



Hotel Booking Cancellations Prediction

Supervised by Eng. Mahmoud Talaat



Our Team



**Farah Yehia Ismail
Awaad**

**Malak Mohamed
Abdelmoniem**

**Mariam Mohamed
Ali El Nakeeb**

**Malak Khaled
Abdelaziz Hamed**

**Jana Mostafa
Mohamed Sabry**

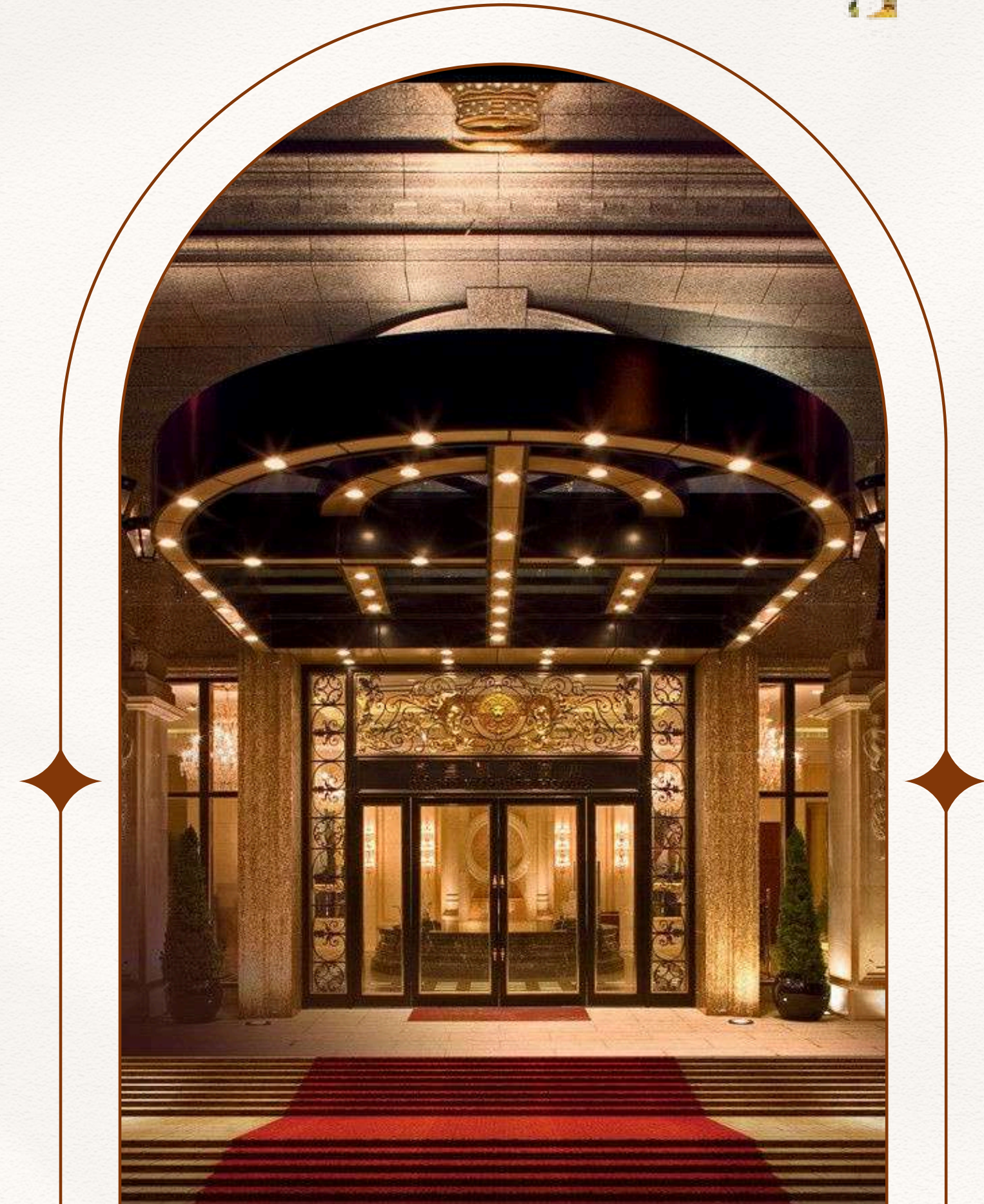
**Youssef Mohamed
Nabil**





Table of Contents

- Introduction
- Problem Statement
- Proposed Solution
- Methodology
- Data Analysis & Visualization
- Data Cleaning & Pre-Processing
- Machine Learning Results
- Conclusion



Introduction

The hospitality industry has undergone a digital transformation, with online booking platforms generating vast amounts of reservation data daily. However, hotel booking cancellations remain a critical challenge that significantly impacts:

- Revenue forecasting and financial planning
- Staff scheduling and resource allocation
- Room inventory management and occupancy optimization

Unexpected cancellations create serious operational challenges:

- Empty rooms and wasted resources
- Revenue losses and reduced profitability
- Inefficient resource planning across departments



Problem Statement

This project aims to build a **predictive model** to determine whether a hotel booking will be canceled—a crucial capability for effective revenue management and operational planning.

The dataset contains diverse booking-related features including lead time, deposit type, special requests, and customer demographics. The complexity lies in the comprehensive data preprocessing required: handling missing values, feature selection and engineering, and managing noise in the data.

Our approach involves training multiple machine learning models, evaluating their performance using appropriate metrics, and interpreting results by identifying the key features that drive cancellation behavior. By leveraging predictive analytics, hotels can make **informed, data-driven decisions** to minimize losses and optimize operations.



Proposed Solution

To address the critical challenge of hotel booking cancellations, we propose an **AI-powered predictive system** that enables hotels to proactively manage cancellations and optimize operations.

Smart Cancellation Risk Assessment

The system analyzes booking data in real-time to predict cancellation likelihood before it happens. By evaluating key indicators such as lead time, deposit type, customer history, and booking patterns, hotels can identify high-risk reservations early and take preventive action.

AI-Driven Decision Support

Our machine learning model provides hotels with actionable intelligence to:

- **Optimize revenue management:** Adjust pricing strategies and overbooking policies based on predicted cancellation rates
- **Enhance resource planning:** Allocate staff and resources efficiently by anticipating actual occupancy
- **Improve customer retention:** Identify at-risk bookings for targeted communication and incentives

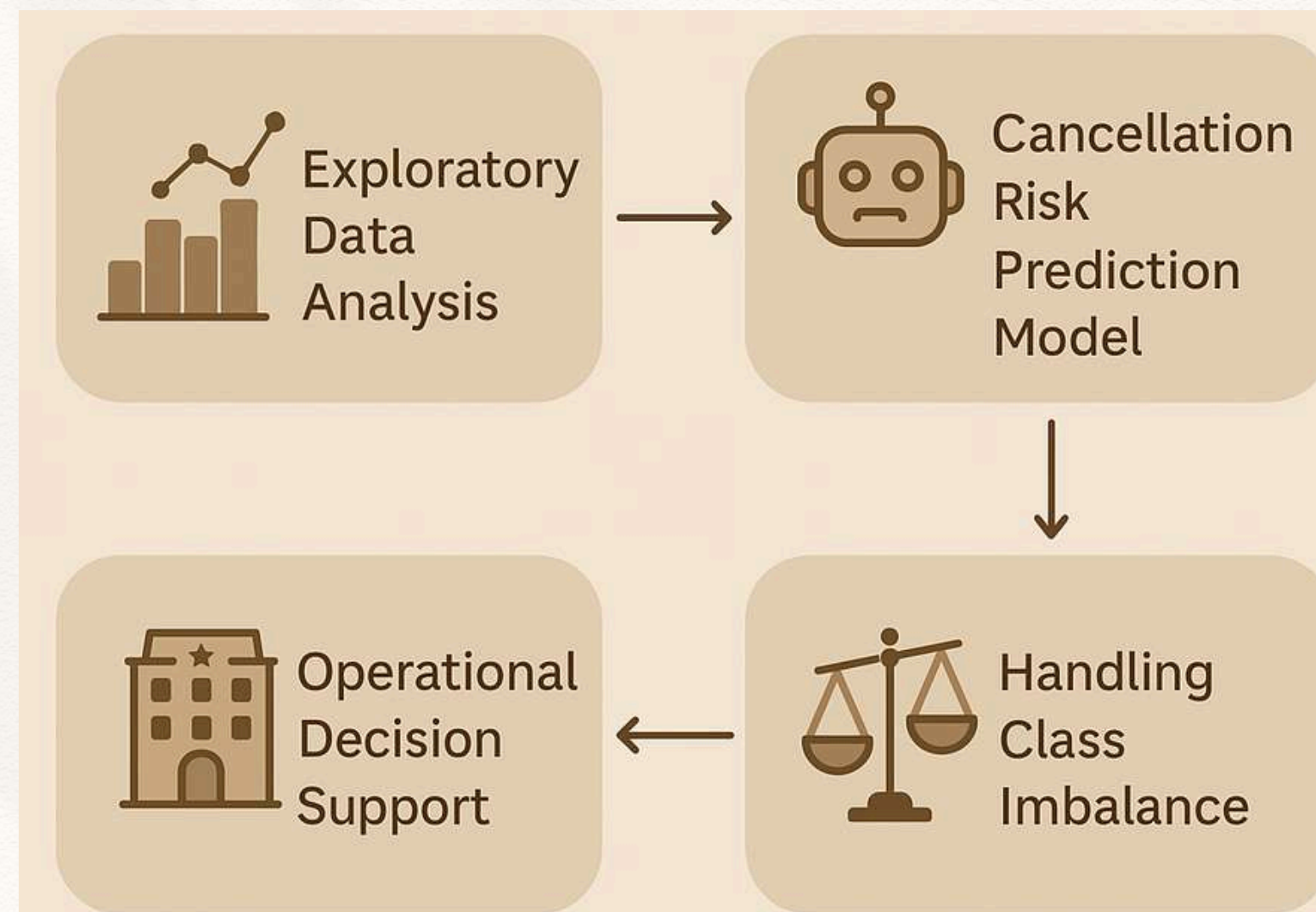


Proposed Solution

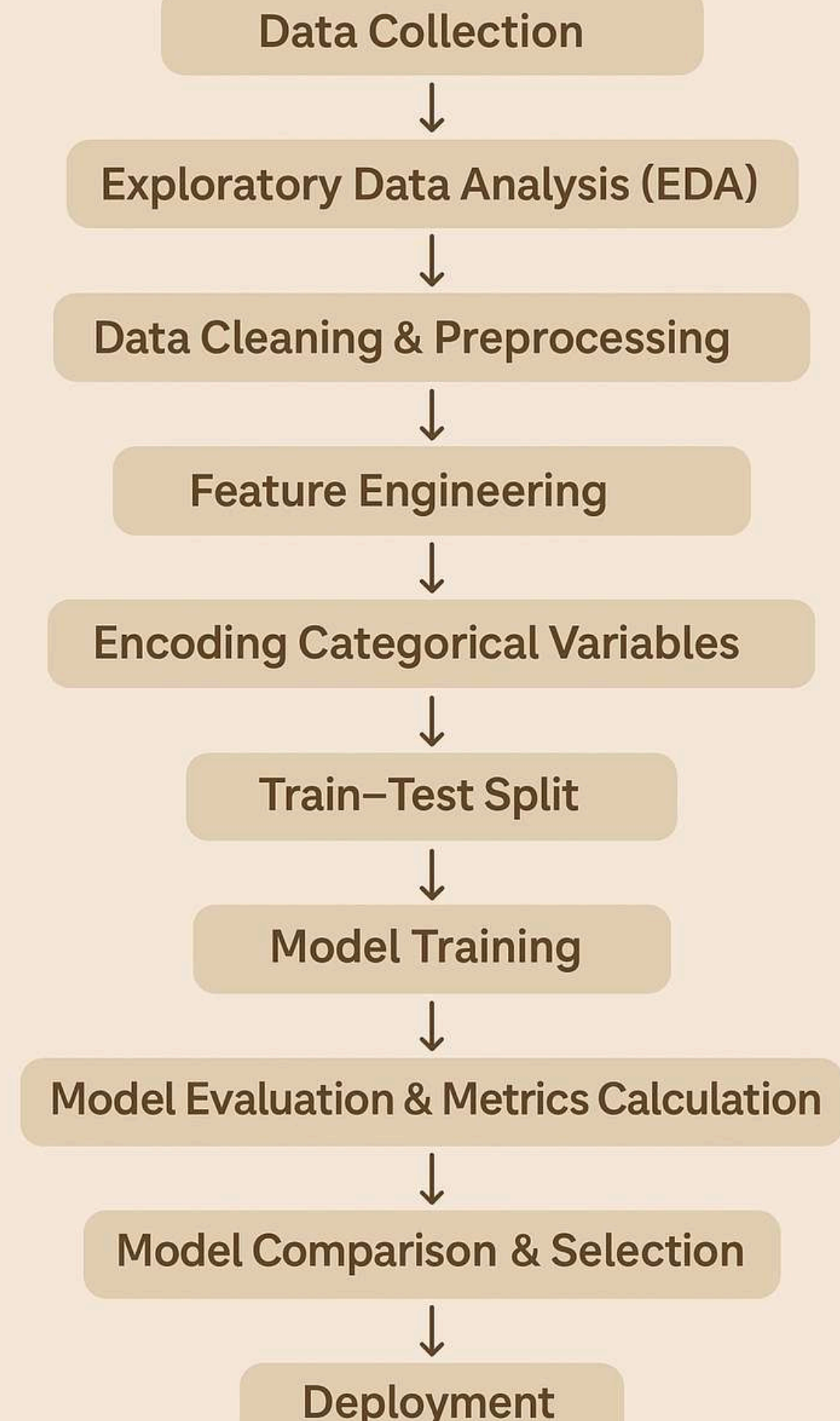
Operational Benefits

This solution empowers hotel management to:

- **Minimize revenue losses** from last-minute cancellations
- **Maximize room occupancy** through data-driven overbooking strategies
- **Reduce operational costs** by aligning staffing with accurate demand forecasts
- **Strengthen customer relationships** through proactive engagement with guests likely to cancel



Methodology



Data Collection

Collected a comprehensive hotel booking dataset containing 36 features describing booking characteristics, customer profiles, and cancellation status

Source:

<https://www.kaggle.com/datasets/arezaei81/hotel-bookingscv>

Booking Characteristics:

- **lead_time:** Days between booking and arrival (longer lead times → higher cancellation risk)
- **deposit_type:** Deposit status (non-refundable deposits → lower cancellations)
- **booking_changes:** Number of modifications (frequent changes → unstable bookings)
- **adr:** Average daily rate (pricing impact on cancellation behavior)

Customer Profile:

- **customer_type:** Transient, group, or repeat customer
- **previous_cancellations:** Past cancellation history
- **country:** Customer origin and travel patterns
- **adults, children, babies:** Guest composition

Stay Details:

- **hotel_type:** City vs. resort hotel
- **arrival_date (year/month/day):** Seasonal trends
- **stays_in_weekend_nights / stays_in_week_nights:** Stay duration
- **reserved_room_type / assigned_room_type:** Room allocation (mismatches may increase cancellations)

Additional Indicators:

- **meal:** Meal plan type
- **total_of_special_requests:** Guest commitment level
- **required_car_parking_spaces:** Service requirements

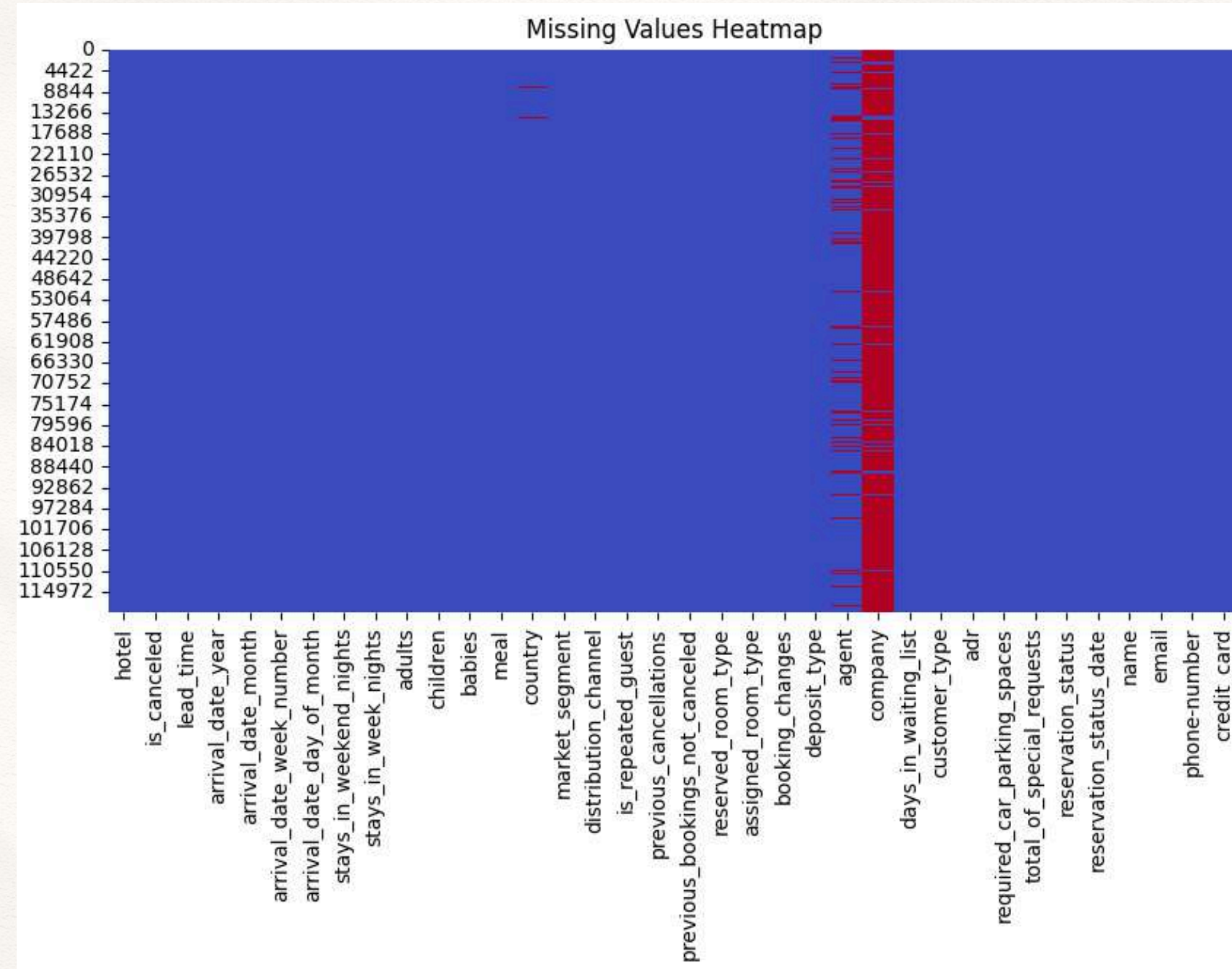
Target Variable:

- **is_canceled:** Cancellation status (0 = Not Cancelled, 1 = Cancelled)

Exploratory Data Analysis

Missing Values Detection:

The heatmap shows the distribution of missing values across the dataset. Missing values are highlighted in a different color to make it easier to identify which columns have incomplete data.



Exploratory Data Analysis

Class Imbalance:

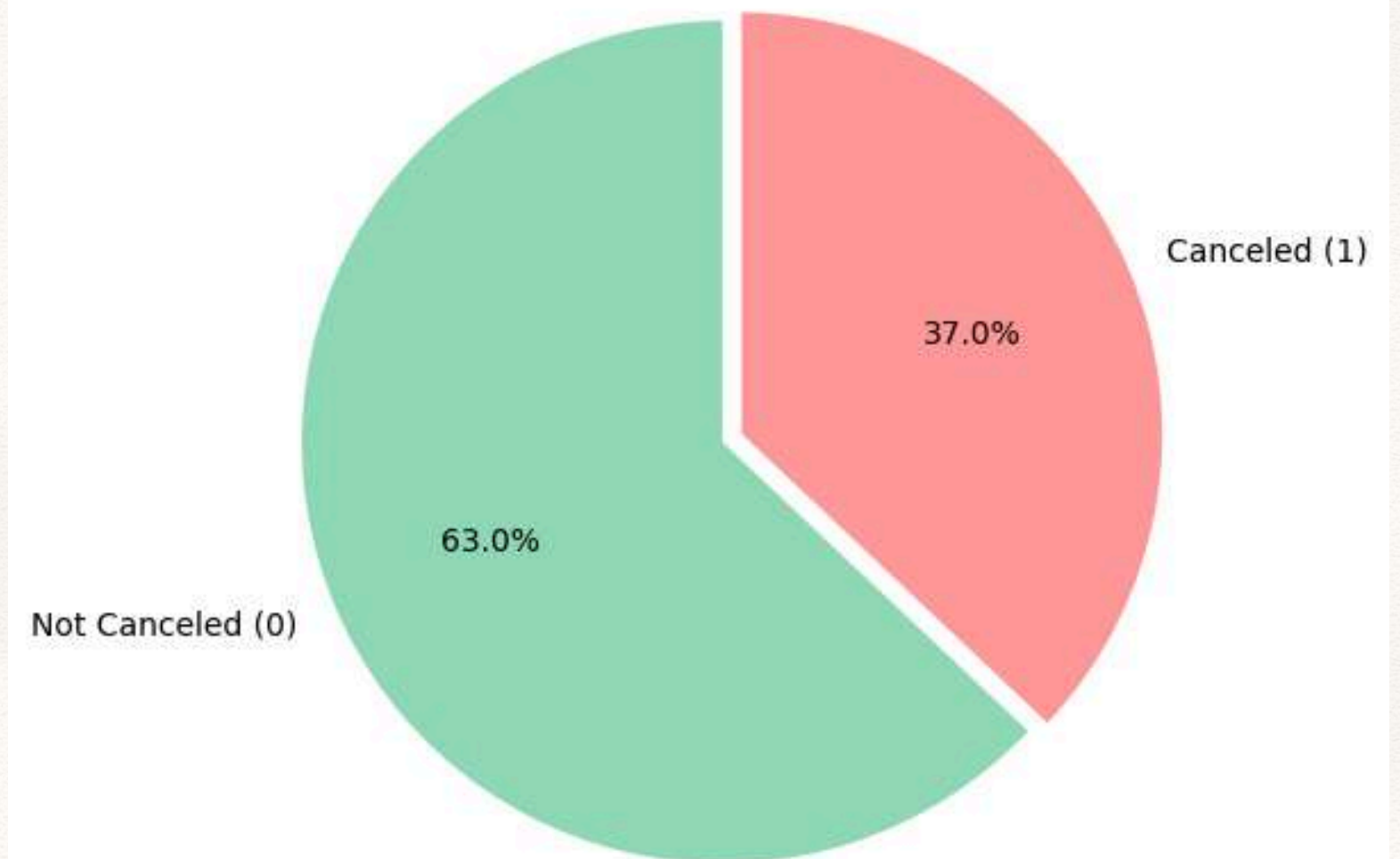
In this section, we check the distribution of the target variable **is_canceled** to see how many bookings were canceled (1) and how many were not canceled (0).

This helps us understand whether the dataset is balanced or imbalanced, which is important for model training.

```
df['is_canceled'].value_counts(normalize=True) * 100
```

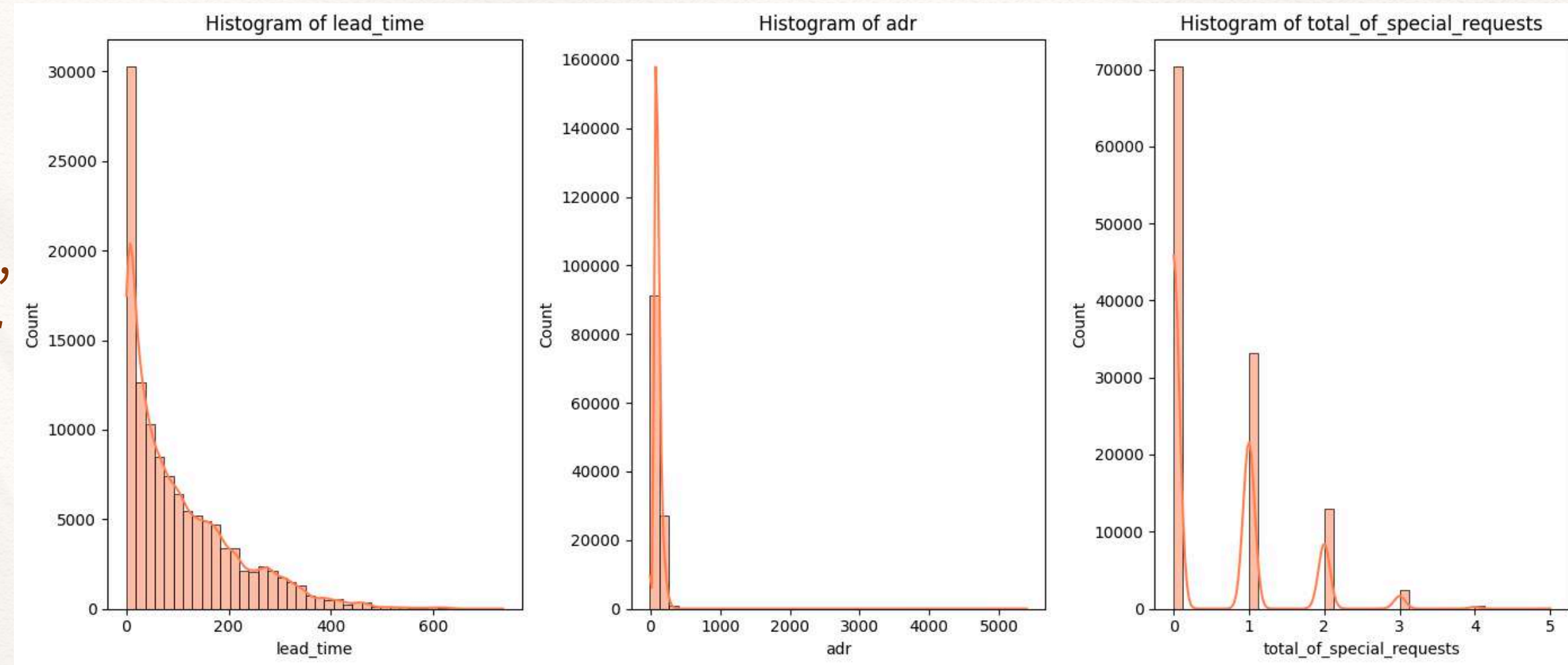
proportion	
is_canceled	
0	62.958372
1	37.041628

Booking Cancellation Distribution



Histograms for Distribution

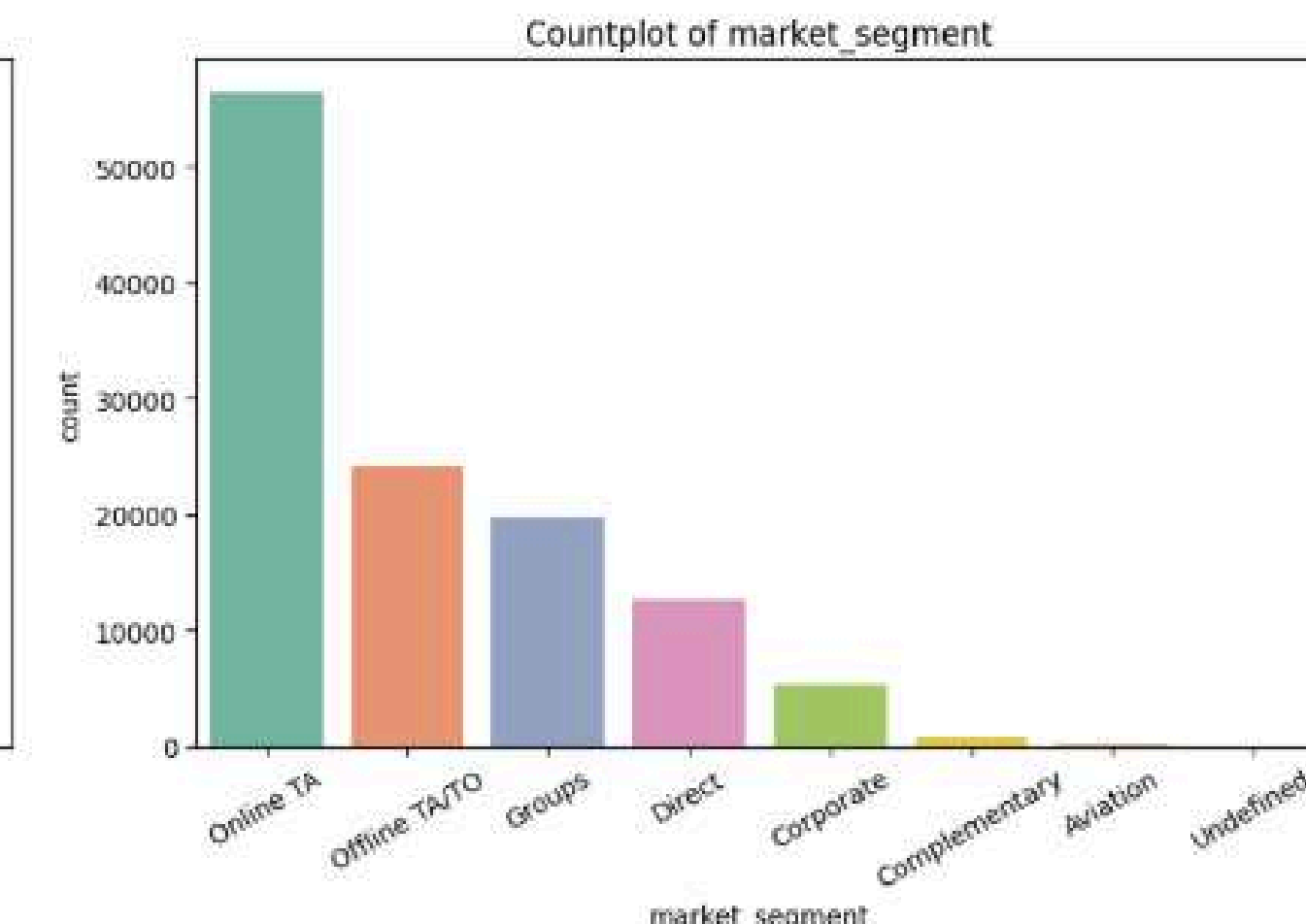
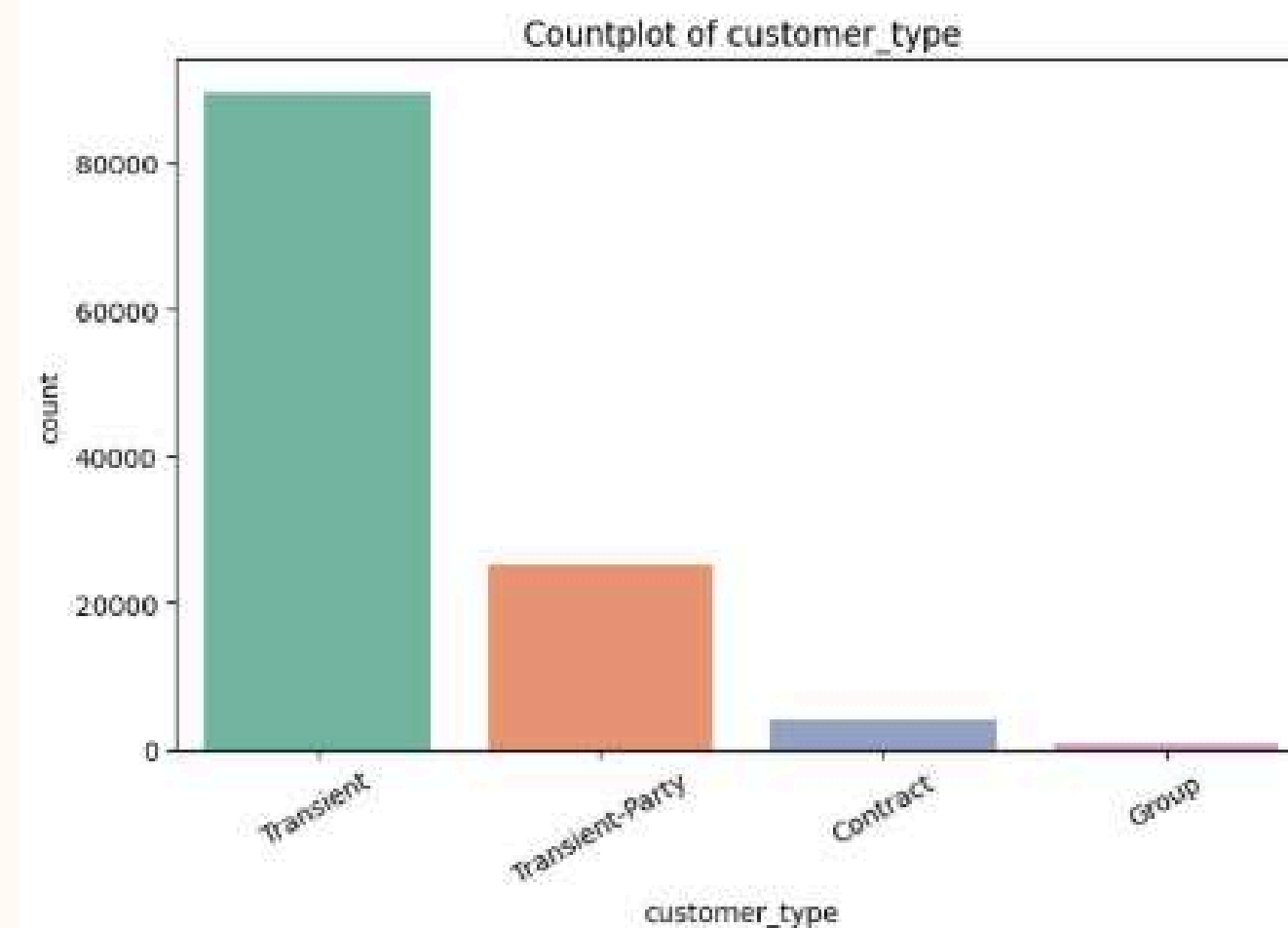
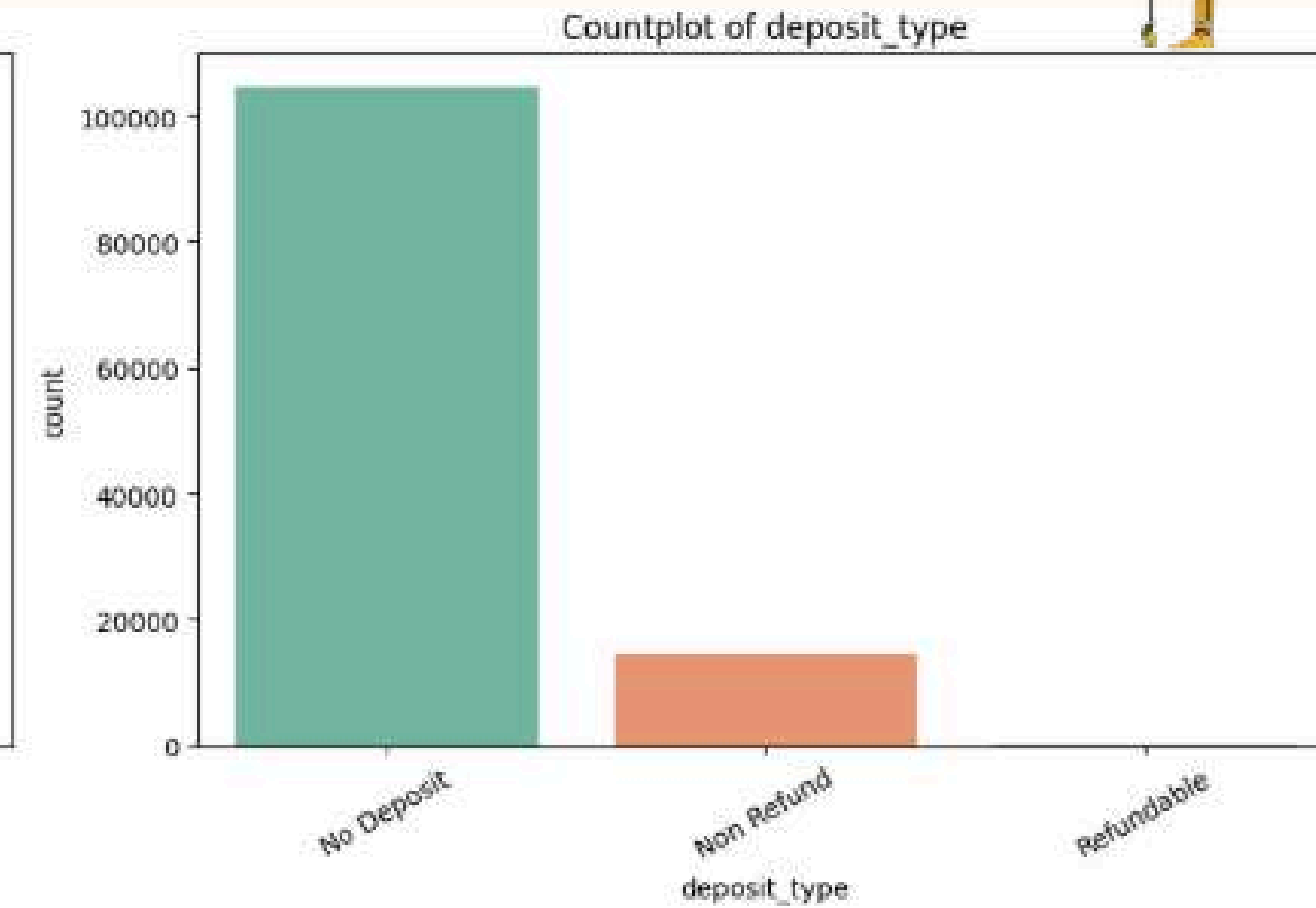
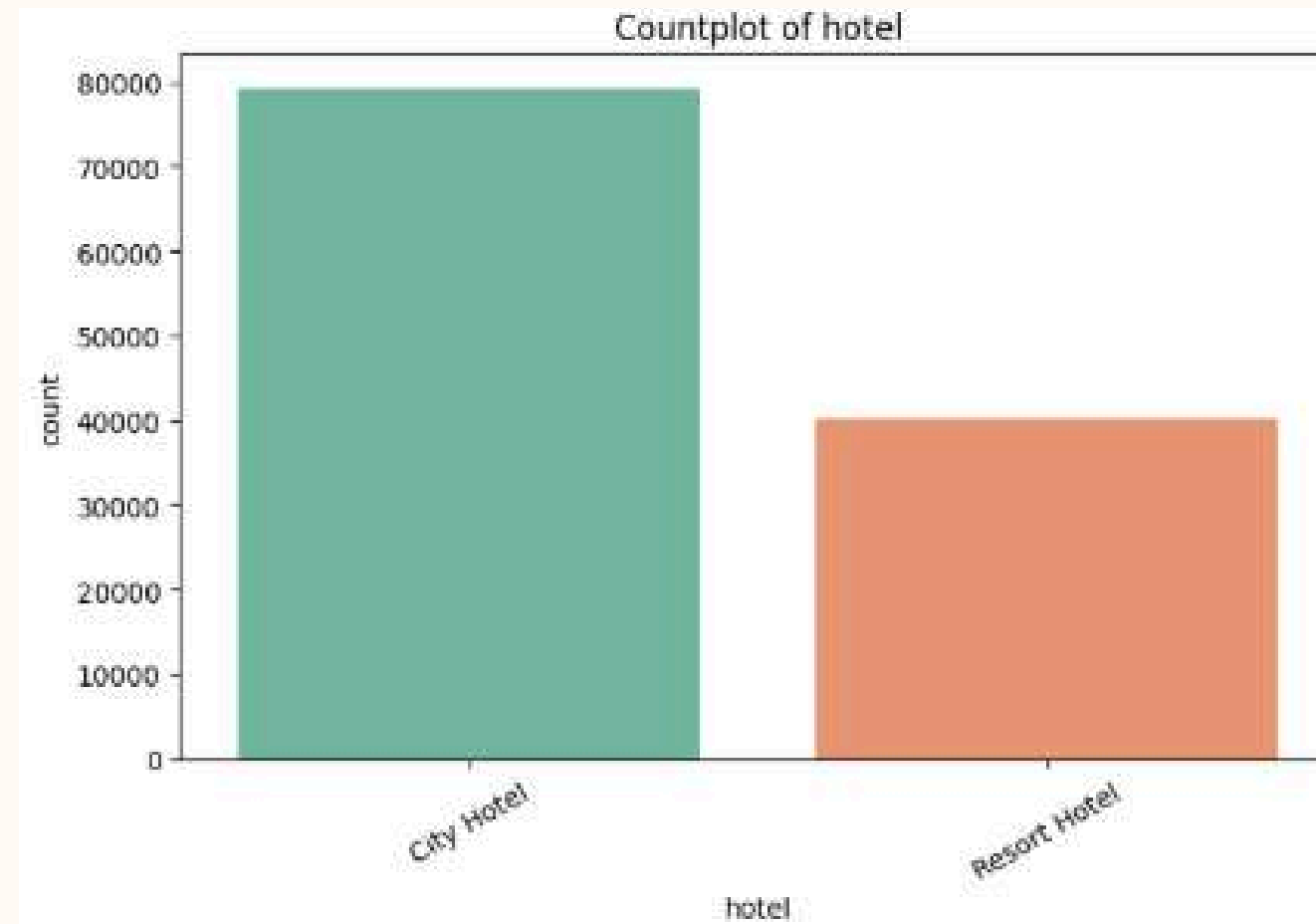
The Histograms above show that **most bookings** have **short lead times**, **ADR** values are skewed toward **lower prices**, and most customers make only **a few special requests**.



Exploratory Data Analysis

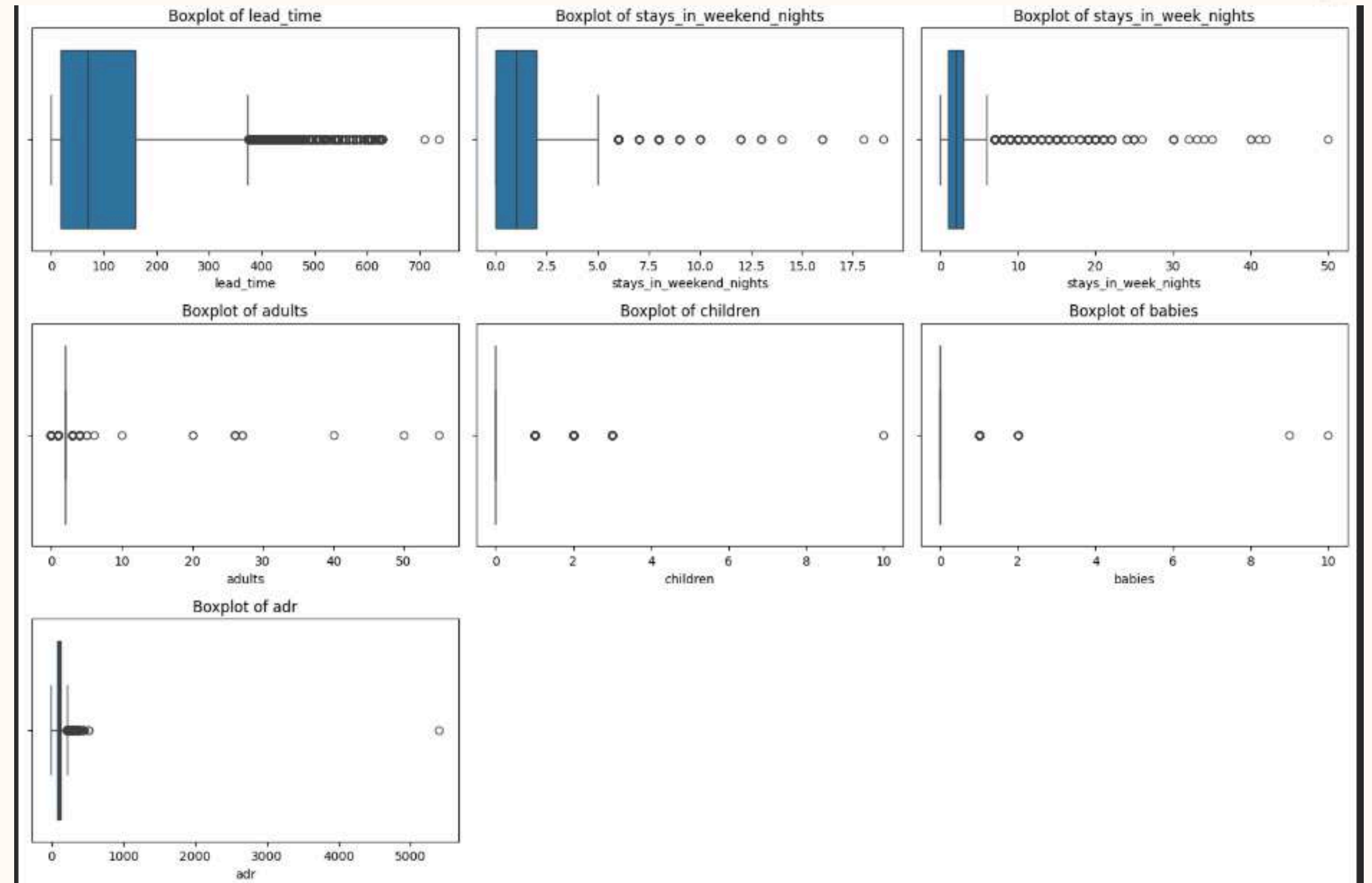


- The **Countplots** indicate that most bookings are from **City Hotels**, most customers pay with **No Deposit**, and the majority are Transient guests. **Online TA** is the dominant market segment.



Outlier Detection:

From the boxplots, we can clearly see extreme values in **adr** and **lead_time**, as well as discrete jumps in **adults** and **children**. These plots visually confirm the findings of the **IQR method**.



[illegible]

1. Handling Missing Values:

After handling missing values, all columns now show **0 missing entries**.
This confirms that the dataset is clean from null values

```
# 1) Company & Agent → missing means no company/agent, so we replace with 0
df['company'] = df['company'].fillna(0)
df['agent'] = df['agent'].fillna(0)

# 2) Country → replace missing with the mode (most frequent country)
df['country'] = df['country'].fillna(df['country'].mode()[0])

# 3) Children → very few missing values, so we use median
df['children'] = df['children'].fillna(df['children'].median())
```

2. Remove Duplicates:

Before cleaning: 8,031 duplicate rows were found in the dataset.
After cleaning: 0 duplicate rows remain.

All duplicates have been successfully removed, ensuring dataset uniqueness.

```
# Check number of duplicate rows
duplicates_count = df.duplicated().sum()
duplicates_count

# Drop duplicate rows
df = df.drop_duplicates()

# Re-check after dropping
new_duplicates_count = df.duplicated().sum()
new_duplicates_count
```

Removing Directly Related & Irrelevant Features:

We drop columns that either:
Directly reveal the target variable
(is_canceled) → data leakage risk.

Contain sensitive or irrelevant information
**(name, email, phone-number,
credit_card).**

This ensures the dataset only includes
meaningful, non-leaking features for
modeling

```
# Drop directly related & irrelevant columns
df = df.drop([
    'reservation_status',
    'reservation_status_date',
    'name',
    'email',
    'phone-number',
    'credit_card'
], axis=1)
```

Feature Engineering:

We create new meaningful features to enrich the dataset:

Total Guests: Sum of adults, children, and babies in the booking.

Total Nights: Sum of weekend nights and week nights.

Is Family: Binary flag (**1** if children or babies are included, **0** otherwise).

Stay Duration Category: Categorizes the total stay as 'Short', 'Medium', or 'Long'.

Season: Identifies the season of arrival (Winter, Spring, Summer, Autumn).

```
# 1. Total Guests
df["total_guests"] = df["adults"] + df["children"].fillna(0) + df["babies"].fillna(0)

# 2. Total Nights
df["total_nights"] = df["stays_in_weekend_nights"] + df["stays_in_week_nights"]

# 3. Is Family (1 if booking has children or babies, else 0)
df["is_family"] = ((df["children"].fillna(0) + df["babies"].fillna(0)) > 0).astype(int)

# 4. Stay Duration Category
df["stay_duration_category"] = pd.cut(
    df["total_nights"],
    bins=[0, 2, 7, 100],
    labels=["Short", "Medium", "Long"],
    right=False
)

# 5. Season based on arrival month
season_map = {
    "December": "Winter", "January": "Winter", "February": "Winter",
    "March": "Spring", "April": "Spring", "May": "Spring",
    "June": "Summer", "July": "Summer", "August": "Summer",
    "September": "Autumn", "October": "Autumn", "November": "Autumn"
}
df["season"] = df["arrival_date_month"].map(season_map)
```

Feature Engineering:

We create new meaningful features to enrich the dataset:

Total Guests: Sum of adults, children, and babies in the booking.

Total Nights: Sum of weekend nights and week nights.

Is Family: Binary flag (1 if children or babies are included, 0 otherwise).

Stay Duration Category: Categorizes the total stay as 'Short', 'Medium', or 'Long'.

Season: Identifies the season of arrival (Winter, Spring, Summer, Autumn).

```
# 1. Total Guests
df["total_guests"] = df["adults"] + df["children"].fillna(0) + df["babies"].fillna(0)

# 2. Total Nights
df["total_nights"] = df["stays_in_weekend_nights"] + df["stays_in_week_nights"]

# 3. Is Family (1 if booking has children or babies, else 0)
df["is_family"] = ((df["children"].fillna(0) + df["babies"].fillna(0)) > 0).astype(int)

# 4. Stay Duration Category
df["stay_duration_category"] = pd.cut(
    df["total_nights"],
    bins=[0, 2, 7, 100],
    labels=["Short", "Medium", "Long"],
    right=False
)

# 5. Season based on arrival month
season_map = {
    "December": "Winter", "January": "Winter", "February": "Winter",
    "March": "Spring", "April": "Spring", "May": "Spring",
    "June": "Summer", "July": "Summer", "August": "Summer",
    "September": "Autumn", "October": "Autumn", "November": "Autumn"
}
df["season"] = df["arrival_date_month"].map(season_map)
```

Data Preprocessing

Splitting The Data:

```
# Define the features (X) and the output labels (y)
X = df.drop('is_canceled', axis=1)
y = df['is_canceled']

# Splitting data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0, stratify=y)
```

- We separate the dataset into features (**X**) and the target label (**y**).
- **X** contains all input variables except the **is_canceled** column.
- **y** contains only the **is_canceled** column, which is what we want the model to predict.
- Then we split the data into training and testing sets using **train_test_split**.
- We use **stratify=y** to keep the same class distribution in both sets, ensuring **fair training**.

Baseline Model – Random Forest Classifier:

The model achieved **89%** accuracy on unseen test data, which is a good baseline result. Training performance was perfect (**100%**), suggesting slight **overfitting**. **Precision (0.89)** and **Recall (0.81)** show the model is able to identify most of the canceled bookings while maintaining good precision.

Classification report for test set					
	precision	recall	f1-score	support	
0	0.89	0.94	0.92	15033	
1	0.89	0.81	0.85	8845	
accuracy			0.89	23878	
macro avg	0.89	0.88	0.88	23878	
weighted avg	0.89	0.89	0.89	23878	

Dataset	Accuracy	Precision	Recall	F1-score	Observation
Training Set	1.00	1.00	1.00	1.00	Model performed perfectly on training data, indicating potential overfitting.
Test Set	0.89	0.89	0.81	0.85	Strong generalization performance with minor signs of overfitting.

Models Training

- Logistic Regression
- Random Forest
- XGBoost
- LightGBM

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC (%)
Random Forest	86.05	80.74	81.88	81.3	93.69
XGBoost	85.84	78.22	85.61	81.75	94.32
LightGBM	85.64	77.8	85.66	81.54	94.26
Logistic Regression	80.99	73.05	77.13	75.03	89.42



Model Comparison Summary (Random Forest vs XGBoost vs LGBM vs Logistic Regression)

✓ **Final Model Selection: XGBoost (Original Data)**

After evaluating multiple machine learning models under different sampling strategies, **XGBoost** trained on the **original (unbalanced)** dataset was selected as **the final and optimal model for deployment**.

COMPLETE RESULTS COMPARISON						
Sampling	Model	Accuracy	Precision	Recall	F1-Score	AUC
Original	Random Forest	86.05	80.74	81.88	81.30	93.69
Original	XGBoost	85.84	78.22	85.61	81.75	94.32
Original	LightGBM	85.64	77.80	85.66	81.54	94.26
Original	Logistic Regression	80.99	73.05	77.13	75.03	89.42
SMOTE	Random Forest	86.32	81.61	81.40	81.51	93.62
SMOTE	XGBoost	83.29	71.81	90.34	80.02	93.96
SMOTE	LightGBM	83.57	72.12	90.73	80.36	94.04
SMOTE	Logistic Regression	81.10	74.82	73.80	74.31	89.26
Undersampling	Random Forest	86.09	80.45	82.49	81.46	93.80
Undersampling	XGBoost	82.48	69.87	92.66	79.67	94.27
Undersampling	LightGBM	82.13	69.39	92.61	79.33	94.20
Undersampling	Logistic Regression	81.07	73.18	77.16	75.12	89.41
SMOTE+Tomek	Random Forest	86.07	81.25	81.11	81.18	93.49
SMOTE+Tomek	XGBoost	83.53	72.39	89.80	80.16	94.00
SMOTE+Tomek	LightGBM	83.52	72.17	90.34	80.24	93.93
SMOTE+Tomek	Logistic Regression	81.12	74.85	73.83	74.34	89.28



Hyperparameter Tunning (XGboost):

After selecting **XGBoost** as the best-performing model, hyperparameter tuning was applied to further improve accuracy and overall performance.

We defined a range of values for each hyperparameter to find the best combination for our XGBoost model:

```
# Define parameter grid
param_distributions = {
    'n_estimators': [100, 200, 300],
    'max_depth': [4, 6, 8],
    'learning_rate': [0.01, 0.05, 0.1],
    'subsample': [0.7, 0.8, 0.9],
    'colsample_bytree': [0.7, 0.8, 0.9],
    'min_child_weight': [1, 3],
    'gamma': [0, 0.1, 0.2],
    'reg_alpha': [0, 0.01, 0.1],
    'reg_lambda': [1, 1.5, 2]
}
```



Hyperparameter Tunning (XGboost):

RandomizedSearchCV Setup:

RandomizedSearchCV tests 20 random combinations of hyperparameters and selects the one with the best F1 score

After running the **hyperparameter tuning process**, we can check which combination of parameters yielded the best performance based on our chosen metric (F1-Score).

✅ Hyperparameter tuning complete!

🏆 Best Parameters Found:

- subsample: 0.8
- reg_lambda: 1
- reg_alpha: 0.01
- n_estimators: 300
- min_child_weight: 1
- max_depth: 8
- learning_rate: 0.1
- gamma: 0
- colsample_bytree: 0.8

📊 Best CV F1-Score: 0.8419

```
random_search = RandomizedSearchCV(  
    estimator=xgb_base,  
    param_distributions=param_distributions,  
    n_iter=20,  
    scoring=f1_scorer,  
    cv=cv_strategy,  
    verbose=1,  
    random_state=42,  
    n_jobs=-1  
)  
  
random_search.fit(X_train, y_train)
```

Final Model Performance – Baseline vs Tuned(XGboost):

Classification Report Summary:

- **Not Canceled (0):** F1-Score = 0.902 → excellent detection
- **Canceled (1):** F1-Score = 0.843 → significant improvement after tuning

```
..
```

Metric	Baseline	Tuned	Improvement
Accuracy	85.84	87.90	2.40
Precision	78.22	81.23	3.85
Recall	85.61	87.55	2.27
F1-Score	81.75	84.27	3.09
AUC	94.32	95.63	1.39

	precision	recall	f1-score	support
Not Canceled (0)	0.9232	0.8810	0.9016	15033
Canceled (1)	0.8123	0.8755	0.8427	8845
accuracy			0.8790	23878
macro avg	0.8678	0.8783	0.8722	23878
weighted avg	0.8822	0.8790	0.8798	23878

Interpretation: The **tuned XGBoost** model performs better in identifying canceled bookings while maintaining high accuracy for non-canceled cases.



Deployment

The input form for the Staff:

Hotel staff can enter customer and booking details into the input form to predict whether a guest is likely to cancel their reservation

Hotel Booking Cancellation Prediction

Predict cancellation risk using Machine Learning

Guest Information

Adults *	Children	Babies
<input type="text" value="2"/>	<input type="text" value="5"/>	<input type="text" value="0"/>
Country	Repeated Guest?	
<input type="text" value="United Kingdom (GBR)"/>	<input type="text" value="Yes"/>	

Hotel & Room Details

Hotel Type *	Meal Plan *	Room Type *
<input type="text" value="Resort Hotel"/>	<input type="text" value="Half Board (HB)"/>	<input type="text" value="Type D"/>

Booking Details

Lead Time (days) *	Arrival Month *	Arrival Year
<input type="text" value="30"/>	<input type="text" value="August"/>	<input type="text" value="2024"/>
Days between booking and arrival		
Weekend Nights *	Week Nights *	Booking Changes
<input type="text" value="1"/>	<input type="text" value="2"/>	<input type="text" value="0"/>

Market & Distribution

Market Segment *	Distribution Channel *	Customer Type *
<input type="text" value="Direct"/>	<input type="text" value="Direct"/>	<input type="text" value="Transient"/>

Guest History

Previous Cancellations	Previous Bookings (Not Canceled)
<input type="text" value="0"/>	<input type="text" value="5"/>

Deployment

Based on the prediction, the system shows the likelihood of cancellation risk along with suggested actions to minimize revenue loss or manage room allocation:

The output of Prediction:

Financial & Special Requests

Average Daily Rate (€) *

150

Deposit Type *

Refundable

Parking Spaces

1

Special Requests

2

Days in Waiting List

0

 Predict Cancellation Risk




Low Cancellation Risk (Low)

99.86%

Booking Looks Good:

- ✓ Guest likely to honor booking
- ✓ Standard confirmation recommended
- ✓ Proceed with normal preparation

 Make Another Prediction

Conclusion

In this project, we built a machine learning model to predict hotel booking cancellations, helping hotels reduce losses and improve planning. After **exploring the data**, we identified **key factors** such as **lead time**, **ADR**, **booking changes**, and **customer type** that strongly influence cancellations.

To address **class imbalance**, we applied **SMOTE** and other **resampling methods**, which improved the model's ability to detect high-risk bookings. We tested **several algorithms**, and **after tuning**, **XGBoost** delivered **the best** overall performance, achieving high accuracy, recall, and AUC.

Overall, this project shows that **machine learning** can **provide** hotels with a **reliable decision-support tool**, enabling smarter overbooking strategies, better resource allocation, and more effective revenue management.





Thank You

Any Questions?

