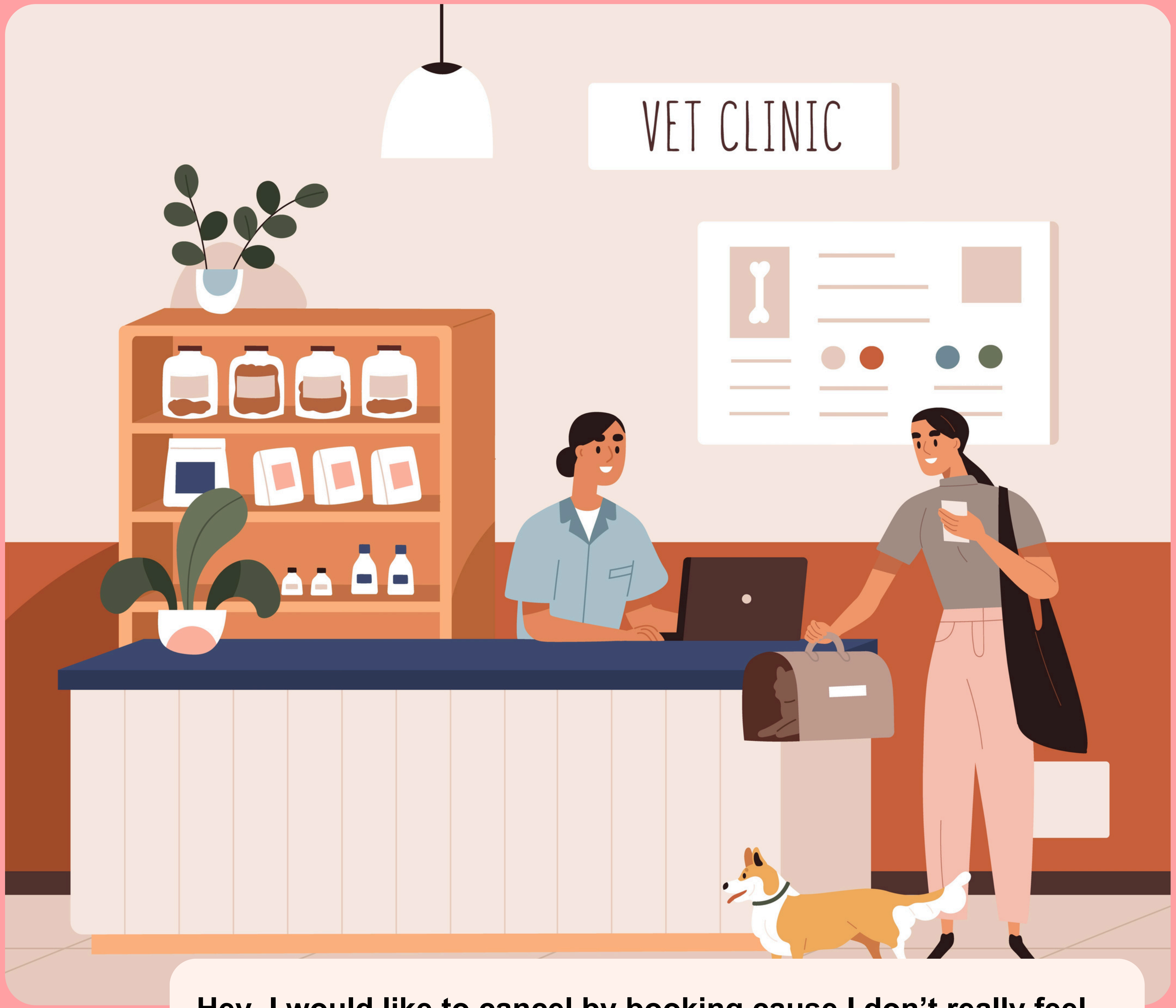
HOTEL BOOKING CANCELLATION

ECDS GROUP 11

The hospitality sector is witnessing a shift in booking dynamics, marked by increasing last-minute cancellations , causing significant revenue losses and poor demand forecasting

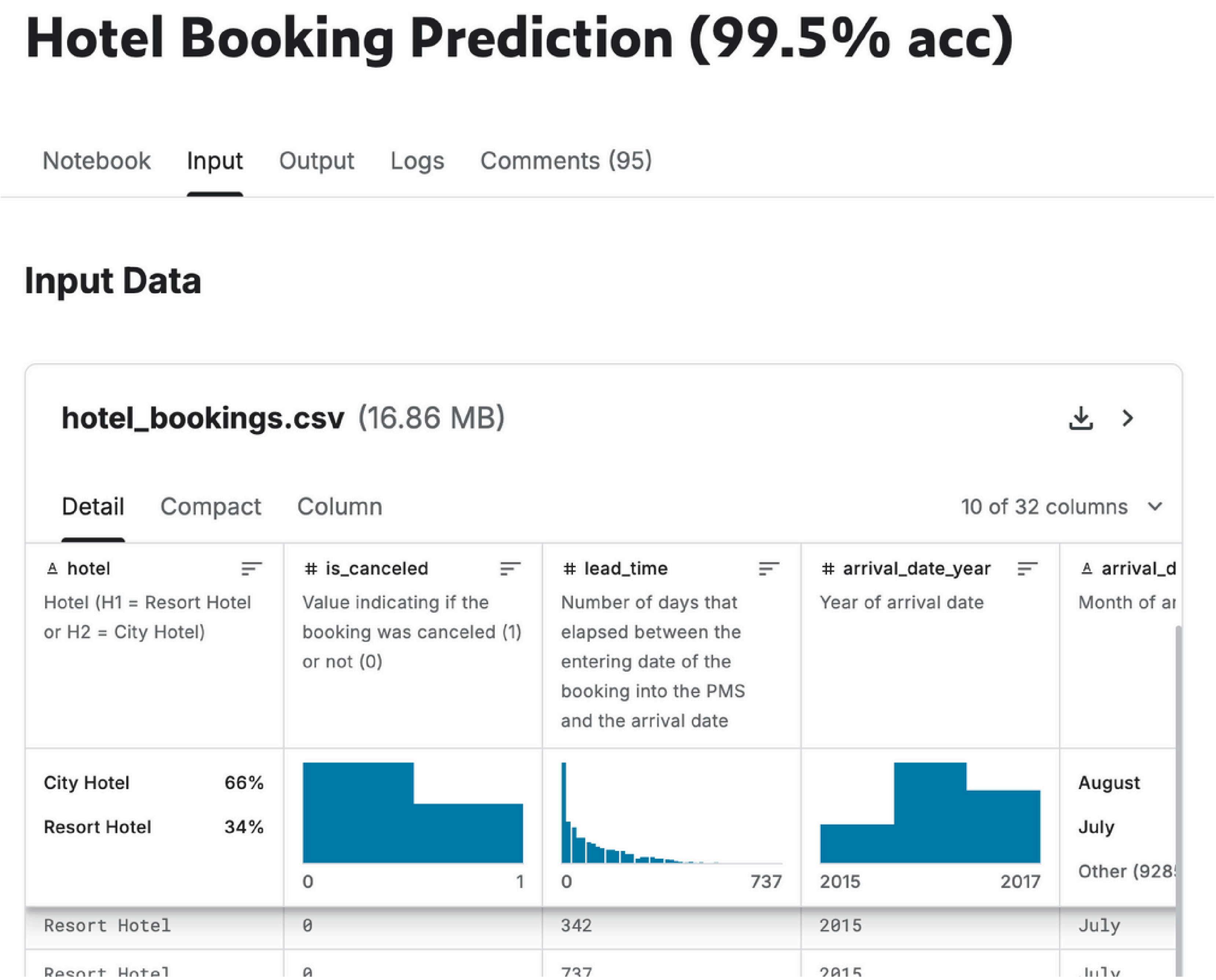


Hey, I would like to cancel by booking cause I don’t really feel like staying here anymore. But the service was great though!

Problem Definition:

Can we predict if a hotel booking will be canceled at the time of booking, using customer and booking information

DATA SET



26 MONTHS

JULY 2015

AUGUST 2017

119,000 booking records

32 FEATURES

DATA PREPARATION, CLEANING AND FEATURE ENGINEERING

IRREGULARITIES

EXPLORATORY DATA ANALYSIS

DATA PREPEARATION, CLEANING AND FEATURE ENGINEERING

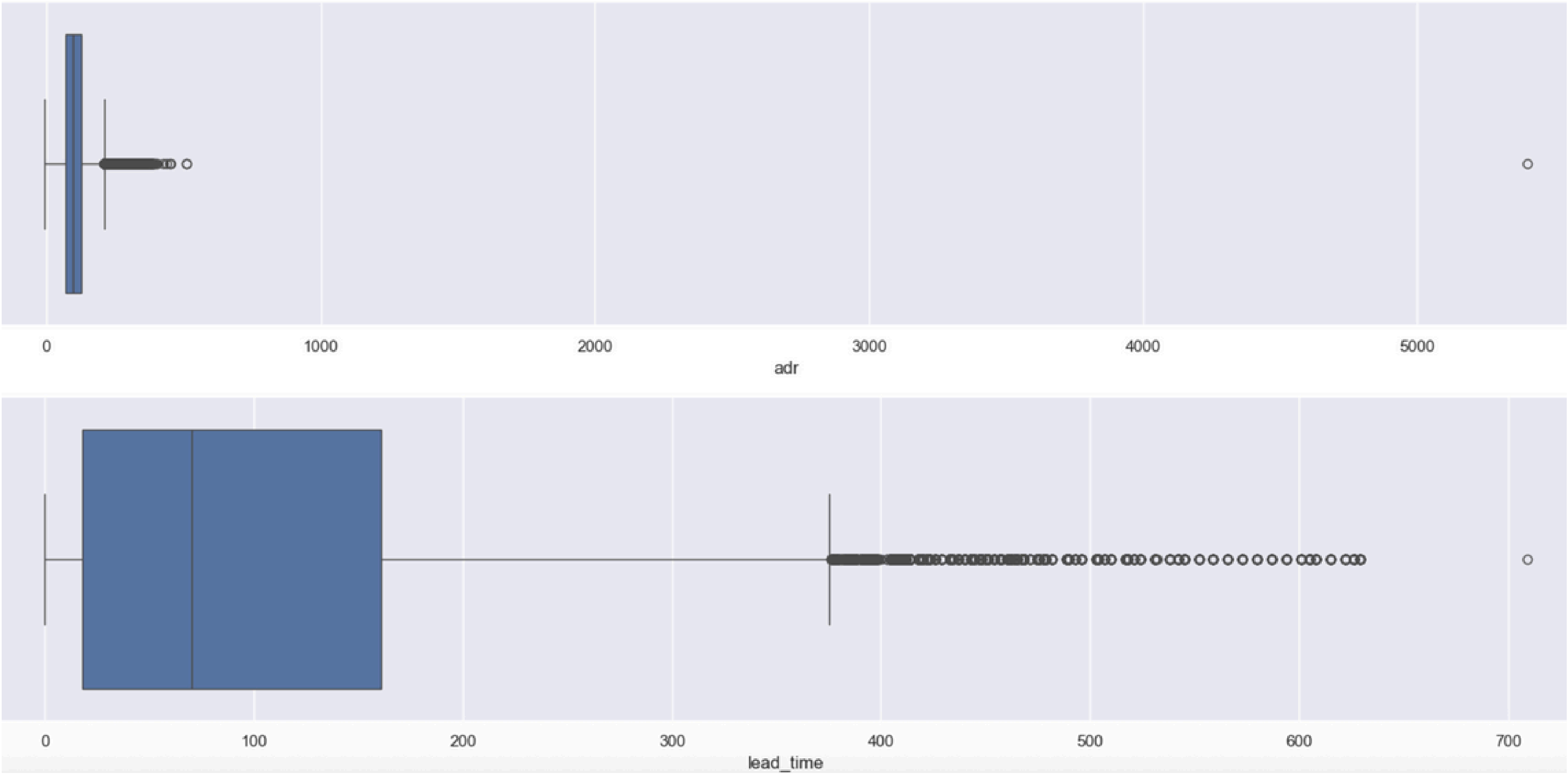
Handled missing values by removing rows with nulls in ‘children’ and imputing values for ‘country’, ‘agent’, and ‘company’.

Transformed 'agent' and 'company' to categorical strings, encoded categorical variables using one-hot encoding, and dropped irrelevant columns.

Created new features — 'stay_duration', 'total_guests', and 'Has_agent' — and checked for class imbalance to inform model strategy.

IRREGULARITIES

Outliers

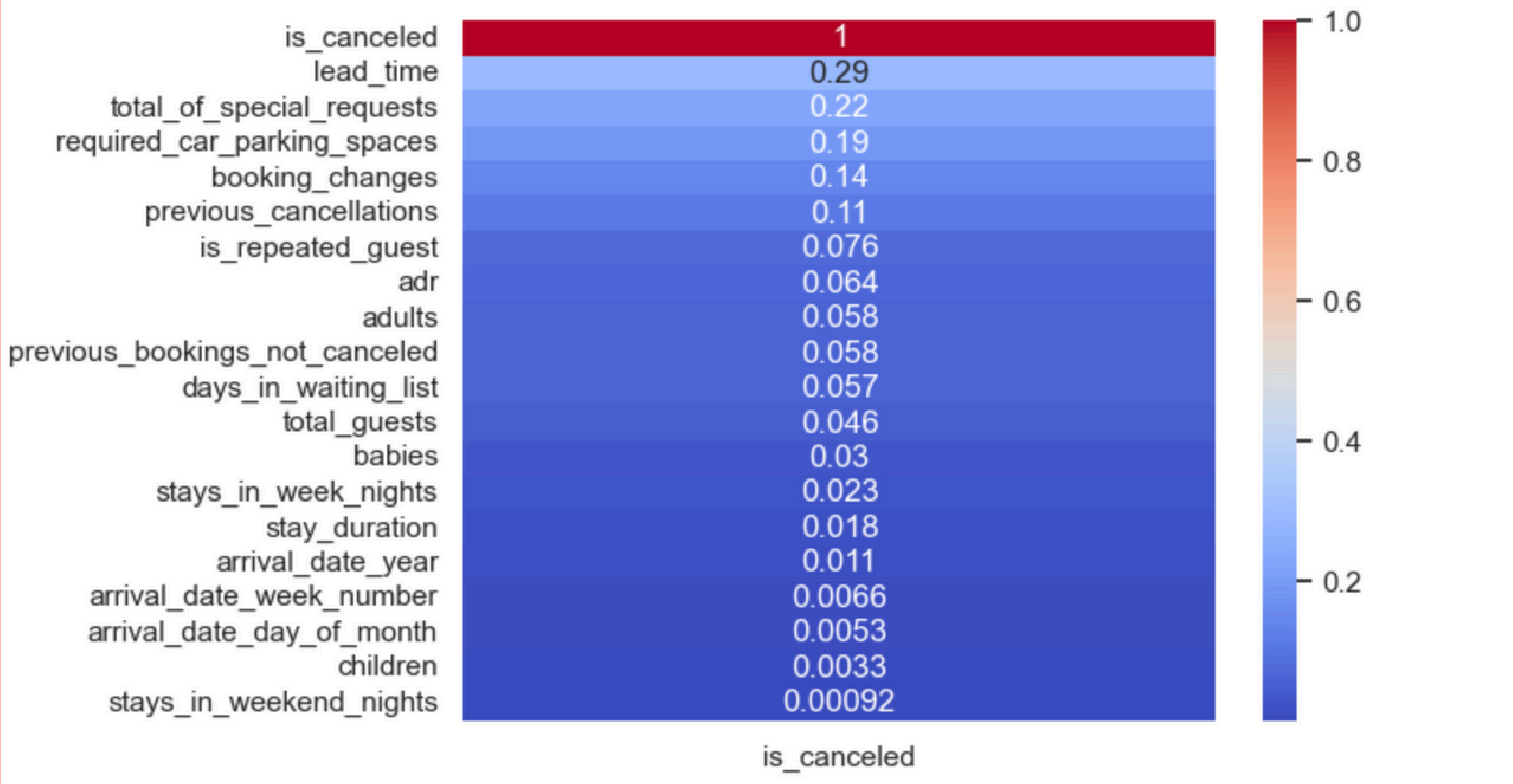


Inter Quartile Range

```
a = ['adr','lead_time']
for i in a:
    var = data[i]

    Q1 = var.describe().loc["25%"]
    Q3 = var.describe().loc["75%"]
    IQR = Q3-Q1


    Low = (Q1-(1.5*IQR))
    Upper = (Q3 + (1.5*IQR))

    data = data[(data[i] >= Low ) & (data[i] <Upper)]
```

Exploitary Data Analysis

From our initial data analysis we realised that the correlation between the numeric variables are low … With the highest being 0.29, between lead_time and is_cancelled



Exploitary Data Analysis

Use of Cramer's V to obtain coerralation between categorical variables

```python
from scipy.stats import chi2_contingency

def cramers_v(confusion_matrix):
    """Calculate Cramer's V (association strength between two categorical variables)."""
    chi2 = chi2_contingency(confusion_matrix)[0]
    n = confusion_matrix.sum().sum()
    phi2 = chi2 / n
    r, k = confusion_matrix.shape
    return np.sqrt(phi2 / min(k-1, r-1))


categorical_cols = data.select_dtypes(include=['object', 'category']).columns.tolist()


cramers_v_scores = {}

for col in categorical_cols:
    confusion_matrix = pd.crosstab(data[col], data['is_canceled'])
    score = cramers_v(confusion_matrix)
    cramers_v_scores[col] = score


cramers_v_sorted = pd.Series(cramers_v_scores).sort_values(ascending=False)
print("Coerralation of categorical Variables with 'is_cancelled'")
print()
print(cramers_v_sorted)
```

Output

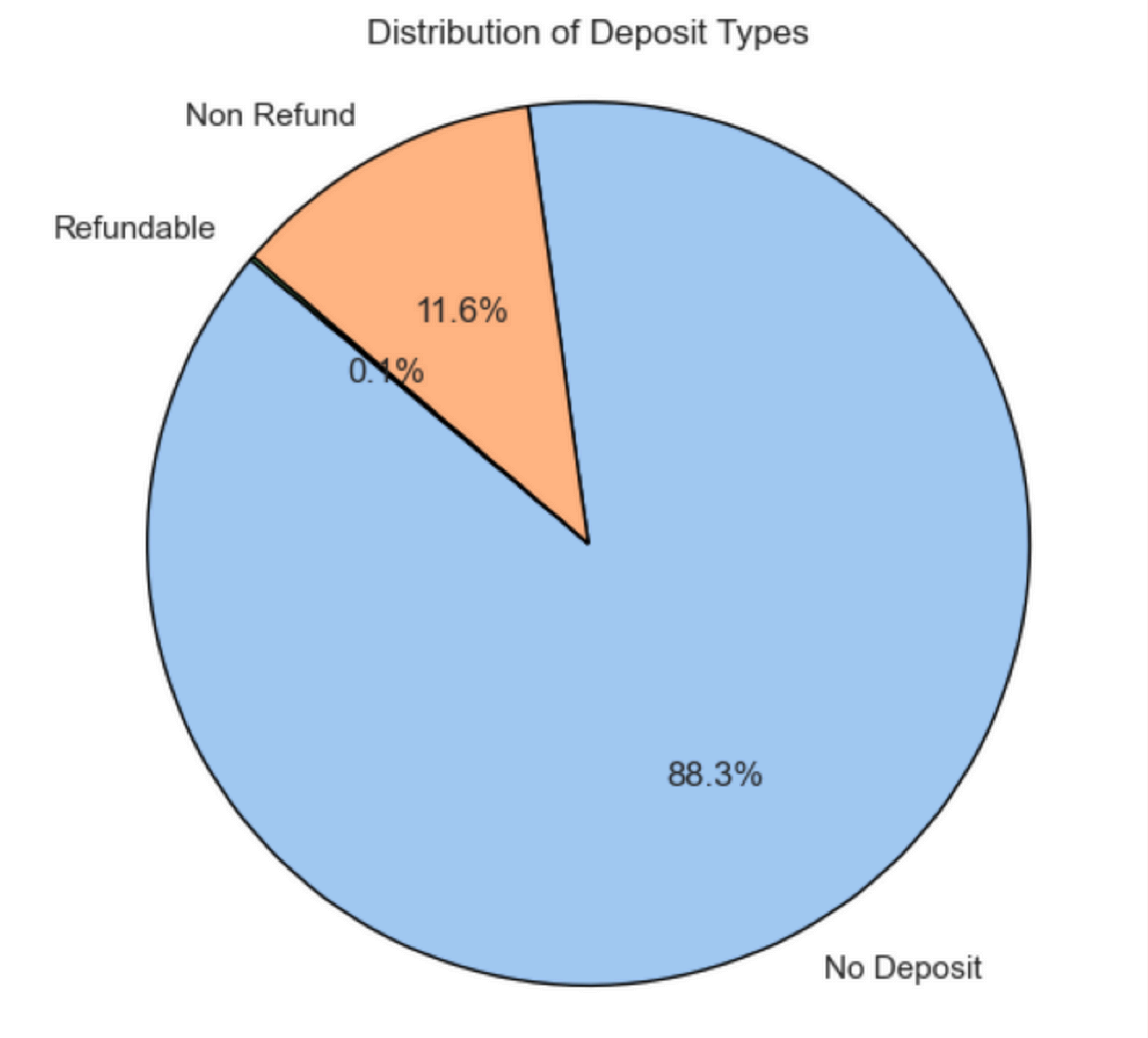
```
Coerralation of categorical Variables with 'is_cancelled'

deposit_type            0.472156
agent                   0.376709
country                 0.358607
market_segment          0.255142
assigned_room_type      0.200957
distribution_channel    0.171346
company                 0.142737
hotel                   0.136757
customer_type           0.125814
Has_Agent               0.097748
arrival_date_month      0.074611
reserved_room_type      0.071304
meal                    0.053124
dtype: float64
```

Exploratory Data Analysis

In the case of categorical data we managed to obtain better relationship between the variables. Some of the key factors that we found were :

`Country, agent  and`

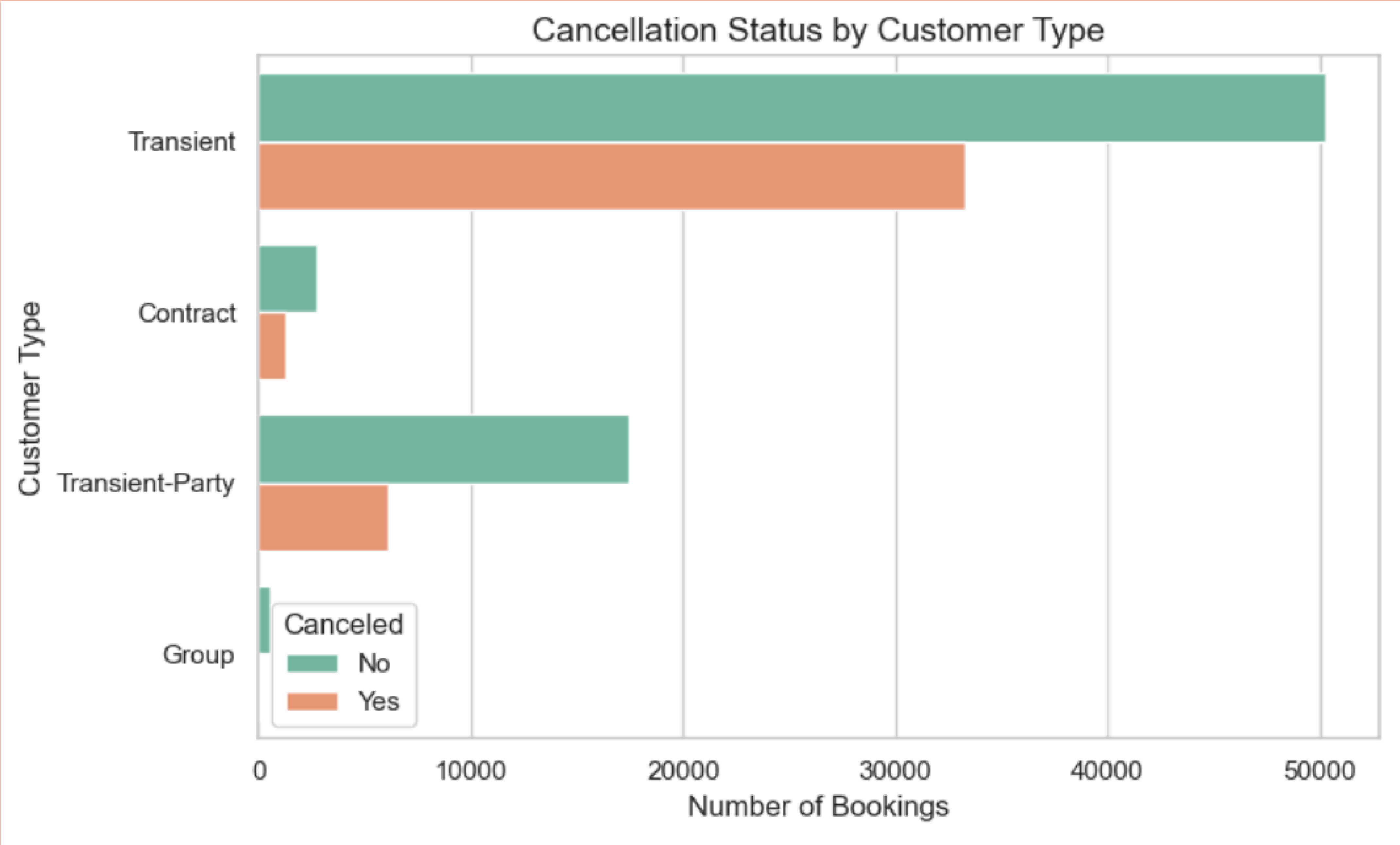
`Refundable deposit_type`



Exploratory Data Analysis

In the case of categorical data we managed to obtain better relationship between the variables. Some of the key factors that we found were :

`Country, Customer_type and Refundable deposit_type`

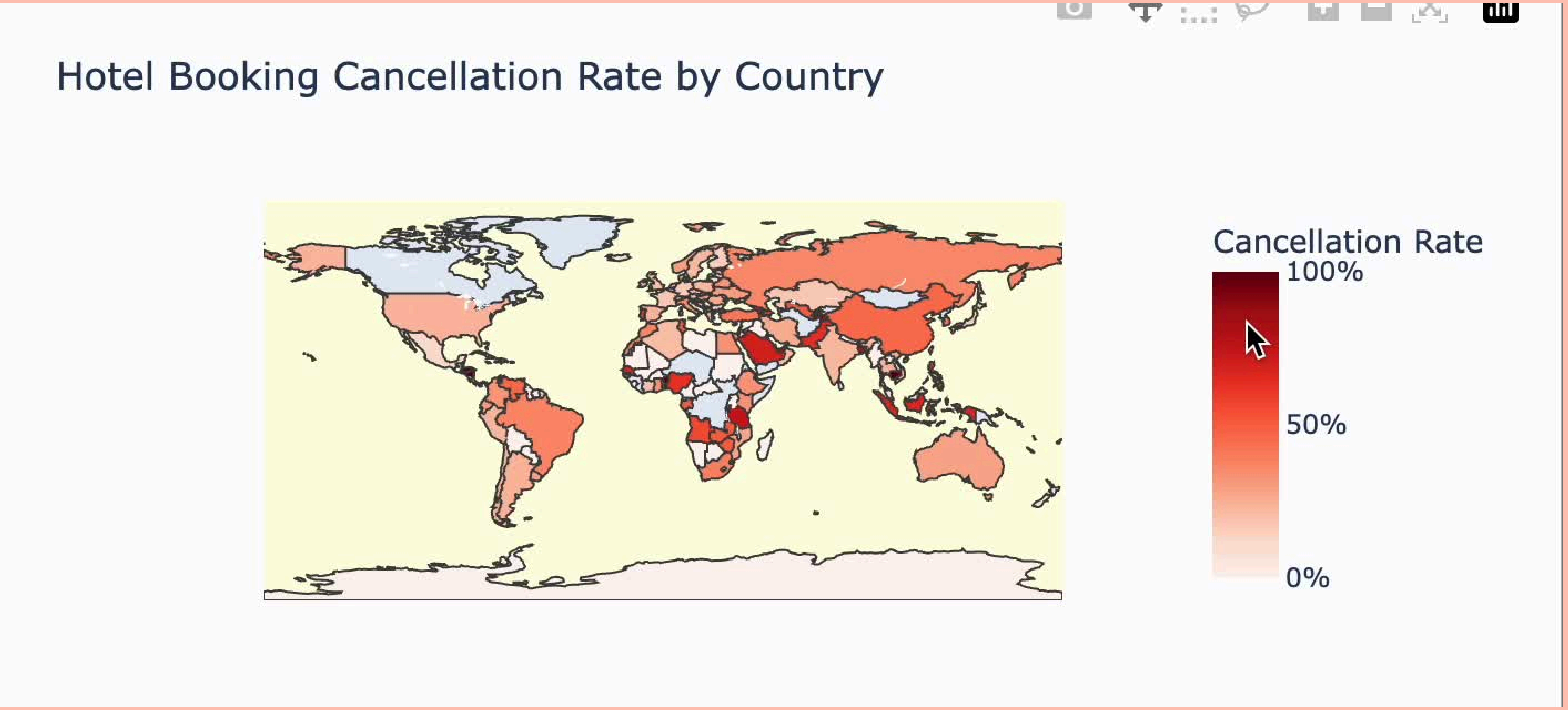


Exploratory Data Analysis

In the case of categorical data we managed to obtain better relationship between the variables. Some of the key factors that we found were :

`Country, agent and`

`Refundable deposit_type`



Which variables to choose

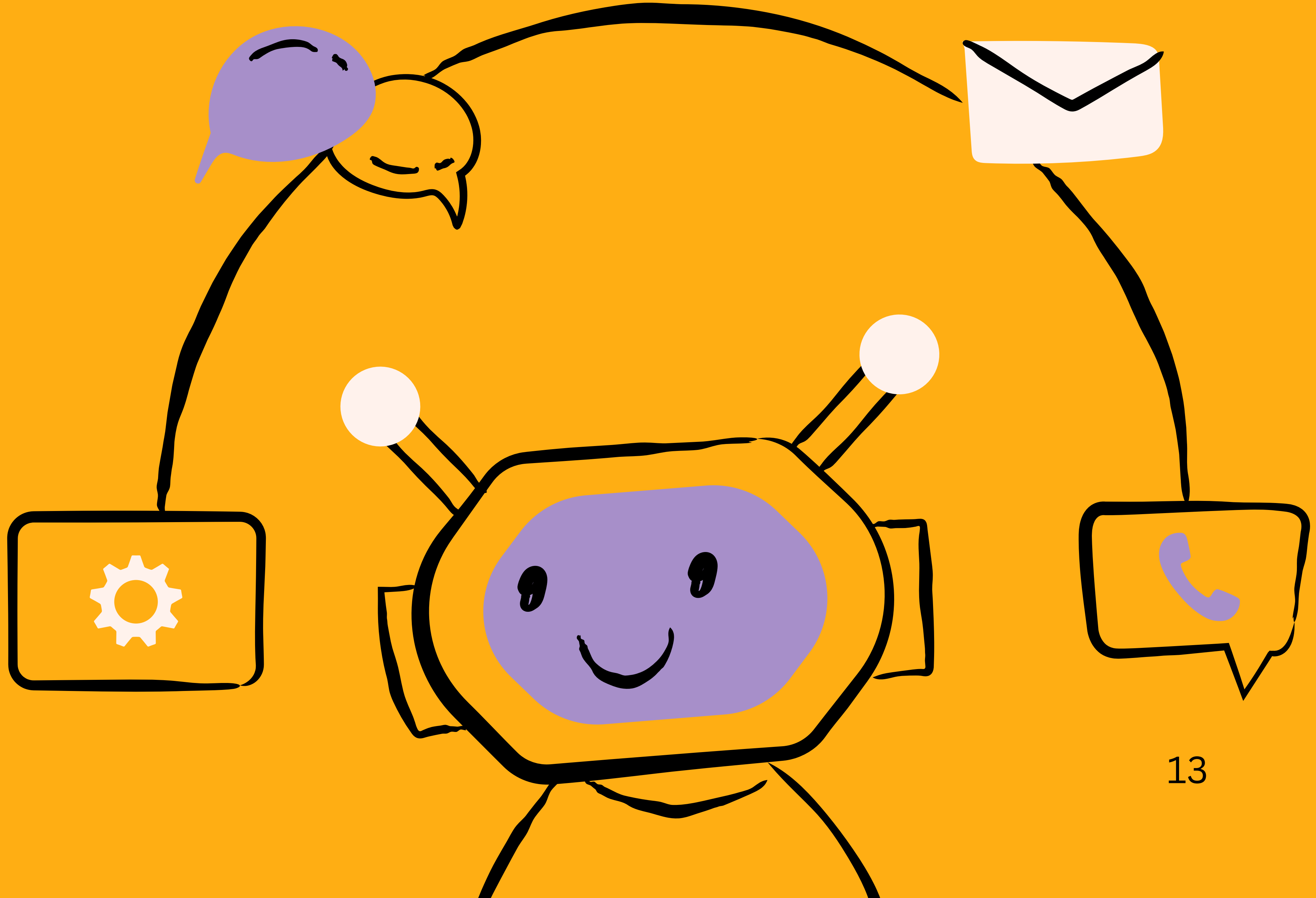
Numerical Variables :

`lead_time`, `total_of_special_requests`

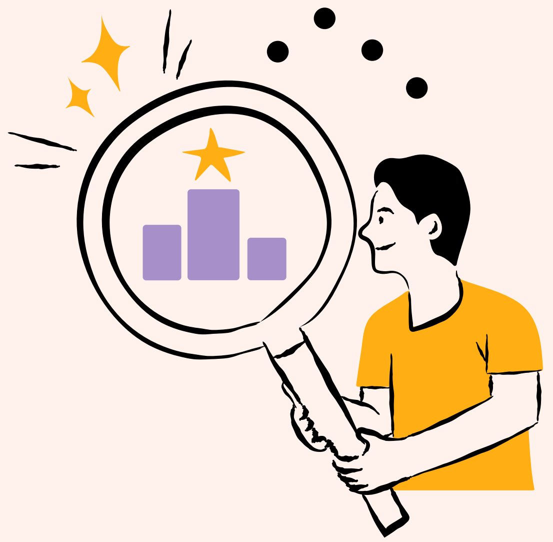
Categorical Variables :

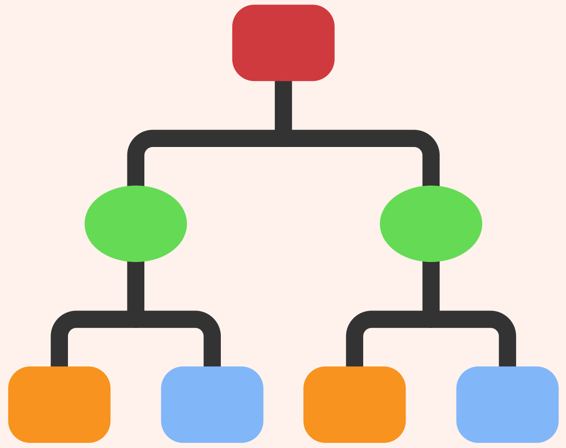
`deposit_type`, `agent`, `market_segment`

Using a combination of variables improves our model's predictive performance and reflects the complexity of real-world decision-making

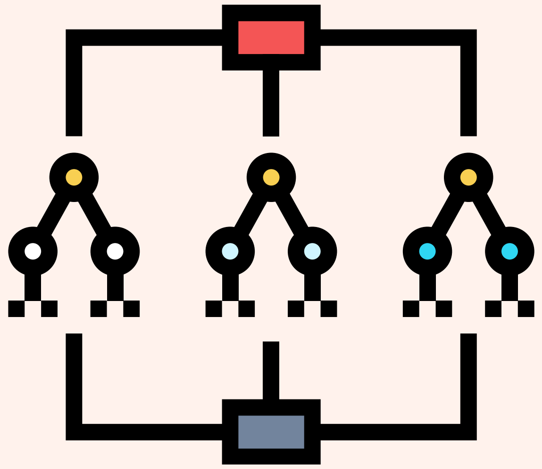


MODELS USED

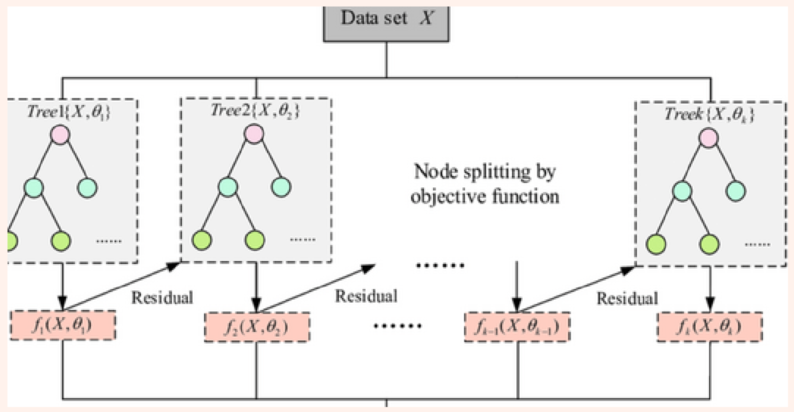




DECISION TREE



RANDOM FOREST



XGBOOST

DECISION TREE

```
from sklearn.model_selection import train_test_split
'''from sklearn.linear_model import LinearRegression'''
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score, confusion_matrix, classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
```

```
dectree = DecisionTreeClassifier(class_weight='balanced', random_state=42)
dectree.fit(X_train, y_train)

y_train_pred_dtc = dectree.predict(X_train)
y_test_pred_dtc = dectree.predict(X_test)
```

DECISION TREE

KEY INSIGHTS

```
Accuracy Score of Decision Tree Classifier : 0.7890827740449217
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.81      0.83     14261
           1       0.69      0.76      0.72      8089

    accuracy                           0.79     22350
   macro avg       0.77      0.78      0.78     22350
weighted avg       0.80      0.79      0.79     22350
```

RANDOM FOREST

KEY INSIGHTS

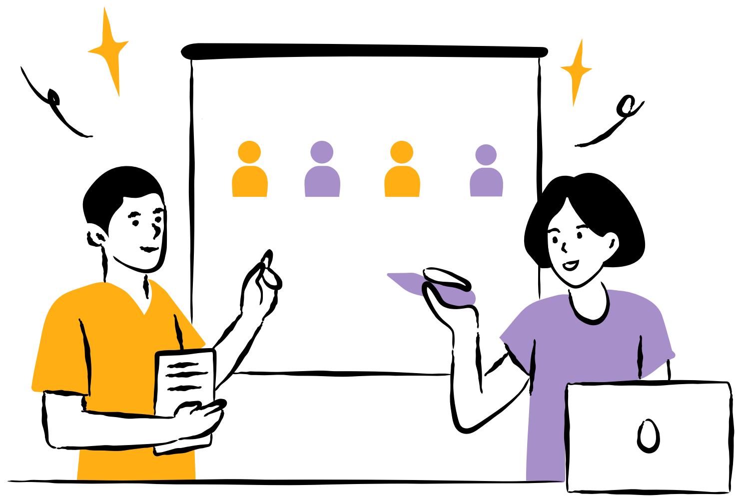
```python
from sklearn.ensemble import RandomForestClassifier

randforest = RandomForestClassifier(n_estimators=500,
    max_depth=15,
    min_samples_split=2,
    class_weight='balanced',
    random_state=42
)
randforest.fit(X_train, y_train)

y_train_pred_rfc = randforest.predict(X_train)
y_test_pred_rfc = randforest.predict(X_test)
```

AN ENSEMBLE METHOD THAT BUILDS MULTIPLE DECISION TREES AND AVERAGES THEIR PREDICTIONS TO IMPROVE ACCURACY AND REDUCE OVERFITTING.

RANDOM FOREST

KEY INSIGHTS

```
Accuracy Score of Random Forest Classifier : 0.7935570469798657
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.87      0.84     14261
           1       0.75      0.65      0.70      8089

    accuracy                           0.79     22350
   macro avg       0.78      0.76      0.77     22350
weighted avg       0.79      0.79      0.79     22350
```

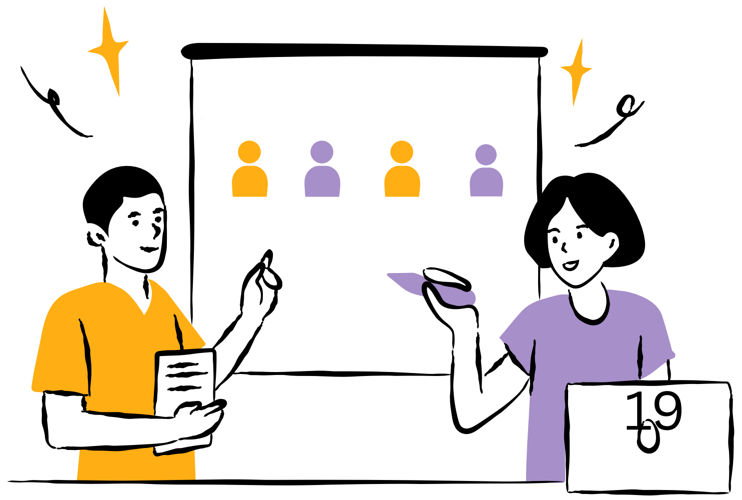


XGBOOST

KEY INSIGHTS

```python
from xgboost import XGBClassifier

xgb = XGBClassifier(
    n_estimators=500,
    max_depth=10,
    learning_rate=0.1,
    subsample=0.8,
    colsample_bytree=0.8,
    eval_metric='logloss',
    random_state=42
)
xgb.fit(X_train, y_train)

y_train_pred_xgb = xgb.predict(X_train)
y_test_pred_xgb = xgb.predict(X_test)
```

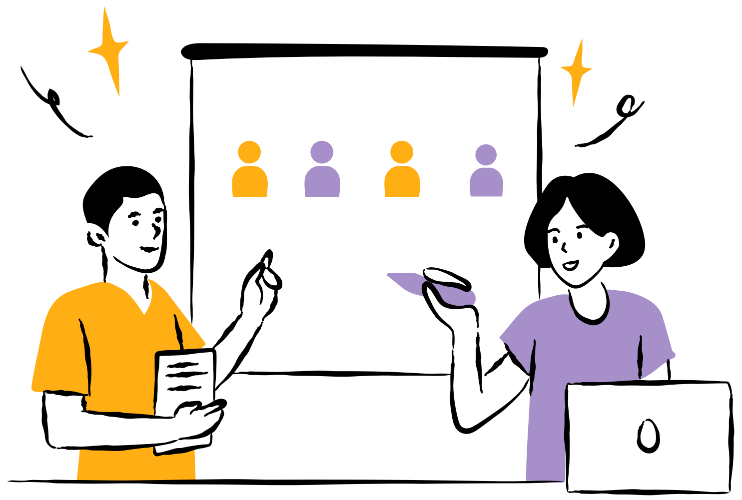
A POWERFUL BOOSTING ALGORITHM THAT BUILDS TREES SEQUENTIALLY AND OPTIMIZES ERRORS FROM PREVIOUS TREES. IT'S KNOWN FOR BEING FAST AND ACCURATE.



XGBOOST

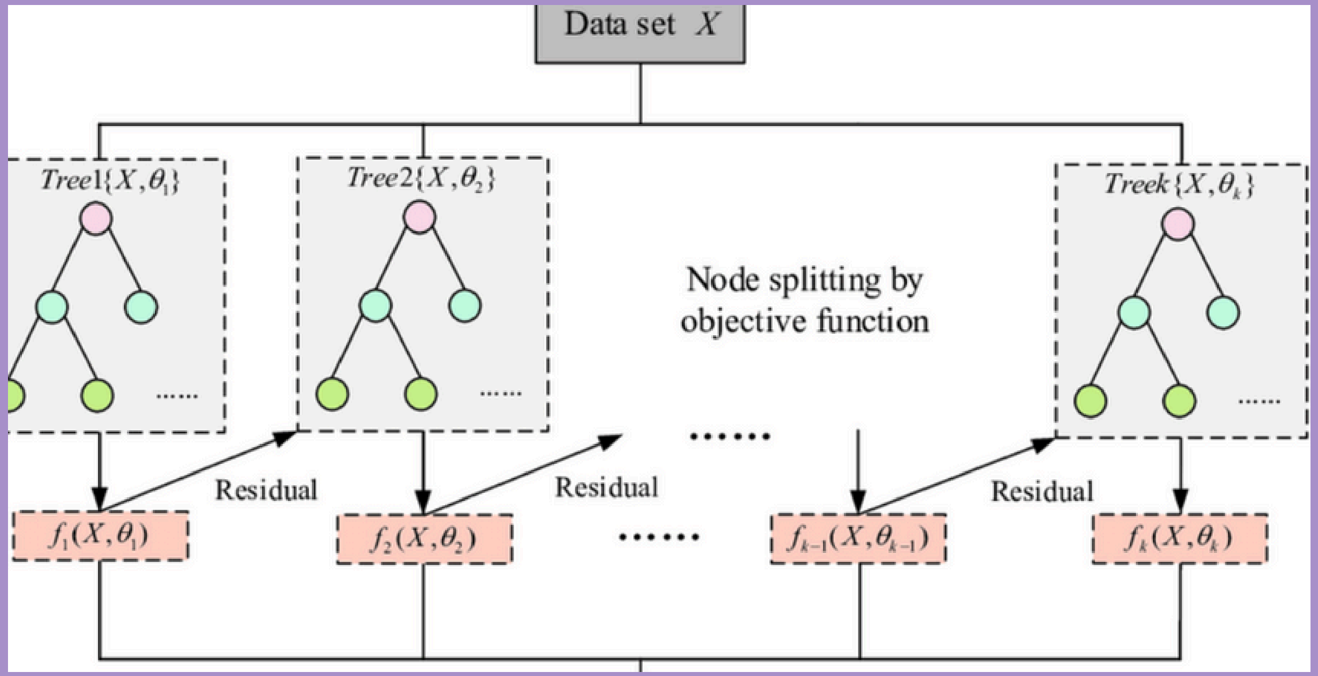
KEY INSIGHTS

```
Accuracy Score of XGB Classifier : 0.8170917225950783
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.92      0.87     14261
           1       0.82      0.64      0.72      8089

    accuracy                           0.82     22350
   macro avg       0.82      0.78      0.79     22350
weighted avg       0.82      0.82      0.81     22350
```



COMPARISON OF METRICS

MODEL	ACCURACY	PRECISION	RECALL	F1 SCORE
DECISION TREE	79	80	79	79
RANDOM FOREST	80	79	79	79
XGBOOST	82	82	82	81



XGBOOST

RECOMENDATION

Conclusion :

With reliable predictions, hotels can safely overbook slightly to compensate for expected cancellations

This maximises their revenue while still maintaining customer satisfaction

THANK YOU