel. The required parking space graph also tells about the mode of transportation used by the guests. Most of the guests used public transport or taxi to reach the hotel.

Lead time (days between date of reservation and date of arrival)

In [ ]:
```
sns.histplot(x = 'lead_time', data = df, bins=100).set_title('Lead Time in days'
```

Out[ ]:
```
Text(0.5, 1.0, 'Lead Time in days')
```

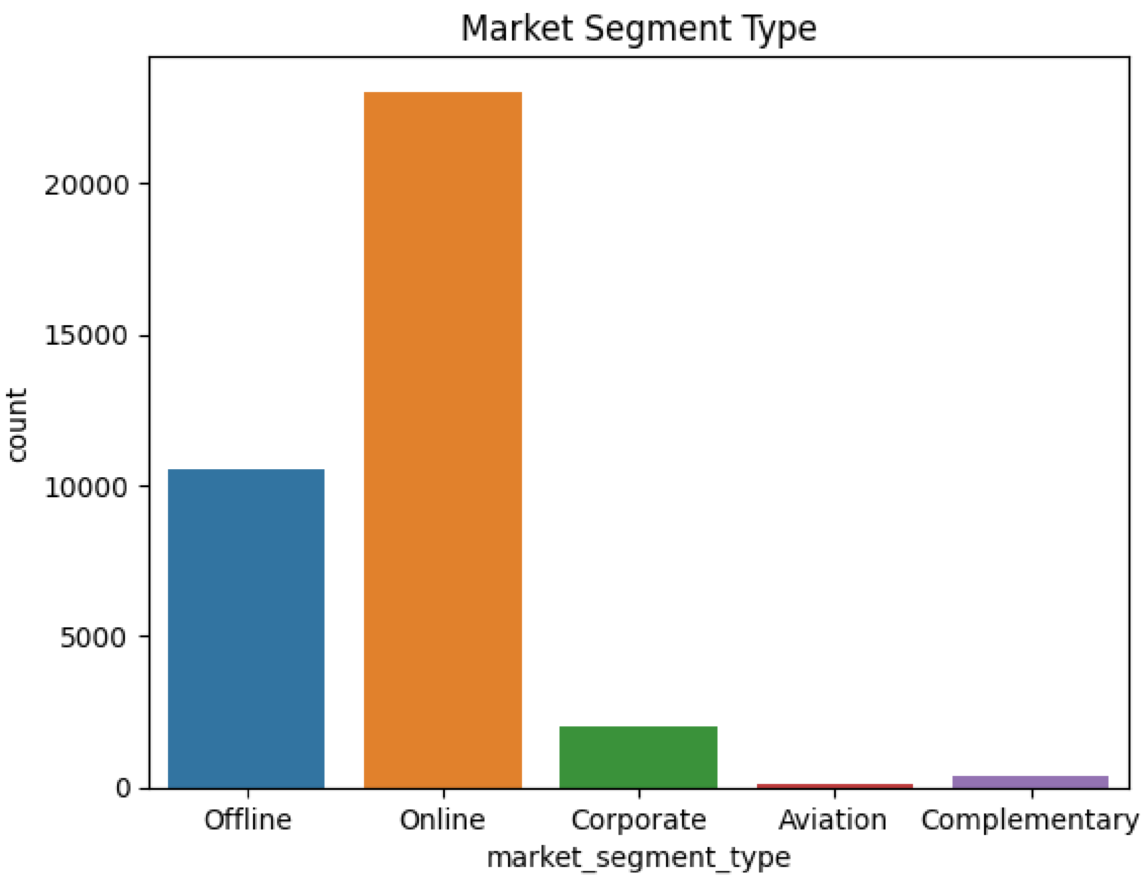


This graph shows that significant number of reservations were made just one day before or on the day of arrival. In addtion to that most of the reservations were made 1 to 2 weeks before the date of arrival. However, there were also reservations made 2-3 months before the date of arrival. From this histogram, I made ab hypothesis that, the guest who have lead time very less are less likely to cancel the reservation as compared to the guest who have more lead time.

Market Segment

In [ ]:
```
sns.countplot(x = 'market_segment_type', data = df).set_title('Market Segment Ty
```

Out[ ]:
```
Text(0.5, 1.0, 'Market Segment Type')
```

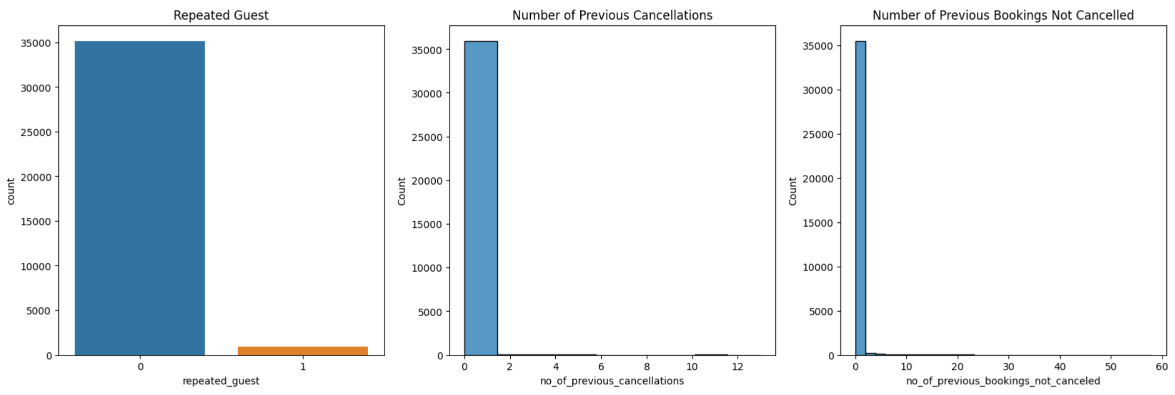


This graph shows the major gateways through which reservations were made at the hotel. Makority of the reservations were made through online platforms which means the hotel company has more presence on travel booking platforms. The second most common way of booking was through offline, which could be on arrival at the hotel or through a travel agent. The third most common way of booking was corporate, which could be through a company. Vert few of the bookings were made by aviation companies which highlights possibility of an airport near the hotel.

Guest's previous experience with the hotel

In [ ]:
```
fig, ax = plt.subplots(1,3,figsize=(20,6))

sns.countplot(x = 'repeated_guest', data = df, ax=ax[0]).set_title('Repeated Gue

sns.histplot(x = 'no_of_previous_cancellations', data = df, ax=ax[1], bins = 9).

sns.histplot(x = 'no_of_previous_bookings_not_canceled', data = df, ax=ax[2], bi
```

Out[ ]:
```
Text(0.5, 1.0, 'Number of Previous Bookings Not Cancelled')
```

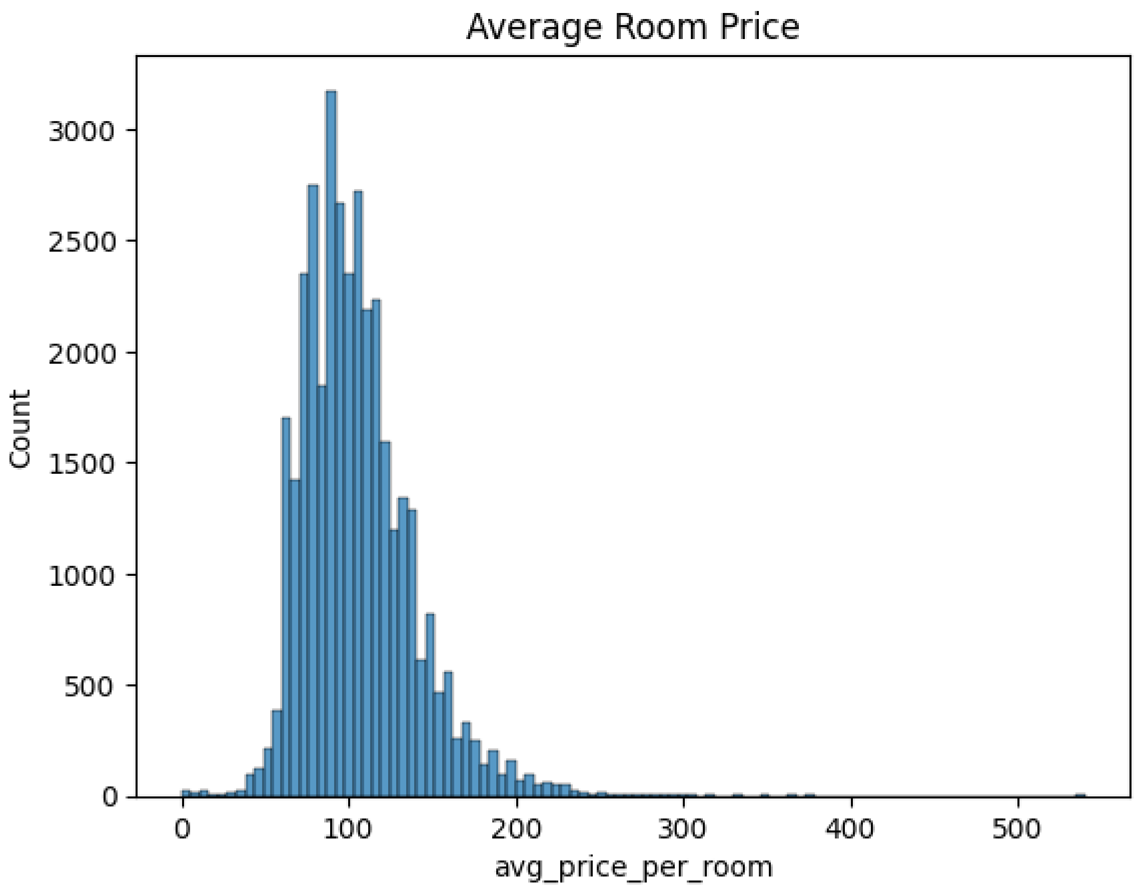


Majority of the reservations made at the hotel are by new guest, very few are the repeated guests at the hotel. This highlights the problem in the customer retention at the hotel. The hotel should focus on providing better services to the guests so that they would like to visit the hotel again. Since majority of the guest are new, so majority of the dataset has 0 pervious bookings cancellation. However, on a little bit closer look, we can see that there some guests who have cancelled their previous bookings.

Average room price

```
In [ ]: sns.histplot(x = 'avg_price_per_room', data = df, bins = 100).set_title('Average
```

```
Out[ ]: Text(0.5, 1.0, 'Average Room Price')
```



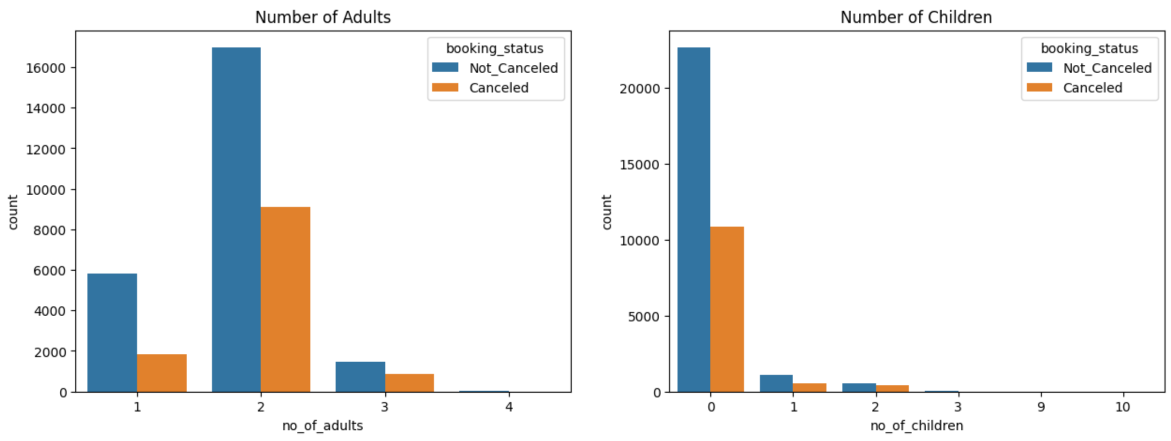
This graph shows the distribution of the room price. Majority of the reservations made had room price between 75 to 150. Very few of the reservations had room price more than 200.

Till now, I have plotted the distribution of data in all the variables and made some hypotheis around it. Now, I will look at the relationship between the independent variables and the target variable, to check the hypothesis.

Guest Information and Cancellation

In [ ]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
sns.countplot( x = 'no_of_adults', data = df, ax=ax[0], hue= 'booking_status').s
sns.countplot( x = 'no_of_children', data = df, ax=ax[1], hue = 'booking_status'
```

Out[ ]:
```
Text(0.5, 1.0, 'Number of Children')
```

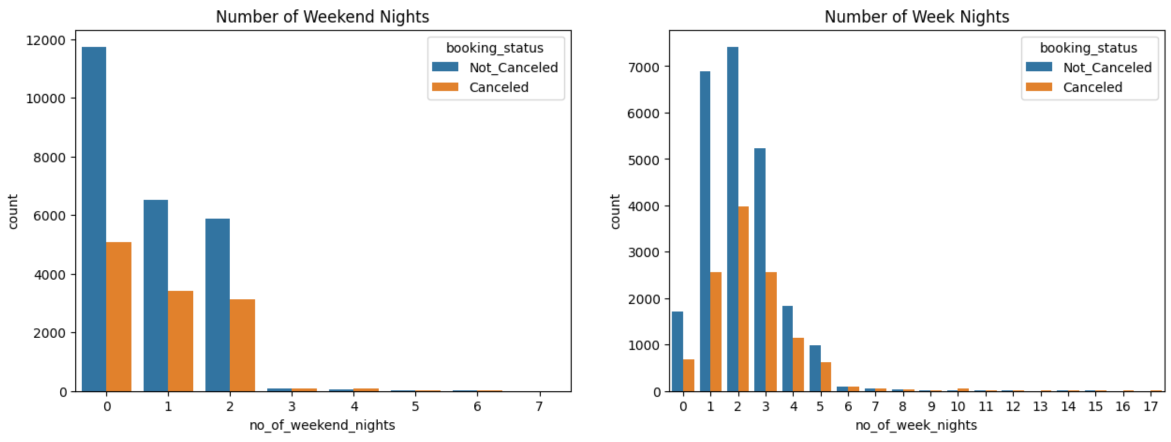


Majority of the reservation cancellations were made when the reservation was made for two adults, probably with no children. The second most common cancellation was made when the reservation was made for one adult. However, the number of cancellation reduces, when the reservation includes children and had more than 2 adults.

Time Spent at Hotel and Cancellation

In [ ]:
```
fig, ax = plt.subplots(1,2,figsize=(15,5))
sns.countplot(x = 'no_of_weekend_nights', data = df, ax=ax[0], hue = 'booking_st
sns.countplot(x = 'no_of_week_nights', data = df, ax=ax[1], hue = 'booking_statu
```

Out[ ]:
```
Text(0.5, 1.0, 'Number of Week Nights')
```

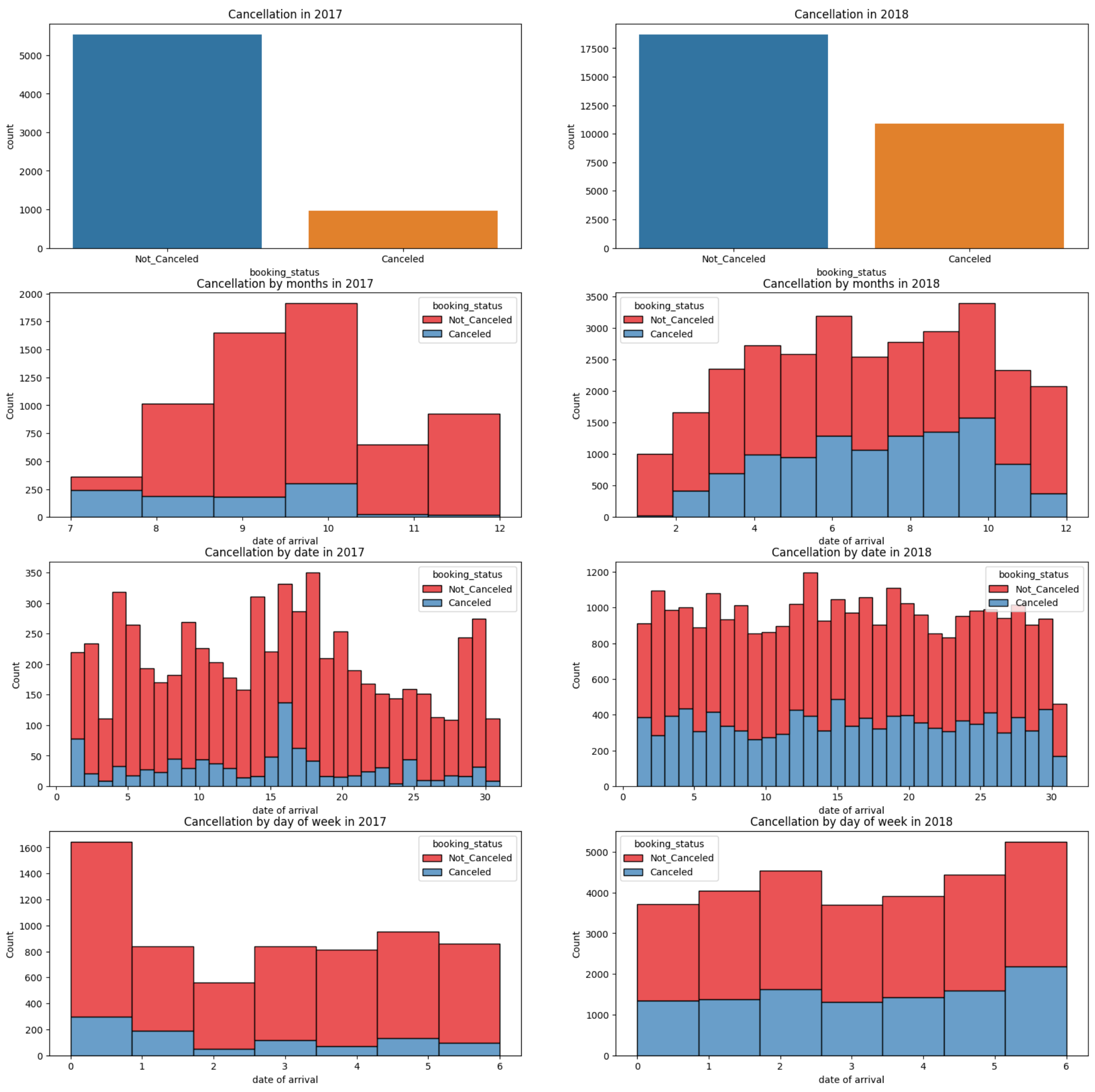


These graphs reveal interesting facts about reservation cancellation. The reservations made to spend 1 or 2 weekends nights have lower count of being cancelled. As compared to the reservations made to spend 2 weekdays at the hotel had the highest cancellation count followed by 1 and 3 week days. This could mean that guest could cancel their reservation, if they were palnning to stay during the week days and for less than 3 days. However, this count is lower, when reservations are made for weekends.

Date of Arrival and Cancellation

In [ ]:
```
fig,ax = plt.subplots(4,2,figsize=(20,20))
df_2017 = df[df['date of arrival'].dt.year == 2017]
df_2018 = df[df['date of arrival'].dt.year == 2018]

#year wise
sns.countplot(x = df_2017['booking_status'], data = df_2017, ax=ax[0,0]).set_tit
sns.countplot(x = df_2018['booking_status'], data = df_2018, ax=ax[0,1]).set_tit

#month wise
sns.histplot(x = df_2017['date of arrival'].dt.month, data = df_2017, ax=ax[1,0]
sns.histplot(x = df_2018['date of arrival'].dt.month, data = df_2018, ax=ax[1,1]

#date wise
sns.histplot(x = df_2017['date of arrival'].dt.day, data = df_2017, ax=ax[2,0],
sns.histplot(x = df_2018['date of arrival'].dt.day, data = df_2018, ax=ax[2,1],

#day of week wise
sns.histplot(x = df_2017['date of arrival'].dt.dayofweek, data = df_2017, ax=ax[
sns.histplot(x = df_2018['date of arrival'].dt.dayofweek, data = df_2018, ax=ax[
```

Out[ ]:
```
Text(0.5, 1.0, 'Cancellation by day of week in 2018')
```



The above graphs visualizes the reservation cancellation based on the dates the reservations were made. As we know that dataset mostly has reservations from 2018, despite of that, the number of reservations cancelled in 2018 is way higher than 2017. In 2017, nearly 5500 reservations were not cancelled and nearly 1000 where cancelled. However in 2018, 17500 reservations were not cancelled and more than 10000 reservations were cancelled. This shows that rate of reservation cancellation was much higher in 2018

Coming to the reservation cancellation according to the months, in 2017 reservations made in July and October had the highest. In addtion to that July had the least number of reservations made but still it has highest cancellation, which points some particular reason, not specified by the data. In 2018, June and October had the highest number of reservations made.

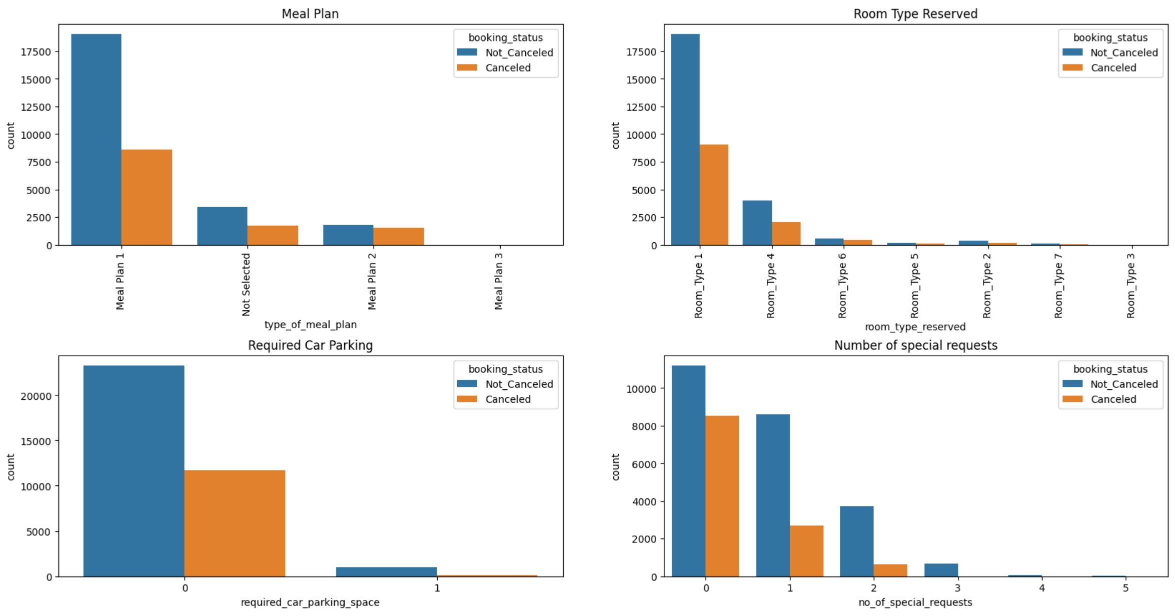
Now, we will look at the reservation cancellations by date of the month. In 2017, most reservations were cancelled for 16th and peculiarly on 1st of the month. In 2018, the number of cancellations were more in second and first week.

Coming to the day wise cancellation, in 2017 Sundays had the highest number of cancellations. In 2018, Saturdays had the highest number of cancellations.

Services and Cancellation

In [ ]:
```
fig, ax = plt.subplots(2,2,figsize=(20,10))
fig.subplots_adjust(hspace=0.5)

sns.countplot(x = 'type_of_meal_plan', data = df, ax=ax[0,0], hue = 'booking_sta
ax[0,0].xaxis.set_tick_params(rotation=90)

sns.countplot(x = 'room_type_reserved', data = df, ax=ax[0,1], hue = 'booking_st
ax[0,1].xaxis.set_tick_params(rotation=90)

sns.countplot(x = 'required_car_parking_space', data = df, ax=ax[1,0], hue = 'bo

sns.countplot(x = 'no_of_special_requests', data = df, ax=ax[1,1], hue = 'bookir
```

Out[ ]:
```
Text(0.5, 1.0, 'Number of special requests')
```

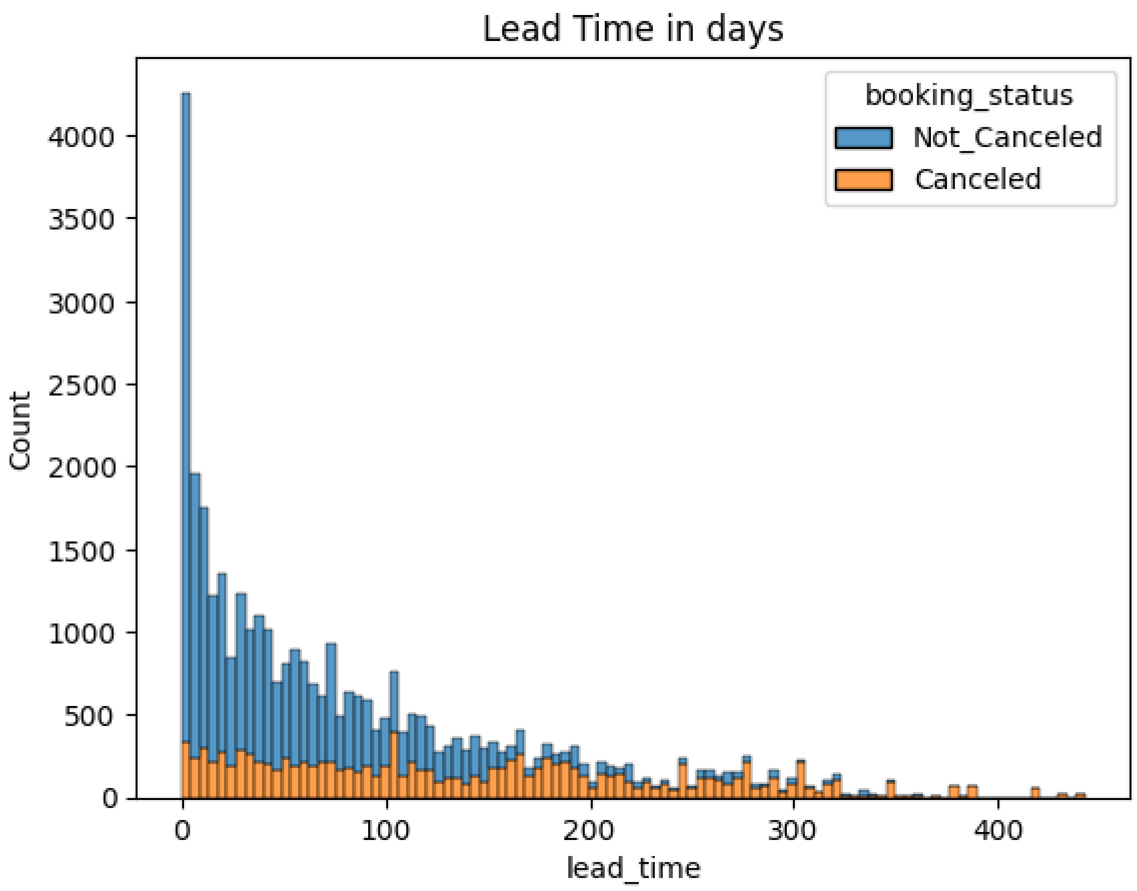


In the above graphs, we can see that the ratio of cancelled and not cancelled reservations is almost same for all the services. Therefore, I can conclude that the services provided by the hotel does not have any impact on the reservation cancellation.

Lead time and Cancellation

In [ ]:
```
sns.histplot(x = 'lead_time', data = df, bins=100, hue = 'booking_status', multi
```

Out[ ]:
```
Text(0.5, 1.0, 'Lead Time in days')
```

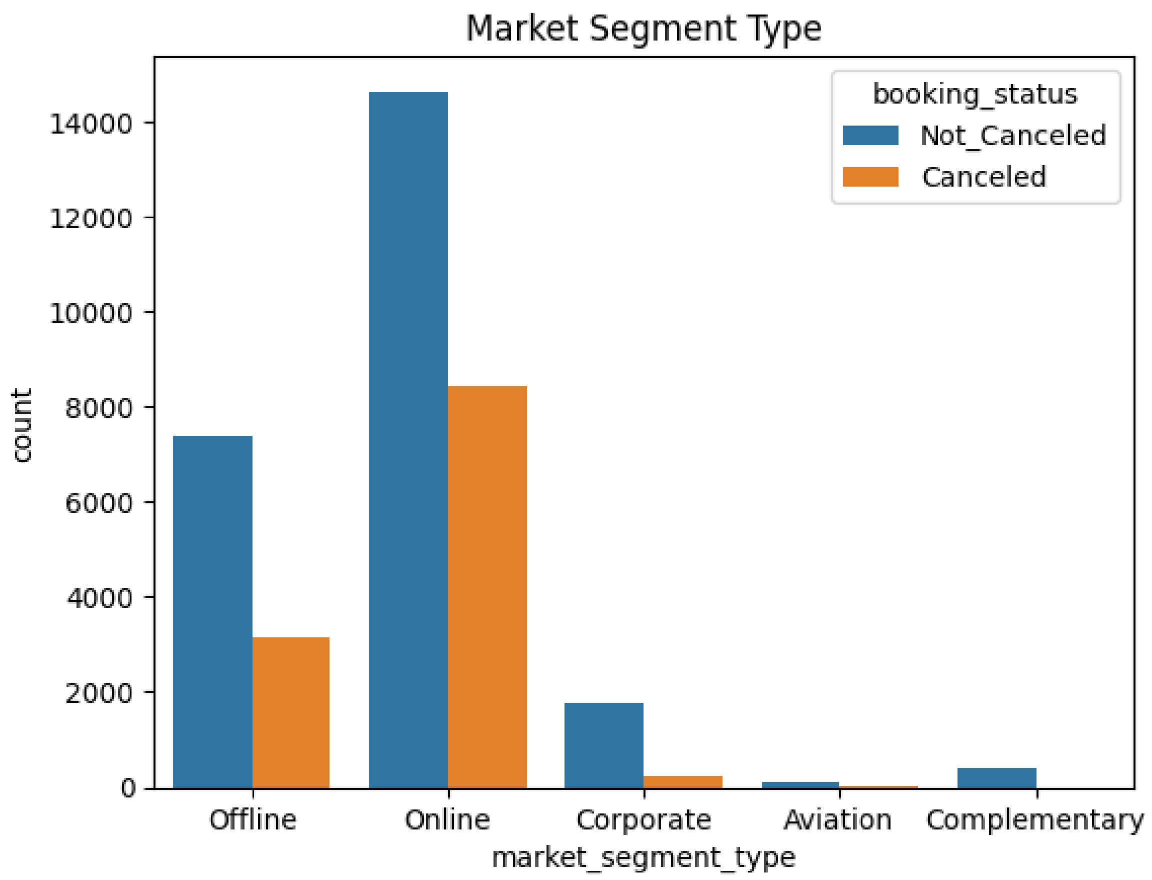


My hypothesis was true. With increase in lead time, the number of reservation cancellations also increases. The differnce in the number of reservations cancelled and not cancelled decreases as the lead time increases. This could mean that the guest who have lead time very less are less likely to cancel the reservation as compared to the guest who have more lead time.

Market Segment and Cancellation

In [ ]:
```
sns.countplot(x = 'market_segment_type', data = df, hue = 'booking_status').set_
```

Out[ ]:
```
Text(0.5, 1.0, 'Market Segment Type')
```

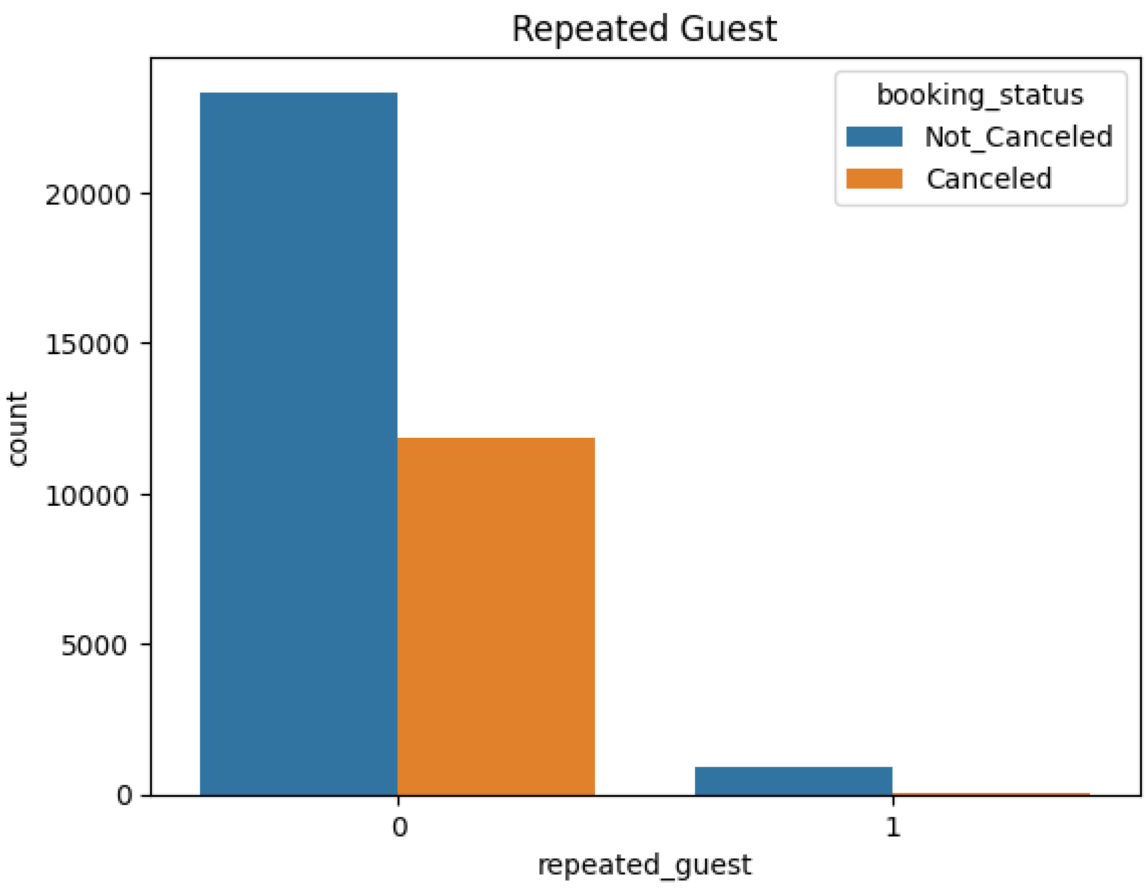


This graph shows the market segment of reservations and cancellation. Here most of the reservations are made through online platforms and thus it has the most number of cancellations. The second most common market segment is offline, which has the second most number of cancellations.

Guest's previous experience and Cancellation

```
In [ ]: sns.countplot(x = 'repeated_guest', data = df, hue = 'booking_status').set_title
```

```
Out[ ]: Text(0.5, 1.0, 'Repeated Guest')
```

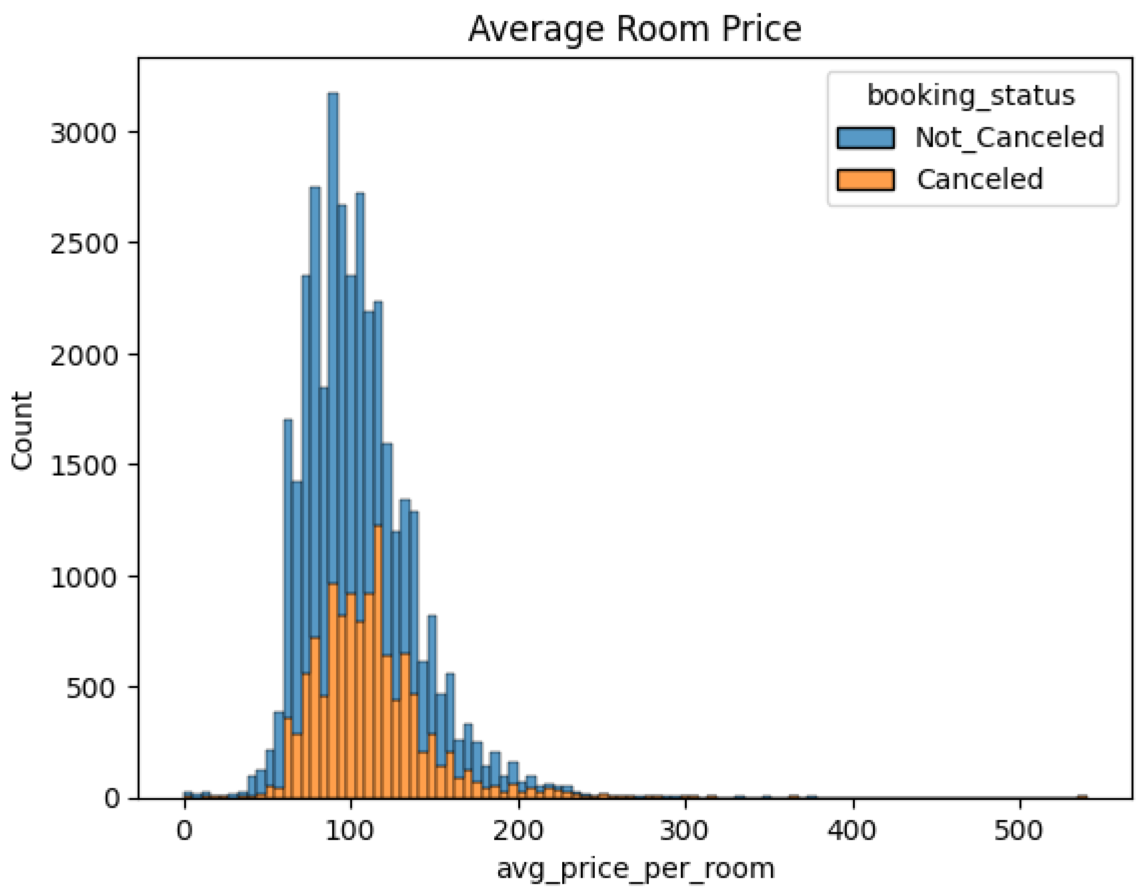


The previous guest are much less likely to cancel the reservation as compared to the new guest.

Average room price and Cancellation

```
In [ ]: sns.histplot(x = 'avg_price_per_room', data = df, bins = 100, hue = 'booking_sta
```

```
Out[ ]: Text(0.5, 1.0, 'Average Room Price')
```



Most of the room prices are between 75-150 and the number of reservations cancellation mostly occur in the same range. Therefore, there is no relation between the room price and reservation cancellation.

Data Preprocessing Part 2

Outlier Removal using IQR

```
In [ ]: #columns for outlier removal
cols = ['lead_time', 'avg_price_per_room']

Q1 = df[cols].quantile(0.25)
Q3 = df[cols].quantile(0.75)
IQR = Q3 - Q1

#removing outliers
df = df[~((df[cols] < (Q1 - 1.5 * IQR)) |(df[cols] > (Q3 + 1.5 * IQR))).any(axis
```

Label Encoding

```
In [ ]: from sklearn.preprocessing import LabelEncoder
#label encoding object
le = LabelEncoder()

#columns to be encoded
cols = ['type_of_meal_plan', 'room_type_reserved', 'market_segment_type', 'booki
```

```
#label encoding
for col in cols:
    le.fit(df[col])
    df[col] = le.transform(df[col])
    print(col, df[col].unique())
```

```
type_of_meal_plan [0 3 1 2]
room_type_reserved [0 3 5 4 1 6 2]
market_segment_type [3 4 2 0 1]
booking_status [1 0]
```

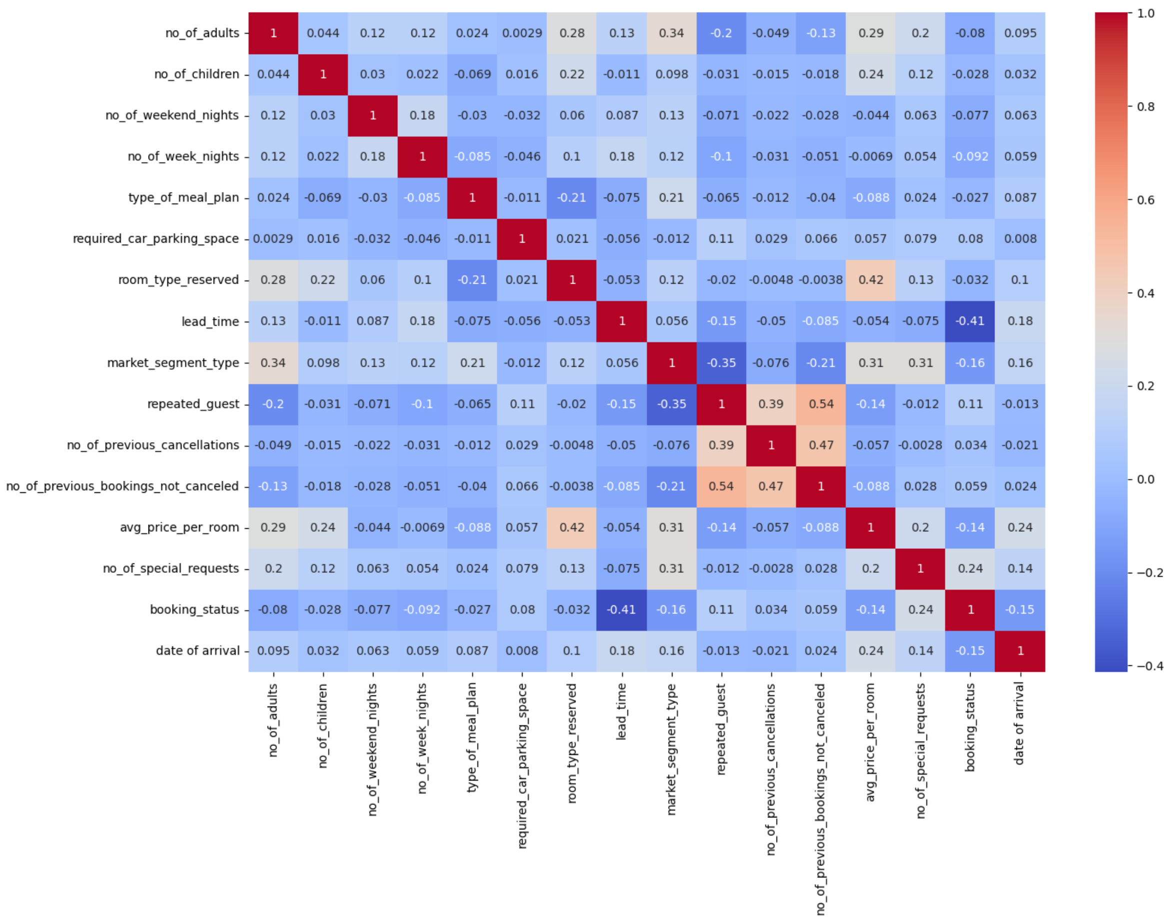
Feature Scaling

In [ ]:
```
from sklearn.preprocessing import StandardScaler
#standardizing the data
scaler = StandardScaler()
df[['lead_time', 'avg_price_per_room']] = scaler.fit_transform(df[['lead_time',
```

Correlation Matrix Heatmap

In [ ]:
```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```

Out[ ]: `<Axes: >`



In [ ]:
```
# Having issue with the model training due to this column
df.drop(columns=['date of arrival'], inplace=True)
```

Train Test Split

```
In [ ]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(df.drop('booking_status', ax
```

Model Building

I will be using the following classification models:

- Decision Tree Classifier
- Random Forest Classifier
- Logistic Regression
- Support Vector Classifier

Decision Tree Classifier

```
In [ ]: from sklearn.tree import DecisionTreeClassifier

        #decision tree classifier Object
        dtree = DecisionTreeClassifier()
```

Hyperparameter Tuning using GridSearchCV

```
In [ ]: from sklearn.model_selection import GridSearchCV

        #grid search parameters
        grid_param = {
            'max_depth': [2,4,6,8],
            'min_samples_leaf': [2,4,6,8],
            'min_samples_split': [2,4,6,8],
            'criterion': ['gini', 'entropy'],
            'random_state' : [0,42]
        }

        #grid search object
        grid_search = GridSearchCV(estimator=dtree, param_grid=grid_param, cv=5, n_jobs=

        #fitting the grid search object to the training data
        grid_search.fit(X_train, y_train)

        #best parameters
        print(grid_search.best_params_)
```

```
{'criterion': 'entropy', 'max_depth': 8, 'min_samples_leaf': 4, 'min_samples_spli
t': 2, 'random_state': 0}
```

```
In [ ]: #decision tree classifier object with best parameters
        dtree = DecisionTreeClassifier(criterion='entropy', max_depth=8, min_samples_lea

        #Training the model
        dtree.fit(X_train, y_train)

        #Training accuracy
        print(dtree.score(X_train, y_train))
```

```
#Predicting the test set results
d_pred = dtree.predict(X_test)
```

```
0.85635687732342
```

Random Forest Classifier

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier

#random forest classifier object
rfc = RandomForestClassifier()
```

Hyperparameter Tuning using GridSearchCV

In [ ]:
```
from sklearn.model_selection import GridSearchCV

#grid search parameters
grid_param = {
    'max_depth': [2,4,6,8],
    'min_samples_leaf': [2,4,6,8],
    'min_samples_split': [2,4,6,8],
    'criterion': ['gini', 'entropy'],
    'random_state' : [0,42]
}

#grid search object
grid_search = GridSearchCV(estimator=rfc, param_grid=grid_param, cv=5, n_jobs=-1

#fitting the grid search object to the training data
grid_search.fit(X_train, y_train)

#best parameters
print(grid_search.best_params_)
```

```
{'criterion': 'gini', 'max_depth': 8, 'min_samples_leaf': 4, 'min_samples_split':
2, 'random_state': 0}
```

In [ ]:
```
#random forest classifier object with best parameters
rfc = RandomForestClassifier(criterion='entropy', max_depth=8, min_samples_leaf=

#Training the model
rfc.fit(X_train, y_train)

#Training accuracy
print(rfc.score(X_train, y_train))

#Predicting the test set results
r_pred = rfc.predict(X_test)
```

```
0.850185873605948
```

Logistic Regression

In [ ]:
```
from sklearn.linear_model import LogisticRegression

#logistic regression object
logreg = LogisticRegression()
```

Hyperparameter Tuning using GridSearchCV

In [ ]:
```
from sklearn.model_selection import GridSearchCV

#grid search parameters
grid_param = {
    'penalty': ['l1', 'l2', 'elasticnet', 'none'],
    'C': [0.001,0.01,0.1,1,10,100,1000],
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
    'random_state' : [0,42]
}

#grid search object
grid_search = GridSearchCV(estimator=logreg, param_grid=grid_param, cv=5, n_jobs

#fitting the grid search object to the training data
grid_search.fit(X_train, y_train)

#best parameters
print(grid_search.best_params_)
```

```
{'C': 1, 'penalty': 'l2', 'random_state': 0, 'solver': 'liblinear'}
```

In [ ]:
```
#logistic regression object with best parameters
logreg = LogisticRegression(C=1, penalty='l2', random_state=0, solver='liblinear

#Training the model
logreg.fit(X_train, y_train)

#Training accuracy
print(logreg.score(X_train, y_train))

#Predicting the test set results
l_pred = logreg.predict(X_test)
```

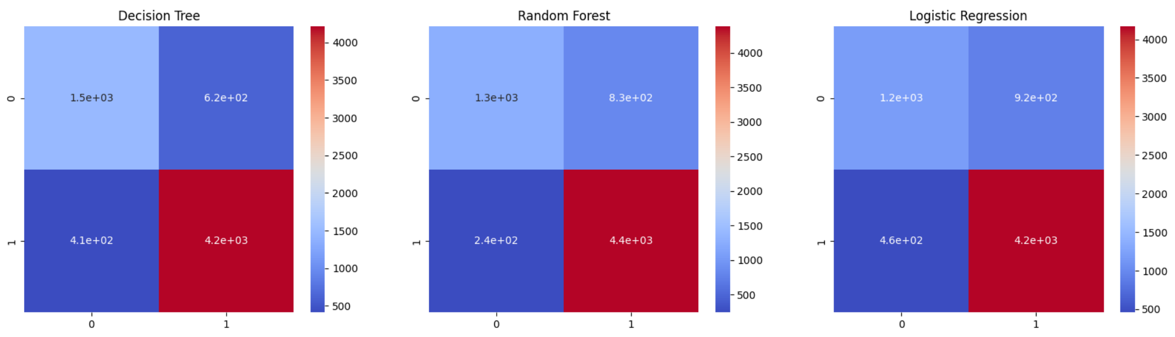
```
0.7956877323420074
```

Model Evaluation

Confusion Matrix Heatmap

In [ ]:
```
from sklearn.metrics import confusion_matrix

fig, ax = plt.subplots(1,3,figsize=(20,5))

#decision tree
sns.heatmap(confusion_matrix(y_test, d_pred), annot=True, cmap='coolwarm', ax=ax
#random forest
sns.heatmap(confusion_matrix(y_test, r_pred), annot=True, cmap='coolwarm', ax=ax
#logistic regression
sns.heatmap(confusion_matrix(y_test, l_pred), annot=True, cmap='coolwarm', ax=ax
```

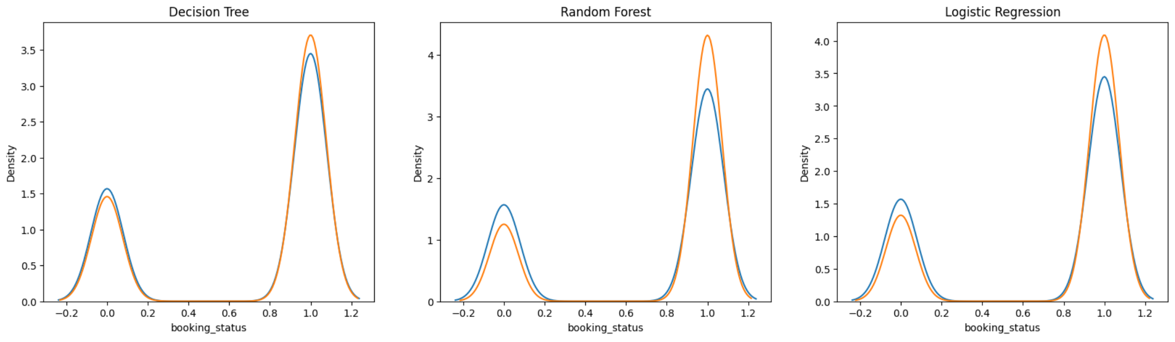
Out[ ]:
```
Text(0.5, 1.0, 'Logistic Regression')
```



Distribution Plot

In [ ]:
```
fig, ax  = plt.subplots(1,3,figsize=(20,5))

#decision tree
sns.distplot(y_test, ax=ax[0], hist= False).set_title('Decision Tree')
sns.distplot(d_pred, ax=ax[0], hist = False)

#random forest
sns.distplot(y_test, ax=ax[1], hist= False).set_title('Random Forest')
sns.distplot(r_pred, ax=ax[1], hist = False)

#logistic regression
sns.distplot(y_test, ax=ax[2], hist= False).set_title('Logistic Regression')
sns.distplot(l_pred, ax=ax[2], hist = False)
```

Out[ ]:
```
<Axes: title={'center': 'Logistic Regression'}, xlabel='booking_status', ylabel
='Density'>
```



Classification Report

In [ ]:
```
from sklearn.metrics import classification_report

#decision tree
print('Decision Tree')
print(classification_report(y_test, d_pred))
#random forest
print('Random Forest')
print(classification_report(y_test, r_pred))
#logistic regression
print('Logistic Regression')
print(classification_report(y_test, l_pred))
```

```
Decision Tree
              precision    recall  f1-score   support

           0       0.78      0.71      0.74      2101
           1       0.87      0.91      0.89      4624

    accuracy                           0.85      6725
   macro avg       0.83      0.81      0.82      6725
weighted avg       0.84      0.85      0.85      6725

Random Forest
              precision    recall  f1-score   support

           0       0.84      0.60      0.70      2101
           1       0.84      0.95      0.89      4624

    accuracy                           0.84      6725
   macro avg       0.84      0.78      0.80      6725
weighted avg       0.84      0.84      0.83      6725

Logistic Regression
              precision    recall  f1-score   support

           0       0.72      0.56      0.63      2101
           1       0.82      0.90      0.86      4624

    accuracy                           0.80      6725
   macro avg       0.77      0.73      0.75      6725
weighted avg       0.79      0.80      0.79      6725
```

Model Metrics

In [ ]:

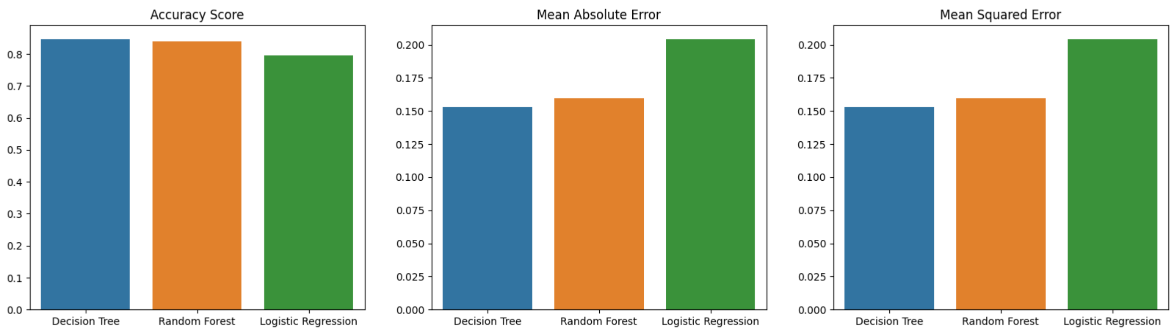
```
from sklearn.metrics import accuracy_score, mean_absolute_error, mean_squared_er

#decision tree
print('Decision Tree')
print('Accuracy Score: ', accuracy_score(y_test, d_pred))
print('Mean Absolute Error: ', mean_absolute_error(y_test, d_pred))
print('Mean Squared Error: ', mean_squared_error(y_test, d_pred))

print('\n')

#random forest
print('Random Forest')
print('Accuracy Score: ', accuracy_score(y_test, r_pred))
print('Mean Absolute Error: ', mean_absolute_error(y_test, r_pred))
print('Mean Squared Error: ', mean_squared_error(y_test, r_pred))

print('\n')

#logistic regression
print('Logistic Regression')
print('Accuracy Score: ', accuracy_score(y_test, l_pred))
print('Mean Absolute Error: ', mean_absolute_error(y_test, l_pred))
print('Mean Squared Error: ', mean_squared_error(y_test, l_pred))
```

```
Decision Tree
Accuracy Score:  0.8472862453531599
Mean Absolute Error:  0.15271375464684014
Mean Squared Error:  0.15271375464684014


Random Forest
Accuracy Score:  0.840446096654275
Mean Absolute Error:  0.15955390334572492
Mean Squared Error:  0.15955390334572492


Logistic Regression
Accuracy Score:  0.7955390334572491
Mean Absolute Error:  0.20446096654275092
Mean Squared Error:  0.20446096654275092
```

Model Comparison

In [ ]:
```python
fig, ax = plt.subplots(1,3,figsize=(20,5))
#Accuracy Score
sns.barplot(x = ['Decision Tree', 'Random Forest', 'Logistic Regression'], y = [
#Mean Absolute Error
sns.barplot(x = ['Decision Tree', 'Random Forest', 'Logistic Regression'], y = [
#Mean Squared Error
sns.barplot(x = ['Decision Tree', 'Random Forest', 'Logistic Regression'], y = [
```

Out[ ]:
```
Text(0.5, 1.0, 'Mean Squared Error')
```

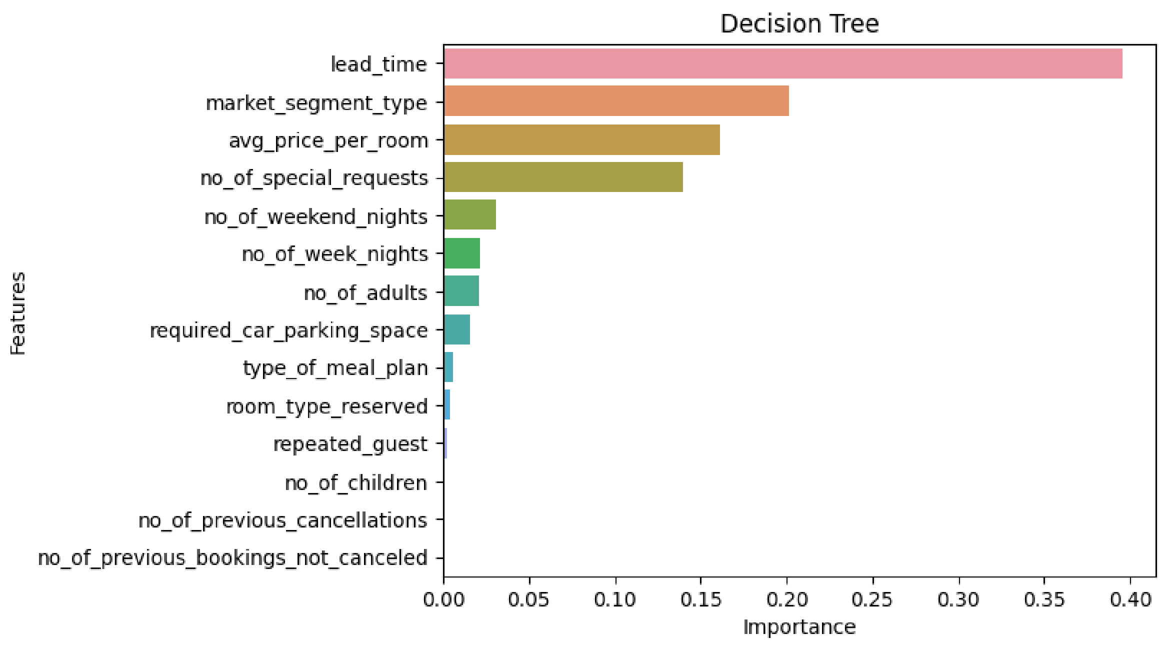


Feature Importance

Feature Importance from best two models

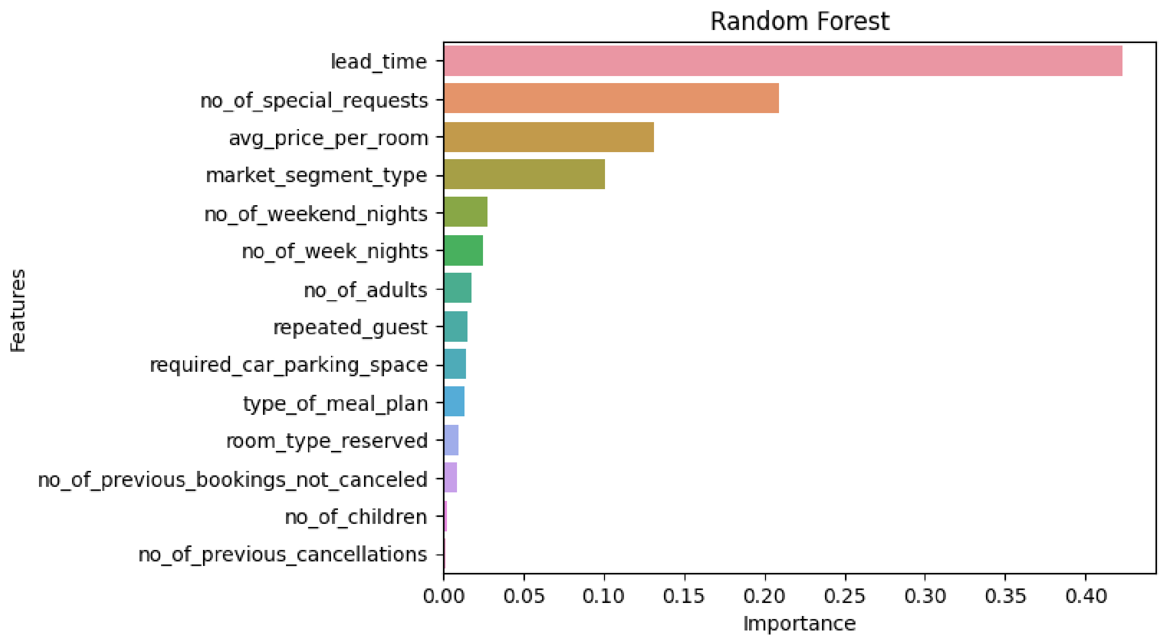
In [ ]:
```python
#decision tree
feature_importance = pd.DataFrame({'Features': X_train.columns, 'Importance': dt
feature_importance.sort_values(by='Importance', ascending=False, inplace=True)
feature_importance.reset_index(drop=True, inplace=True)
sns.barplot(x = 'Importance', y = 'Features', data = feature_importance).set_tit
```

Out[ ]:
```
Text(0.5, 1.0, 'Decision Tree')
```



```
In [ ]: #random forest
        feature_importance = pd.DataFrame({'Features': X_train.columns, 'Importance': rf
        feature_importance.sort_values(by='Importance', ascending=False, inplace=True)
        feature_importance.reset_index(drop=True, inplace=True)
        sns.barplot(x = 'Importance', y = 'Features', data = feature_importance).set_tit
```

```
Out[ ]: Text(0.5, 1.0, 'Random Forest')
```



Conclusion

From the Exploratory Data Analysis, I came to know that, the most of the reservations wjere made for 2 adults with no children which could probably for a couple had highest cancellation count. In addtion to that, the cancellation count of reservations decreases when there are children involved. Most of the reservations were made for week nights and had exponentially higher cancellations as compared to those made for weekend nights.

The year 2018 had higher cancellation rate as compared to 2017, with most of the cancellation done in month of July and October. Upon visualization of the services opted during reservation with booking status, it was found that the services opted during reservation does not have any impact on the reservation cancellation.

The lead time had a huge impact on the reservation cancellation, which has been evident from feature importance as well. The guest who have lead time very less are less likely to cancel the reservation as compared to the guest who have more lead time. Therefore, with increased lead time, the guests have more time to think about the reservation and thus they are more likely to cancel the reservation. So, the hotel should try to take reservations for shorter lead time.

The market segment of the reservation also had an impact on the reservation cancellation. The reservations made through online platforms had the highest number of cancellations. This highlights the hotel's reputation and presence on online platforms. The hotel should try to improve its reputation on online platforms to reduce the reservation cancellation.

Coming to the classification models, I have used Decision Tree Classifier, Random Forest Classifier, Logistic Regression for predicting the reservation cancellation. The Decision Tree Classifier had the highest accuracy i.e. 85% among all the models.