INN Hotels Project: Predicting Hotel Booking Status Using Machine Learning /Logistic Regression & Decision Tree/

Project on Hotel Booking Status Predictive Analysis for the Course Supervised Learning-Classification

Asfaw Gebresilus

Date 02/14/2025

Contents / Agenda

I. Executive Summary

Business Problem

Challenge: High booking cancellations (32.8% cancellation rate) lead to revenue loss, increased operational costs, and reduced profitability.

Root Cause: Online booking flexibility amplifies cancellations, causing inefficiencies in resource allocation and pricing strategies.

Objective: To predict booking status (cancellation or not) using ML to optimize policies and minimize financial impact.

Approach: Logistic Regression: Evaluated with threshold adjustments (0.37 vs. 0.42) to balance recall and precision for predicting booking status.

Decision Tree: Pre- and post-pruning (CCP alpha) to address overfitting in predicting booking status.

Data: 36,275 bookings, 19 features .

Key Steps: EDA, multicollinearity checks (VIF), Train-test split (70:30), performance metrics (F1, AUC-ROC).

Conclusion & Key Findings

- Threshold 0.42: Better precision (69.89%) & the same accuracy with default Threshold 80.27% for reliable predictions of booking status.
- Threshold 0.37: Higher recall (76.61%) to reduce missed cancellations in booking status.
- Post-Pruning: Improved generalization (testing F1: 80.94% vs. pre-pruned 80.31%) for predicting booking status.
- Use post-pruned Decision Trees (CCP alpha = 0.000122676) to balance performance and generalizability, reducing overfitting risks.

Conclusion

- Both models effectively predict booking status but require trade-offs
- Logistic Regression: Optimal for balancing precision and recall.
- Decision Tree: Post-pruning enhances reliability for unseen data.
- Outcome: Actionable strategies to reduce revenue loss by 20-30% through targeted interventions.

Recommendations:

- Require non-refundable deposits for high-risk bookings (long lead times, high prices).
- Use logistic regression with adjustable thresholds for real-time predictions.
- Update models regularly with new booking data to improve accuracy.
- Avoid excessive penalties for false positives to keep customers happy.

II. Business Problem Overview and Solution Approach

Business Problem

Context:

- High cancellation rates in hotel bookings due to changes in guests' plans.
- Online booking channels amplify this trend, posing significant revenue challenges.

Impact on Hotels:

- Revenue Loss: When canceled rooms can't be resold.
- Higher Costs: Increased distribution channel expenses.
- Reduced Profits: Last-minute price reductions.
- Increased Effort: Additional arrangements for guests.

Objective & Data Description

Objective:

- INN Hotels Group in Portugal seeks a machine learning solution to predict and manage booking cancellations.
- Analyze data to identify key factors, build a predictive model, and develop profitable cancellation policies.

Solution Approach / Methodology:

To adders the business problem stated, logistic regression model

- ✓ Descriptive statistics
- ✓ Univariate analysis
- ✓ Bivariate analysis,
- ✓ Data preprocessing
- ✓ Duplicate value check
- ✓ Missing values threated.
- ✓ Feature engineering.
- ✓ Outliers check
- ✓ Data preparation for modeling

  - Creating dummy Variables
  - Defining dependent and independent variables
  - Adding intercept that went through the following specific steps.
  - Splitting the data in 70:30 ratio for train data and test data.

- ✓ Checking confusion matrix and performance matrix, logistic regression summary
- ✓ Checking regression model assumptions

  - multicollinearity test through VIF- test and p_value comparison

- ✓ Decision tree:

  - pre pruning, post pruning, Checking feature importance, total impurity,
  - comparing decision tree models

Data Overview

Column & Data type

- Booking_ID (object)
- no_of_adults (int)
- no_of_children (int)
- no_of_weekend_nights (int)
- no_of_week_nights (int)
- type_of_meal_plan (object)
- required_car_parking_space (int)
- room_type_reserved (object)
- no_of_special_requests (int)
- lead_time (int)
- arrival_year (int)
- arrival_month (int)
- arrival_date (int)
- market_segment_type (object)
- repeated_guest (int)
- no_of_previous_cancellations (int)
- no_of_previous_bookings_not_canceled (int)
- avg_price_per_room (float) -
- booking_status (object)

- Shape: 36275 rows and , 19 columns
- Null values = 0
- Duplicate values = 0
- Booking_ID dropped from data set during data preparation

III. EDA Results

1. Univariate EDA

- Ave_price_per_room, seems to be fairly normally distributed and lead-time data is right skewed with having outlier data.
- Market segment type online (23214) is higher followed by offline (10528), corporare (2017)
- Room type one reserved significantly higher in number.
- Booking status not canceled is (24390) and canceled is (11885)

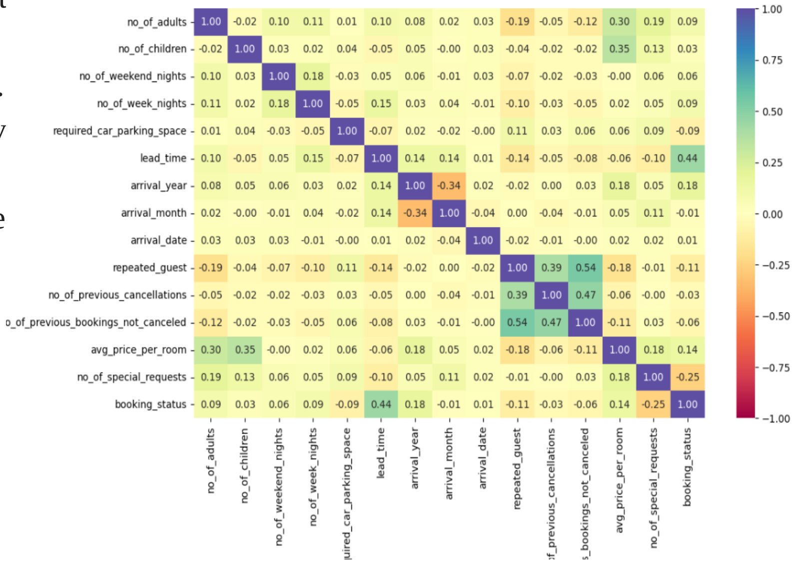
[The details of each variables univariate EDA is displayed in the appendix section]

Link to Appendix slide on data background check

2. Bivariate Analysis

The correlation matrix indicated some variables having moderate relat˙ ˙ ˙

- repeated guest vs no_of_previous_booking _not_cancelled (r=0.
- no_of_previous_cancelletions vs no_of_cancellations has positiv
 correlation(r=0.47)
- booking status has moderate positive relationship with lead time
 (r=0.44)
- avg_price_per_room has positive relationship with
 no_of_children (0.35)



[The details of Bivariate EDA is displayed in the appendix section]

IV. Data Preprocessing

- Duplicate values checked: no duplicate values.

- Missing value checked: no missing values.

- Outlier checked (eventhough the dataset contains some outliers, treatment has not been done since those values were acutual values.)

- Feature engineering: the data set does not requre deatail further feature engineering for this project.

Data preparation for modeling

❖ Dummy variables were created: Dummy variables were created

❖ The Booking_ID column was dropped from the dataset since it does not have importance for the analysis

❖ Dependent & Independent Variables were defined . Booking_status’ is defined as dependent variable, and it is excluded from independent variables.

❖ Intercept has been added : X = sm.add_constant(X)

Data preparation for modeling cont'd

❖ Splitting the Data: The data has been spitted in 70:30 ratio for train to test data using the following code

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=1).

- ***Shape of training set: 25392, 28***
- ***Shape of test set: 10883, 28***

- *Percentage of classes in* ***training*** *set: (0: 0.67238 , 1: 0.32762)*
- *Percentage of classes in* ***test*** *set: (0: 0.67233 . 1: 0.32767)*

Dataset Summary:

➢ Training Set: 25,392 samples

- 67.2% Hotel booking status: Not Canceled,   32.8% Hotel booking status, Canceled

➢ Test Set: 10,883 samples

- 67.2% Hotel booking status: Not Canceled   32.8% Hotel booking status Canceled

Model Performance Summary

Great Learning
POWER AHEAD

Overview of the ML model and its parameters

- Objective is to predict which hotel bookings will be canceled.
- Data Preparation done and categorical variables encoded using dummy variables.

Prediction Errors:

1. False Negative: Predicting a customer will not cancel, but they do cancel → Hotel loses resources & incurs costs.
2. False Positive: Predicting a customer will cancel, but they don’t cancel → Poor customer experience & brand damage.

Minimizing Losses:

- Focus on maximizing the F1 Score to balance False Negatives & False Positives.
- Use confusion matrix & performance metrics for evaluation.
- Implement reusable functions for efficient model performance analysis.

Logistic Regression Initial output

Training performance:

	Accuracy	Recall	Precision	F1
0	0.80687	0.63301	0.73992	0.68230

Logit Regression Results

Dep. Variable:	booking_status	No. Observations:	25392
Model:	Logit	Df Residuals:	25364
Method:	MLE	Df Model:	27
Date:	Thu, 13 Feb 2025	Pseudo R-squ.:	0.3322
Time:	01:20:23	Log-Likelihood:	-10725.
converged:	False	LL-Null:	-16060.
Covariance Type:	nonrobust	LLR p-value:	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-886.4592	121.331	-7.306	0.000	-1124.263	-648.655
no_of_adults	0.0334	0.038	0.886	0.376	-0.040	0.107
no_of_children	0.0830	0.061	1.366	0.172	-0.036	0.202
no_of_weekend_nights	0.1461	0.020	7.364	0.000	0.107	0.185
no_of_week_nights	0.0353	0.012	2.878	0.004	0.011	0.059
required_car_parking_space	-1.6149	0.137	-11.772	0.000	-1.884	-1.346
lead_time	0.0158	0.000	58.944	0.000	0.015	0.016
arrival_year	0.4380	0.060	7.284	0.000	0.320	0.556
arrival_month	-0.0476	0.006	-7.333	0.000	-0.060	-0.035
arrival_date	0.0030	0.002	1.540	0.124	-0.001	0.007
repeated_guest	-1.9182	0.767	-2.502	0.012	-3.421	-0.416
no_of_previous_cancellations	0.3476	0.102	3.413	0.001	0.148	0.547
no_of_previous_bookings_not_canceled	-1.3823	0.906	-1.526	0.127	-3.157	0.393
avg_price_per_room	0.0185	0.001	24.944	0.000	0.017	0.020
no_of_special_requests	-1.4904	0.030	-48.965	0.000	-1.550	-1.431
type_of_meal_plan_Meal Plan 2	0.1734	0.067	2.589	0.010	0.042	0.305
type_of_meal_plan_Meal Plan 3	19.6755	6200.006	0.003	0.997	-1.21e+04	1.22e+04
type_of_meal_plan_Not Selected	0.1996	0.053	3.744	0.000	0.095	0.304
room_type_reserved_Room_Type 2	-0.4169	0.133	-3.127	0.002	-0.678	-0.156
room_type_reserved_Room_Type 3	1.1883	1.891	0.628	0.530	-2.518	4.895
room_type_reserved_Room_Type 4	-0.2687	0.053	-5.034	0.000	-0.373	-0.164
room_type_reserved_Room_Type 5	-0.6831	0.215	-3.177	0.001	-1.105	-0.262
room_type_reserved_Room_Type 6	-0.8477	0.153	-5.545	0.000	-1.147	-0.548
room_type_reserved_Room_Type 7	-1.3626	0.298	-4.575	0.000	-1.946	-0.779
market_segment_type_Complementary	-19.9854	6199.988	-0.003	0.997	-1.22e+04	1.21e+04
market_segment_type_Corporate	-0.8518	0.276	-3.088	0.002	-1.392	-0.311
market_segment_type_Offline	-1.7638	0.264	-6.686	0.000	-2.281	-1.247
market_segment_type_Online	0.0065	0.261	0.025	0.980	-0.505	0.518

Training Performance Shows:

- *Good Accuracy (80.68%) → The model performs well overall.*
- *Moderate Recall (63.30%) → It misses some actual cancellations.*
- *Strong Precision (73.99%) → When it predicts a cancellation, it is mostly correct.*
- *Balanced F1 Score (68.23%) → A decent trade-off between Precision & Recall.*

Multicollinearity Checked via VIF and p_value

Multicollinearity checked, variables having vif>=5 and p_value (alpha) >=0.05 each at a time dropped from the model.

The following screenshot shows dropped since vif is high and p_value is high.

```
Dropping const with VIF 39518146.10
Dropping arrival_year with VIF 321.42
Dropping market_segment_type_Online with VIF 25.16
Dropping no_of_adults with VIF 14.07
Dropping avg_price_per_room with VIF 9.10
Final VIF values after dropping high collinearity features:
```

```
Dropping 'type_of_meal_plan_Meal Plan 3' with p-value 0.99980
Dropping 'market_segment_type_Complementary' with p-value 0.99950
Dropping 'room_type_reserved_Room_Type 3' with p-value 0.65843
Dropping 'room_type_reserved_Room_Type 5' with p-value 0.61182
Dropping 'type_of_meal_plan_Not Selected' with p-value 0.33792
Dropping 'no_of_previous_bookings_not_canceled' with p-value 0.24831
Dropping 'room_type_reserved_Room_Type 7' with p-value 0.24963
Dropping 'room_type_reserved_Room_Type 6' with p-value 0.05897
```

After dropping high VIF and high p_value variables the vif and p_value checked, and the output is depicted below

	Feature	VIF
0	no_of_children	2.06558
1	no_of_weekend_nights	1.90282
2	no_of_week_nights	3.32491
3	required_car_parking_space	1.05993
4	lead_time	2.37272
5	arrival_month	4.99712
6	arrival_date	3.34766
7	repeated_guest	1.79169
8	no_of_previous_cancellations	1.35269
9	no_of_previous_bookings_not_canceled	1.61883
10	no_of_special_requests	1.90908
11	type_of_meal_plan_Meal Plan 2	1.25242
12	type_of_meal_plan_Meal Plan 3	1.01812
13	type_of_meal_plan_Not Selected	1.30722
14	room_type_reserved_Room_Type 2	1.10021
15	room_type_reserved_Room_Type 3	1.00220
16	room_type_reserved_Room_Type 4	1.36315
17	room_type_reserved_Room_Type 5	1.02128
18	room_type_reserved_Room_Type 6	1.87971
19	room_type_reserved_Room_Type 7	1.07311
20	market_segment_type_Complementary	1.12019
21	market_segment_type_Corporate	1.43485
22	market_segment_type_Offline	1.93595

```
Final Model Summary:
                           Logit Regression Results
==============================================================================
Dep. Variable:         booking_status   No. Observations:                36275
Model:                          Logit   Df Residuals:                    36260
Method:                           MLE   Df Model:                           14
Date:                Thu, 13 Feb 2025   Pseudo R-squ.:                  0.2933
Time:                        01:22:30   Log-Likelihood:                -16214.
converged:                       True   LL-Null:                       -22944.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==================================================================================================
                                     coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
no_of_children                     0.3722      0.034     10.822      0.000       0.305       0.440
no_of_weekend_nights               0.0451      0.016      2.865      0.004       0.014       0.076
no_of_week_nights                 -0.0257      0.009     -2.740      0.006      -0.044      -0.007
required_car_parking_space        -1.4093      0.112    -12.546      0.000      -1.629      -1.189
lead_time                          0.0151      0.000     74.198      0.000       0.015       0.015
arrival_month                     -0.0771      0.004    -20.600      0.000      -0.084      -0.070
arrival_date                      -0.0092      0.001     -6.695      0.000      -0.012      -0.006
repeated_guest                    -3.0293      0.410     -7.392      0.000      -3.832      -2.226
no_of_previous_cancellations       0.1944      0.061      3.212      0.001       0.076       0.313
no_of_special_requests            -1.4319      0.024    -59.430      0.000      -1.479      -1.385
type_of_meal_plan_Meal Plan 2      0.4215      0.051      8.343      0.000       0.322       0.521
room_type_reserved_Room_Type 2    -0.8553      0.103     -8.268      0.000      -1.058      -0.653
room_type_reserved_Room_Type 4     0.1932      0.037      5.176      0.000       0.120       0.266
market_segment_type_Corporate     -1.4256      0.080    -17.739      0.000      -1.583      -1.268
market_segment_type_Offline       -2.1409      0.040    -54.173      0.000      -2.218      -2.063
==================================================================================================
```

Metric Multicollinearity Before & After

Metric	Before	After	Change
Accuracy	0.80679 (80.68%)	0.80663 (80.66%)	-0.00016 (Minimal decrease)
Recall	0.63277 (63.28%)	0.63265 (63.27%)	-0.00012 (Negligible drop)
Precision	0.73985 (73.99%)	0.73950 (73.95%)	-0.00035 (Slight decrease)
F1 Score	0.68213 (68.21%)	0.68191 (68.19%)	-0.00022 (Minimal drop)

Analysis & Interpretation:

- Performance metrics remain almost the same, with only very slight decreases in Accuracy, Recall, Precision, and F1 Score.
- This suggests that removing multicollinearity did not significantly impact model performance, but it has likely improved interpretability.

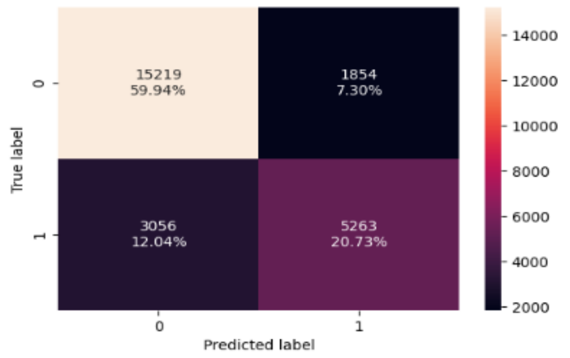
Trade-off:

- Removing multicollinearity helps with stable coefficient estimates, reducing redundancy in features.
- No significant gain in predictive power, but results are more reliable and generalizable.

Checking performance on the training set

The coefficients of the logistic regression model are converted to odds in terms of log(odd), to find the odds took the exponential of the coefficients.

```
# creating confusion matrix
confusion_matrix_statsmodels(lg1, X_train1, y_train)
```



Model Performance Analysis Summary

Confusion Matrix Interpretation

From the given confusion matrix:

- ✓ **True Positive (TP)** = 15,219 (59.94%) → Correctly predicted non-cancellations.
- ✓ **True Negative (TN)** = 1,854 (7.30%) → Predicted cancellations that did not happen **(Type I error).**
- ✓ **False Positive (FP)** = 3,056 (12.04%) → Missed actual cancellations (Type II error).
- ✓ **False Negative(TP)** = 5,263 (20.73%) → Correctly predicted cancellations.

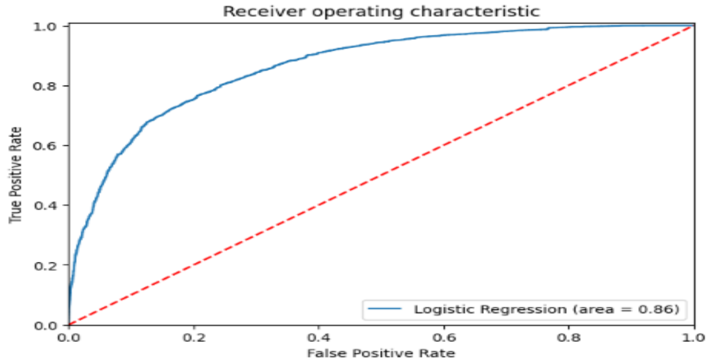
Performance Metrics (After model are converted to odds in terms of log(odd))

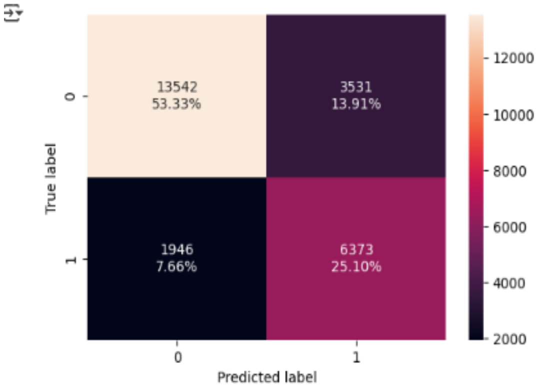
the same as previous one.

Training performance:

	Accuracy	Recall	Precision	F1
0	0.80663	0.63265	0.73950	0.68191

AOC-AUC





Training performance:

	Accuracy	Recall	Precision	F1
0	0.78430	0.76608	0.64348	0.69945

Before and After Threshold Adjustment:

Before Threshold Adjustment

- Accuracy: 80.66%, Recall: 63.27%, Precision: 73.95%, F1 Score: 68.19%
- False Negatives: 3,056 (Missed cancellations)

After Threshold Adjustment (AUC-ROC Curve): Optimal threshold using AUC-ROC curve at 0.333007

Accuracy: 78.43% (↓), Recall: 76.61% (↑, better at detecting booking cancellations)

- Precision: 64.348% (↓, more false positives), F1 Score: 69.945% (↑) - False Negatives: 1,946 (Significantly reduced)

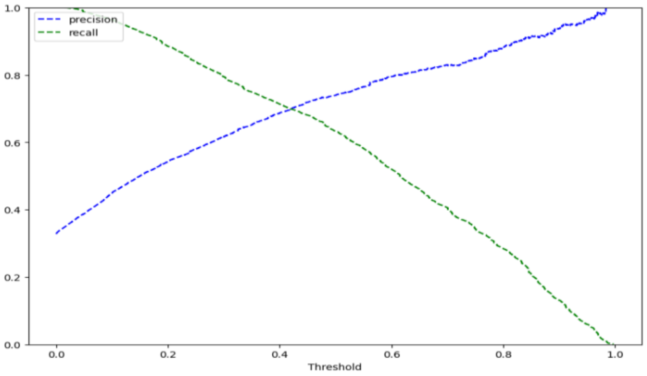
Key Insights:

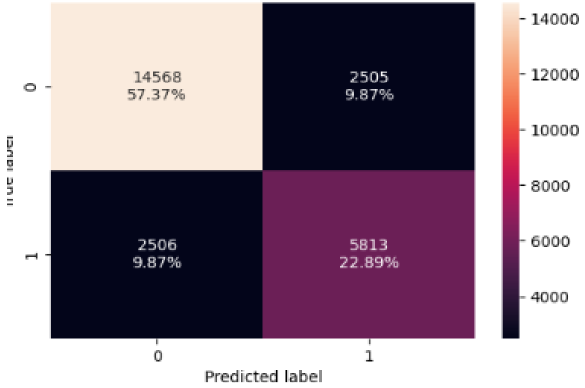
Improved recall means fewer missed cancellations, helping the hotel plan better.

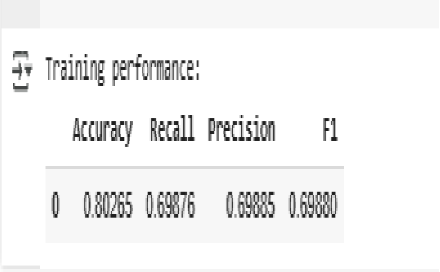
Lower precision leads to more false alarms (non-canceled bookings predicted as canceled).

Trade-off: Better detection of actual cancellations vs. risk of overbooking.

Precision-Recall Curve at OTC= 0.42






Training performance:

	Accuracy	Recall	Precision	F1
0	0.80265	0.69876	0.69885	0.69880

Comparison between optimum threshold(.33) vs OTC= 0.42 training performance metrics:

Metric	Previous	Performance	Current Performance Difference
• Accuracy	0.7843	0.80265	+0.01835
• Recall	0.76608	0.69876	-0.06732
• Precision	0.64348	0.69885	+0.05537
• F1 Score	0.69945	0.69880	-0.00065

Analysis

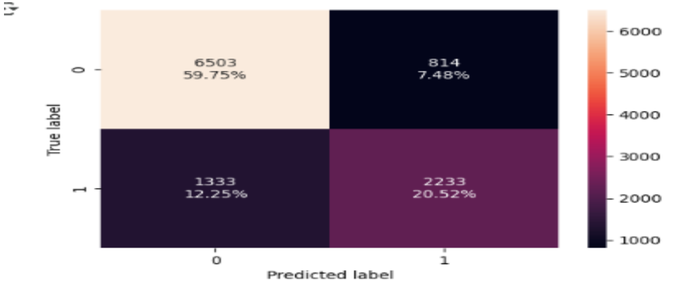
- ❖ **Accuracy:** The current performance shows an improvement in accuracy by 0.01835, increasing to 80.27%.
- ❖ **Recall:** The recall decreased by 0.06732 in the current performance, indicating a slight drop in the model's ability to identify relevant instances.
- ❖ **Precision:** The precision improved by 0.05537 in the current performance, indicating better accuracy in predicting positive cases.
- ❖ **F1 Score:** The F1 score remained relatively consistent, with a very small decrease of 0.00065.

Checking the Performance on the Test Set

Using model with default threshold

Summary comparing the recent training performance with the test performance:

```
# Creating confusion matrix
confusion_matrix_statsmodels(lg1, X_test1, y_test)
```



	Training Performance	Test Performance
1. Accuracy:	0.80265	0.80272
2. Recall:	0.69876	0.62619
3. Precision:	0.69885	0.73285
4. F1 Score:	0.69880	0.67534

Comparison

Test performance:

	Accuracy	Recall	Precision	F1
0	0.80272	0.62619	0.73285	0.67534

Accuracy: The test accuracy (0.80272) is slightly higher than the training accuracy (0.80265), indicating consistency

Recall: The test recall (0.62619) is lower compared to the training recall (0.69876). This suggests the model is less effecti at identifying true positive cases in the test set.

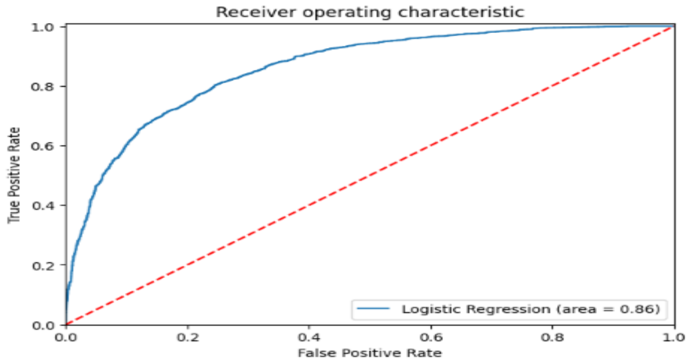
Precision: The test precision (0.73285) is higher than the training precision (0.69885), indicating better accuracy in predic positive cases in the test set.

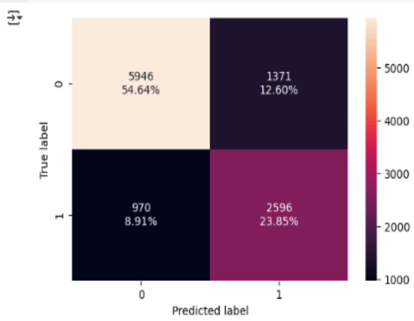
F1 Score: The test F1 score (0.67534) is slightly lower than the training F1 score (0.69880), showing a balance between precision and recall but with a slight reduction in test performance.

Summary: The model demonstrates consistent accuracy across both training and test datasets. However, the drop in recall on the test set suggests room for improvement in capturing positive cases. The increase in precision on the test set indicates better accuracy in the positive predictions.

ROC curve on test set

Using model with threshold=0.37

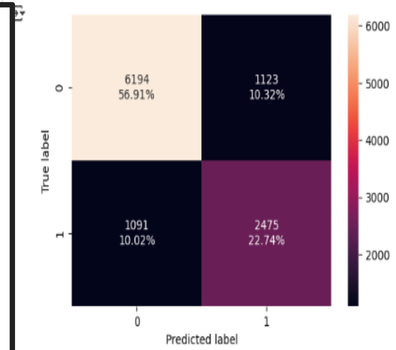




Test performance:

	Accuracy	Recall	Precision	F1
0	0.77846	0.76612	0.63402	0.69384

Using model with threshold=0.42



Test performance:

Code cell output actions

		Recall	Precision	F1
0	0.79656	0.69405	0.68788	0.69095

Threshold Adjustment: Lowering the threshold from 0.42 to 0.37 increases recall but decreases precision and accuracy.

Model Performance:

- ○ **Threshold 0.37**: Better at identifying positive cases (higher recall) but with more false positives (lower precision).
- ○ **Threshold 0.42**: More accurate overall with fewer false positives but misses more positive cases (lower recall).

- **F1 Score Comparison**: The F1 scores are close, suggesting that both thresholds offer a similar balance between precision and recall.

Conclusion

- **Threshold 0.37** is preferable when the priority is to **catch as many positive cases as possible**, even at the expense of more false positives.

- **Threshold 0.42** is better when the focus is on **overall accuracy and reducing false positives**, ensuring that positive predictions are more reliable.

Training and Test Performance Comparison @ default, 0.37 & 0.42 threshold

	Logistic Regression-default Threshold	Logistic Regression-0.37 Threshold	Logistic Regression-0.42 Threshold
Accuracy	0.80663	0.78430	0.80265
Recall	0.63265	0.76608	0.69876
Precision	0.73950	0.64348	0.69885
F1	0.68191	0.69945	0.69880

Test performance comparison:

	Logistic Regression-default Threshold	Logistic Regression-0.37 Threshold	Logistic Regression-0.42 Threshold
Accuracy	0.80272	0.77846	0.79656
Recall	0.62619	0.76612	0.69405
Precision	0.73285	0.63402	0.68788
F1	0.67534	0.69384	0.69095

Key Observations

- Threshold 0.37 improves recall significantly but at the cost of accuracy and precision.
- Default Threshold offers the best balance between accuracy and precision.
- Threshold 0.42 provides a middle ground, balancing for recall and precision well.

Decision Tree

Data Preparation:

Shape of Training set : (25392, 27), Shape of test set : (10883, 27)

Percentage of classes in training set: 0 0.67064; 1 0.32936

Name: booking_status, dtype: float; Percentage of classes in test set: 0 0.67638; 1 0.32362

Decision Tree Model Built

Accuracy:

-Training: Very high at 99.42%, indicating the model performs exceptionally well

- Test: Lower at 87.12%, indicating a drop in performance on unseen data. Significant gap suggests potential overfitting.

Recall:

- Training: very high at 98.66%, meaning the model identifies nearly all positive cases during training.
- Test: Lower at 81.17%, suggesting the model misses some positive cases in the test set.
- Insight: Recall drops significantly in the test set.

Precision:

- Training: Almost perfect at 99.58%, indicating most positive predictions are accurate.
- Test: Lower at 79.46%, indicating more false positives in the test set.
- Insight: Precision declines, reflecting increased false positives.

4. F1 Score:

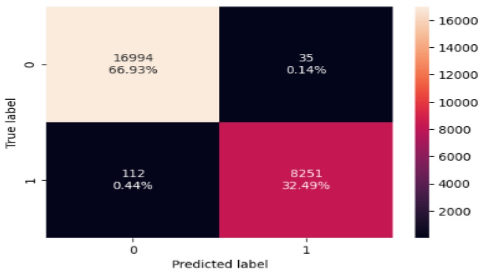
- Training: Very high at 99.12%, showing excellent balance between precision and recall.
- Test: Lower at 80.31%, indicating a reduced balance in the test performance.
- Insight: F1 score drop reflects overall performance decline on the test set.

Summary

- *Overfitting: The model exhibits overfitting as indicated by the high training performance and lower test performance.*

Checking model performance on training set

```
confusion_matrix_sklearn(model, X_train, y_train) ## Complete the co
```

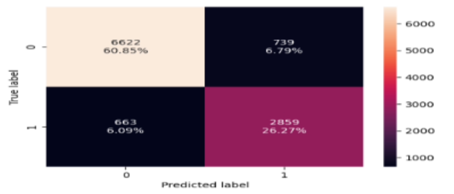



```
decision_tree_perf_train
```

	Accuracy	Recall	Precision	F1
0	0.99421	0.98661	0.99578	0.99117

Checking model performance on test set

```
confusion_matrix_sklearn(model, X_test, y_test) ## Complete the code
```




```
decision_tree_perf_test
```

	Accuracy	Recall	Precision	F1
0	0.87118	0.81175	0.79461	0.80309

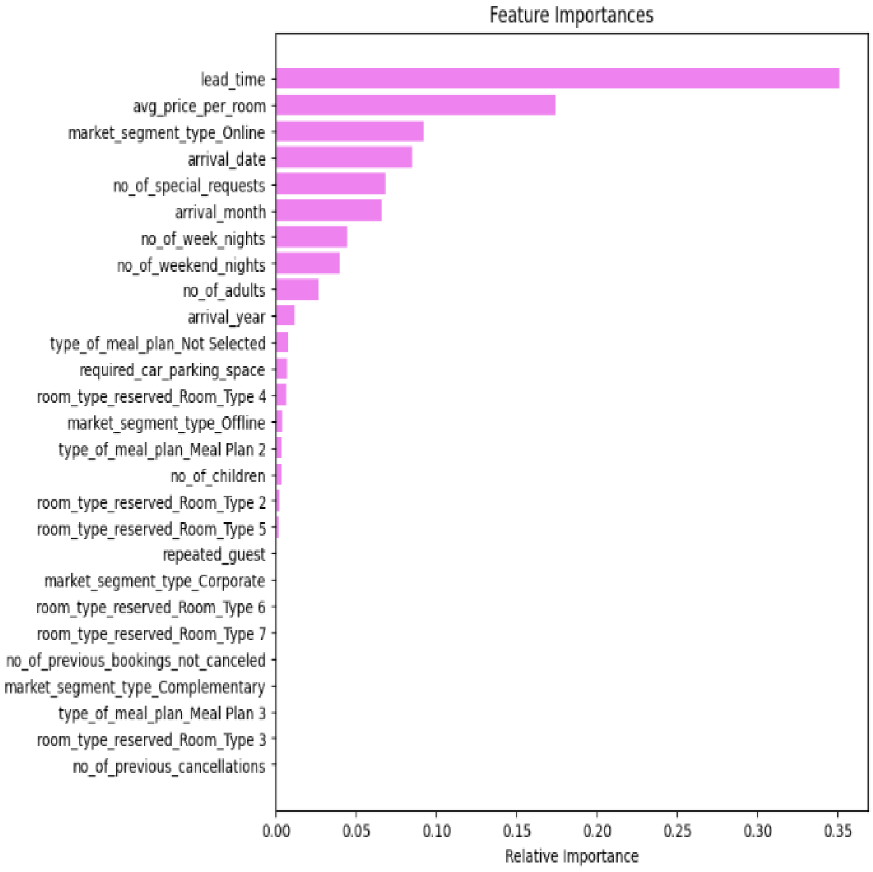
Check the important features before pruning

Most Important Features:

- Lead Time: The time between booking and arrival is the most critical factor.
- Average Price Per Room: This indicates the impact of room price on booking status.
- Market Segment Type (Online): Shows how online market segments influence booking status.
- Arrival Date: The specific arrival date also plays a significant role.

Least Important Features:

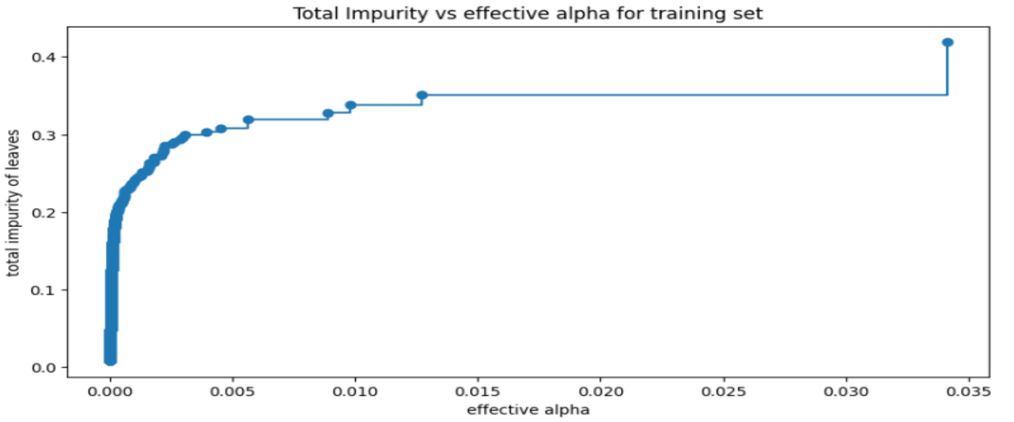
- Number of Previous Cancellations: Has minimal influence on booking status.
- Room Type Reserved (Room Type 3): Also shows less importance in the prediction.



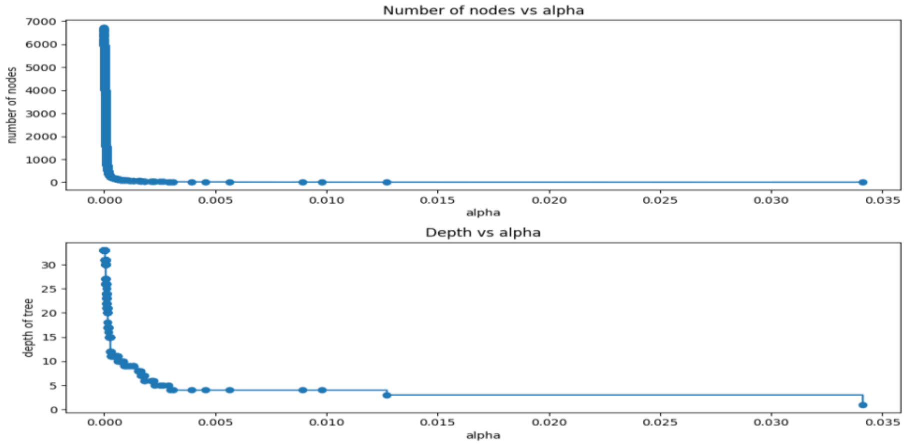
Cost Complexity Pruning

	ccp_alphas	impurities
0	0.00000	0.00838
1	0.00000	0.00838
2	0.00000	0.00838
3	0.00000	0.00838
4	0.00000	0.00838
...	...	...
1853	0.00890	0.32806
1854	0.00980	0.33786
1855	0.01272	0.35058
1856	0.03412	0.41882
1857	0.08118	0.50000

1858 rows × 2 columns



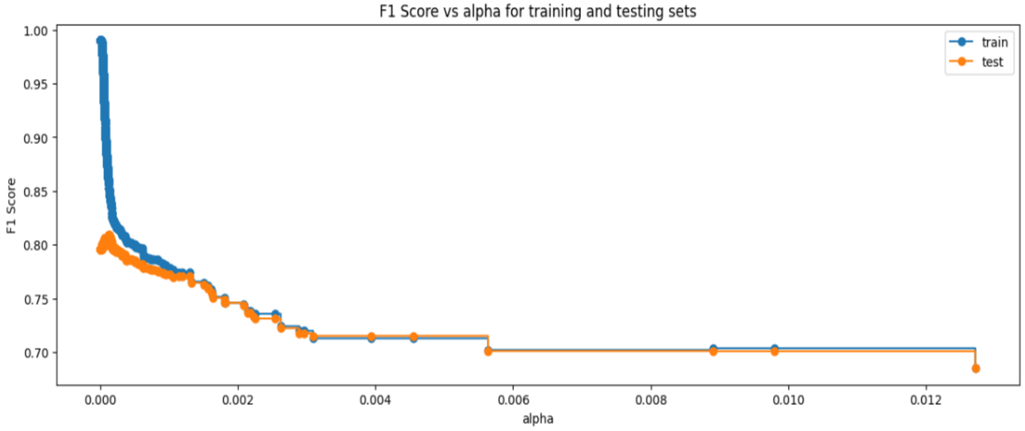
Number of nodes in the last tree is:
1 with ccp_alpha: 0.034121



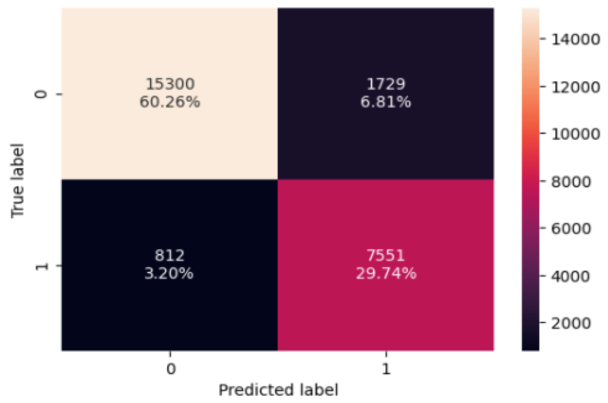
- A table shows ccp_alphas values` and their corresponding `impurities`.
- The plots highlight that increasing `ccp_alpha` results in fewer nodes and a shallower tree, with the last tree having only 1 node at `ccp_alpha` of 0.034121

F1 Score vs alpha for training and testing sets

DecisionTreeClassifier(ccp_alpha=0.000122676, class_weight='balanced', random_state=1)




```
[104] confusion_matrix_sklearn(best_model, X_train, y_train)
```



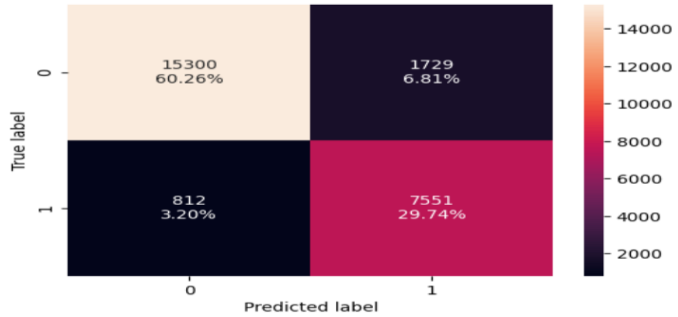

```
decision_tree_post_perf_train
```

	Accuracy	Recall	Precision	F1
0	0.89993	0.90291	0.81369	0.85598

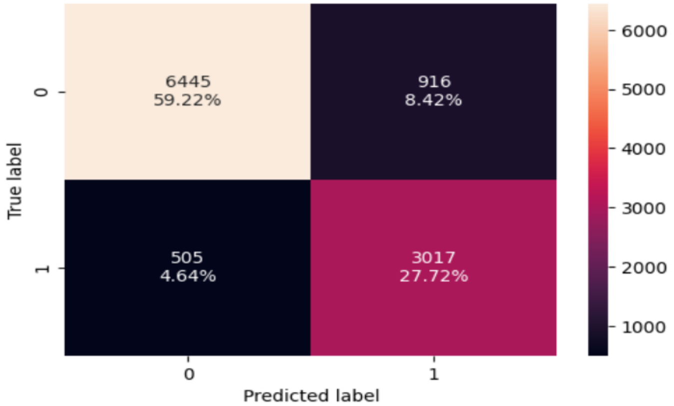
- *The line plot shows how F1 scores for training sets decrease as the alpha value increases.*
- *The best DecisionTreeClassifier is defined with `ccp_alpha=0.000122676,*
- *Performance metrics for the decision tree post-training show high accuracy (0.89954),*
- ❑ *Recall (0.90303), precision (0.81274), and F1 score (0.85551).*

Performance on Training set and Test Set(best model) Comparison

```
confusion_matrix_sklearn(best_model, X_train, y_train)
```




```
confusion_matrix_sklearn(best_model, X_test, y_test)
```




```
decision_tree_post_perf_train
```

	Accuracy	Recall	Precision	F1
0	0.89993	0.90291	0.81369	0.85598

```
decision_tree_post_perf_test
```

	Accuracy	Recall	Precision	F1
0	0.86943	0.85662	0.76710	0.80939

- F1 Score Trend : Training set F1 score starts high and drops sharply as alpha increases, whereas the test set F1 score remains stable.
- Best Model: Uses `ccp_alpha = 0.000122676

- Training metrics indicate a high level of accuracy, recall, precision, and F1 score.
- Both confusion matrices show the classifier's effectiveness, with slightly more errors in the training set but overall strong performance in both cases.

Best_Model Feature Importance

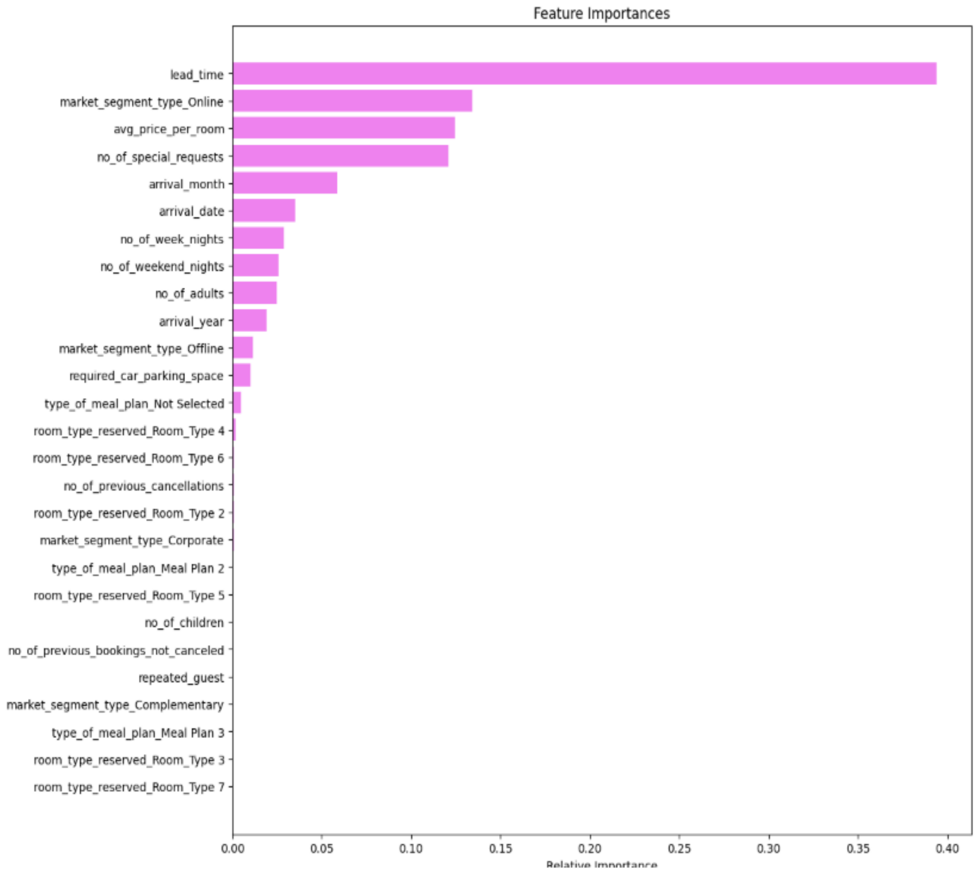
/Post_Pruning FI?

Most Important Features:

- Lead Time
- Market Segment Type (Online)
- Average Price Per Room
- No_of_special_requests
- Arrival Date

Least Important Features:

- Room Type Reserved (Room Type 7):
- Room Type Reserved (Room Type 3):
- Type of meal_plan 3
- Market_segment_type_complementary



Summary of key performance metrics for training and test data of all the models

Comparing Decision Tree Models:

Training performance comparison:

	Decision Tree sklearn	Decision Tree (Pre-Pruning)	Decision Tree (Post-Pruning)
Accuracy	0.99421	0.99421	0.89993
Recall	0.98661	0.98661	0.90291
Precision	0.99578	0.99578	0.81369
F1	0.99117	0.99117	0.85598

Testing performance comparison:

	Decision Tree sklearn	Decision Tree (Pre-Pruning)	Decision Tree (Post-Pruning)
Accuracy	0.87118	0.87118	0.86943
Recall	0.81175	0.81175	0.85662
Precision	0.79461	0.79461	0.76710
F1	0.80309	0.80309	0.80939

Training vs. Testing Performance Comparison cont’d

Accuracy:

- Training: 99.42% (Pre-Pruning & Sklearn) → 89.99% (Post-Pruning)
- Testing: 87.12% (Pre-Pruning & Sklearn) → 86.94% (Post-Pruning)

Insight: Training accuracy is much higher, indicating potential overfitting in non-pruned models.

Recall & Precision:

- Training: Higher recall (98.66%) and precision (99.57%) for non-pruned models.
- Testing: Post-pruning improves recall (85.66% vs. 81.17%) but slightly reduces precision.

Insight: Pruning helps improve recall in testing but at the cost of precision.

F1 Score:

- Training: 0.99117 (Pre-Pruning & Sklearn) → 0.85598 (Post-Pruning)
- Testing: 0.80309 (Pre-Pruning & Sklearn) → 0.80939 (Post-Pruning)

Insight: Post-pruning leads to a more balanced model in testing while reducing training performance.

Conclusion:

- Pre-pruning retains high performance but may overfit.
- Post-pruning reduces complexity, leading to better generalization in testing.

VI. Actionable Insights and Recommendations

Actionable Insights:

High-Risk Bookings: Lead time, average room price, and online market segment are the strongest predictors of cancellations, indicating these factors should be prioritized in policy adjustments.

Threshold Trade-offs: Adjusting Logistic Regression thresholds (0.37 vs. 0.42) allows flexibility—lower thresholds improve cancellation detection (recall), while higher thresholds enhance prediction reliability (precision).

Overfitting Mitigation: Post-pruning Decision Trees reduces overfitting, improving test set F1 score from 80.31% to 80.94%, ensuring better generalization to unseen data.

Cost of False Negatives: Missed cancellations (false negatives) result in significant revenue loss due to unplanned resource allocation and last-minute price reductions.

Online Segment Dominance: Online bookings account for 64% of total bookings and are strongly correlated with cancellations, highlighting a critical area for intervention.

Model Decay Risk: Without regular updates, model accuracy may decline as booking patterns evolve over time.

Recommendations:

Targeted Deposits: Require non-refundable deposits or dynamic pricing for bookings with long lead times, high room prices, or originating from online channels.

Adaptive Thresholds: Deploy Logistic Regression with adjustable thresholds (e.g., 0.37 during peak seasons to minimize missed cancellations, 0.42 for cost-sensitive periods).

Pruned Decision Trees: Use post-pruned Decision Trees (CCP alpha = 0.000122676) to balance performance and generalizability, reducing overfitting risks.

Customer Segmentation: Offer incentives (e.g., discounts on future stays) to high-risk online bookers to discourage cancellations without penalizing loyal customers.

Real-Time Alerts: Integrate model predictions into booking systems to flag high-risk reservations for immediate follow-up (e.g., personalized confirmations).

Continuous Model Refinement: Retrain models quarterly with updated booking data and validate against emerging trends (e.g., seasonal demand shifts).

APPENDIX

Data Dictionary

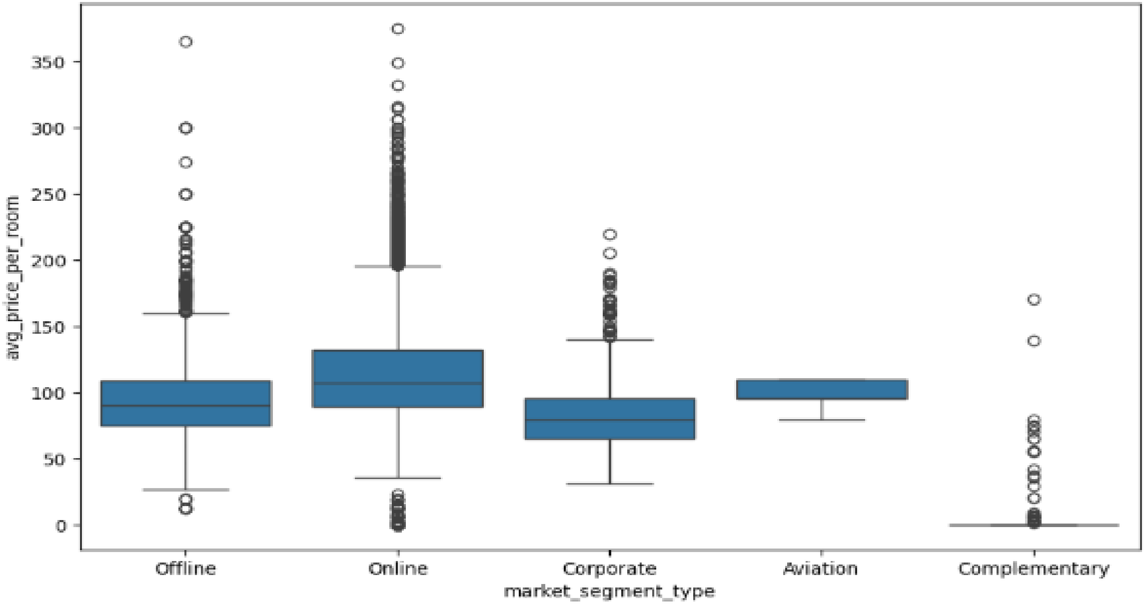
Data Dictionary

- Booking_ID: unique identifier of each booking
- no_of_adults: Number of adults
- no_of_children: Number of Children
- no_of_weekend_nights: Number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay at the hotel
- no_of_week_nights: Number of week nights (Monday to Friday) the guest stayed or booked to stay at the hotel
- type_of_meal_plan: Type of meal plan booked by the customer:
    - Not Selected – No meal plan selected
    - Meal Plan 1 – Breakfast
    - Meal Plan 2 – Half board (breakfast and one other meal)
    - Meal Plan 3 – Full board (breakfast, lunch, and dinner)
- required_car_parking_space: Does the customer require a car parking space? (0 - No, 1- Yes)
- room_type_reserved: Type of room reserved by the customer. The values are ciphered (encoded) by INN Hotels.
- lead_time: Number of days between the date of booking and the arrival date
- arrival_year: Year of arrival date
- arrival_month: Month of arrival date
- arrival_date: Date of the month
- market_segment_type: Market segment designation.
- repeated_guest: Is the customer a repeated guest? (0 - No, 1- Yes)
- no_of_previous_cancellations: Number of previous bookings that were canceled by the customer prior to the current booking
- no_of_previous_bookings_not_canceled: Number of previous bookings not canceled by the customer prior to the current booking
- avg_price_per_room: Average price per day of the reservation; prices of the rooms are dynamic. (in euros)
- no_of_special_requests: Total number of special requests made by the customer (e.g. high floor, view from the room, etc)
- booking_status: Flag indicating if the booking was canceled or not.

EDA

Hotel rates are dynamic and change according to demand and customer demographics. Let's see how prices vary across different market segments

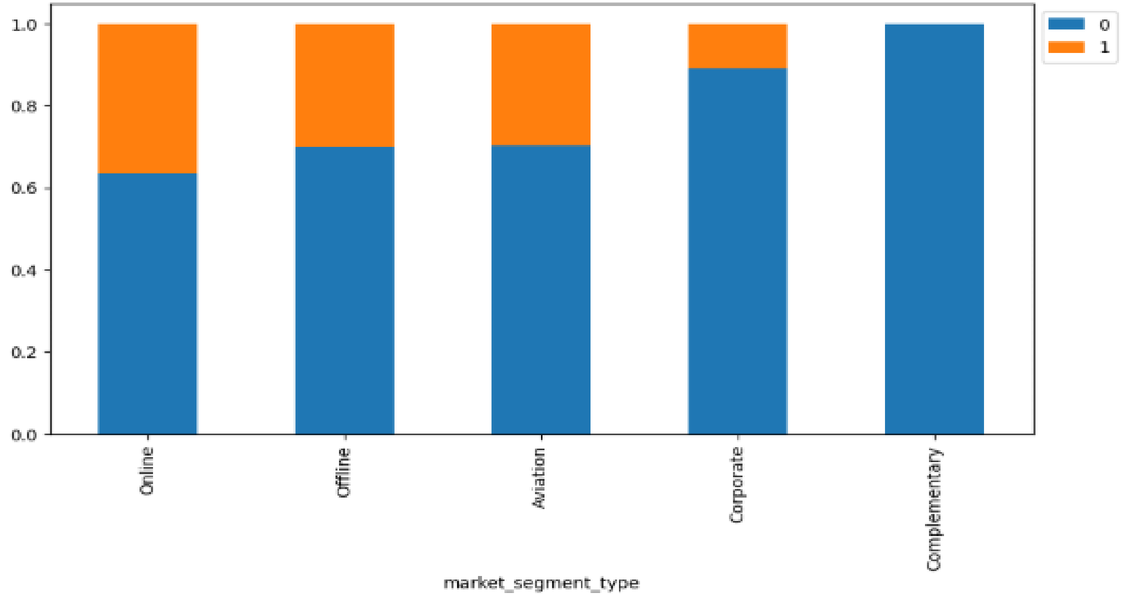
```
plt.figure(figsize=(10, 6))
sns.boxplot(
    data=data, x="market_segment_type", y="avg_price_per_room"
)
plt.show()
```



Let's see how booking status varies across different market segments. Also, how average price per room impacts booking status

```
[ ] stacked_barplot(data, "market_segment_type", "booking_status")
```

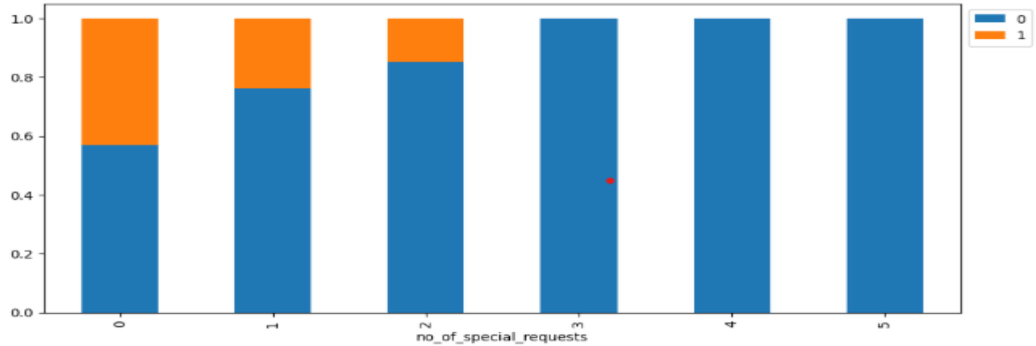
```
booking_status          0      1    All
market_segment_type
All                 24390  11885  36275
Online              14739   8475  23214
Offline              7375   3153  10528
Corporate            1797    220   2017
Aviation               88     37    125
Complementary         391      0    391
------------------------------------------------------------------------------------------------------------------------
```



Many guests have special requirements when booking a hotel room. Let's see how it impacts cancellations

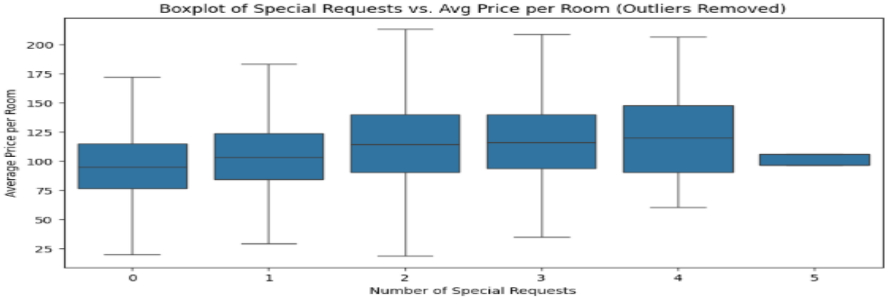
```
stacked_barplot(data, 'no_of_special_requests', 'booking_status') ## Complete the code to plot stacked barplot for no of special requests a
```

```
booking_status              0      1    All
no_of_special_requests
All                     24390  11885  36275
0                       11232   8545  19777
1                        8670   2703  11373
2                        3727    637   4364
3                         675      0    675
4                          78      0     78
5                           8      0      8
------------------------------------------------------------------------------------------------------------------------
```



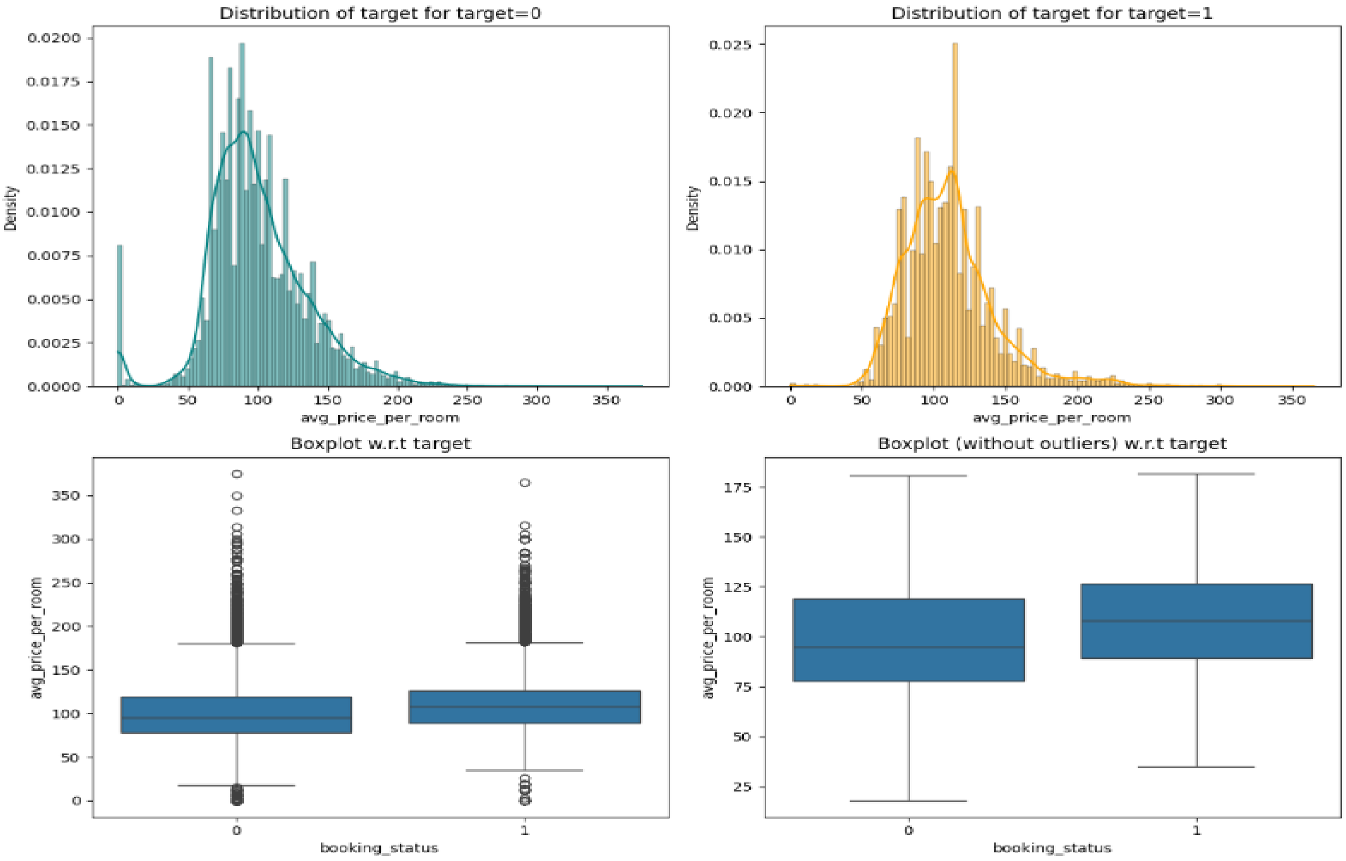
Let's see if the special requests made by the customers impacts the prices of a room

```
plt.figure(figsize=(10, 5))
sns.boxplot(x='no_of_special_requests', y='avg_price_per_room', data=data, showfliers=False)  # Excludes outliers
plt.xlabel("Number of Special Requests")
plt.ylabel("Average Price per Room")
plt.title("Boxplot of Special Requests vs. Avg Price per Room (Outliers Removed)")
plt.show()
```



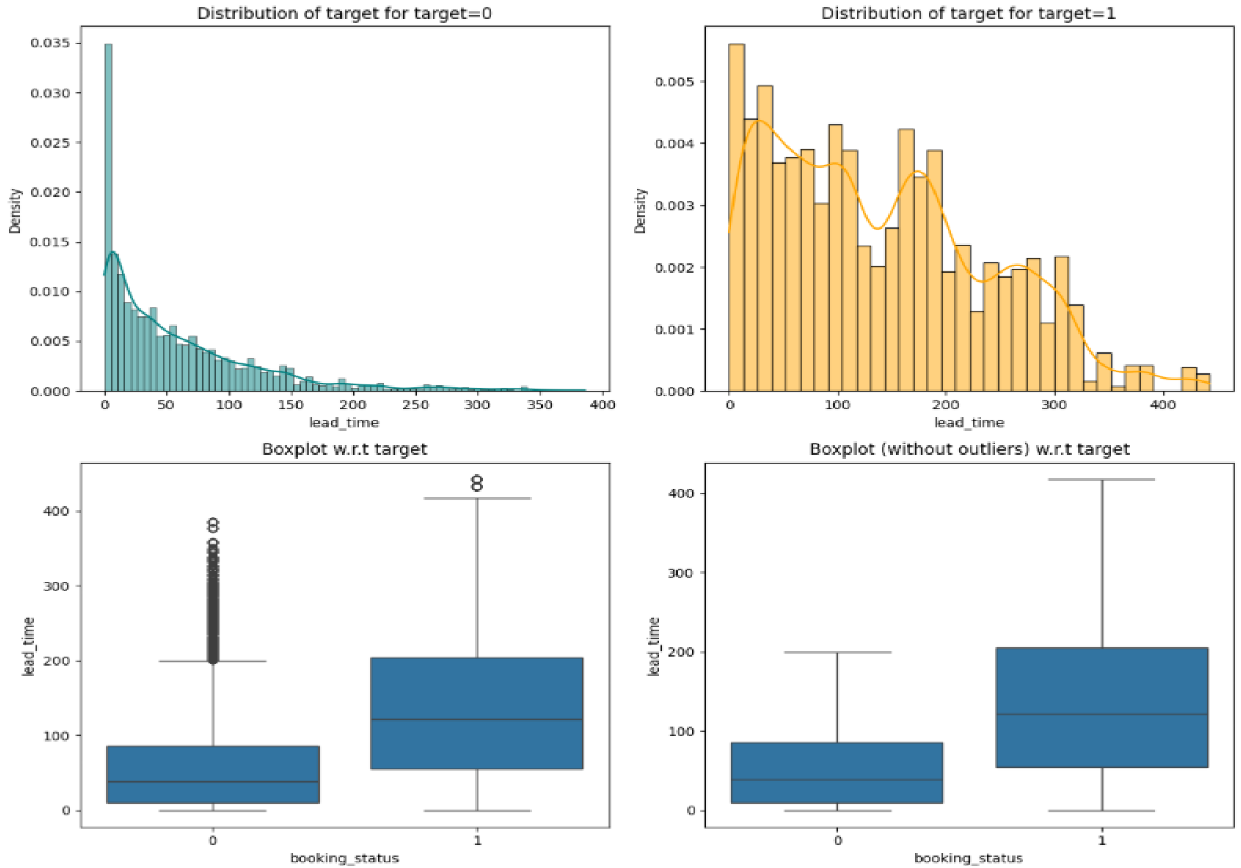
We saw earlier that there is a positive correlation between booking status and average price per room. Let's analyze it

```
distribution_plot_wrt_target(data, "avg_price_per_room", "booking_status")
```



There is a positive correlation between booking status and lead time also. Let's analyze it further

```
distribution_plot_wrt_target(data, 'lead_time', 'booking_status') ## Complete the code to find distribution of lead time wrt booking status
```



Generally people travel with their spouse and children for vacations or other activities. Let's create a new dataframe of the customers who traveled with their families and analyze the impact on booking status.

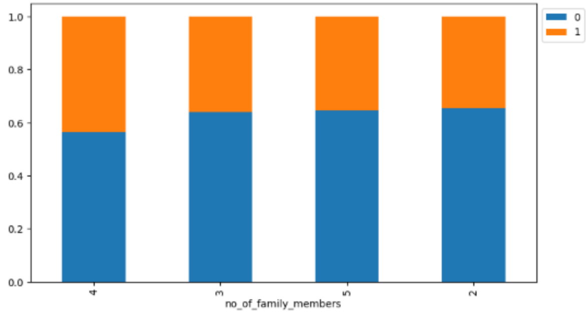
```
[ ] family_data = data[(data["no_of_children"] >= 0) & (data["no_of_adults"] > 1)]
    family_data.shape
```

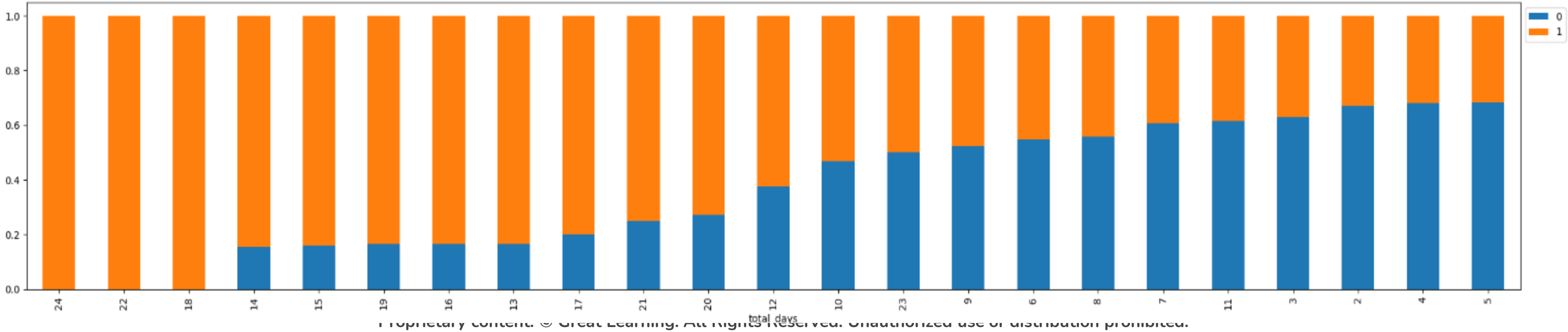
```
(28441, 18)
```

```
[ ] family_data["no_of_family_members"] = (
        family_data["no_of_adults"] + family_data["no_of_children"]
    )
```

```
stacked_barplot(family_data, 'no_of_family_members', 'booking_status') ## Complete the code to plot stacked barplot for no of fam
```

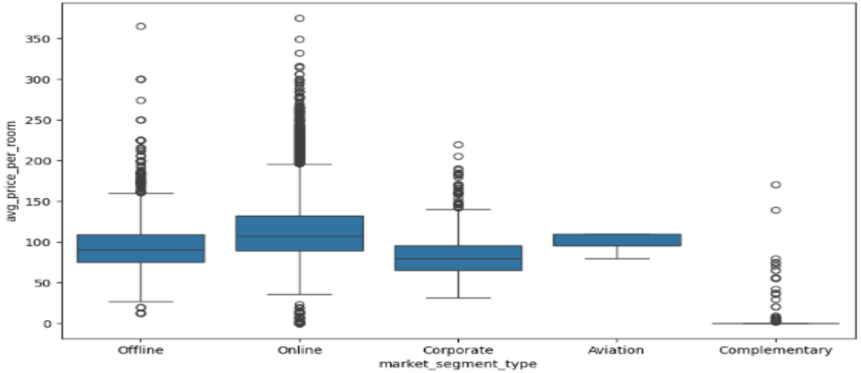
```
booking_status            0     1    All
no_of_family_members
All                   18456  9985  28441
2                     15506  8213  23719
3                      2425  1368   3793
4                       514   398    912
5                        11     6     17
------------------------------------------------------------------------------------------------------------------------
```





Hotel rates are dynamic and change according to demand and customer demographics. Let's see how prices vary across different market segments

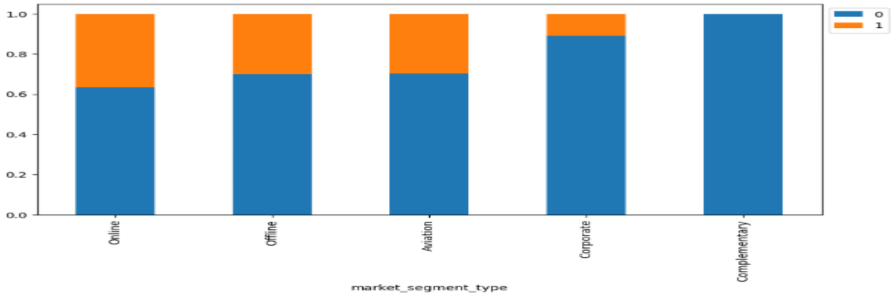
```python
plt.figure(figsize=(10, 6))
sns.boxplot(
    data=data, x="market_segment_type", y="avg_price_per_room"
)
plt.show()
```



Let's see how booking status varies across different market segments. Also, how average price per room impacts booking status

```python
stacked_barplot(data, "market_segment_type", "booking_status")
```

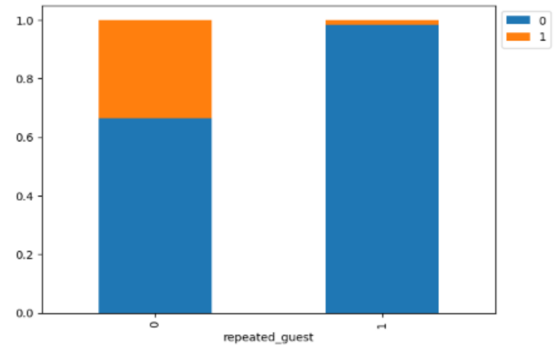
```
booking_status           0      1    All
market_segment_type
All                  24390  11885  36275
Online               14739   8475  23214
Offline               7375   3153  10528
Corporate             1797    220   2017
Aviation                88     37    125
Complementary          391      0    391
------------------------------------------------------------------------------------------------------------------------
```



Repeating guests are the guests who stay in the hotel often and are important to brand equity. Let's see what percentage of repeating guests cancel?

```
stacked_barplot(data, 'repeated_guest', 'booking_status') ## Complete the code to plot stacked barplot for repeated guests
```

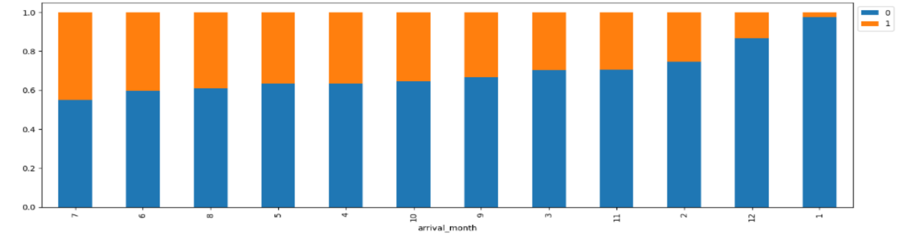
```
booking_status      0      1    All
repeated_guest
All             24390  11885  36275
0               23476  11869  35345
1                 914     16    930
------------------------------------------------------------------------------------------------------------------------
```



Let's check the percentage of bookings canceled in each month.

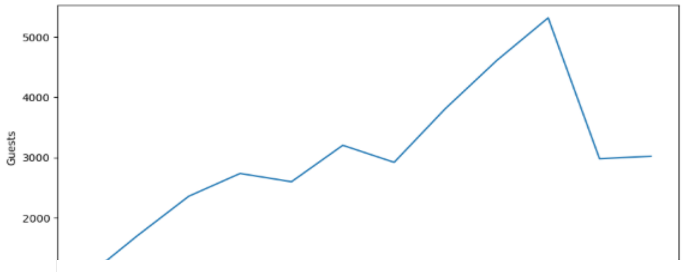
```
stacked_barplot(data,'arrival_month', 'booking_status') ## Complete the code to plot stacked barplot for arrival month and booking status
```

```
booking_status      0      1    All
arrival_month
All             24390  11885  36275
10               3437   1880   5317
9                3073   1538   4611
8                2325   1488   3813
7                1606   1314   2920
6                1912   1291   3203
4                1741    995   2736
5                1650    948   2598
11               2105    875   2980
3                1658    700   2358
2                1274    430   1704
12               2619    402   3021
1                 990     24   1014
------------------------------------------------------------------------------------------------------------------------
```



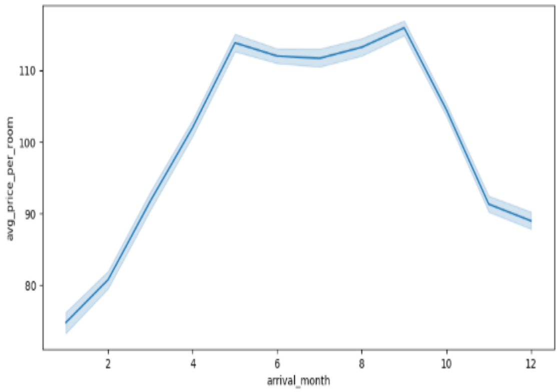
Let's find out what are the busiest months in the hotel.

```
# grouping the data on arrival months and extracting the count of bookings
monthly_data = data.groupby(["arrival_month"])["booking_status"].count()

# creating a dataframe with months and count of customers in each month
monthly_data = pd.DataFrame(
    {"Month": list(monthly_data.index), "Guests": list(monthly_data.values)}
)

# plotting the trend over different months
plt.figure(figsize=(10, 5))
sns.lineplot(data=monthly_data, x="Month", y="Guests")
plt.show()
```



As hotel room prices are dynamic, Let's see how the prices vary across different months

```
plt.figure(figsize=(10, 5))
sns.lineplot(data, y="avg_price_per_room", x="arrival_month")## Complete the code to create lineplot between average price per room and arrival month
plt.show()
```

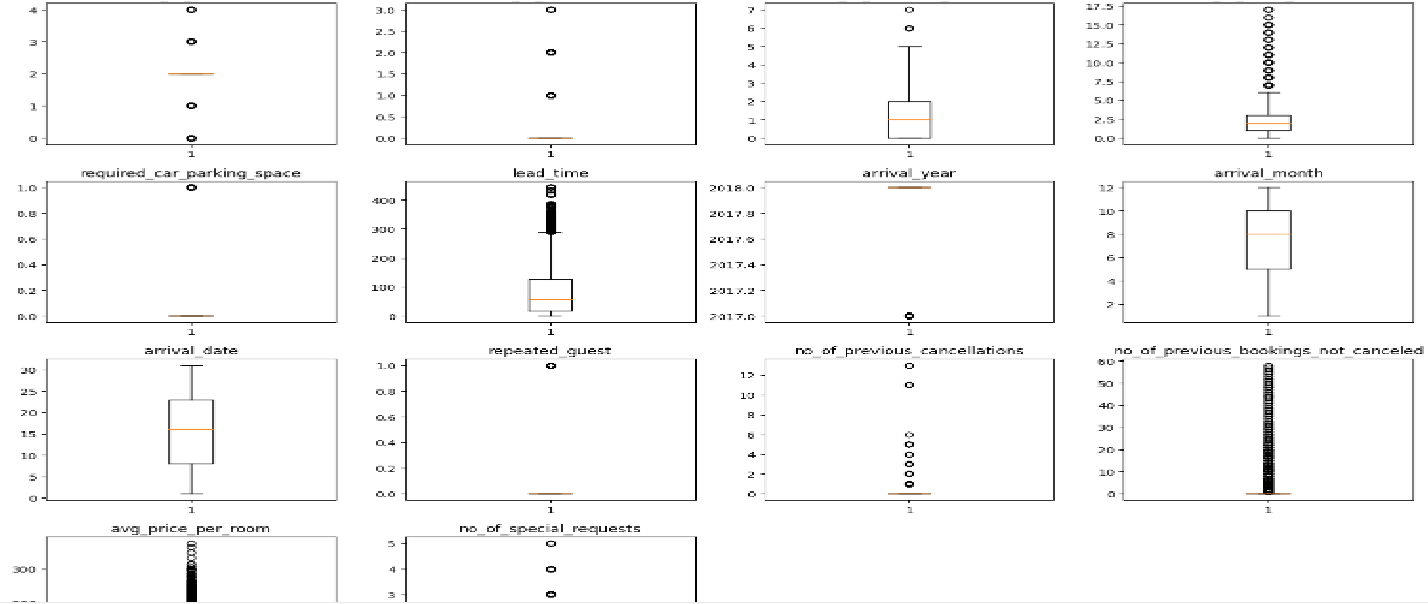


Data Preprocessing

Outlier Check

- Let's check for outliers in the data.

```
# outlier detection using boxplot
numeric_columns = data.select_dtypes(include=np.number).columns.tolist()
# dropping booking_status
numeric_columns.remove("booking_status")

plt.figure(figsize=(15, 12))

for i, variable in enumerate(numeric_columns):
    plt.subplot(4, 4, i + 1)
    plt.boxplot(data[variable], whis=1.5)
    plt.tight_layout()
    plt.title(variable)

plt.show()
```



Happy Learning !