PREDICTIVE MAINTAINANCE FOR INDUSTRIAL EQUIPMENT

**MINI PROJECT REPORT**

***Submitted By***

**HIRESH V BERIA - 211701020**

**BONDIL ADITYA SINGH - 211701010**

***In partial fulfilment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND DESIGN**



**RAJALAKSHMI ENGINEERING COLLEGE ANNA UNIVERSITY, CHENNAI-600 025**

**NOVEMBER 2024**

RAJALAKSHMI ENGINEERING COLLEGE

**BONAFIDE CERTIFICATE**

Certified that this Report titled **”HOTEL RESERVATIONS CANCELLATION”** is the bonafide work of **“HIRESH V BERIA (211701020) & BONDIL ADITYA SINGH (211701010)”** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE**

**ABSTRACT**

The increasing use of online booking platforms has revolutionized the hospitality industry but has also introduced challenges like frequent cancellations. These cancellations, driven by flexible policies and changing customer plans, result in revenue losses and operational inefficiencies for hotels. This project aims to predict hotel reservation cancellations by analyzing customer booking data and identifying key factors influencing cancellation behavior.

A dataset containing customer demographics, booking details, and other relevant variables was utilized. Data preprocessing steps included cleaning missing values, handling outliers, and encoding categorical features. Exploratory Data Analysis (EDA) revealed significant patterns, such as the impact of lead time, booking channels, and guest history on cancellations. Machine learning models, including Decision Tree Classifier, Random Forest Classifier, and Logistic Regression, were employed to build predictive models.

The findings provide actionable recommendations for hotel management, such as dynamic pricing strategies and targeted marketing efforts. By implementing predictive analytics, hotels can optimize operations, improve revenue management, and enhance customer satisfaction.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** |  |
| **1.** | **INTRODUCTION** |  |
|  | OVERVIEW OF THE PROBLEM STATEMENT | 1 |
|  | OBJECTIVES | 1 |
| **2.** | **DATASET DESCRIPTION** |  |
|  | DATASET SOURCE | 2 |
|  | DATASET SIZE AND STRUCTURE | 2 |
|  | DATASET FEATURES DESCRIPTION | 3 |
| **3.** | **DATA ACQUISITION AND INITIAL ANALYSIS** |  |
|  | DATA LOADING | 4 |
|  | INITIAL OBSERVATIONS | 4 |
| **4.** | **DATA CLEANING AND PREPROCESSING** |  |
|  | HANDLING MISSING VALUES | 5 |
|  | FEATURE ENGINEERING | 5 |

|  |  |  |
| --- | --- | --- |
|  | DATA TRANSFORMATION | 6 |
| **5.** | **EXPLORATORY DATA ANALYSIS** |  |
|  | DATA INSIGHTS DESCRIPTION | 7 |
|  | DATA INSIGHTS VISUALIZATION | 8 - 9 |
| **6.** | **PREDICTIVE MODELING** |  |
|  | MODEL SELECTION AND JUSTIFICATION | 10 - 11 |
|  | DATA PARTITIONING | 12 |
|  | MODEL TRAINING AND HYPERPARAMETER TUNING | 12-13 |
| 7. | **MODEL EVALUATION AND OPTIMIZATION** |  |
|  | PERFORMANCE ANALYSIS | 14 |
|  | FEATURE IMPORTANCE | 14 - 15 |
|  | MODEL REFINEMENT | 16 - 17 |
| **8.** | **DISCUSSION AND CONCLUSION** |  |
|  | SUMMARY OF FINDINGS | 18 - 19 |
|  | CHALLENGES AND LIMITATIONS: | 20 - 21 |
|  | **APPENDIX** | 22 - 44 |
|  | **REFERENCES** | 45 |

**CHAPTER 1 INTRODUCTION**

**OVERVIEW OF THE PROBLEM STATEMENT :**

Frequent hotel reservation cancellations pose significant challenges to the hospitality industry, affecting revenue management, resource allocation, and operational efficiency. These cancellations are often driven by flexible booking policies, changes in customer plans, or competitive pricing dynamics. While beneficial for customers, such flexibility leads to unpredictable revenue streams and logistical complications for hotels. Traditional methods of handling cancellations, such as overbooking, can further alienate customers if mishandled.Understanding the factors influencing cancellations and predicting their likelihood is crucial for optimizing hotel operations. This project addresses the problem by utilizing machine learning models to analyze customer booking data and identify patterns. Predictive insights can empower hotel management to implement proactive strategies, enhancing profitability and customer retention.

**OBJECTIVES :**

The primary objective of this project is to predict hotel reservation cancellations by leveraging customer booking data and advanced machine learning techniques. This involves analyzing booking patterns and customer behavior to identify the factors most influencing cancellations, such as lead time, market segment, and previous booking history. By developing and evaluating predictive models, the project aims to accurately forecast cancellations and assess the significance of key features in the dataset. Ultimately, the project seeks to provide actionable insights to hotel management, enabling them to proactively manage room allocations, optimize pricing strategies, and improve customer retention while minimizing revenue losses associated with cancellations.

**CHAPTER 2**

**DATA DESCRIPTION**

**DATASET SOURCE :**

In a hotel reservation cancellation project, the dataset can be used to predict cancellation patterns by analyzing various factors. **Booking ID** serves as a unique identifier, allowing tracking of individual reservations. Features like **number of adults** and **number of children** help explore how family size affects cancellations, while **number of weekend nights** and **number of week nights** reveal how the length of stay impacts cancellation rates. The **type of meal plan** and **required car parking space** can provide insights into customer preferences and their likelihood of canceling. **Room type reserved** and **average price per room** help assess the relationship between room selection and cancellation likelihood. **Lead time**, **arrival details**, and **market segment type** provide context for predicting cancellation behavior based on timing and customer type. Features like **repeated guest**, **previous cancellations**, and **special requests** give deeper insights into guest behavior, which can be used to predict future cancellations. Ultimately, these features are used to build predictive models that forecast cancellations and optimize hotel booking strategies.

**DATASET SIZE AND STRUCTURE:**

The Predictive Maintenance Dataset consists of 36,500 rows and 19 columns. The dataset includes columns such as:

* Numerical Columns: no\_of\_adults, no\_of\_children, no\_of\_weekend\_nights, no\_of\_week\_nights,lead\_time,arrival\_year,arrival\_month,arrival\_date, no\_of\_previous\_cancellations,no\_of\_previous\_bookings\_not\_canceled, avg\_price\_per\_room, no\_of\_special\_requests.
* Categorical Columns:Booking\_ID, type\_of\_meal\_plan, required\_car\_parking\_space, room\_type\_reserved,market\_segment\_type,repeated\_guest,booking\_status.

**DATASET FEATURES DESCRIPTION**

**Booking\_ID**: A unique identifier for each booking.

**no\_of\_adults**: The number of adults in the reservation.

**no\_of\_children**: The number of children in the reservation.

**no\_of\_weekend\_nights**: The number of weekend nights booked.

**no\_of\_week\_nights**: The number of weekday nights booked.

**type\_of\_meal\_plan**: The type of meal plan (e.g., Full Board, Half Board, etc.).

**required\_car\_parking\_space**: Whether the guest requires parking (1 = Yes, 0 = No).

**room\_type\_reserved**: The type of room reserved (e.g., Single, Double, Suite, etc.).

**lead\_time** (Integer): The number of days between booking and arrival.

**arrival\_year**: The year of arrival.

**arrival\_month**: The month of arrival (1 to 12).

**arrival\_date**: The day of the month when the guest is arriving.

**market\_segment\_type**: The type of market segment (e.g., Online Travel Agent, Corporate, Direct).

**repeated\_guest**: Whether the guest is a repeated customer (1 = Yes, 0 = No).

**no\_of\_previous\_cancellations**: The number of times the guest has canceled a booking in the past.

**no\_of\_previous\_bookings\_not\_canceled**: The number of times the guest has made bookings and not canceled.

**avg\_price\_per\_room**: The average price of the reserved room.

**no\_of\_special\_requests**: The number of special requests made by the guest (e.g., extra pillows, early check-in).

**booking\_status**: The target variable indicating whether the booking was canceled.

**CHAPTER 3**

**DATA ACQUISITION AND INITIAL ANALYSIS**

**DATA LOADING:**

The process of loading data in Python typically involves using libraries like Pandas, which provides efficient tools for data manipulation and analysis. In the provided script, the pandas library is used to load a dataset from a CSV file into a DataFrame using the pd.read\_csv function. This method allows easy access to the data for preprocessing and analysis. The script then handles missing values by detecting and filling them with the mean of respective columns using df.fillna(df.mean(), inplace=True). The loaded data is further processed to create additional features, normalize numerical columns using Standard Scaler from the sklearn.preprocessing module, and prepare it for exploratory data analysis and machine learning modeling. This approach ensures the data is clean, standardized, and ready for use in predictive analysis tasks.

**INITIAL OBSERVATIONS:**

Initial observations from the hotel reservation dataset indicate that it contains a mix of numerical and categorical features. Numerical data such as **lead\_time**, **no\_of\_weekend\_nights**, and **avg\_price\_per\_room** provide measurable aspects of each booking, which can help predict cancellations based on booking characteristics and price. Categorical features like **room\_type\_reserved**, **type\_of\_meal\_plan**, and **market\_segment\_type** offer insights into customer preferences and booking channels. **Booking\_status**, the target variable, indicates whether the reservation was canceled or not, which is crucial for building predictive models. Initial analysis would focus on understanding the distribution of cancellations, correlations between features, and checking for any missing or anomalous values that might affect model performance. This exploration will guide further preprocessing steps.

**CHAPTER 4**

**DATA CLEANING AND PREPROCESSING**

**HANDLING MISSING VALUES:**

Handling missing values in the dataset involves several steps to ensure data quality. First, missing values are identified using functions like isnull(), allowing us to determine the extent of missing data for each column. For categorical columns, missing values can be replaced with the most frequent category (mode). For numerical columns, mean or median imputation is commonly used, depending on the skewness of the data, while interpolation methods can address missing sequential data. Columns or rows with excessive missing values may be dropped if they don’t significantly impact analysis.

**FEATURE ENGINEERING:**

Feature engineering for the hotel reservation cancellation enhances the dataset's predictive power. Date-related features can include identifying the season, the day of the week, or calculating the time window between booking and arrival. Customer behavior can be quantified with a reliability score derived from **repeated\_guest**, **no\_of\_previous\_cancellations**, and **no\_of\_previous\_bookings\_not\_canceled**, while normalizing **no\_of\_special\_requests** by stay length provides insight into request frequency. Stay patterns can be analyzed by combining **no\_of\_weekend\_nights** and **no\_of\_week\_nights** into total nights or calculating the weekend-to-total-night ratio. Price-related features like cost per night (dividing **avg\_price\_per\_room** by total nights) and flags for discounted bookings add further context to the pricing structure. These features improve model performance by capturing meaningful trends.

**DATA TRANSFORMATION:**

Data transformation is essential in preparing the hotel reservation dataset for analysis and modeling. Numerical features like **lead\_time** and **avg\_price\_per\_room** can be normalized and to ensure uniformity and improve model performance. Categorical variables such as **type\_of\_meal\_plan**, **room\_type\_reserved**, and **market\_segment\_type** can be encoded using techniques label encoding, depending on the modeling requirements. Date variables like **arrival\_year**, **arrival\_month**, and **arrival\_date** can be transformed into cyclic features to capture seasonal patterns. These transformations ensure that the data is clean, consistent, and ready for machine learning models.

**CHAPTER - 5**

**EXPLORATORY DATA ANALYSIS**

**DATA INSIGHTS DESCRIPTION**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **DATA INSIGHT** | **DESCRIPTION** |
| **1.** | Guest Information | Most cancellations occur for reservations with two adults, followed by one adult,children or larger groups. |
| **2.** | Time Spent At Hotel | Indicated by weekend and week nights, correlates with commitment, with longer stays reducing cancellations. |
| **3.** | Date Of Arrival | Reflects booking seasonality, with higher cancellations in off-peak periods and fewer cancellations during peak seasons. |
| **4.** | Services | Services like room type, meal plan, parking, and special requests reflect guest preferences, with more requests often correlating with reduced cancellations. |
| **5.** | Lead Time | Measures booking-to-arrival duration; longer lead times often correlate with higher cancellation rates, especially for distant bookings. |
| **6.** | Market Segment | Identifies booking channels (e.g., corporate, online, offline); cancellations vary across segments, with business travelers often canceling less. |
| **7.** | Guest's Previous Experience | Including past cancellations and bookings, significantly influences cancellation likelihood, with frequent cancellers showing higher future cancellations. |
| **8.** | Average Room Price | Indicates booking value; higher prices typically correlate with lower cancellation rates due to guest commitment. |

**DATA INSIGHTS VISUALIZATION:**

# 

# Data Visualization: Confusion Matrix Heatmap

* **Inference:** The heatmap shows how well the model predicts cancellations and non-cancellations, highlighting areas of misclassification, especially false positives or negatives.
* **Observation:** If false positives (non-cancellations predicted as cancellations) are high, it suggests the model is overly conservative, predicting more cancellations than necessary.
* **Implication:** Misclassifications, especially false positives, could lead to unnecessary actions, like overbooking or underestimating capacity, affecting hotel revenue and operations.
* **Recommendation:** Improve model accuracy by tuning hyperparameters, addressing imbalanced classes, or using more advanced techniques like ensemble methods.

# Data Visualization: Correlation Matrix Heatmap

* **Inference:** The heatmap shows how features like **lead\_time**, **no\_of\_adults**, and **avg\_price\_per\_room** relate to the target variable **booking\_status** and other variables
* **Observation:** Strong positive correlations between features like **no\_of\_previous\_bookings\_not\_canceled** and **booking\_status** suggest previous booking behavior strongly influences cancellation likelihood.
* **Implication:** Highly correlated features may indicate redundancy, and excluding one can reduce model complexity and improving performance.
* **Recommendation:** Focus on the influential features for model building, while combining highly correlated to streamline the dataset.

**Data Visualization: Distribution Plot**

* **Inference:** Different models show varied distributions—Logistic Regression have smooth outputs, while Decision Trees are more concentrated.
* **Observation:** Logistic Regression produce continuous probability distributions, while Decision Trees and Random Forests show discrete clusters.
* **Implication:** Discreet, clustered predictions may suggest overfitting or poor generalization, requiring calibration, pruning, or hyperparameter adjustments for stability.
* **Recommendation:** Optimize models like Random Forest for stable performance, prune decision trees, and calibrate logistic regression.

# Data Visualization: Model Comparison

* **Inference:** The visualization shows how models differ in terms of predictive accuracy and error rates, allowing a quick performance comparison.
* **Observation:** Models like **Random Forest** typically show higher accuracy, while **Logistic Regression** and might have lower performance.
* **Implication:** High-performing models indicate better generalization, while lower-performing models may need further tuning or different hyperparameters for optimization.
* **Recommendation:** Prioritize **Random Forest** for accuracy, but consider **Logistic Regression** for simpler, interpretable results. Tune carefully for high-dimensional datasets

**CHAPTER 6**

**PREDICTIVE MODELING**

**MODEL SELECTION AND JUSTIFICATION:**

In the predictive maintenance analysis, three machine learning models were selected: Decision Tree, Random Forest Classifier and Logistic Regression. Each model was chosen based on its ability to address specific characteristics of the dataset and meet the hotel reservation cancellation prediction.

**Decision Tree Algorithm**: It is used to predict booking cancellations by recursively splitting the data into subsets based on specific conditions. Each split in the tree maximizes information gain or minimizes impurity, helping to classify bookings as either canceled or not. The tree builds a flowchart-like structure, where each internal node represents a decision based on a feature, and the leaves indicate the predicted outcome. The model continues splitting until it reaches pure nodes, making the decision process easy to understand and interpret.

**Random Forest Classifier**: It is used to predict cancellations by building an ensemble of decision trees. Each tree is trained on a random subset of the data, and predictions from all trees are aggregated to make a final decision. The algorithm works by randomly selecting subsets of features and samples, reducing overfitting and increasing model robustness. Each decision tree in the forest votes for a predicted class, and the majority vote is chosen as the final prediction. Random Forests are effective for handling large, complex datasets and improving prediction accuracy.

**Logistic Regression**: It is used to predict the probability of booking cancellations. It models the relationship between the input features and the binary outcome (canceled or not canceled) by fitting a logistic function. The algorithm estimates the probability of cancellation by calculating a weighted sum of the input features, passed through a sigmoid function that outputs values between 0 and 1. The model is trained using historical booking data, and the coefficients represent the impact of each feature on the likelihood of cancellation. Logistic regression is simple, interpretable, and efficient for binary classification tasks.

**Justification**: The models selected for the hotel reservation cancellation project each bring unique strengths. **Logistic Regression** is used for its simplicity and interpretability, effectively modeling binary outcomes with linear relationships. **Decision Trees** are chosen for their ability to capture non-linear patterns, though they can overfit, which is addressed by using **Random Forest**—an ensemble method that increases accuracy and stability by combining multiple trees. Together, these models provide a comprehensive approach to predicting hotel reservation cancellations.

**DATA PARTITIONING:**

**Data partitioning** is a crucial step to ensure the model’s ability to generalize well to unseen data. The dataset is divided into three primary sets: **training**, **validation**, and **test**. First, the dataset is randomly split into a **training set** (typically 70-80% of the data) and a **test set** (the remaining 20-30%). The **training set** is used to build the models, allowing them to learn from the data by adjusting model parameters. The **validation set** is used during model training to fine-tune hyperparameters and avoid overfitting. Once the model is optimized, the **test set** is used to evaluate the final model’s performance and check its generalization ability. This partitioning ensures that the model is not trained and tested on the same data, providing a more reliable estimate of its real-world performance.

**MODEL TRAINING AND HYPERPARAMETER TUNING:**

For model training and hyperparameter tuning, each of the selected models—Decision, Random Forest Classifier and Logistic Regression—underwent training on the training set to learn from the data patterns, followed by tuning to improve their performance.

**Decision Tree Algorithm**: The Decision Tree model was initially trained with default parameters. Hyperparameter tuning was conducted using **GridSearchCV** to identify the optimal settings. The key parameters tuned included the **maximum depth of the tree, minimum samples required to split a node, minimum samples required at a leaf node**, and the **criterion** used to evaluate splits. The tuning aimed to enhance model accuracy by preventing overfitting and ensuring better generalization. Cross-validation was also incorporated to evaluate performance consistently and improve the model's robustness.

**Random Forest Algorithm**: The Random Forest model was trained initially with default parameters. Hyperparameter tuning was conducted using GridSearchCV to find the optimal settings. Parameters tuned included the number of trees, maximum depth of trees, minimum samples to split a node (min\_samples\_split), and minimum samples at each leaf node. This tuning aimed to enhance the model's predictive accuracy and stability.

**Logistic Regression**: The Logistic Regression model was trained using a standard configuration, and then tuning was performed by adjusting the regularization strength parameter (C) and the solver type. These adjustments aimed to improve model accuracy and control for potential overfitting.

**CHAPTER 7**

**MODEL EVALUATION AND OPTIMIZATION**

**PERFORMANCE ANALYSIS:**

The models' performances were evaluated using a set of metrics relevant to classification tasks, such as accuracy, precision, recall, F1-score.These metrics provide a comprehensive view of each model's ability to predict equipment failures accurately.

**Decision Tree Algorithm**: The Decision Tree model demonstrated higher **accuracy** than Logistic Regression. While the model achieved balanced **precision** and **recall**, it required tuning to prevent overfitting and ensure better generalization on unseen data.

**Random Forest Algorithm**: The Random Forest model achieved high accuracy, reflecting its capability to capture complex patterns in the data. Model demonstrated higher **accuracy** than Logistic Regression. While the model achieved balanced **precision** and **recall**, it required tuning to prevent overfitting and ensure better generalization on unseen data.

**Logistic Regression**: Logistic Regression showed good accuracy but slightly lower performance on recall compared to Random Forest, suggesting that it may miss some failure cases.Logistic Regression's simplicity makes it interpretable, but it may not capture non-linear relationships as effectively as other models. It performed well in terms of **precision** but had lower **recall**, particularly for predicting cancellations.

**sFEATURE IMPORTANCE:**

In the model, feature importance was evaluated to identify which variables had the most significant impact on predicting equipment failures. Determine which input features have the most influence on the model’s predictions. It helps identify which features are the most predictive of the target variable and should be prioritized for model training. Feature importance can be computed using various methods, such as measuring the reduction in impurity in decision trees, examining the magnitude of coefficients in linear models.

**Arrival Year, Month, and Day**: The time of year, month, and day of arrival significantly impact cancellation rates. Seasonal factors, such as holidays or peak tourist seasons, tend to reduce cancellations, as guests are more committed to traveling during these times. Off-season or weekdays may see higher cancellations, as people are more likely to change plans. Understanding arrival trends helps hotels anticipate fluctuations in booking behavior and plan accordingly for cancellations.

**Previous Cancellations**: The number of previous cancellations by a guest is a strong predictor of future behavior. If a guest has canceled multiple bookings in the past, they are more likely to cancel future reservations. This feature acts as a historical indicator, signaling that a guest may not be as committed to honoring their bookings.

**Lead Time**:Lead time measures the number of days between a reservation and the guest’s arrival. Longer lead times often result in higher cancellation rates as guests may change plans. It indicates uncertainty in the booking and can highlight the likelihood of cancellation, as early bookings might not be as firm.

**Number of Adults**: The number of adults in a reservation influences cancellation behavior. Reservations made for more adults might reflect a higher commitment to the booking, leading to fewer cancellations. Larger groups, especially in the case of family bookings, tend to show more certainty in their plans. However, single adult bookings or smaller groups may have a higher cancellation likelihood, as they could be more likely to make last-minute changes or adjustments to their plans.

**MODEL REFINEMENT**:

**Model Refinement** involves improving the performance of the machine learning models to better predict cancellations. It focuses on optimizing the model's parameters, features, and algorithms to achieve higher accuracy, precision, recall.

**Hyperparameter Tuning**: Hyperparameter tuning was performed using GridSearchCV to optimize the model parameters and improve accuracy. This involved testing different values for parameters such as the number of estimators in Random Forest, the depth of the trees in Decision Tree, and the regularization strength in Logistic Regression. By evaluating all possible combinations of hyperparameters, the model could be fine-tuned to achieve the best performance, minimizing overfitting and underfitting while improving the model’s generalization to unseen data.

**Confusion Matrix**: **Confusion matrix** was used to assess the performance of the classification models by comparing the predicted results with the actual outcomes. It provides insights into the true positives, false positives, true negatives, and false negatives. This matrix helps evaluate the model’s accuracy, precision, recall, and F1-score, highlighting its strengths and weaknesses in predicting cancellations versus non-cancellations. It also guides further refinement by identifying areas for model improvement, such as reducing false positives or improving recall for cancellations.

**Distribution Plot**: A **distribution plot** was used to visualize the spread and distribution of key features, such as lead time, room type, or number of adults, within the dataset. It provided valuable insights into how these features vary across different reservation cancellations. By analyzing the distribution of features, we were able to detect outliers, skewed distributions, or imbalances, which helped in data preprocessing and feature engineering. This visualization was crucial in identifying the most influential features for model refinement and ensuring that they were well-represented in the training process.

**Classification Report**: **Classification report** provided a detailed summary of the model’s performance, including precision, recall, and F1-score for each class (canceled vs. not canceled). This report was essential for evaluating the model's ability to correctly classify reservation cancellations. It helped identify whether the model was biased towards predicting one class over the other, especially in cases of class imbalance. By analyzing the classification report, we could fine-tune the model further, focusing on improving metrics like recall for cancellation predictions and overall classification accuracy.

**CHAPTER 8**

**DISCUSSION AND CONCLUSION**

**SUMMARY OF FINDINGS:**

This project focused on analyzing hotel reservation data to predict booking cancellations, aiming to optimize hotel operations and improve revenue management. Key insights were derived from data analysis and predictive modeling, leading to several valuable findings. By understanding patterns in booking behavior, such as the impact of lead time, number of adults, and room type, we were able to predict cancellations with higher accuracy. These insights can guide proactive strategies for overbooking, customer engagement, and targeted offers, helping the hotel minimize revenue loss from cancellations and improve overall operational efficiency.

**Data Analysis Insights**: Exploratory Data Analysis (EDA) in the hotel reservation cancellation project uncovered significant patterns and relationships within the dataset. Features like **lead time**, **number of adults**, and **room type reserved** showed clear differences between canceled and non-canceled bookings, indicating a strong link to reservation stability. Engineered features, such as **previous cancellations** and **special requests**, revealed intricate relationships between guest behavior and cancellation likelihood. Additionally, **market segment** and **seasonal trends** highlighted external factors influencing cancellations.

**Impact of Key Features**: Feature importance analysis in the hotel reservation cancellation project identified **lead time**, **number of adults, room type reserved**, and engineered features like **previous cancellations** and **special requests** as the most impactful in predicting cancellations. These findings highlight the significance of monitoring these factors, as they provide early indicators of booking behavior and cancellation risks. By understanding these key features, hotels can proactively manage reservations, implement targeted strategies to reduce cancellations, and improve overall operational efficiency.

**Model Performance**: Among the models tested, **Random Forest** and **Decision tree** achieved the best results, showing high accuracy, precision, and recall in predicting hotel reservation cancellations. Both models excelled in identifying key patterns and accurately predicting cancellations, which is crucial for effective booking management. **Logistic Regression**, while serving as a useful benchmark, was less effective in capturing complex interactions between features compared to the other models. This highlights the importance of using more advanced algorithms like Random Forest and Decision Tree for better performance in predicting cancellations.

**Refinement and Optimization**: Model performance in the hotel reservation cancellation project was further improved through **confusion matrix**, **hyperparameter tuning**, **Distribution plot and classification report**. These refinements ensured the models provided accurate predictions while minimizing overfitting. By fine-tuning parameters and adjusting for class imbalance, the models not only performed well on the training data but also generalized effectively to unseen data. This made the models more reliable and applicable for real-world scenarios, helping the hotel better predict cancellations and optimize their operations for improved revenue management and customer service.

**CHALLENGES AND LIMITATIONS:**

This project on hotel reservation cancellation prediction faced several challenges and limitations, primarily related to data quality, feature complexity, and model optimization. Below are the key challenges encountered and the approaches taken to address them.

**Data Quality and Missing Values** Missing data, particularly in features like number of special requests or market segment type, posed a challenge. To address this, missing values were imputed using the mean for numerical columns. However, this approach assumes randomness in missingness and may overlook patterns in missing data, which could limit model accuracy.

**Class Imbalance**: he dataset had an imbalance between canceled and non-canceled reservations, with more non-cancellations. This imbalance risked model bias towards the majority class. To mitigate this, class weights were adjusted in models like Random Forest and Logistic Regression, and metrics like precision, recall were prioritized for better performance on the minority class.

**Feature Complexity**: Identifying meaningful features was complex due to intricate relationships within the dataset. Feature engineering, such as creating interaction terms and examining historical cancellation data, helped uncover hidden patterns. However, the complexity of guest behavior may still require advanced techniques to capture deeper relationships.

**Computational Costs**: Hyperparameter tuning, especially using GridSearchCV, was computationally expensive and time-consuming. To manage this, the parameter search space was reduced based on domain knowledge, optimizing performance while lowering computational costs.

**Model Generalization**: Ensuring the models generalized well to new, unseen data was a challenge, particularly in preventing overfitting. Cross-validation techniques were employed to ensure robustness, but reliance on a single dataset limited generalizability across different hotel types or seasons.

**Limited Interpretability in Complex Models**: Models like Decision Tree and Random Forest while highly accurate, lack transparency compared to simpler models like Logistic Regression. This made explaining predictions to stakeholders more difficult. Feature importance scores and visualizations were used to enhance interpretability, though further efforts are needed to improve clarity and communication with non-technical stakeholders.

**APPENDIX**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

df = pd.read\_csv('Hotel Reservations.csv')

df.head()

# DATA PREPROCESSING – 1 :

df.shape

df.drop(['Booking\_ID'], axis=1, inplace=True)

df['date of arrival'] = df['arrival\_year'].astype(str) + '/' + df['arrival\_month'].astype(str) + '/' + df['arrival\_date'].astype(str)

df['date of arrival'] = pd.to\_datetime(df['date of arrival'],format='mixed', infer\_datetime\_format=True, errors='coerce',yearfirst=True)

df.drop(columns=['arrival\_date', 'arrival\_month', 'arrival\_year'], inplace=True)

df.isnull().sum()

df.dropna(inplace=True)

df.reset\_index()

df.dtypes

df.nunique()

df.describe()

df['avg\_price\_per\_room'].replace(0,df['avg\_price\_per\_room'].mean(), inplace=True)

df.drop(df[df['no\_of\_adults'] == 0].index, inplace = True)

df.head()

# 

# EXPLORATORY DATA ANALYSIS (EDA) :

# 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.

fig, ax = plt.subplots(1,2,figsize=(15,5))

sns.countplot( x = 'no\_of\_adults', data = df, ax=ax[0]).set\_title('Number of Adults')

sns.countplot( x = 'no\_of\_children', data = df, ax=ax[1]).set\_title('Number of Children')

fig, ax = plt.subplots(1,2,figsize=(15,5))

sns.countplot(x = 'no\_of\_weekend\_nights', data = df, ax=ax[0]).set\_title('Number of Weekend Nights')

sns.countplot(x = 'no\_of\_week\_nights', data = df, ax=ax[1]).set\_title('Number of Week Nights')

fig, ax = plt.subplots(2,2,figsize=(20,10))

ax[0,0].pie(df['date of arrival'].dt.year.value\_counts(), labels = [2018,2017], autopct='%1.1f%%', shadow=True, startangle=90)

ax[0,0].set\_title('Year of arrival')

sns.histplot(x = df['date of arrival'].dt.month, ax=ax[0,1], bins=12, hue = df['date of arrival'].dt.year, palette = 'Set1').set\_title('Month of arrival')

sns.histplot(x = df['date of arrival'].dt.day, ax=ax[1,0], bins=31, hue = df['date of arrival'].dt.year, palette = 'Set1').set\_title('Day of arrival')

sns.histplot(x = df['date of arrival'].dt.dayofweek, ax=ax[1,1], bins=7, hue = df['date of arrival'].dt.year, palette = 'Set1').set\_title('Day of week of arrival')

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 Plan')

ax[0,0].xaxis.set\_tick\_params(rotation=90)

sns.countplot(x = 'room\_type\_reserved', data = df, ax=ax[0,1]).set\_title('Room Type Reserved')

ax[0,1].xaxis.set\_tick\_params(rotation=90)

sns.countplot(x = 'required\_car\_parking\_space', data = df, ax=ax[1,0]).set\_title('Required Car Parking')

sns.countplot(x = 'no\_of\_special\_requests', data = df, ax=ax[1,1]).set\_title('Number of special requests')

sns.histplot(x = 'lead\_time', data = df, bins=100).set\_title('Lead Time in days')

sns.countplot(x = 'market\_segment\_type', data = df).set\_title('Market Segment Type')

fig, ax = plt.subplots(1,3,figsize=(20,6))

sns.countplot(x = 'repeated\_guest', data = df, ax=ax[0]).set\_title('Repeated Guest')

sns.histplot(x = 'no\_of\_previous\_cancellations', data = df, ax=ax[1], bins = 9).set\_title('Number of Previous Cancellations')

sns.histplot(x = 'no\_of\_previous\_bookings\_not\_canceled', data = df, ax=ax[2], bins = 30).set\_title('Number of Previous Bookings Not Cancelled')

sns.histplot(x = 'avg\_price\_per\_room', data = df, bins = 100).set\_title('Average Room Price')

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.

fig, ax = plt.subplots(1,2,figsize=(15,5))

sns.countplot( x = 'no\_of\_adults', data = df, ax=ax[0], hue= 'booking\_status').set\_title('Number of Adults')

sns.countplot( x = 'no\_of\_children', data = df, ax=ax[1], hue = 'booking\_status').set\_title('Number of Children')

fig, ax = plt.subplots(1,2,figsize=(15,5))

sns.countplot(x = 'no\_of\_weekend\_nights', data = df, ax=ax[0], hue = 'booking\_status').set\_title('Number of Weekend Nights')

sns.countplot(x = 'no\_of\_week\_nights', data = df, ax=ax[1], hue = 'booking\_status').set\_title('Number of Week Nights')

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]

sns.countplot(x = df\_2017['booking\_status'], data = df\_2017, ax=ax[0,0]).set\_title('Cancellation in 2017')

sns.countplot(x = df\_2018['booking\_status'], data = df\_2018, ax=ax[0,1]).set\_title('Cancellation in 2018')

sns.histplot(x = df\_2017['date of arrival'].dt.month, data = df\_2017, ax=ax[1,0], bins=6, hue = df\_2017['booking\_status'], palette = 'Set1', multiple = 'stack').set\_title('Cancellation by months in 2017')

sns.histplot(x = df\_2018['date of arrival'].dt.month, data = df\_2018, ax=ax[1,1], bins=12, hue = df\_2018['booking\_status'], palette = 'Set1', multiple ='stack').set\_title('Cancellation by months in 2018')

sns.histplot(x = df\_2017['date of arrival'].dt.day, data = df\_2017, ax=ax[2,0], bins=31, hue = df\_2017['booking\_status'], palette = 'Set1', multiple='stack').set\_title('Cancellation by date in 2017')

sns.histplot(x = df\_2018['date of arrival'].dt.day, data = df\_2018, ax=ax[2,1], bins=31, hue = df\_2018['booking\_status'], palette = 'Set1', multiple ='stack').set\_title('Cancellation by date in 2018')

sns.histplot(x = df\_2017['date of arrival'].dt.dayofweek, data = df\_2017, ax=ax[3,0], bins=7, hue = df\_2017['booking\_status'], palette = 'Set1', multiple = 'stack').set\_title('Cancellation by day of week in 2017')

sns.histplot(x = df\_2018['date of arrival'].dt.dayofweek, data = df\_2018, ax=ax[3,1], bins=7, hue = df\_2018['booking\_status'], palette = 'Set1', multiple = 'stack').set\_title('Cancellation by day of week in 2018')

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\_status').set\_title('Meal Plan')

ax[0,0].xaxis.set\_tick\_params(rotation=90)

sns.countplot(x = 'room\_type\_reserved', data = df, ax=ax[0,1], hue = 'booking\_status').set\_title('Room Type Reserved')

ax[0,1].xaxis.set\_tick\_params(rotation=90)

sns.countplot(x = 'required\_car\_parking\_space', data = df, ax=ax[1,0], hue = 'booking\_status').set\_title('Required Car Parking')

sns.countplot(x = 'no\_of\_special\_requests', data = df, ax=ax[1,1], hue = 'booking\_status').set\_title('Number of special requests')

sns.histplot(x = 'lead\_time', data = df, bins=100, hue = 'booking\_status', multiple = 'stack').set\_title('Lead Time in days')

sns.countplot(x = 'market\_segment\_type', data = df, hue = 'booking\_status').set\_title('Market Segment Type')

sns.countplot(x = 'repeated\_guest', data = df, hue = 'booking\_status').set\_title('Repeated Guest')

sns.histplot(x = 'avg\_price\_per\_room', data = df, bins = 100, hue = 'booking\_status', multiple = 'stack').set\_title('Average Room Price')

**OUTLIER REMOVAL USING IQR:**

cols = ['lead\_time', 'avg\_price\_per\_room']

Q1 = df[cols].quantile(0.25)

Q3 = df[cols].quantile(0.75)

IQR = Q3 - Q1

df = df[~((df[cols] < (Q1 - 1.5 \* IQR)) |(df[cols] > (Q3 + 1.5 \* IQR))).any(axis=1)]

**LABEL ENCODING :**

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

cols = ['type\_of\_meal\_plan', 'room\_type\_reserved', 'market\_segment\_type', 'booking\_status']

for col in cols:

    le.fit(df[col])

    df[col] = le.transform(df[col])

    print(col, df[col].unique())

**FEATURE SCALING :**

from sklearn.preprocessing import StandardScaler

#standardizing the data

scaler = StandardScaler()

df[['lead\_time', 'avg\_price\_per\_room']] = scaler.fit\_transform(df[['lead\_time', 'avg\_price\_per\_room']])

**CORRELATION MATRIX HEATMAP :**

plt.figure(figsize=(15,10))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

df.drop(columns=['date of arrival'], inplace=True)

**TRAIN TEST SPLIT :**

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df.drop('booking\_status', axis=1), df['booking\_status'], test\_size=0.2, random\_state=42)

**MODEL BUILDING :**

**DECISION TREE CLASSIFIER :**

from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()

**HYPERPARAMETER TUNING USING GRIDSEARCHCV :**

from sklearn.model\_selection import GridSearchCV

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 = GridSearchCV(estimator=dtree, param\_grid=grid\_param, cv=5, n\_jobs=-1, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

print(grid\_search.best\_params\_)

dtree = DecisionTreeClassifier(criterion='entropy', max\_depth=8, min\_samples\_leaf=4, min\_samples\_split=2, random\_state=0)

dtree.fit(X\_train, y\_train)

print(dtree.score(X\_train, y\_train))

d\_pred = dtree.predict(X\_test)

**RANDOM FOREST CLASSIFIER:**

from sklearn.ensemble import RandomForestClassifier

#random forest classifier object

rfc = RandomForestClassifier()

**HYPERPARAMETER TUNING USING GRIDSEARCHCV :**

from sklearn.model\_selection import GridSearchCV

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 = GridSearchCV(estimator=rfc, param\_grid=grid\_param, cv=5, n\_jobs=-1)

grid\_search.fit(X\_train, y\_train)

print(grid\_search.best\_params\_)

rfc = RandomForestClassifier(criterion='entropy', max\_depth=8, min\_samples\_leaf=4, min\_samples\_split=2, random\_state=0)

rfc.fit(X\_train, y\_train)

print(rfc.score(X\_train, y\_train))

r\_pred = rfc.predict(X\_test)

**LOGISTIC REGRESSION:**

from sklearn.linear\_model import LogisticRegression

logreg = LogisticRegression()

**HYPERPARAMETER TUNING USING GRIDSEARCHCV :**

from sklearn.model\_selection import GridSearchCV

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 = GridSearchCV(estimator=logreg, param\_grid=grid\_param, cv=5, n\_jobs=-1)

grid\_search.fit(X\_train, y\_train)

print(grid\_search.best\_params\_)

logreg = LogisticRegression(C=1, penalty='l2', random\_state=0, solver='liblinear')

logreg.fit(X\_train, y\_train)

print(logreg.score(X\_train, y\_train))

l\_pred = logreg.predict(X\_test)

**MODEL EVALUATION :**

**CONFUSION MATRIX HEATMAP :**

from sklearn.metrics import confusion\_matrix

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

sns.heatmap(confusion\_matrix(y\_test, d\_pred), annot=True, cmap='coolwarm', ax=ax[0]).set\_title('Decision Tree')

sns.heatmap(confusion\_matrix(y\_test, r\_pred), annot=True, cmap='coolwarm', ax=ax[1]).set\_title('Random Forest')

sns.heatmap(confusion\_matrix(y\_test, l\_pred), annot=True, cmap='coolwarm', ax=ax[2]).set\_title('Logistic Regression')

**DISTRIBUTION PLOT :**

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

sns.distplot(y\_test, ax=ax[0], hist= False).set\_title('Decision Tree')

sns.distplot(d\_pred, ax=ax[0], hist = False)

sns.distplot(y\_test, ax=ax[1], hist= False).set\_title('Random Forest')

sns.distplot(r\_pred, ax=ax[1], hist = False)

sns.distplot(y\_test, ax=ax[2], hist= False).set\_title('Logistic Regression')

sns.distplot(l\_pred, ax=ax[2], hist = False)

**CLASSIFICATION REPORT :**

from sklearn.metrics import classification\_report

print('Decision Tree')

print(classification\_report(y\_test, d\_pred))

print('Random Forest')

print(classification\_report(y\_test, r\_pred))

print('Logistic Regression')

print(classification\_report(y\_test, l\_pred))

**MODEL METRICS :**

from sklearn.metrics import accuracy\_score, mean\_absolute\_error, mean\_squared\_error

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')

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')

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))

**MODEL COMPARISON :**

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

sns.barplot(x = ['Decision Tree', 'Random Forest', 'Logistic Regression'], y = [accuracy\_score(y\_test, d\_pred), accuracy\_score(y\_test, r\_pred), accuracy\_score(y\_test, l\_pred)], ax=ax[0]).set\_title('Accuracy Score')

sns.barplot(x = ['Decision Tree', 'Random Forest', 'Logistic Regression'], y = [mean\_absolute\_error(y\_test, d\_pred), mean\_absolute\_error(y\_test, r\_pred), mean\_absolute\_error(y\_test, l\_pred)], ax=ax[1]).set\_title('Mean Absolute Error')

sns.barplot(x = ['Decision Tree', 'Random Forest', 'Logistic Regression'], y = [mean\_squared\_error(y\_test, d\_pred), mean\_squared\_error(y\_test, r\_pred), mean\_squared\_error(y\_test, l\_pred)], ax=ax[2]).set\_title('Mean Squared Error')

**FEATURE IMPORTANCE:**

feature\_importance = pd.DataFrame({'Features': X\_train.columns, 'Importance': dtree.feature\_importances\_})

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\_title('Decision Tree')

feature\_importance = pd.DataFrame({'Features': X\_train.columns, 'Importance': rfc.feature\_importances\_})

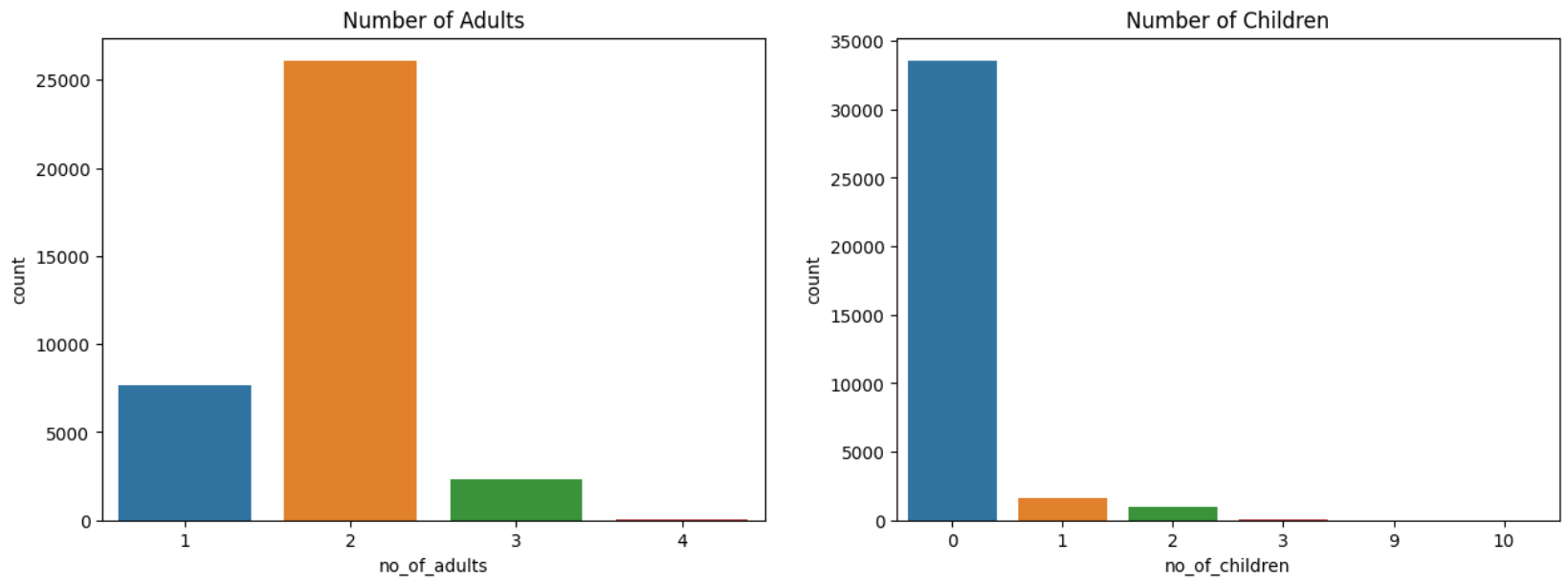
feature\_importance.sort\_values(by='Importance', ascending=False, inplace=True)

feature\_importance.reset\_index(drop=True, inplace=True)

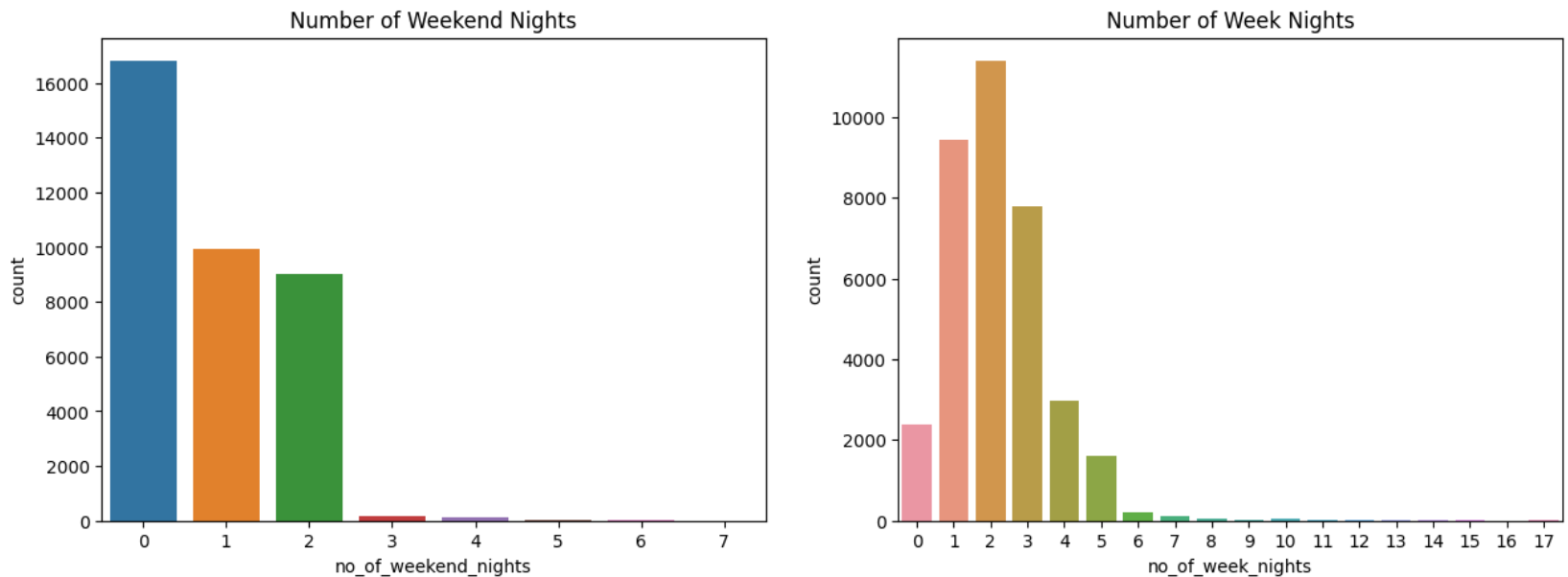
sns.barplot(x = 'Importance', y = 'Features', data = feature\_importance).set\_title('Random Forest')

**OUTPUT SCREENSHOTS:**

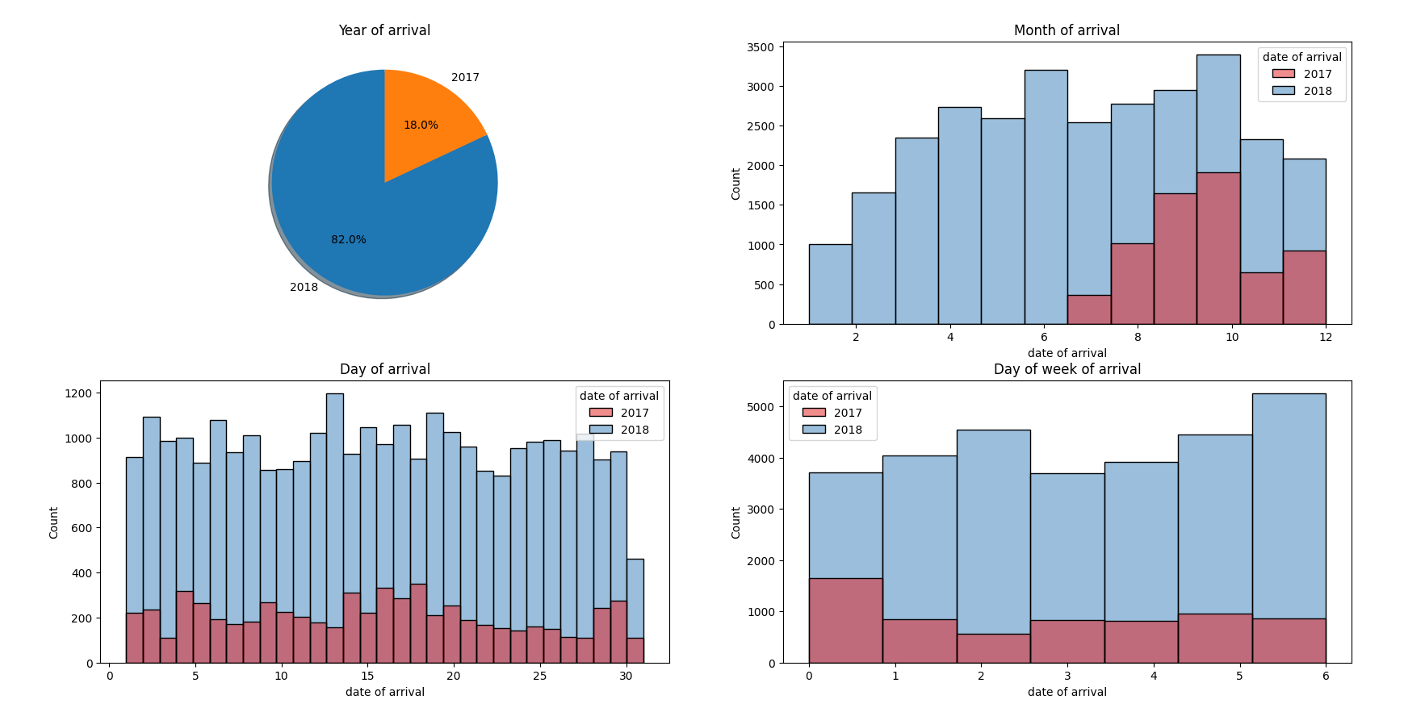
**GUEST INFORMATION :**

****

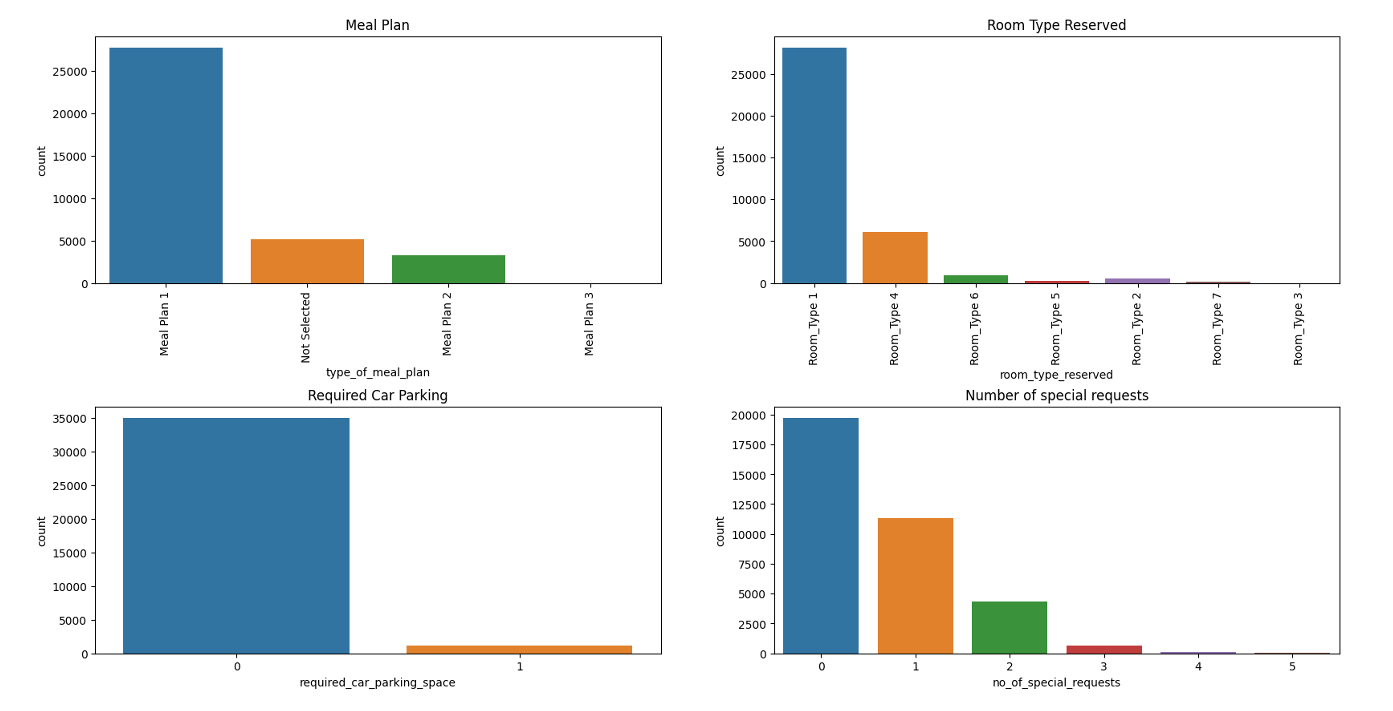
**TIME SPEND AT HOTEL :**



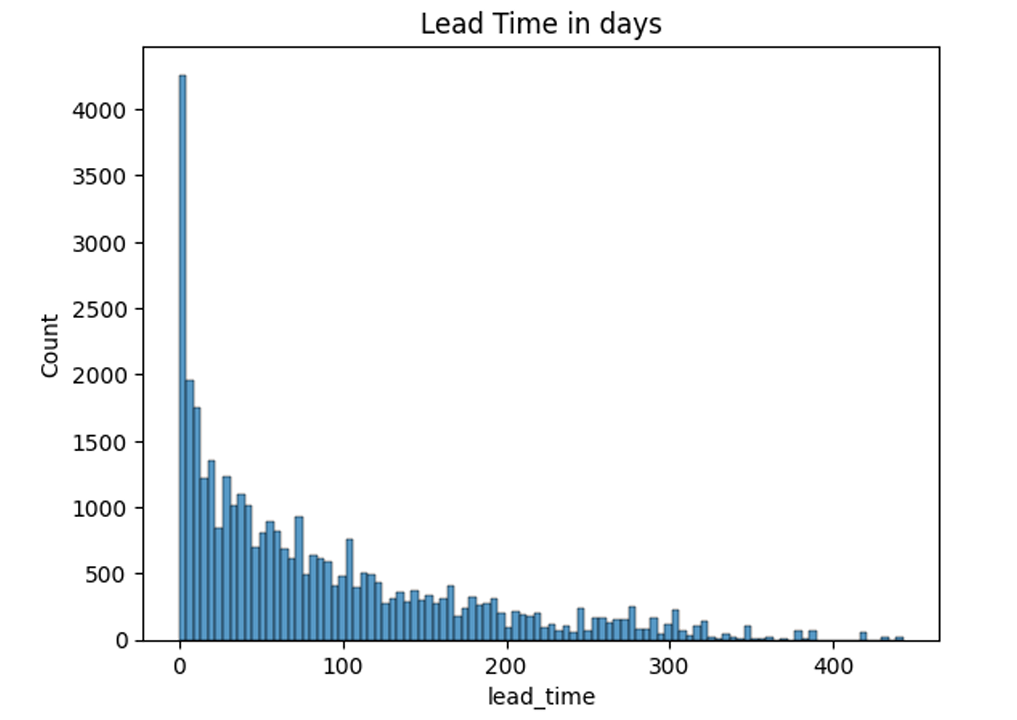
**DATE OF ARRIVAL :**

****

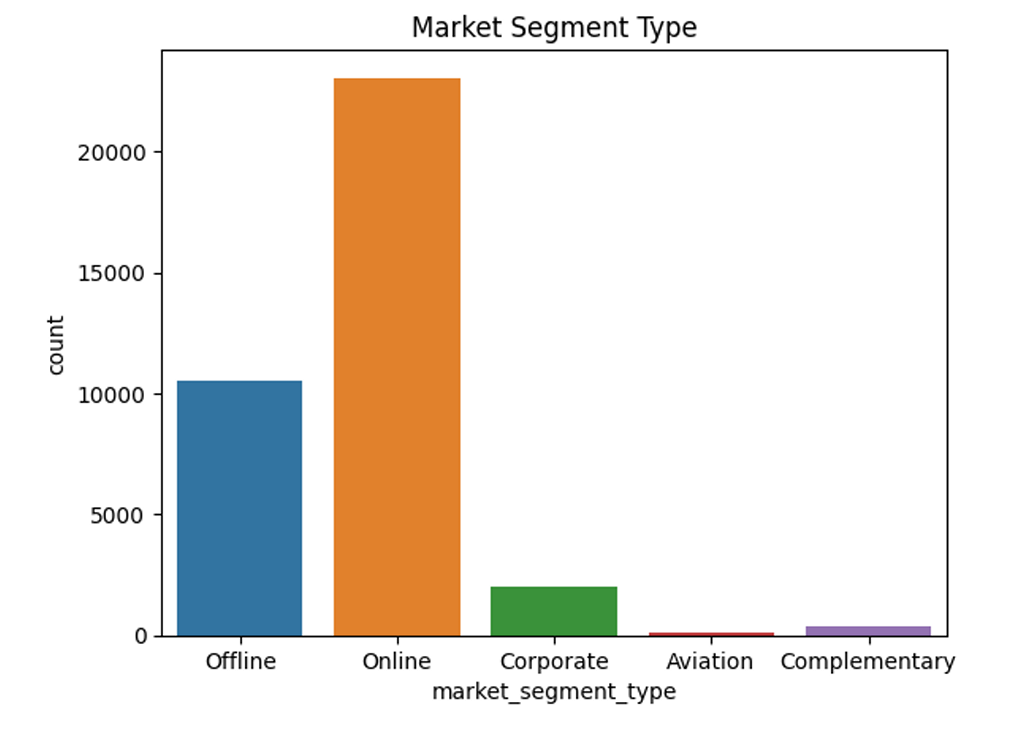
**SERVICES :**



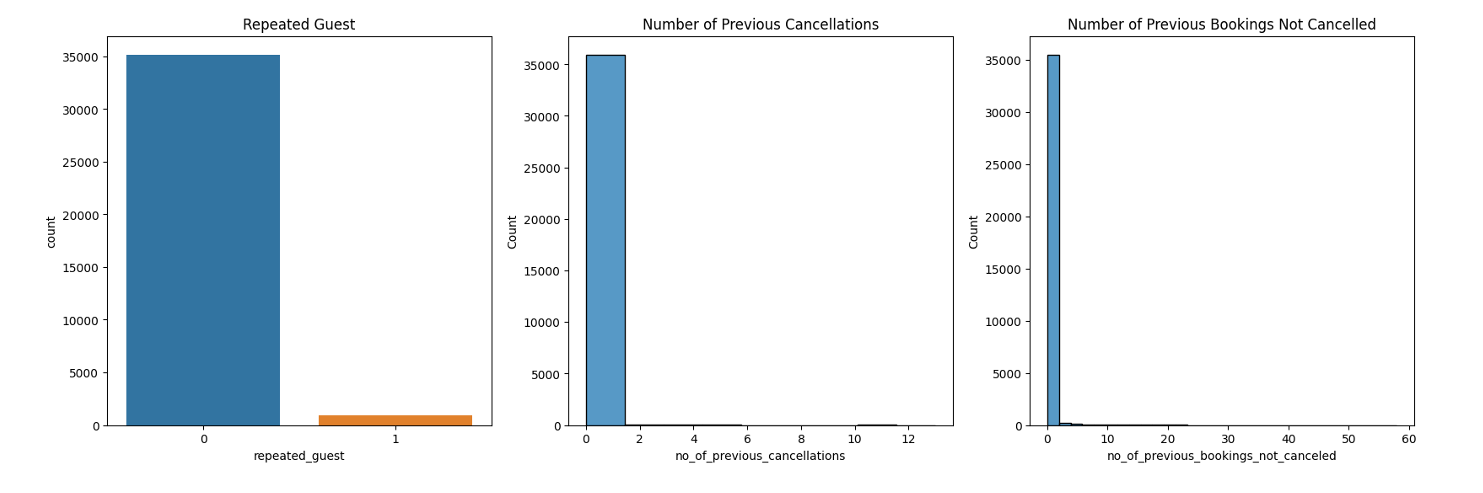
**LEAD TIME (days between date of reservation and date of arrival) :**

****

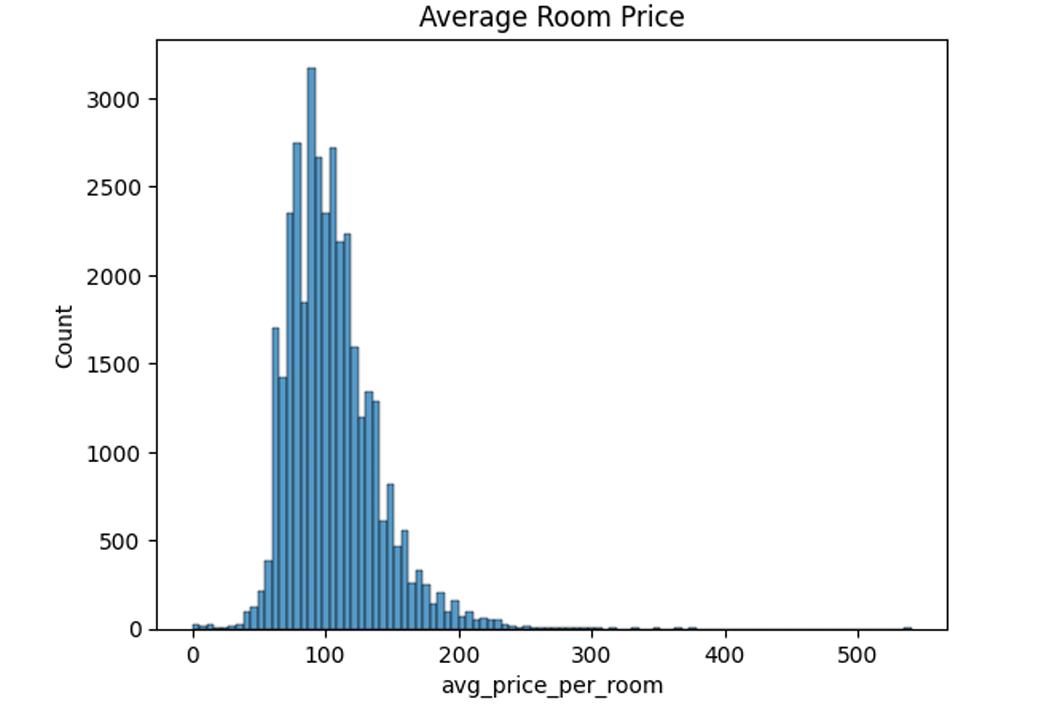
**MARKET SEGMENT :**

****

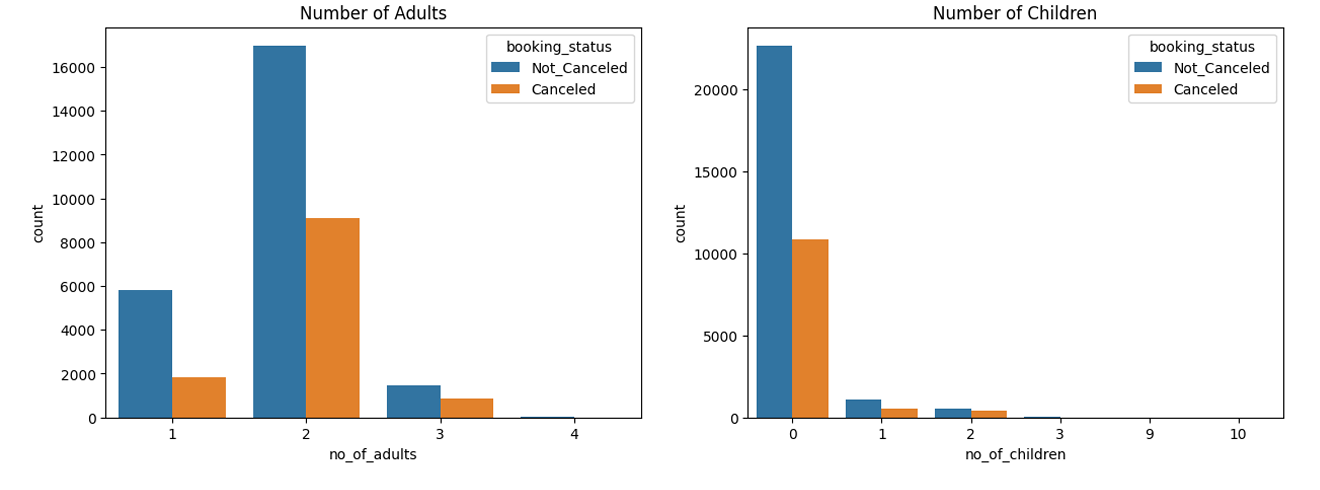
**GUEST'S PREVIOUS EXPERIENCE WITH THE HOTEL :**

****

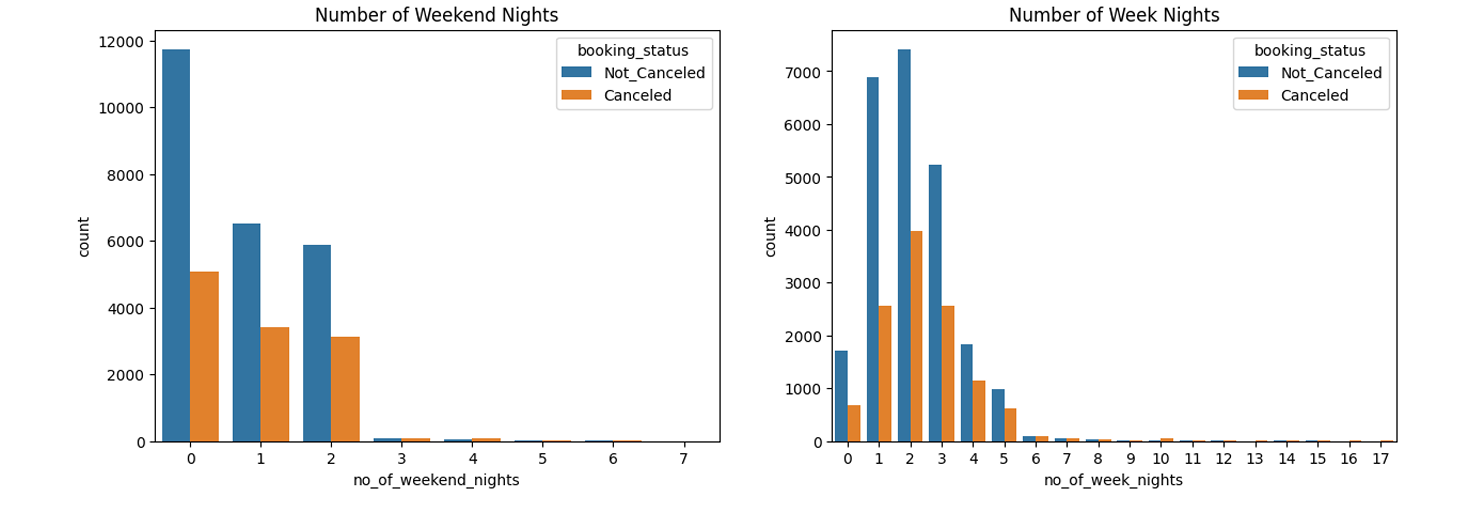
**AVERAGE ROOM PRICE :**

****

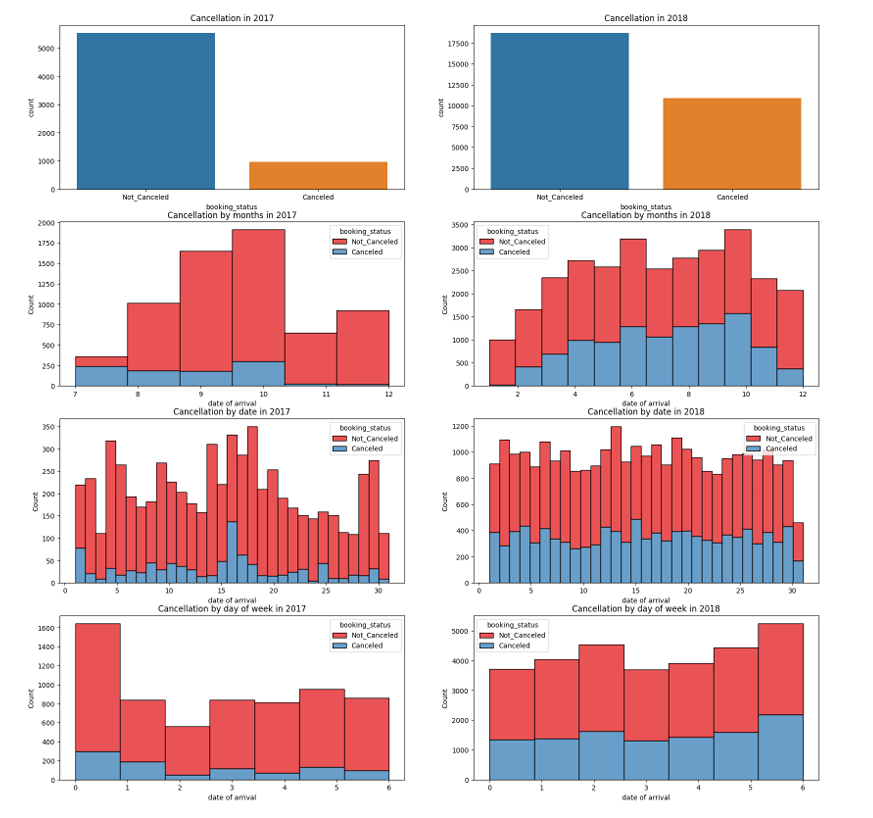
**GUEST INFORMATION AND CANCELLATION :**

****

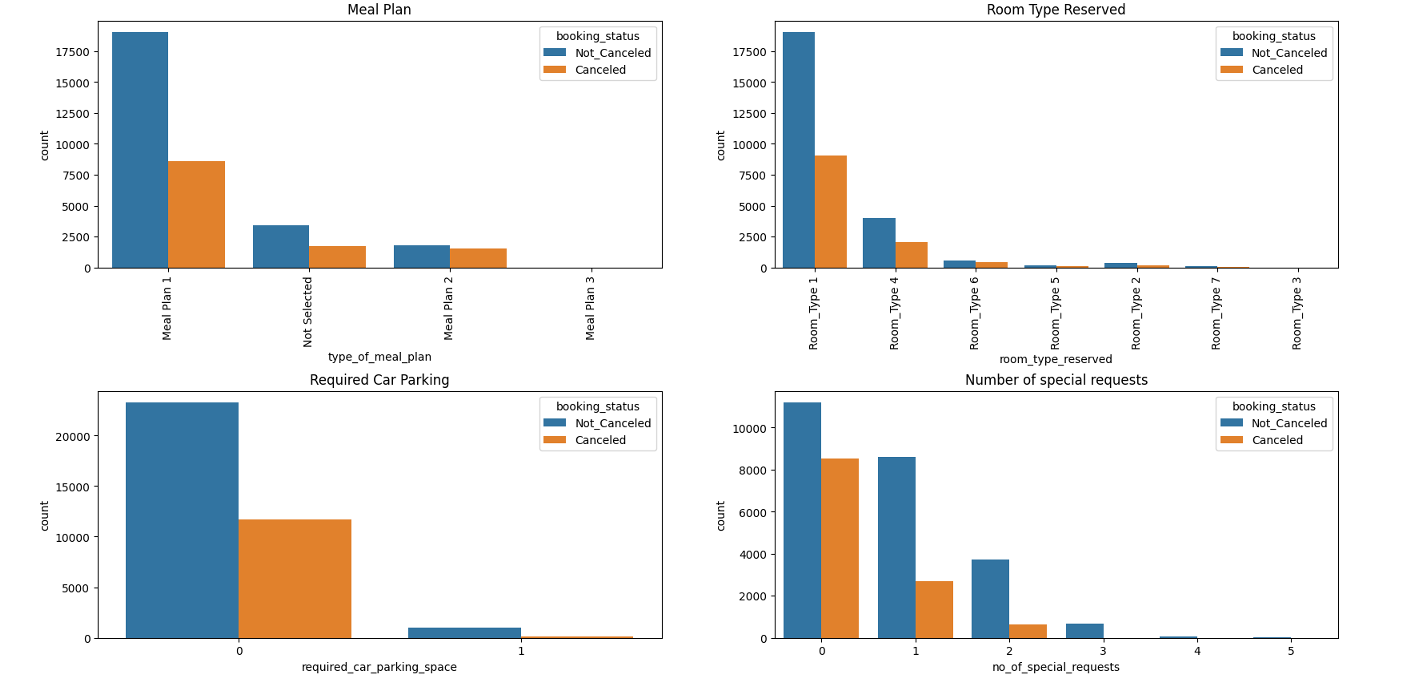
**TIME SPEND AT HOTEL AND CANCELLATION :**

****

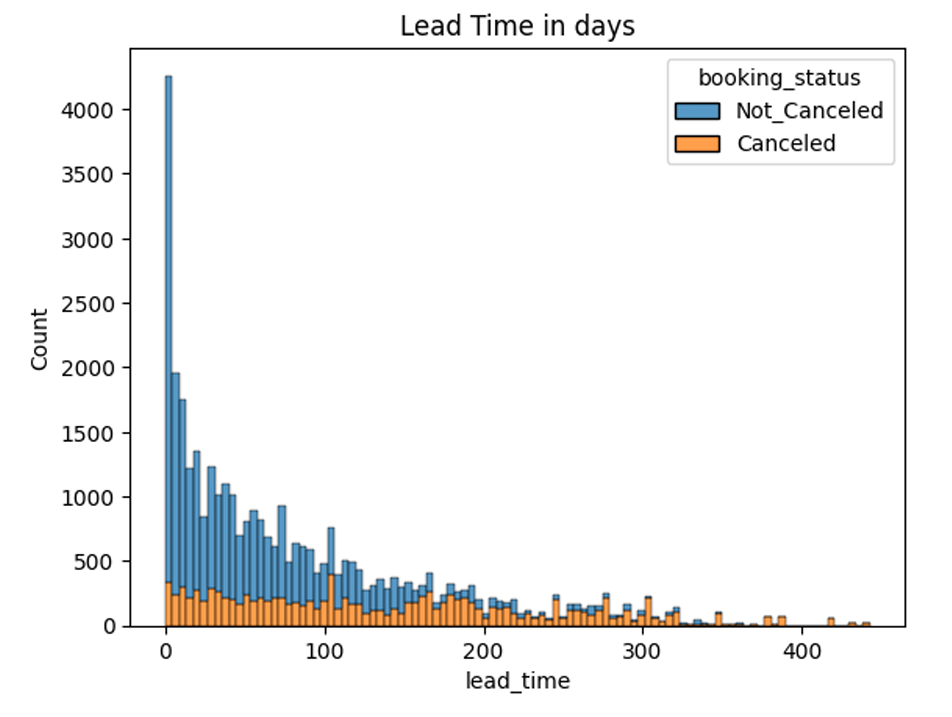
**DATE OF ARRIVAL AND CANCELLATION :**



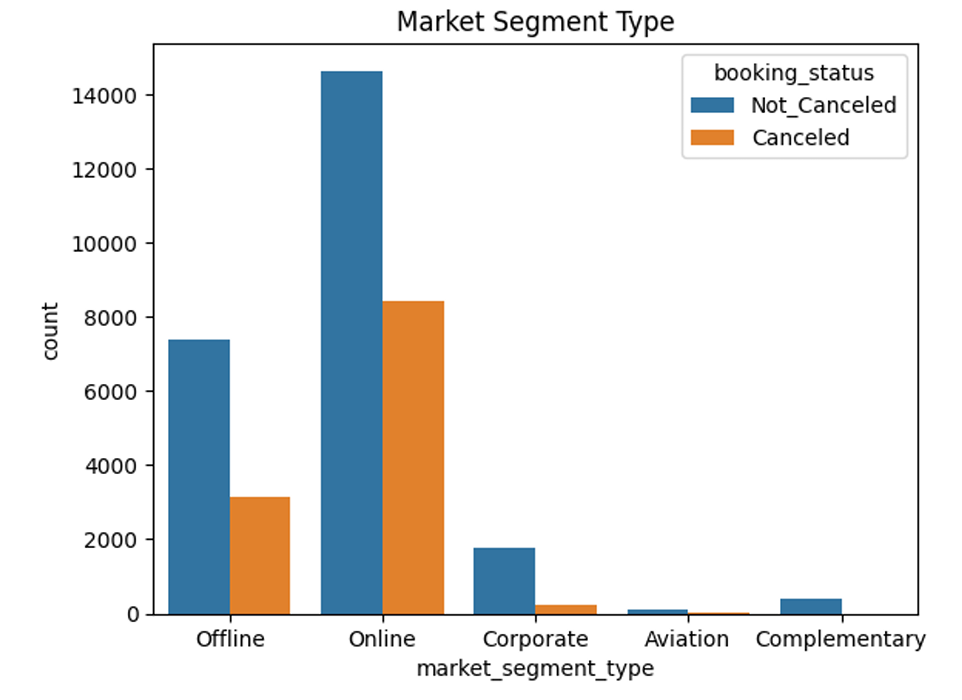
**SERVICES AND CANCELLATION :**

****

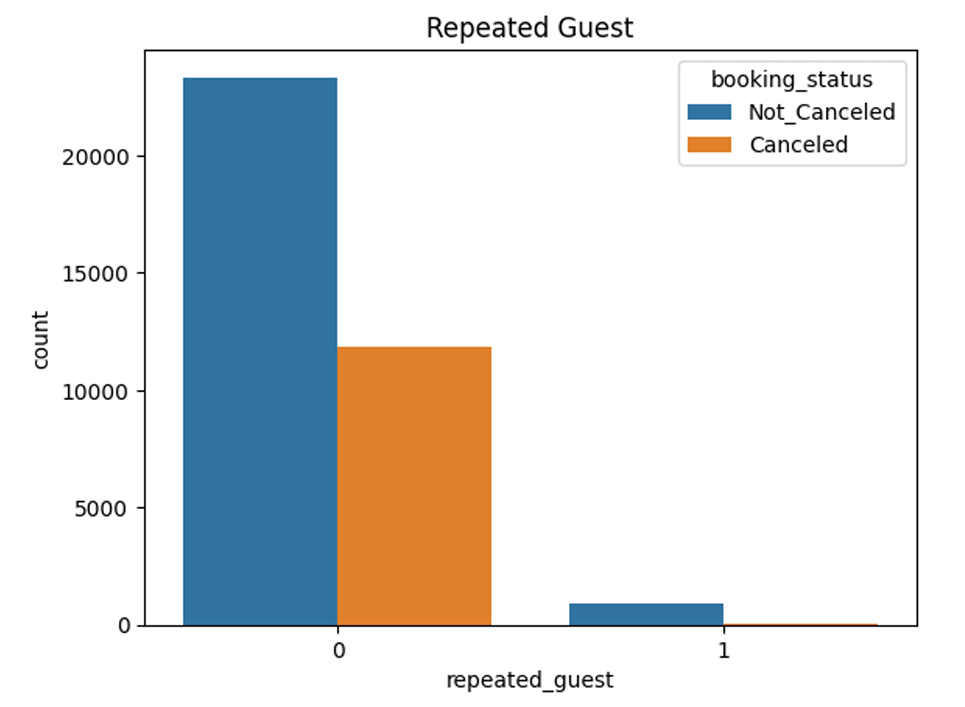
**LEAD TIME AND CANCELLATION :**



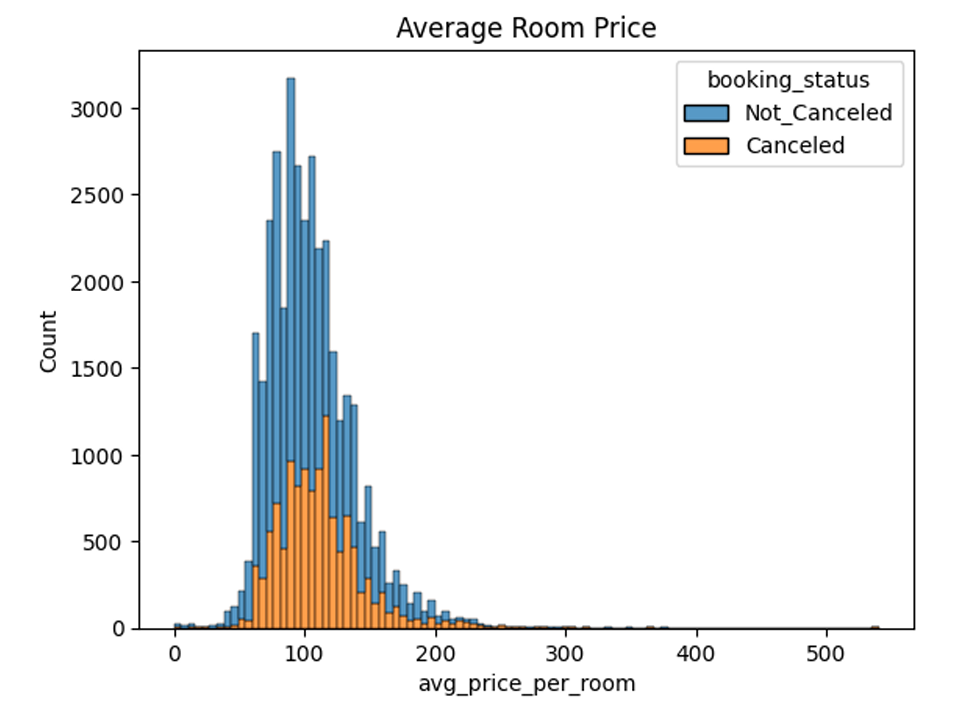
**MARKET SEGMENT AND CANCELLATION :**

****

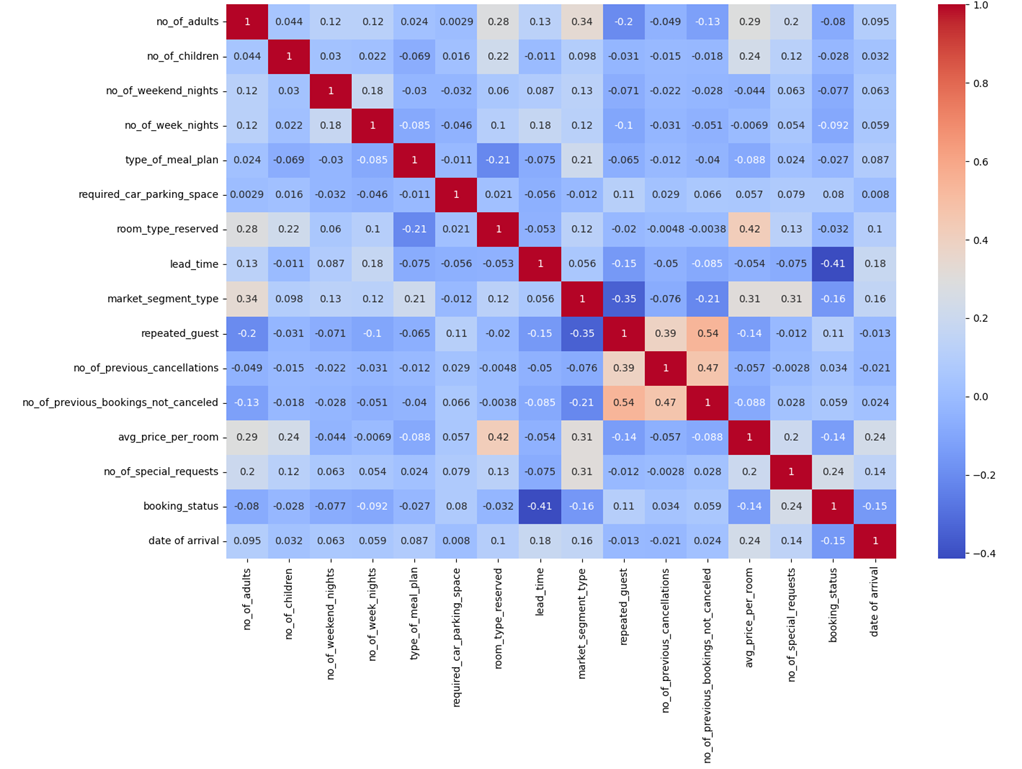
**GUEST'S PREVIOUS EXPERIENCE AND CANCELLATION :**

****

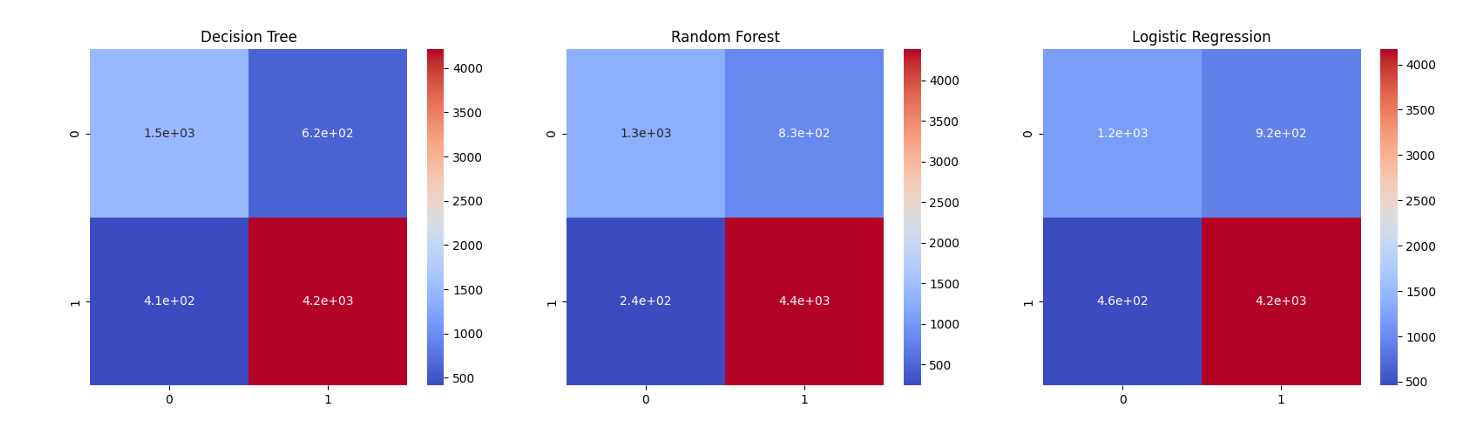
**AVERAGE ROOM PRICE AND CANCELLATION :**

****

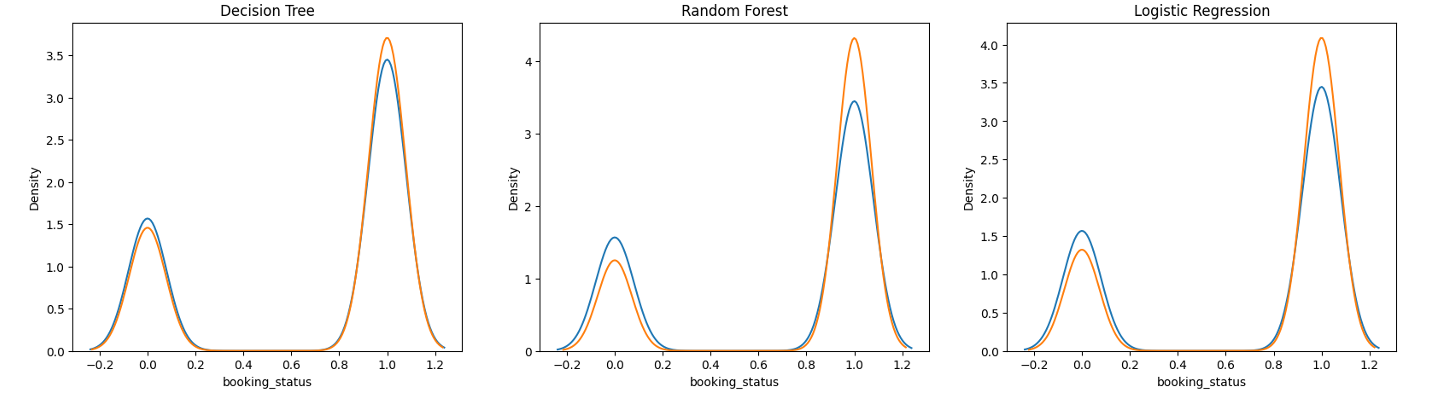
**CORRELATION MATRIX HEATMAP :**

****

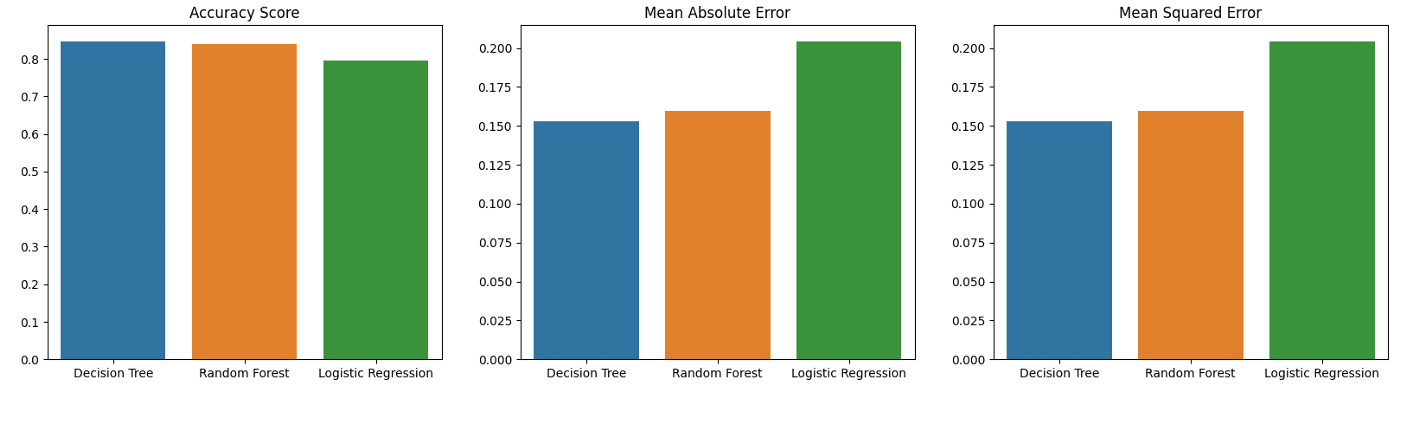
**CONFUSION MATRIX HEATMAP :**

****

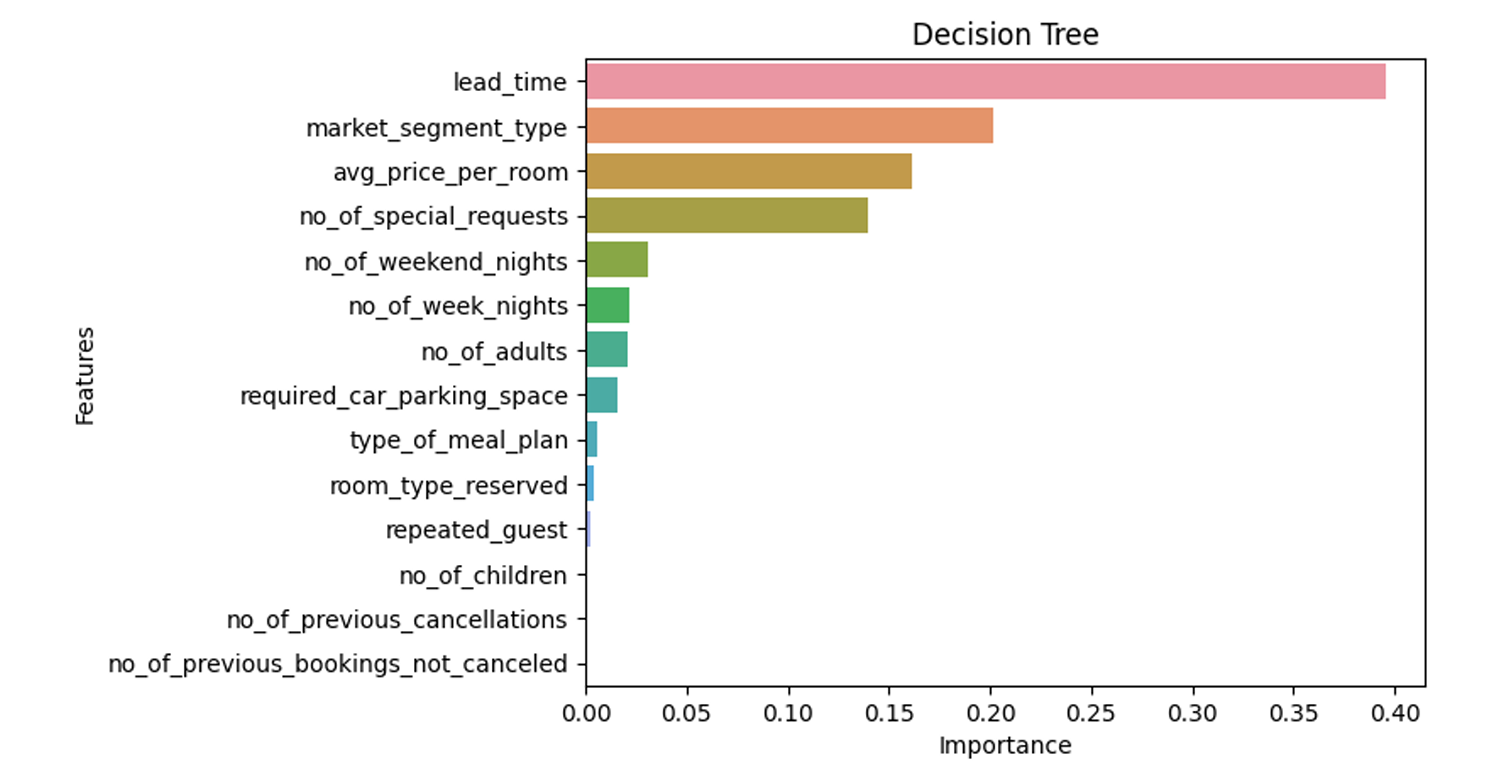
**DISTRIBUTION PLOT :**

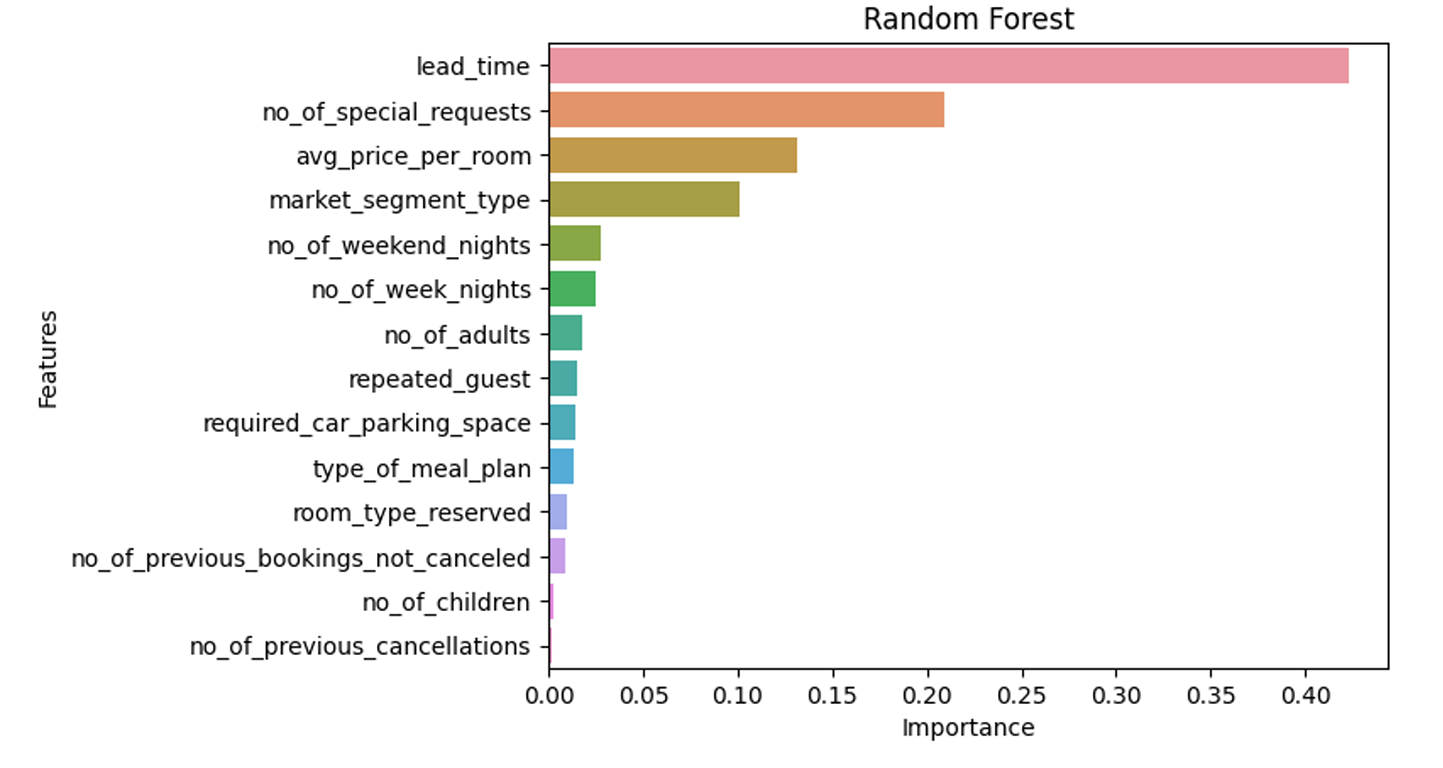
****

**MODEL COMPARISON :**



**FEATURE IMPORTANCE :**





**REFERENCES**

1. Yaqi Lin, "Research on the Influencing Factors of Cancellation of Hotel", July 2023.
2. Aditya Dole, "An Analysis of Hotel Booking Cancellations and Factors Affecting Revenue Generation" , January 2023.
3. Zharfan Akbar Andriawan, "Prediction of Hotel Booking Cancellation using CRISP-DM " , November 2020.
4. Shuixia Chen and Yaoyao Ku, "Prediction of hotel booking cancellations: Integration of machine learning and probability model based on interpretable feature interaction, July 2023.
5. Yopi Yuda Febrian, "Hotel Reservation Cancellation Prediction using Boosting Model", February 2024.
6. Abhishek Bhilare , Abhayjeet Singh , Nitin Jutlor and Payel Thakur, "Implementation of hotel booking cancellation using machine learning algorithms", April 2022.
7. Yukyeong Choi and Jeong-Yoo Kim, "A signaling theory of reservation cancellation policies" , January 2024.
8. Anita Herrera, "Forecasting hotel cancellations through machine learning", April 2024
9. Martin Falk and Markku Vieru, "Modelling the cancellation behaviour of hotel guests", October 2018.
10. Nuno Antonio, Ana de Almeida and Luis Nunes, "Predicting hotel booking cancellations to decrease uncertainty and increase revenue", April 2017.