HOTEL BOOKING CANCELLATION PREDICTION REPORT

Trendalytix - Team 5

Date of Submission: 30 June, 2025

S.no	Name	Role
1	Varshini Chilakala	Intern (TL)
2	Konda Sai Sreekar Reddy	Intern
3	Diti Solanki	Intern

Project Title: Predicting Hotel Booking Cancellations using Data Analytics and Machine Learning.

Objective: To identify patterns and build a predictive model to determine whether a hotel booking will be canceled, enabling hotel management to optimize operations, reduce overbooking, and improve customer satisfaction.

Dataset:

Source: Hotel Booking Demand Dataset

https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand
Size: 119,390 rows x 32 columns
Columns: Hotel type, lead time, booking dates, number of guests, special requests, cancellation status, etc.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 32 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   hotel                           119390 non-null  object
 1   is_canceled                     119390 non-null  int64
 2   lead_time                       119390 non-null  int64
 3   arrival_date_year               119390 non-null  int64
 4   arrival_date_month              119390 non-null  object
 5   arrival_date_week_number        119390 non-null  int64
 6   arrival_date_day_of_month       119390 non-null  int64
 7   stays_in_weekend_nights         119390 non-null  int64
 8   stays_in_week_nights            119390 non-null  int64
 9   adults                          119390 non-null  int64
 10  children                        119386 non-null  float64
 11  babies                          119390 non-null  int64
 12  meal                            119390 non-null  object
 13  country                         118902 non-null  object
 14  market_segment                  119390 non-null  object
 15  distribution_channel            119390 non-null  object
 16  is_repeated_guest               119390 non-null  int64
 17  previous_cancellations          119390 non-null  int64
 18  previous_bookings_not_canceled  119390 non-null  int64
 19  reserved_room_type              119390 non-null  object
 20  assigned_room_type              119390 non-null  object
 21  booking_changes                 119390 non-null  int64
 22  deposit_type                    119390 non-null  object
 23  agent                           103050 non-null  float64
 24  company                         6797 non-null    float64
 25  days_in_waiting_list            119390 non-null  int64
 26  customer_type                   119390 non-null  object
 27  adr                             119390 non-null  float64
 28  required_car_parking_spaces     119390 non-null  int64
 29  total_of_special_requests       119390 non-null  int64
 30  reservation_status              119390 non-null  object
 31  reservation_status_date         119390 non-null  object
dtypes: float64(4), int64(16), object(12)
memory usage: 29.1+ MB
```

Tools & Technologies Used:

- **Languages**: Python (Pandas, NumPy, Scikit-learn, XGBoost, Seaborn, Matplotlib)
- **Visualization**: Power BI
- **Notebook**: Jupyter Notebook, Google Colab
- **Version Control**: Git, GitHub

Process Followed:

1. Problem Understanding

Understanding the hotel operations problem where cancellations impact occupancy planning and revenue.

The problem is focused on analyzing hotel booking behavior using data to derive actionable insights.

The goal is to help the hotel enhance its business strategy, such as:

- Reducing cancellations
- Improving revenue
- Optimizing room availability and pricing
- Understanding customer preferences and behavior patterns

Data Understanding Requirements: Examine variables like:

- Booking dates
- Arrival dates
- Customer type (e.g., repeated guest, first-time)
- Number of adults/children
- Market segment (e.g., corporate, online travel agent)
- Room type reserved vs. assigned
- Special requests
- Deposit type
- Cancellation flag

Analysis Goals:

- Identify factors that lead to cancellations
- Understand the peak booking periods
- Determine which market segments generate the most revenue or cancellations

● Profile customer types (e.g., what type of customer books early, requests more services, etc.)

● Examine gaps between reserved room type and assigned room type

● Understand how lead time affects booking behavior and cancellations

Constraints:

● Data may be imbalanced, especially for cancellations vs. confirmed bookings

● Missing or inconsistent data (e.g., missing children count, mismatched room types)

Business constraints:

● Maximize revenue while reducing cancellations

● Improve customer satisfaction

● Dataset might not include all real-time influencing factors (e.g., events, weather)

2. Data Cleaning & Preprocessing

A comprehensive data preparation phase was carried out to ensure quality and reliability of the dataset:

- **Duplicate Records**: Identified and removed duplicate rows to eliminate data redundancy.

```
# removing duplicate values

print("Duplicates before:",df.duplicated().sum())

df.drop_duplicates(inplace=True)

print("Duplicates after:",df.duplicated().sum())
```

```
Duplicates before: 31994
Duplicates after: 0
```

- **Missing Values**: Addressed missing values using a thoughtful strategy:
  - `agent` and `company` columns had a high percentage of missing values, which were filled with **0** and a new feature (`is_company_known`) was created to retain that information.

```
[14]:
# agent --16340 missing -- fill with 0 for no agent

df['agent']=df['agent'].fillna(0).astype(int)
```

```
[15]:
# company -- 112593 missing
#creating a column (binary flag column) for missing company values so later maybe we can use

df['is_company_missing']=df['company'].isnull().astype(int)

print(df[['company','is_company_missing']].head(10))
```

```
   company  is_company_missing
0      NaN                   1
1      NaN                   1
2      NaN                   1
3      NaN                   1
4      NaN                   1
5      NaN                   1
6      NaN                   1
7      NaN                   1
8      NaN                   1
9      NaN                   1
```

  - `children` column missing values were filled with the **mode** (most frequent value).

```
#handle missing values

#children - 4 missing values -- filling with most commmon value

df['children']=df['children'].fillna(df['children'].mode()[0])
```

- `country` missing values were filled with **"Unknown"** to preserve row information.

```
#country--488 missing -- categorical value--fill with unknown value

df['country']=df['country'].fillna('Unknown')
```

- **Data Type Conversion**:
  - Converted columns like `children`, `reservation_status_date` to categorical/integer types where appropriate.

```
# fix data types (if needed)

#children is float64 -- changing to int

df['children']=df['children'].astype(int)
```

```
[21]:
# reservation_status_date from object to datetime

df['reservation_status_date']=pd.to_datetime(df['reservation_status_date'])
```

  - Transformed date-related components (`arrival_date_year`, `arrival_date_month`, `arrival_date_day_of_month`) into a unified **datetime column** called `arrival_date`.

```
#arrival date
if {'arrival_date_year', 'arrival_date_month', 'arrival_date_day_of_month'}.issubset(df.columns):
    df['arrival_date_month'] = df['arrival_date_month'].apply(lambda x: str(x))
    df['arrival_date'] = pd.to_datetime(
        df['arrival_date_year'].astype(str) + '-' +
        df['arrival_date_month'] + '-' +
        df['arrival_date_day_of_month'].astype(str),
        errors='coerce'
    )
```

- **Feature Engineering**:
  - `total_stay`: Sum of `stays_in_weekend_nights` and `stays_in_week_nights`
  - `total_guests`: Sum of `adults`, `children`, and `babies`
  - `is_family`: True if guests include children or babies
  - `is_repeated_customer`: Transformed from `is_repeated_guest`

```
#Feature engineering

#total stay duration
df['total_stay']=df['stays_in_weekend_nights']+df['stays_in_week_nights']

#total guests
df['total_guests']=df['adults']+df['children']+df['babies']

#is family booking?
df['is_family']=df['total_guests'].apply(lambda x:1 if x>1 else 0)

#is weekend stay
df['is_weekend_stay']=df['stays_in_weekend_nights'].apply(lambda x:1 if x>0 else 0)

#season feature
month_season_map={
    'November':'Winter','December':'Winter','January':'Winter','February':'Winter',
    'March':'Spring','April':'Spring',
    'May':'Summer','June':'Summer','July':'Rainy','August':'Rainy',
    'September':'Fall','October':'Fall',
}
df['season'] = df['arrival_date_month'].map(month_season_map)

#repeated customer
df['is_repeated_customer']=df['is_repeated_guest'].apply(lambda x:1 if x>0 else 0)
```

- **Outlier Handling**:
  - Visualized numerical columns using boxplots
  - Applied **capping (clipping)** technique to variables like `lead_time`, `adr`,

previous_cancellations, booking_changes, etc. to control outlier impact

```
#handling outliers

#numerical columns

num_cols=df.select_dtypes(include=['int64','float64']).columns
print("Numerical Columns:",num_cols.tolist())
```

```
Numerical Columns: ['is_canceled', 'lead_time', 'arrival_date_year', 'arrival_date_week_number', 'arriva
l_date_day_of_month', 'stays_in_weekend_nights', 'stays_in_week_nights', 'adults', 'babies', 'is_repeate
d_guest', 'previous_cancellations', 'previous_bookings_not_canceled', 'booking_changes', 'company', 'day
s_in_waiting_list', 'adr', 'required_car_parking_spaces', 'total_of_special_requests']
```

```
#finding outliers

for col in num_cols:
    Q1=df[col].quantile(0.25)
    Q3=df[col].quantile(0.75)
    IQR=Q3-Q1
    lower_bound=Q1-1.5*IQR
    upper_bound=Q3+1.5*IQR
    outlier_count=df[(df[col]<lower_bound)|(df[col]>upper_bound)].shape[0]
    print(f"{col}:{outlier_count} outliers")
```

```
is_canceled:0 outliers
lead_time:2396 outliers
arrival_date_year:0 outliers
arrival_date_week_number:0 outliers
arrival_date_day_of_month:0 outliers
stays_in_weekend_nights:220 outliers
stays_in_week_nights:1531 outliers
adults:22899 outliers
babies:914 outliers
is_repeated_guest:3415 outliers
previous_cancellations:1685 outliers
previous_bookings_not_canceled:3545 outliers
booking_changes:15902 outliers
company:0 outliers
days_in_waiting_list:860 outliers
adr:2490 outliers
required_car_parking_spaces:7313 outliers
total_of_special_requests:2673 outliers
```

[28]:
```python
#remove ouliers

def remove_outliers_iqr(df,col):
    Q1=df[col].quantile(0.25)
    Q3=df[col].quantile(0.75)
    IQR=Q3-Q1
    lower=Q1-1.5*IQR
    upper=Q3+1.5*IQR
    return df[(df[col]>=lower) & (df[col]<=upper)]

cols_to_remove=['adr','lead_time','stays_in_week_nights','stays_in_weekend_nights','babies','required_car

for cols in cols_to_remove:
    df=remove_outliers_iqr(df,col)

#cap outliers

#cap adults at 4(rare to have >4 in one room)
df['adults']=df['adults'].clip(upper=4)

#cap booking_changes at 5
df['booking_changes']=df['booking_changes'].clip(upper=5)

# cap special requests at 5
df['total_of_special_requests']=df['total_of_special_requests'].clip(upper=5)
```

- **Date Feature Integration**:
  - Created `arrival_date` using combined year, month, and day values
  - Dropped the original separate date columns after transformation

This robust cleaning ensured consistency and optimized data quality for modeling.

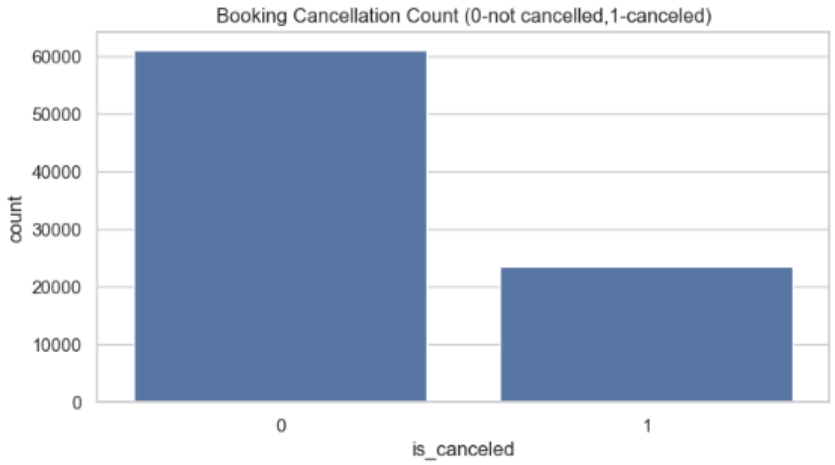
3. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was conducted in two major phases — **Univariate Analysis** and **Bivariate Analysis** — to understand the underlying distribution, relationships, and patterns that influence hotel booking cancellations.

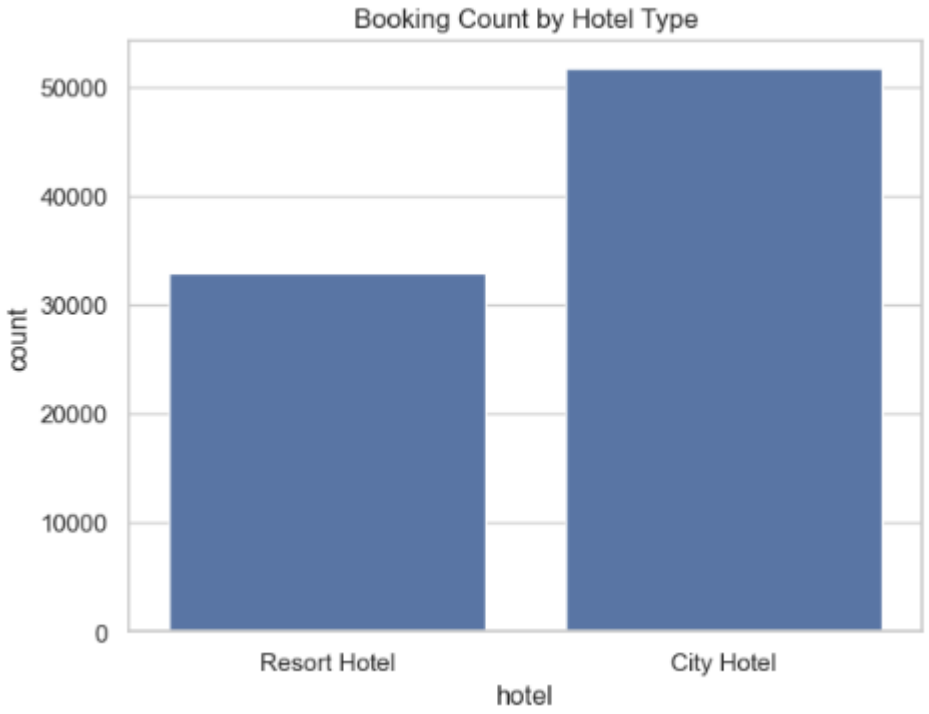
Univariate Analysis

Focused on analyzing individual features to understand their distributions and categorical counts:

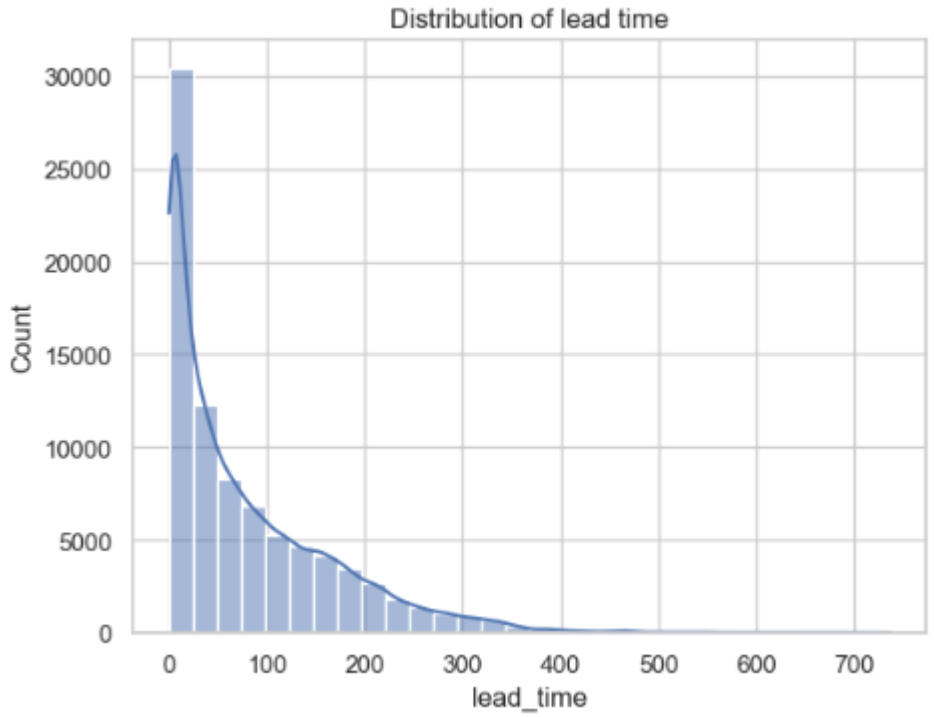
- **Target Variable - `is_canceled`**:
  - Countplot is used.
  - Clear class imbalance observed.



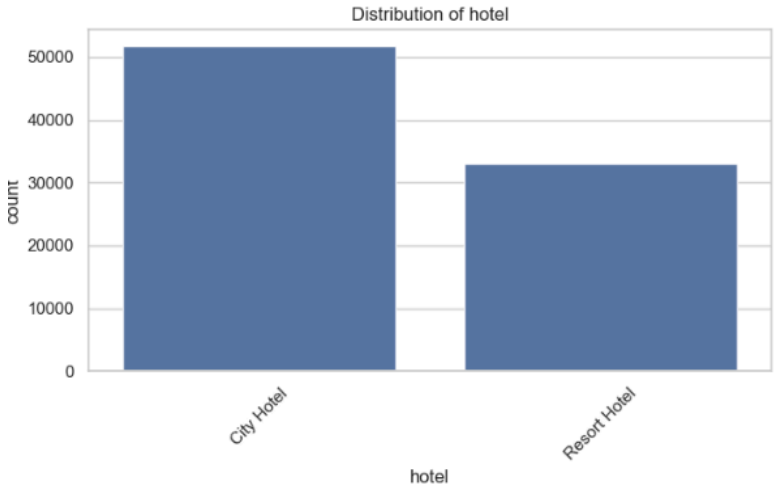
- **Hotel Type Distribution**:
  - Two categories: `City Hotel` and `Resort Hotel`
  - City hotels had a higher number of bookings overall.

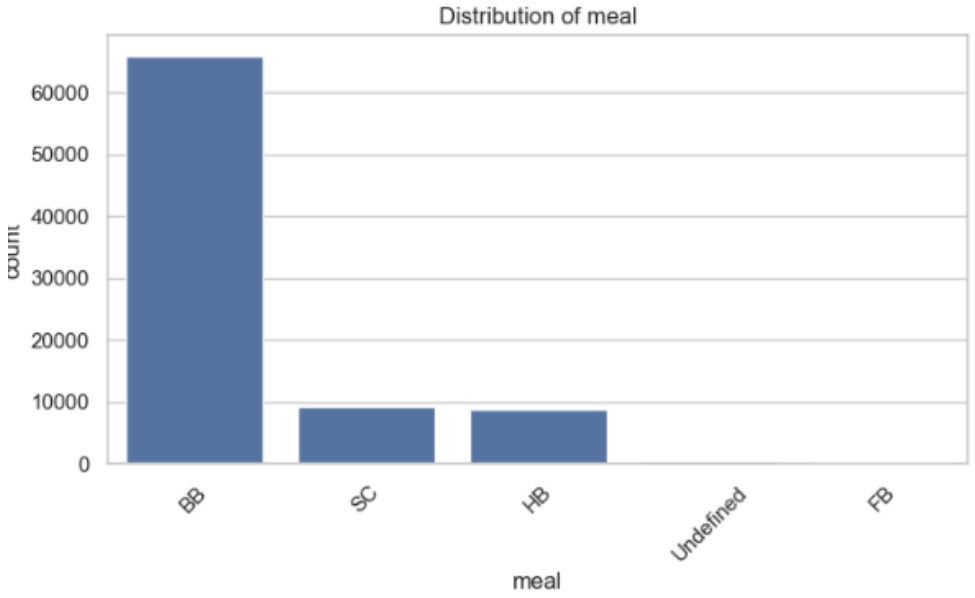


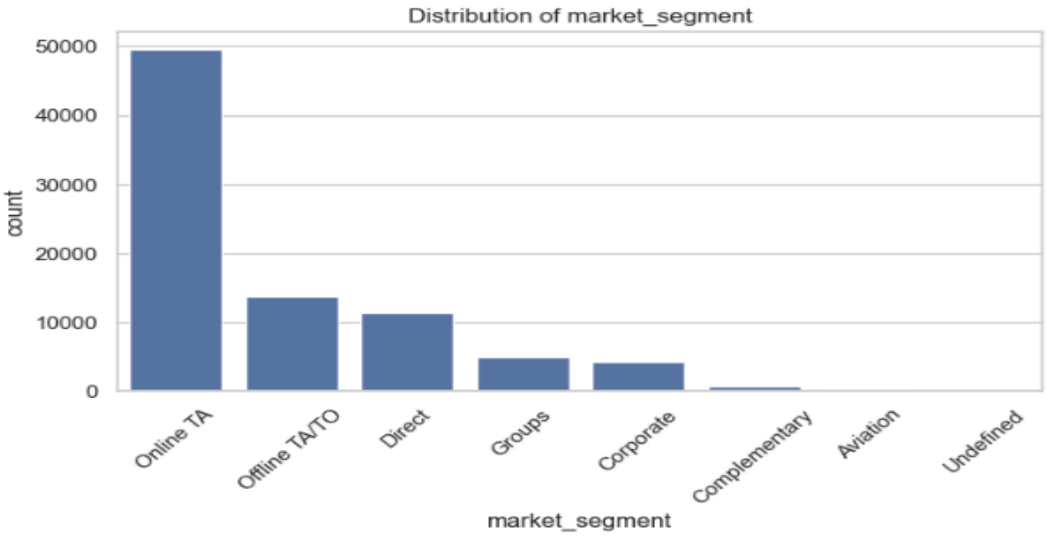
- **Lead Time Distribution**:
  - Highly skewed right; some bookings were made almost a year in advance.
  - Plotted histogram and boxplot revealed heavy outliers.

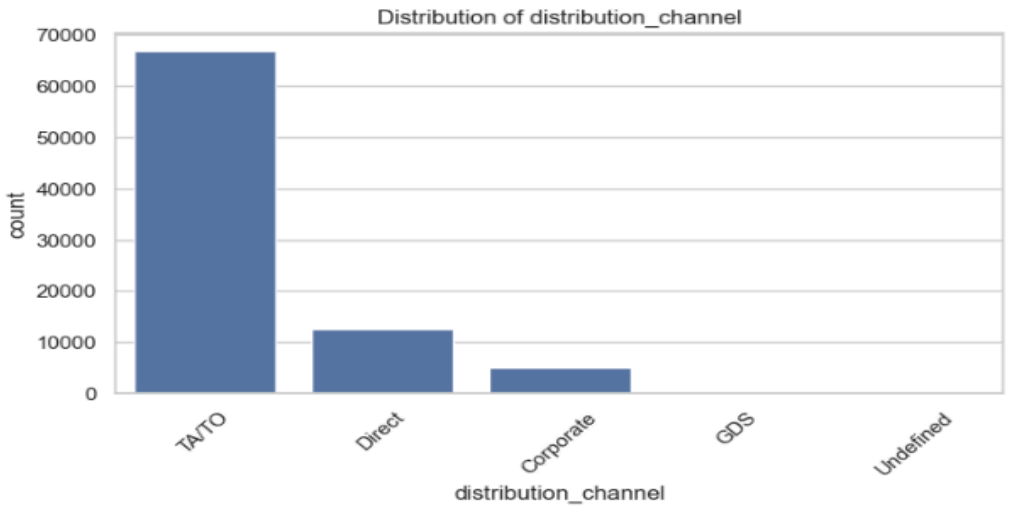


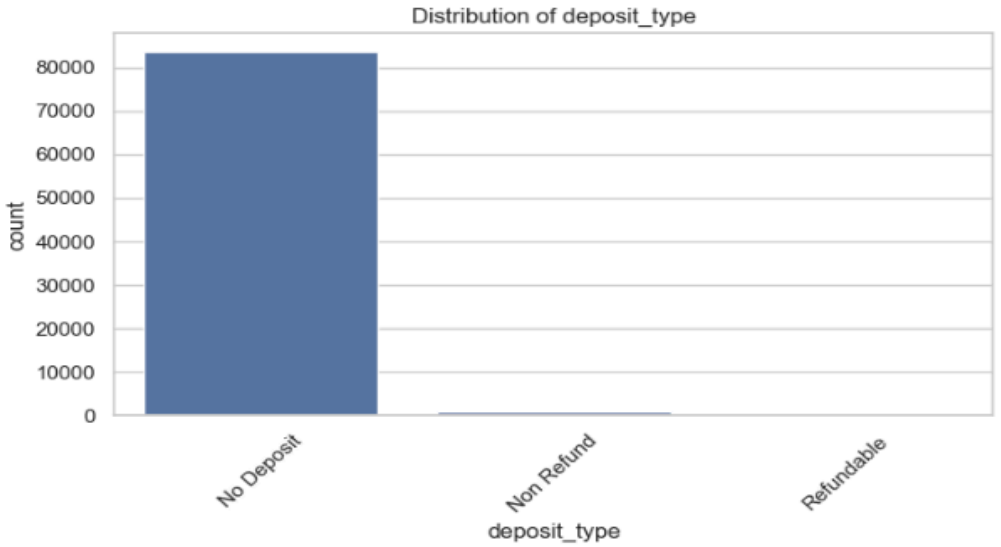
- **Categorical Variables**: Used countplots for the following features to visualize booking preferences:
  - `hotel`
  - `meal`
  - `market_segment`
  - `distribution_channel`
  - `deposit_type`
  - `customer_type`

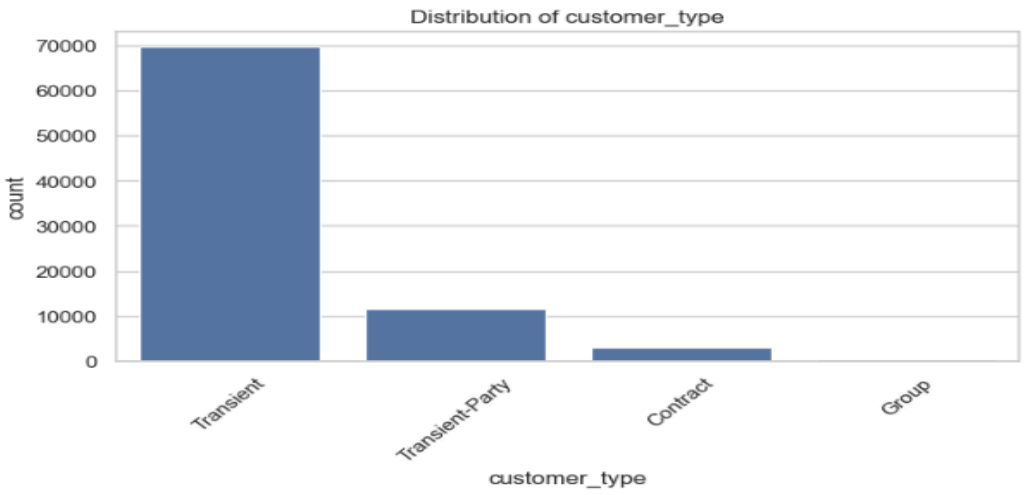

Distribution of hotel
count
50000
40000
30000
20000
10000
0
City Hotel
Resort Hotel
hotel


Distribution of meal
count
60000
50000
40000
30000
20000
10000
0
BB
SC
HB
Undefined
FB
meal


Distribution of market_segment
count
50000
40000
30000
20000
10000
0
Online TA
Offline TA/TO
Direct
Groups
Corporate
Complementary
Aviation
Undefined
market_segment


Distribution of distribution_channel
70000
60000
50000
40000
30000
20000
10000
0
count
TA/TO
Direct
Corporate
GDS
Undefined
distribution_channel


Distribution of deposit_type
80000
70000
60000
50000
40000
30000
20000
10000
0
count
No Deposit
Non Refund
Refundable
deposit_type


Distribution of customer_type
70000
60000
50000
40000
30000
20000
10000
0
count
Transient
Transient-Party
Contract
Group
customer_type

- **Observations**:
    - Most guests chose "No Deposit"
    - "Online TA" dominated as market segment
    - "Transient" was the most common customer type
    - Most meals were booked under "BB" (Bed & Breakfast)

Bivariate Analysis

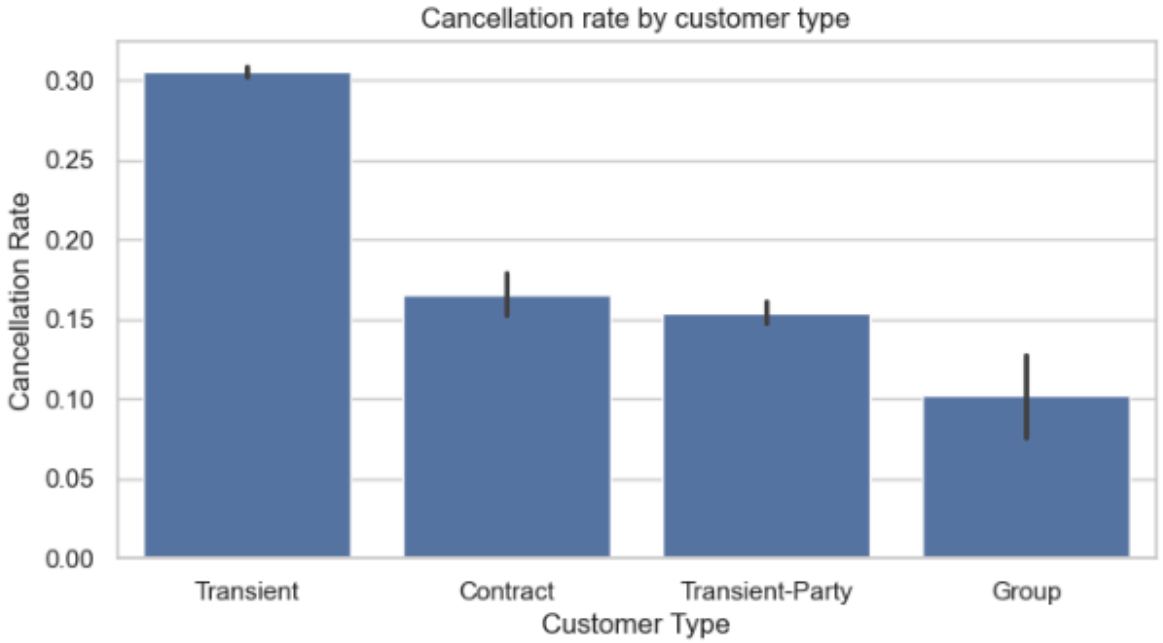
Assessed relationships between variables to understand patterns associated with cancellations:

- **Hotel Type vs Cancellation**:
    - City Hotel had significantly higher cancellation rates compared to Resort Hotels.

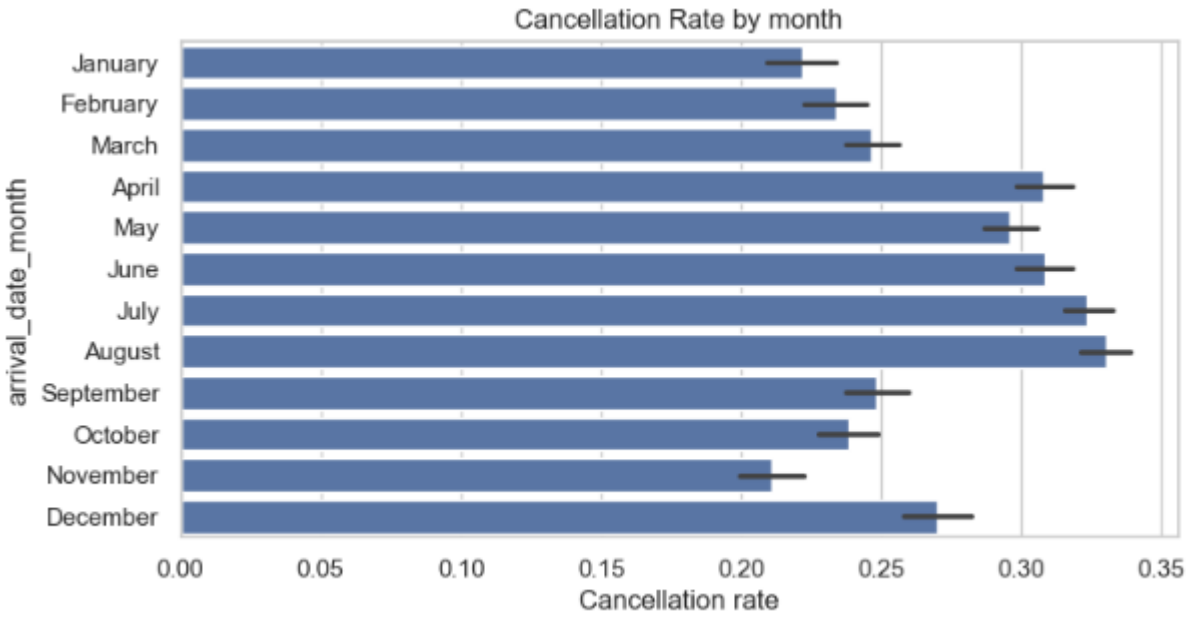
- **Deposit Type vs Cancellation**:
    - Guests with "No Deposit" had the highest cancellation rate.

    - "Non-refundable" bookings had very low cancellation rates.

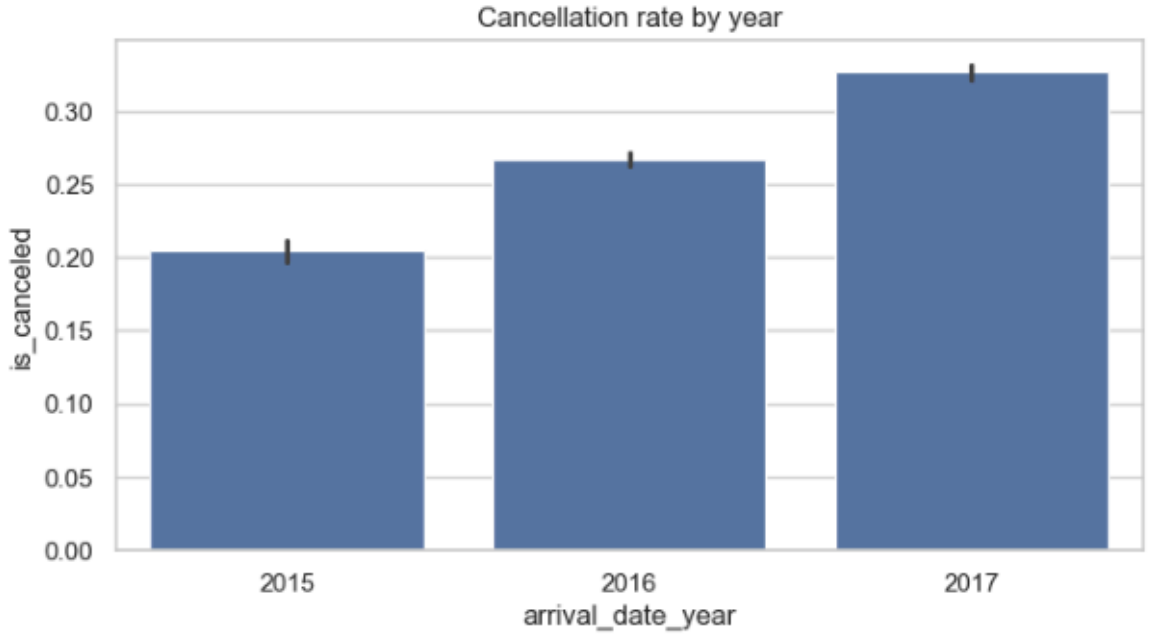
- **Customer Type vs Cancellation**:
    - "Transient" customers had the highest share of cancellations.



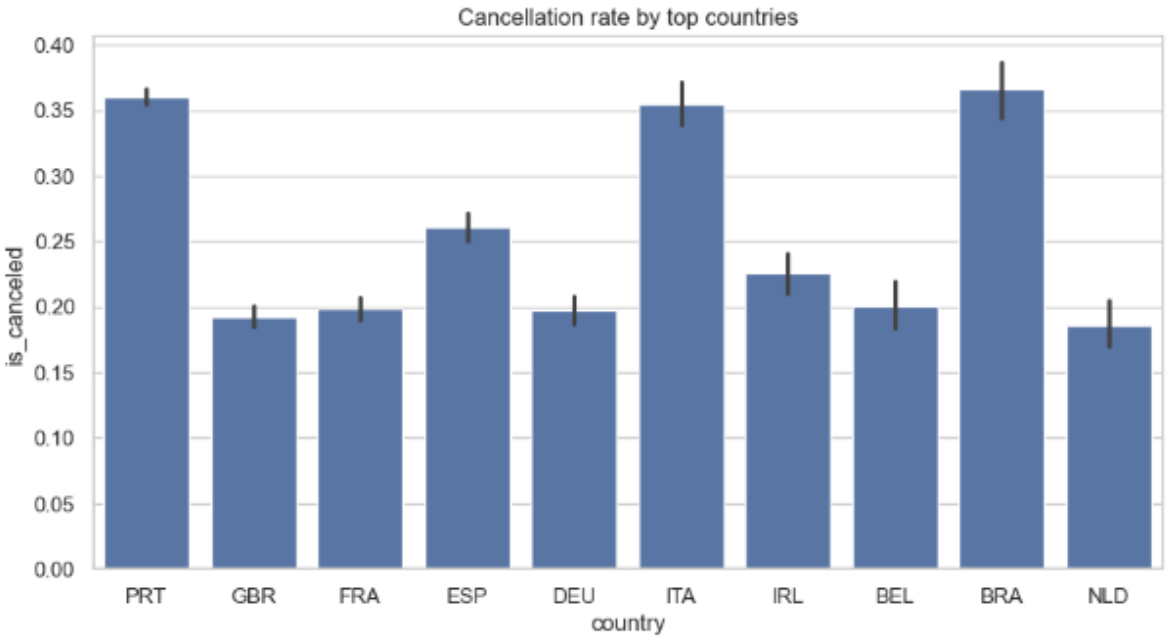
- **Seasonal Patterns (Month-wise Trends)**:
    - Cancellation rate peaked around July–August.
    - Resort Hotels had more stable bookings during winter months.



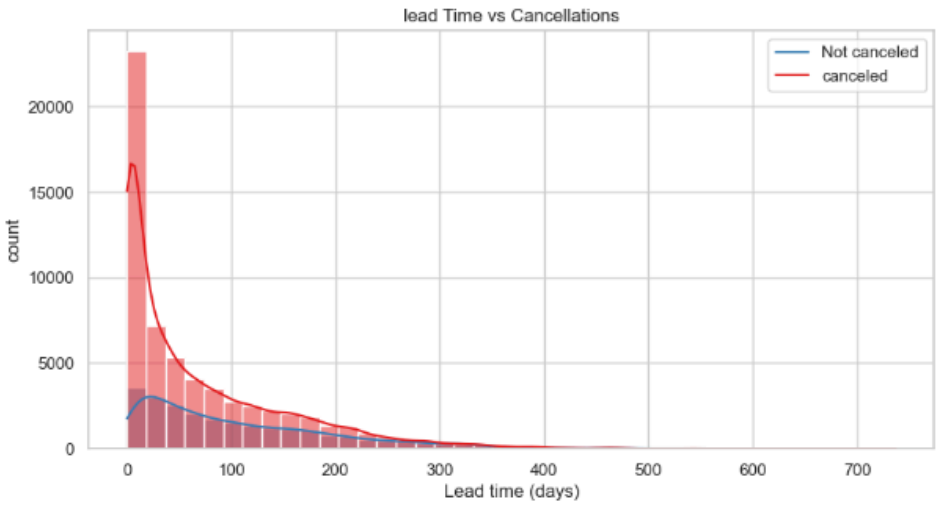
- **Year-wise Trends**:
  - Compared 2015, 2016, and 2017 bookings.
  - Cancellations gradually increased each year.



- **Country-wise Trends**:
  - Identified top 10 countries by number of bookings.
  - Portugal, the United Kingdom, France, and Spain topped the list.
  - Created bar plots showing country vs cancellation count.



- **Lead Time vs Cancellations**:
  - Canceled bookings had longer average lead times.
  - Bookings made far in advance were more likely to be canceled.



4. Model Development

To predict booking cancellations, we formulated this as a binary classification problem (`is_canceled`: 0 or 1). Several supervised learning models were trained, validated, and compared based on key performance metrics.

Models Evaluated

We tested the following classification models:

1. **Random Forest Classifier**
    - Ensemble-based, non-linear model that handled both numerical and categorical features effectively.
    - Showed strong performance with balanced accuracy and robustness to outliers.

```
#one hot encoding
df_encoded=pd.get_dummies(df,drop_first=True)

#separate features and target
target=df_encoded['is_canceled']
features=df_encoded.drop('is_canceled',axis=1)

print("features shape:",features.shape)
print("Target distribution:",target.value_counts())
```

```
features shape: (84723, 73)
Target distribution: is_canceled
0    61131
1    23592
Name: count, dtype: int64
```

•[98]:

```
# Drop datetime columns from feature set before training
cols=['reservation_status_date','company','arrival_date']
for col in cols:
    if col in features.columns:
        features = features.drop(col, axis=1)
```

[100]:

```
#split data into train and test sets

from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(features,target,test_size=0.2,random

print("Training set shape:",X_train.shape)
print("Test set shape:",X_test.shape)
```

```
Training set shape: (67778, 70)
Test set shape: (16945, 70)
```

[101]:

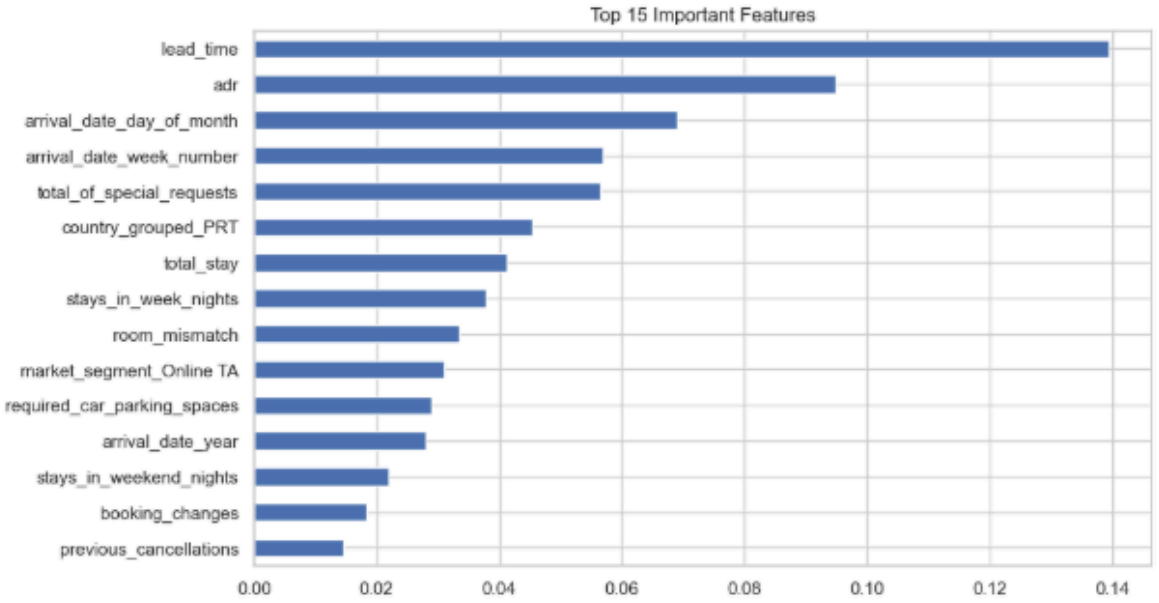
```
#random forest classifier

from sklearn.ensemble import RandomForestClassifier

model=RandomForestClassifier(n_estimators=100,random_state=42)

#train a model
model.fit(X_train,y_train)
print("Model training complete")
```

```
Model training complete
```

```python
#evaluate the model performance

from sklearn.metrics import classification_report,confusion_matrix

y_pred=model.predict(X_test)

#evaluation
print("Classification Report:\n",classification_report(y_test,y_pred))
print("confusion matrix:\n",confusion_matrix(y_test,y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.92      0.89     12226
           1       0.76      0.64      0.69      4719

    accuracy                           0.84     16945
   macro avg       0.81      0.78      0.79     16945
weighted avg       0.84      0.84      0.84     16945

confusion matrix:
 [[11248   978]
 [ 1687  3032]]
```

```python
#feature importance

feat_importance=pd.Series(model.feature_importances_,index=features.columns)
feat_importance.nlargest(15).sort_values().plot(kind='barh',figsize=(10,6))
plt.title("Top 15 Important Features")
plt.show()
```



2. **Logistic Regression**
    - Linear baseline model.
    - Performed well in precision but struggled with recall due to imbalance and nonlinear patterns.

```python
#Logistic regression

from sklearn.linear_model import LogisticRegression

log_model=LogisticRegression(max_iter=1000,class_weight='balanced',random_state=42)
log_model.fit(X_train,y_train)

y_pred_log=log_model.predict(X_test)

print("Logistic Regression report:\n",classification_report(y_test,y_pred_log))
```

```
Logistic Regression report:
              precision    recall  f1-score   support

           0       0.90      0.73      0.80     12226
           1       0.53      0.79      0.63      4719

    accuracy                           0.74     16945
   macro avg       0.71      0.76      0.72     16945
weighted avg       0.79      0.74      0.75     16945
```

3. **XGBoost Classifier** (*Best Performing Model)*
    - Gradient Boosted Trees model with superior learning ability on tabular data.
    - Handled class imbalance and nonlinear interactions effectively.
    - Fine-tuned using grid search and early stopping.

```python
#XGBoost Classifier - advance tree based model
from xgboost import XGBClassifier

xgb_model=XGBClassifier(use_label_encoder=False,eval_metric='logloss',random_state=
xgb_model.fit(X_train,y_train)

y_pred_xgb=xgb_model.predict(X_test)

print("XGBoost report:\n",classification_report(y_test,y_pred_xgb))
```

```
C:\Python312\Lib\site-packages\xgboost\training.py:183: UserWarning: [18:07:16] WAR
NING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
XGBoost report:
              precision    recall  f1-score   support

           0       0.88      0.91      0.89     12226
           1       0.75      0.67      0.71      4719

    accuracy                           0.84     16945
   macro avg       0.81      0.79      0.80     16945
weighted avg       0.84      0.84      0.84     16945
```

```python
from sklearn.model_selection import GridSearchCV

#fine tuning XGBoost

xgb_base=XGBClassifier(use_label_encoder=False,eval_metric='logloss',random_state=42)

# hyperparameter grid
param_grid={
    'n_estimators':[100,200],
    'max_depth':[3,5,7],
    'learjobsning_rate':[0.01,0.1],
    'subsample':[0.8,1],
    'colsample_bytree':[0.8,1]
}

#grid search
grid_xgb=GridSearchCV(
    estimator=xgb_base,
    param_grid=param_grid,
    scoring='f1',
    cv=3,
    verbose=1,
    n_jobs=-1
)

#fit grid search
grid_xgb.fit(X_train,y_train)

#best model
best_xgb=grid_xgb.best_estimator_
print("Best XGBoost params:",grid_xgb.best_params_)
```

```
Fitting 3 folds for each of 48 candidates, totalling 144 fits
```

```
y_pred_best_xgb=best_xgb.predict(X_test)
print("Fine-tuned XGBoost report:\n",classification_report(y_test,y_pred_best_xgb))
```

```
Fine-tuned XGBoost report:
              precision    recall  f1-score   support

           0       0.88      0.91      0.89     12226
           1       0.74      0.67      0.71      4719

    accuracy                           0.84     16945
   macro avg       0.81      0.79      0.80     16945
weighted avg       0.84      0.84      0.84     16945
```

4. **Decision Tree Classifier**
    - Single tree model used to interpret decision logic.
    - Overfit easily, leading to lower overall accuracy compared to ensemble models.

```
from sklearn.tree import DecisionTreeClassifier

dt_model=DecisionTreeClassifier(max_depth=5,class_weight='balanced',random_state=42)
dt_model.fit(X_train,y_train)

y_pred_dt=dt_model.predict(X_test)

print("Decision Tree report:\n",classification_report(y_test,y_pred_dt))
```

```
Decision Tree report:
              precision    recall  f1-score   support

           0       0.93      0.63      0.76     12226
           1       0.48      0.89      0.62      4719

    accuracy                           0.70     16945
   macro avg       0.71      0.76      0.69     16945
weighted avg       0.81      0.70      0.72     16945
```

Final Performance Metrics (on Test Set)

Model evaluation was done using **accuracy**, **precision**, **recall**, and **F1-score**.

Model	Accuracy	Precision (Canceled)	Recall (Canceled)	F1-Score (Canceled)
Random Forest	0.84	0.76	0.65	0.70
Logistic Regression	0.74	0.53	0.79	0.63
XGBoost (Tuned)	0.84	0.74	0.67	0.71
Decision Tree	0.70	0.48	0.89	0.62

Note: Precision/recall scores shown here are for predicting class 1 (cancellation).

Feature Importance (XGBoost)

The top predictors influencing cancellation according to the trained XGBoost model:

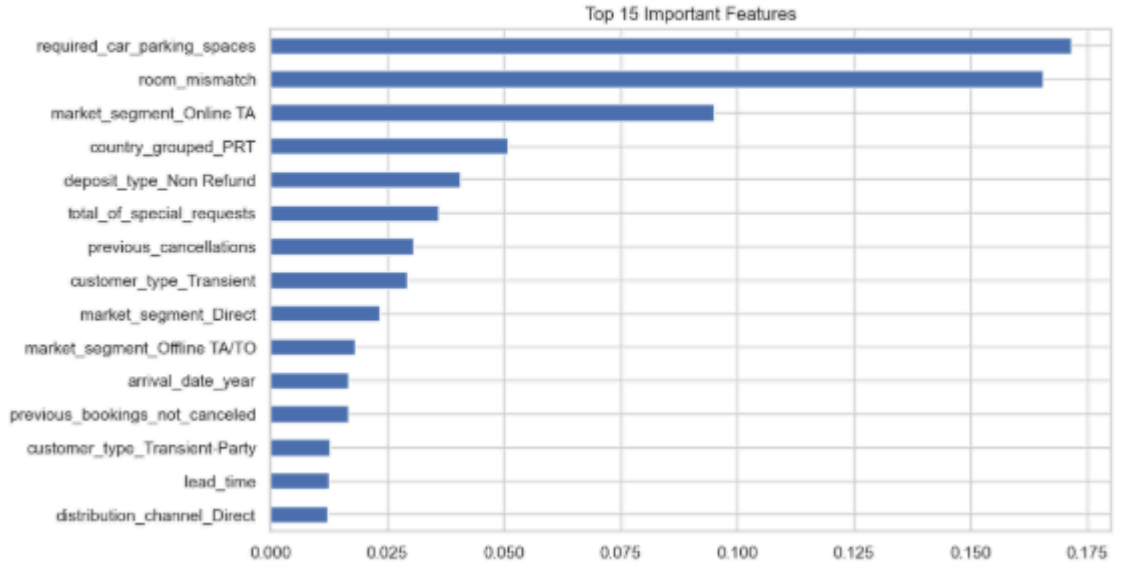
1. `required_car_parking_spaces`
2. `room_mismatch`

3. `market_segment_Online TA`
4. `country_grouped_PRT`
5. `deposit_type_Non Refund`
6. `total_of_special_requests`
7. `previous_cancellations`
8. `customer_type_Transient`
9. `market_segment_Direct`
10. `market_segment_Offline TA/TO`

These features aligned well with insights observed in EDA and business intuition.

Visual feature importance plots from XGBoost were included to highlight interpretability.

```
xgb_feat_importance=pd.Series(best_xgb.feature_importances_,index=features.columns)
xgb_feat_importance.nlargest(15).sort_values().plot(kind='barh',figsize=(10,6))
plt.title("Top 15 Important Features")
plt.show()
```

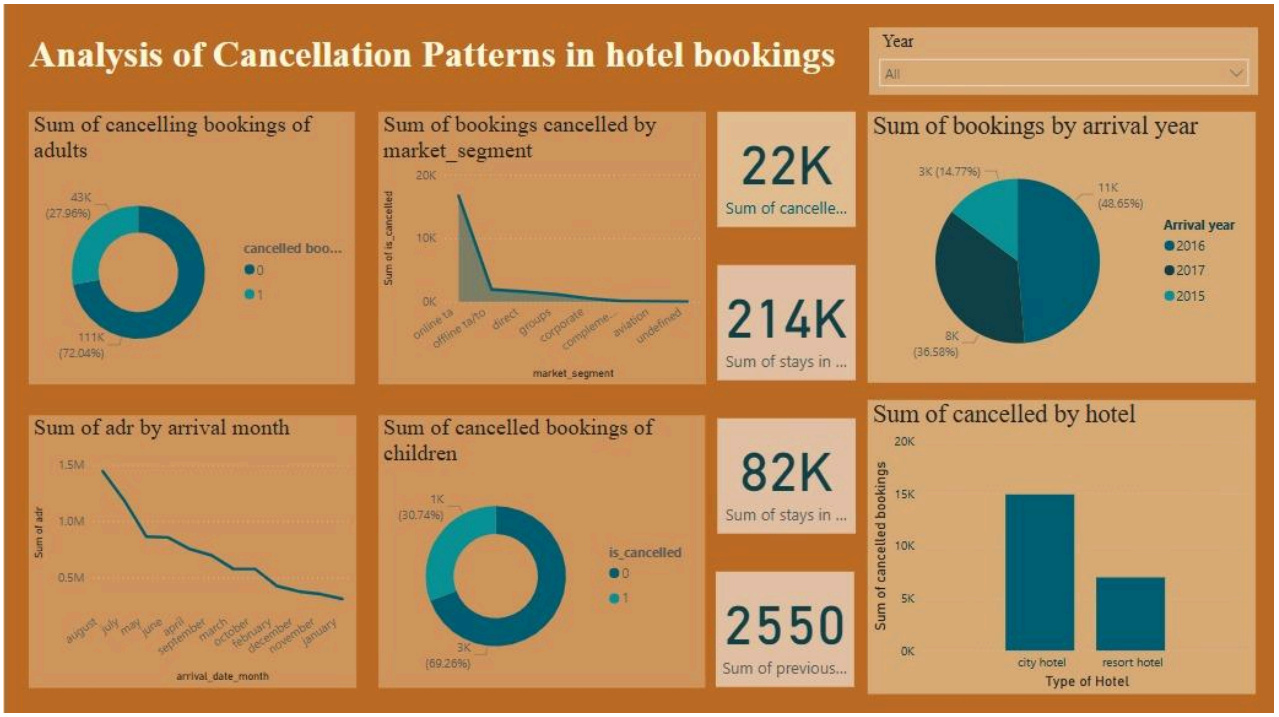


5. Dashboard Insights (Power BI)

To complement the machine learning modeling, we developed an **interactive Power BI dashboard** that visually summarizes key cancellation trends and booking behaviors in the hotel industry.

Overview

The dashboard titled **“Analysis of Cancellation Patterns in Hotel Bookings”** is designed for business stakeholders and hotel management teams to monitor, understand, and react to booking behaviors that affect occupancy and revenue.



Key Visuals and Insights

1. **Cancellation by Adults**
    - Shows a **donut chart** comparing canceled vs. non-canceled bookings based on adult counts.
    - **72.04% of bookings with adults were not canceled**, while **27.96% were canceled**, revealing a strong likelihood of fulfillment when adults are included in the booking.
2. **Cancellations by Market Segment**
    - This vertical bar chart reveals **‘Online TA’ (Online Travel Agents)** as the primary contributor to cancellations with a significantly higher volume than other segments like ‘Direct’ or ‘Corporate’.
    - Indicates possible over-dependence on third-party platforms with less customer commitment.
3. **Cancellations by Hotel Type**
    - City Hotels faced nearly **twice the cancellations** compared to Resort Hotels.
    - Suggests urban areas might have more short-term or volatile bookings, potentially driven

by business travel or convenience-led choices.

4. **Annual Trends in Booking Volume**

    - Pie chart showing booking distribution across **2015, 2016, and 2017**.

    - 2016 recorded **the highest volume of bookings (48.65%)**, suggesting growth during that year or improved data capture.

5. **Monthly ADR (Average Daily Rate) Trend**

    - Line chart of ADR across months reveals **peak pricing during July and August**, aligning with the summer travel season.
    - Gradual decline observed into off-peak months like December and January.

6. **Cancellations by Children Count**
    - Donut chart shows that **30.74% of bookings with children were canceled**, hinting that family travel might be more prone to cancellations (possibly due to logistical changes or seasonal dependencies).

7. **Arrival Month vs. ADR**
    - Clear visualization of how **room pricing fluctuates seasonally**, allowing the revenue team to optimize promotions and policies accordingly.

8. **KPIs and Aggregates (Right Side Cards)**
    - Display high-level statistics such as:
        - **22K** total canceled bookings
        - **214K** stays in hotels
        - **82K** stays in resorts
        - **2550** previous cancellations, indicating repeated patterns from certain guests

6. Results & Observations

This section summarizes the outcomes of both the **predictive machine learning model** and the **interactive Power BI dashboard**, highlighting their effectiveness in addressing real-world challenges in the hotel industry.

Machine Learning Model Outcomes

The best-performing model, **XGBoost**, demonstrated strong predictive capability in classifying booking cancellations, with the following key metrics:

Metric	Value
Accuracy	84%
Precision (Canceled)	0.74
Recall (Canceled)	0.67
F1-Score (Canceled)	0.71

These metrics indicate that the model is:

- **Reliable in identifying likely cancellations**, helping prevent overbooking-related losses.
- **Effective in minimizing false positives**, which means the hotel is less likely to mistakenly treat a legitimate booking as a cancellation.
- **Balanced in recall and precision**, making it suitable for deployment in real-time decision-making systems.

Feature importance analysis further revealed that factors such as:

- `required_car_parking_spaces`,

- `room_mismatch`,
- `market_segment_Online TA`,
- `deposit_type_Non Refund`,
- and `total_of_special_requests`
  play a significant role in predicting whether a booking will be canceled.

Real-world impact: Hotels can now prioritize follow-ups or confirmations for bookings predicted as “likely to cancel,” thereby reducing last-minute cancellations and optimizing room allocation.

Power BI Dashboard Outcomes

The **Power BI dashboard** complemented the machine learning model by providing intuitive and actionable insights through interactive visuals.

Key Strategic Insights:

- **High cancellation rates** were observed in bookings from the *Online Travel Agent* segment, and among customers who didn’t require parking.

- **Room mismatch cases** showed a strong correlation with cancellations, indicating a potential issue in reservation handling or customer satisfaction.

- **City Hotels** experienced significantly more cancellations compared to Resort Hotels, signaling a

need for tailored cancellation policies based on hotel type.

- **Seasonal Trends** revealed peak cancellations during mid-year months, suggesting opportunities for promotional campaigns or flexible policies during high-risk periods.

- **Family Bookings** involving children showed higher instability, underlining the importance of dynamic family-specific packages and retention offers.

Business Benefits:

- **Marketing & Customer Targeting**: Enables managers to identify customer types and seasons associated with high cancellation risk.

- **Revenue Forecasting**: Helps improve revenue prediction accuracy by understanding booking behaviors over time.

- **Operational Planning**: Allows for better room allocation and staffing based on expected cancellations and arrivals.

Combined Value

By integrating machine learning predictions with real-time visual analysis:

- **Hotels gain a competitive advantage** through data-backed decision-making.

- **Customer engagement and retention improve**, as targeted strategies can be applied to risky bookings.

- **Operational costs are reduced**, and **overbooking risk is minimized** through accurate forecasting and smart allocation.

Challenges Faced:

- High imbalance between canceled vs non-canceled bookings
- Missing data in categorical features like `company`, `agent`
- Feature leakage due to date fields like `reservation_status_date`
- Feature importance not always aligned with correlation (nonlinear impact)

Future Work:

- Deploy the model as a web-based prediction app
- Integrate with real-time booking systems
- Include customer review sentiment if available

Deliverables:

- Final Report (this document)
- Power BI Dashboard
- GitHub Repository (cleaned notebook + model code + assets)
- Presentation (submitted separately)

GitHub Repository

Varshini-Chilakala/hotel_booking_cancellation_prediction_project