UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Master's Degree in Artificial Intelligence and Data Engineering

Data Mining and Machine Learning Project

Predicting Hotel Bookings Cancellation With a Machine Learning Classification Model

Author:

Tommaso Falaschi

Project GitHub Repository:
https://github.com/LeBonWskii/DMML_Project

ACADEMIC YEAR 2024/2025

Contents

Chapter 1

Introduction

1.1 Introduction

Booking cancellations represent a critical challenge in the hospitality sector, directly affecting demand forecasting, inventory allocation, and revenue optimization. Their unpredictable nature complicates hotel operations and undermines the effectiveness of pricing strategies. To counteract this uncertainty, hotels often resort to overbooking strategies and rigid cancellation policies. While these approaches can partially mitigate the losses associated with no-shows, they also introduce new risks such as customer dissatisfaction, damage to reputation, and even long-term revenue loss. Accurate prediction of booking cancellations can significantly enhance hotel revenue management practices. By identifying potential cancellations in advance, hotel staff can adopt proactive strategies such as offering targeted incentives, adjusting overbooking thresholds, or reallocating inventory in real time. More importantly, predicting individual booking outcomes enables a refined estimation of net demand—crucial for maximizing occupancy and profitability. In this project, we frame the cancellation prediction task as a supervised classification problem, leveraging machine learning techniques to model complex booking behaviors. Our methodology includes data preprocessing, oversampling techniques to handle class imbalance, feature engineering, model tuning, and explainability analysis. We adopt a time-aware data splitting strategy to mimic real-world deployment and evaluate our models using stratified cross-validation followed by final hold-out validation. Furthermore, we compare default and optimized hyperparameter configurations for each model, using statistical tests to guide selection. The final system not only aims to achieve high predictive accuracy, but also to provide transparent and interpretable outputs that can inform decision-making in a hotel management context.

Chapter 2

Dataset Overview

2.1 Dataset Overview

The dataset [1] used in this project consists of two hotel booking datasets refers to a **resort hotel** (H1) and the other to a **city hotel** (H2). Both datasets share the same structure, including 31 variables. The resort hotel dataset contains 40,060 observations, while the city hotel dataset contains 79,330 observations. The data records bookings scheduled to arrive between July 1, 2015, and August 31, 2017, including both realized bookings and canceled reservations.

2.1.1 Feature Description

The dataset comprises 31 features, which can be broadly grouped into four main categories: **numerical**, **categorical**, **temporal**, and **target-related** variables. These features collectively capture a comprehensive profile of each hotel booking, encompassing customer characteristics, reservation details, and behavioral history.

- **Numerical features** include continuous or countable values such as the number of adults, children, and babies per booking, the lead time (i.e., the number of days between booking and arrival), the total number of nights stayed, and the average daily rate (ADR). These variables often provide key signals for understanding the scale and intent of a reservation.

- **Categorical features** describe qualitative attributes such as the customer type (e.g., transient, contract), deposit type, market segment, distribution channel, room types (reserved and assigned), country of origin. These features help to differentiate customer profiles and the booking context.

- **Temporal features** relate to the arrival date, including the year, month, week number, and day of the week, enabling the analysis of seasonal patterns and trends over time.

- **Target and behavioral features** include the binary variable *Is-Canceled*, which indicates whether the booking was eventually canceled, as well as the number of previous cancellations and previous bookings not canceled.

This heterogeneous mix of features allows for a rich and multidimensional analysis of booking patterns and cancellation behavior, making the dataset well-suited for machine learning classification tasks. Below is a concise description of the key features:

- **ADR (Average Daily Rate)**: Numeric, average room rate per night.

- **Adults**: Integer, number of adults per booking.

- **Agent**: Categorical, ID of the travel agency making the booking.

- **ArrrivalDateDayOfMonth,ArrivalDateMonth, ArrivalDate-WeekNumber, ArrivalDateYear**: Temporal details of arrival. All integer except ArrivalDateMonth whinch is categorical with values from "January" to "December"

- **AssignedRoomType**: Categorical, assigned room type.

- **Babies, Children**: Integer, number of babies and children per booking.

- **BookingChanges**: Integer, number of changes made to booking.

- **Company**: Categorical, ID of the company/entity that made the booking.

- **Country**: Categorical, origin country of the booking.

- **CustomerType**: Categorical, type of customer who made the booking. It includes four possible categories:
  - *Contract*: bookings associated with corporate or business agreements.
  - *Group*: reservations that are part of a group booking, often related to events or tours.
  - *Transient*: individual bookings not part of a group or contract.
  - *Transient-party*: individual bookings similar to *Transient*, but associated with a group or party traveling together.
- **DaysInWaitingList** : Integer, Number of days the booking was in the waiting list before it was confirmed to the customer.
- **DepositType**: Categorical, indicates the type of deposit made at the time of booking. It includes three possible values:
  - *No Deposit*: no deposit was required.
  - *Non Refundable*: a deposit was paid upfront and cannot be refunded.
  - *Refundable*: a deposit was paid, but it is refundable if the booking is canceled.
- **DistributionChannel**: Categorical, distribution channel for booking.
- **IsCanceled**: Binary target variable indicating cancellation.
- **IsRepeatedGuest**: Categorical, value indicating if the booking name was from a repeated guest or not.
- **LeadTime**: Integer, number of days between booking and arrival date.
- **MarketSegment** : Categorical, market segment designation.
- **Meal**: Categorical, specifies the meal plan associated with the booking. Categories include:
  - *SC* (Self Catering / Undefined): no meal included,

  - *BB*: Bed and Breakfast,
  - *HB*: Half Board (breakfast and one additional meal, usually dinner),
  - *FB*: Full Board (breakfast, lunch, and dinner).

  PreviousBookingsNotCanceled, PreviousCancellations: Historical booking behaviors.

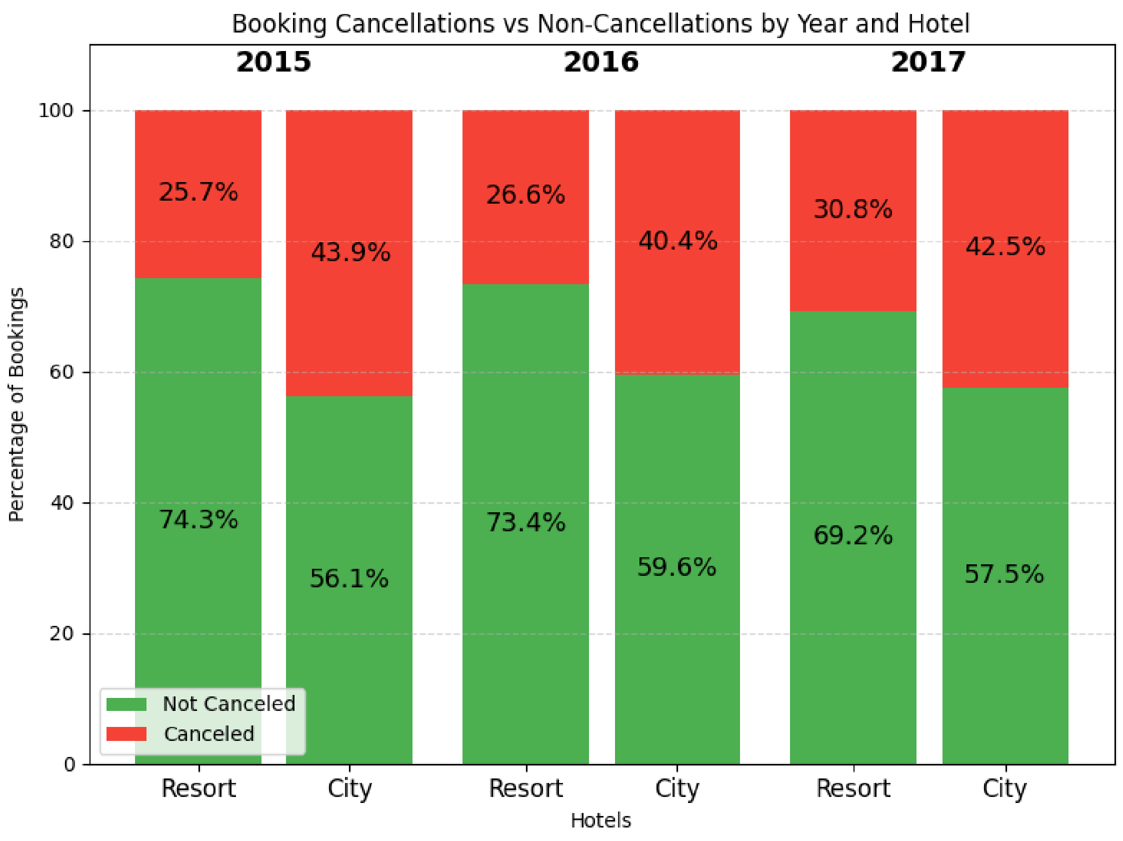
- **RequiredCardParkingSpaces**:Integer, number of car parking spaces required by the customer
- **ReservationStatus, ReservationStatusDate**: Final status and date of the last update.
- **ReservedRoomType**: Categorical, code of room type reserved at the time of booking.
- **StaysInWeekendNights, StaysInWeekNights**: Integer, length of stay.
- **TotalOfSpecialRequests**: Integer, special requests made by customers.

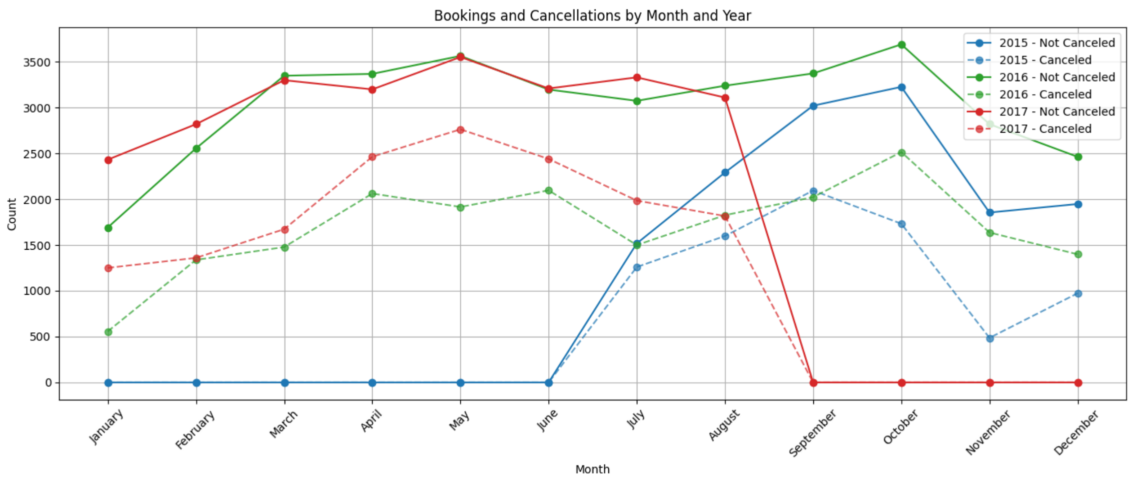
2.2 Exploratory Data Analysis

2.2.1 Data Distribution

The target variable *IsCanceled* indicates whether a hotel booking was canceled (value 1) or not (value 0). Analyzing its distribution reveals some differences between the two hotel types. In the resort hotel, only 27.8% of the bookings were canceled, while 72.2% were successfully fulfilled. In contrast, the city hotel exhibited a higher cancellation rate of 41.7%, with just 58.3% of bookings being honored.

This imbalance in cancellation behavior between hotel types may reflect differences in guest profiles, travel purposes, or booking channels.



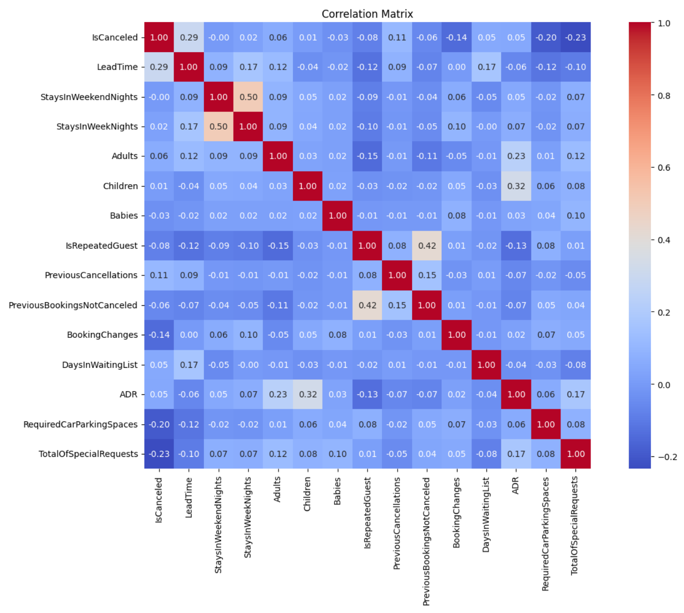


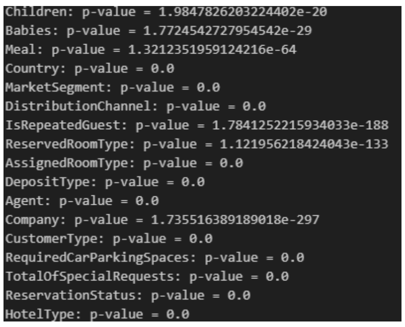
2.2.2 Correlation Analisys

Correlation Analysis

In order to better understand the relationships between the input variables and the target variable *IsCanceled*, both numerical and categorical features were analyzed.

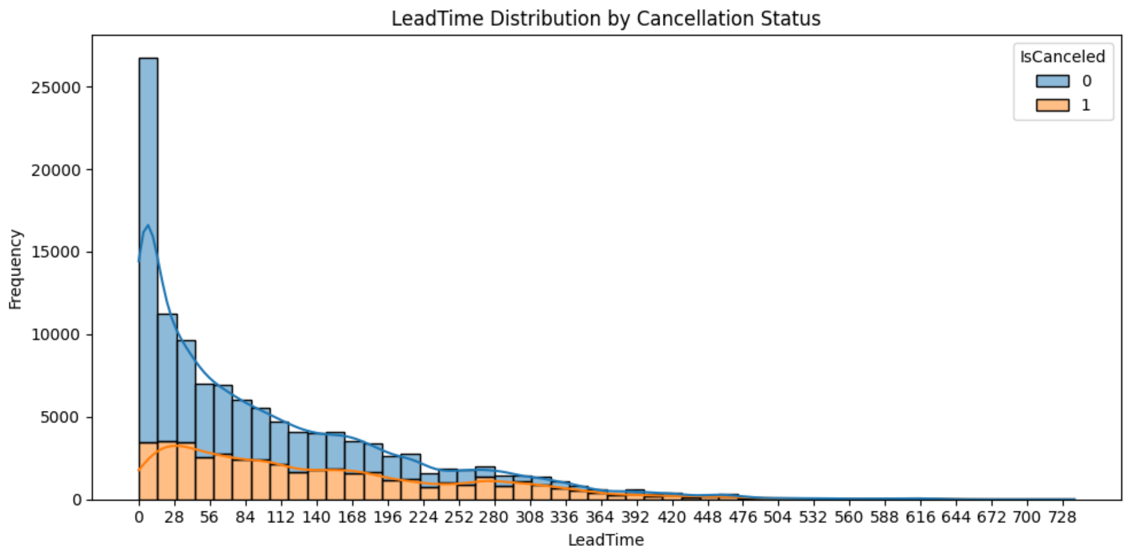
For numerical features, a Pearson correlation matrix was computed. The results show that most numerical variables have weak correlations with the target. Among them, *LeadTime* stands out with the highest positive correlation, suggesting that bookings made far in advance are more likely to be canceled. Conversely, features such as *TotalOfSpecialRequests* and *RequiredCarParkingSpaces* exhibit negative correlations, indicating that customers who request additional services or parking spaces are more likely to honor their bookings. For categorical features, statistical independence tests (Chi-squared) revealed that most of them are significantly associated with booking cancellations. Variables such as *DepositType*, *CustomerType*, *DistributionChannel*, and *MarketSegment* showed particularly strong dependence with the target variable.




```
Children: p-value = 1.9847826203224402e-20
Babies: p-value = 1.7724542727954542e-29
Meal: p-value = 1.3212351959124216e-64
Country: p-value = 0.0
MarketSegment: p-value = 0.0
DistributionChannel: p-value = 0.0
IsRepeatedGuest: p-value = 1.7841252215934033e-188
ReservedRoomType: p-value = 1.121956218424043e-133
AssignedRoomType: p-value = 0.0
DepositType: p-value = 0.0
Agent: p-value = 0.0
Company: p-value = 1.735516389189018e-297
CustomerType: p-value = 0.0
RequiredCarParkingSpaces: p-value = 0.0
TotalOfSpecialRequests: p-value = 0.0
ReservationStatus: p-value = 0.0
HotelType: p-value = 0.0
```

2.2.3 LeadTime Distribution

The distribution of *LeadTime* shows a clear inverse relationship with booking frequency: most bookings are made with short notice, and the frequency gradually decreases as the lead time increases. However, cancellations tend to be more common when the lead time is longer. This suggests that customers who book well in advance are more likely to cancel, possibly due to changes in plans or reduced commitment. In contrast, last-minute bookings are more likely to be confirmed and honored.



2.3 Preprocessing

2.3.1 Handling Missing and Undefined Values

The dataset contains several columns with missing or undefined values, either as actual NaNs or as placeholder strings like "NULL" or "Undefined". The following strategies were applied:

- **Children**: NaN values were replaced with 0, assuming missing entries represent bookings without children.
- **Country**: NaNs were replaced with the string "Unknown".
- **Agent and Company**: These fields often contain "NULL" when bookings are not mediated by an agent or company. As 0 was not used elsewhere, these placeholders were replaced with the integer 0.
- **Meal**: The "Undefined" category was safely replaced with "SC" (Self Catering), which accurately reflects the absence of included meals.
- **Other rare undefined values**: Rows containing undefined values in MarketSegment and DistributionChannel were removed due to their very low frequency and lack of interpretability.
- Bookings with no guests (all of *Adults*, *Children*, and *Babies* equal to 0) were removed, as they are considered invalid entries.

2.3.2 Preventing Data Leakage

Following best practices and findings from the reference paper [2], specific columns were removed to prevent leakage of future information:

- **Country**: Although NaNs were initially replaced with the string "Unknown", further analysis revealed that country values, especially "Portugal", were strongly correlated with booking status due to default system behavior—posing a risk of data leakage. For instance, the value "Portugal" is often automatically assigned in the system and only corrected upon check-in, leading to a higher occurrence in canceled bookings. As a result, the *Country* column was removed.
- **AssignedRoomType**: This feature is removed to prevent data leakage. The *AssignedRoomType* field is typically updated only after check-in, meaning it contains information that is not available at booking time. An analysis of the mismatch between *ReservedRoomType* and *AssignedRoomType* revealed that such mismatches occur much more frequently in bookings that were not canceled. This suggests that *AssignedRoomType* may implicitly encode the outcome of the booking, thus posing a significant risk of leakage if included. Since this information may only be updated after check-in, keeping it risks data leakage.
- **ReservationStatus and ReservationStatusDate**: Directly encode the outcome of the booking, and therefore must be excluded.

2.3.3 Feature Engineering

The *ADR* (Average Daily Rate) is influenced by seasonal and contextual factors. However, because temporal features were removed to avoid leakage, ADR on its own may no longer be meaningful. To address this, a new normalized variable was introduced:

$$ADRThirdQuartileDeviation = \frac{ADR}{Q3_{ADR}} \tag{2.1}$$

where $Q3_{ADR}$ is the third quartile ADR for groups of bookings sharing the same:

- Distribution channel

- Reserved room type
- Expected arrival week and year

This transformation allows ADR to retain its relevance by comparing it to similar bookings within the same market context.

2.3.4 Encoding Categorical Features

Some categorical features have high cardinality with many infrequent values. To address this:

- **Rare Encoding**: Applied to features like *Agent* and *Company*, where categories appearing in fewer than 2% of entries were grouped under a common "Rare" category.

After applying rare encoding, **one-hot encoding** was used on all non-binary categorical variables due to the low cardinality.

2.3.5 Scaling of Numerical Features

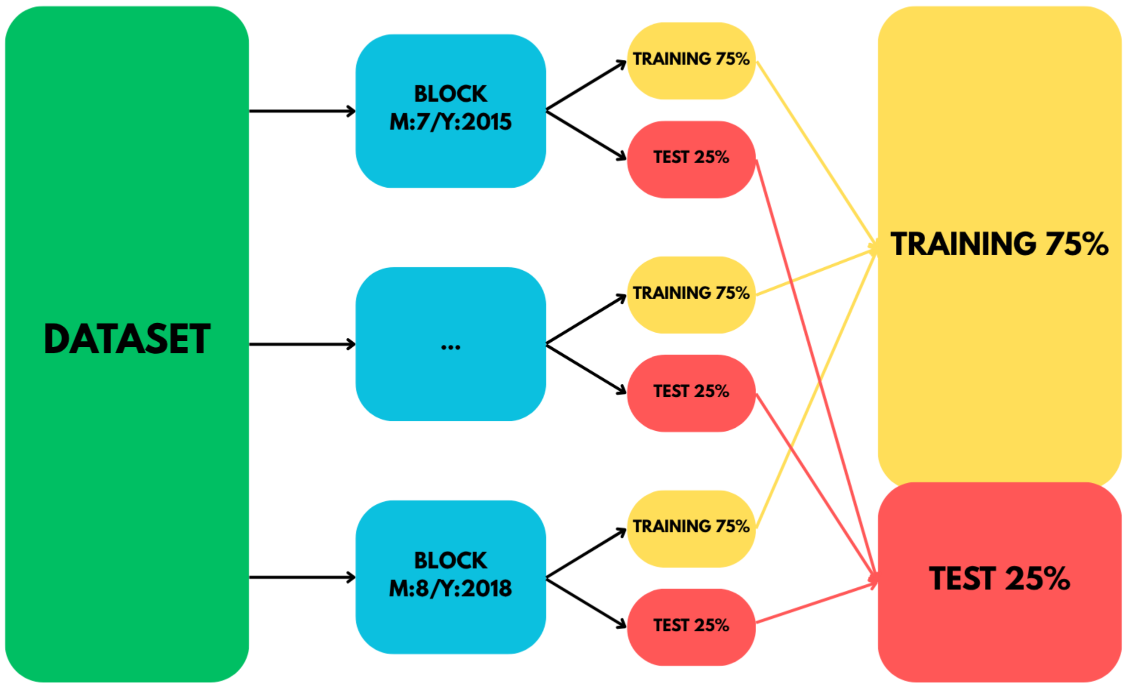
To ensure consistent input ranges and improve the performance of models sensitive to feature magnitude, all numerical features were standardized using the *Standard Scaler*.

2.4 Dataset Splitting

2.4.1 Convenience Splitting

Instead of applying a random or stratified split across the entire dataset, we employed a method known as *convenience splitting*, inspired by best practices in time series analysis. This approach is designed to handle *non-stationary temporal data*, where the statistical properties of the data may change over time.

The process begins by ordering all booking records by their expected arrival date. Then, discrete blocks are created for each unique *month/year* combination. Within each monthly block, a 75% stratified sample is allocated to the training set and the remaining 25% to the test set. This ensures that both sets reflect the temporal dynamics and seasonal trends present in real-world booking patterns. This strategy allows for a more realistic simulation of deployment conditions by avoiding information leakage from future data and preserving the time-dependent structure of the dataset. Additionally, it enables the model to generalize better to evolving booking behaviors and market conditions. Before training the models, all temporal features were removed from both the training and test sets, to prevent future information leakage and ensure models rely only on data available at booking time.



2.4.2 StratifiedKFold

To train the models and tune hyperparameters, a *Stratified K-Fold Cross-Validation* procedure with 5 folds was applied exclusively on the training set. This ensured that the class distribution of the target variable was preserved across each fold, improving model robustness and generalizability. The test set obtained from the convenience splitting was kept untouched and used only for the final hold-out evaluation.

Chapter 3

Classification

3.1 Model Training

3.1.1 Pipeline Building

To assess the impact of class imbalance handling techniques, two distinct machine learning pipelines were initially tested: one including *SMOTENC* and one without it. The target variable *IsCanceled* exhibits a moderate imbalance, with a higher proportion of non-canceled bookings. SMOTENC (Synthetic Minority Over-sampling Technique for Nominal and Continuous features) was selected as it allows oversampling of the minority class while properly handling categorical variables. By comparing performance with and without SMOTENC, we aimed to evaluate whether synthetic balancing leads to more robust models or introduces noise that may degrade generalization. This comparison informed the final choice of pipeline structure used in hyperparameter tuning.

3.1.1.1 Pipeline 1: Without SMOTENC

This pipeline processes the data without any oversampling, maintaining the original class distribution:

- **Feature Engineering:** Creation of the *ADRThirdQuartileDeviation* variable to normalize ADR with respect to similar bookings.
- **Preprocessing:**One-hot encoding for categorical variables, and standard scaling for numerical features.

- **Model Training:** The following baseline classifiers were trained on the original (imbalanced) dataset:
  - Logistic Regression
  - Random Forest
  - XGBoost
  - LightGBM
  - CatBoost
  - AdaBoost
  - Decision Tree
  - K-Nearest Neighbors (KNN)

3.1.1.2 Pipeline 2: With SMOTENC

This version integrates SMOTENC to balance the dataset before model training:

- **Feature Engineering:** Same ADR normalization via *ADRThirdQuartileDeviation*.
- **SMOTENC Oversampling:** Applied to rebalance the classes, using a list of categorical features. Designed specifically for dataset containing categorical variables.
- **Preprocessing:** One-hot encoding, and scaling.
- **Classification:** Same 8 classifiers.

3.1.2 Choise of Pipeline

Table 3.1: With SMOTENC

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**ROC AUC**
Logistic Regression	0.789	0.720	0.704	0.712	0.846
Random Forest	0.841	0.793	0.772	0.782	0.908
XGBoost	0.829	0.780	0.750	0.765	0.896
LightGBM	0.826	0.779	0.740	0.759	0.891
CatBoost	0.832	0.786	0.750	0.768	0.897
AdaBoost	0.793	0.728	0.706	0.717	0.859
Decision Tree	0.796	0.709	0.763	0.735	0.792
KNN	0.807	0.738	0.742	0.740	0.862

Table 3.2: Without SMOTENC

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**ROC AUC**
Logistic Regression	0.814	0.839	0.615	0.710	0.847
Random Forest	0.848	0.836	0.734	0.782	0.909
XGBoost	0.839	0.854	0.684	0.759	0.899
LightGBM	0.836	0.867	0.661	0.750	0.894
CatBoost	0.840	0.860	0.680	0.759	0.899
AdaBoost	0.816	0.850	0.612	0.711	0.861
Decision Tree	0.803	0.731	0.741	0.736	0.792
KNN	0.818	0.789	0.693	0.738	0.862

Table 3.3: Difference (With SMOTENC - Baseline)

Model	Accuracy	Precision	Recall	F1 Score	ROC AUC
Logistic Regression	-0.025	-0.119	+0.089	+0.002	-0.001
Random Forest	-0.007	-0.043	+0.038	+0.000	-0.001
XGBoost	-0.010	-0.074	+0.066	+0.006	-0.003
LightGBM	-0.010	-0.088	+0.079	+0.009	-0.003
CatBoost	-0.008	-0.074	+0.070	+0.009	-0.002
AdaBoost	-0.023	-0.122	+0.094	+0.006	-0.002
Decision Tree	-0.007	-0.022	+0.022	-0.001	+0.000
KNN	-0.011	-0.051	+0.049	+0.002	+0.000

Despite the slight drop in precision and accuracy, we chose to adopt the pipeline with **SMOTENC** due to its consistent improvement in **recall** across most models. In the context of hotel booking management, recall represents the ability to correctly identify *cancellations*, which is a critical business need. Failing to anticipate a cancellation (false negatives) may lead to *unused inventory*, *lost revenue opportunities*, and *inefficient resource allocation*. Therefore, although the baseline pipeline without SMOTENC slightly outperforms in some metrics like precision, the **ability of SMOTENC to better detect cancellations justifies its adoption** for this predictive task.

3.1.3 Hyperparameter Tuning

To optimize model performance, an extensive hyperparameter tuning phase was conducted for each classifier using `GridSearchCV` with 5-fold stratified cross-validation. This process systematically explores combinations of model-specific hyperparameters to identify the configuration that yields the best validation performance, based on the F1-score. Each classifier was paired with a dedicated hyperparameter grid:

- **Logistic Regression**
  - `C`: [0.01, 0.1, 1, 10, 100]
  - `penalty`: ['l2']
  - `solver`: ['lbfgs', 'liblinear']

- **Random Forest**
  - `n_estimators`: [100, 200, 300]
  - `max_depth`: [None, 10, 20, 30]
  - `max_features`: ['sqrt', 'log2']
  - `min_samples_split`: [2, 5, 10]
- **AdaBoost**
  - `n_estimators`: [50, 100, 200]
  - `learning_rate`: [0.01, 0.1, 1.0]
- **Decision Tree**
  - `max_depth`: [None, 10, 20, 30]
  - `min_samples_split`: [2, 5, 10]
  - `criterion`: ['gini', 'entropy']
- **K-Nearest Neighbors**
  - `n_neighbors`: [3, 5, 7, 9]
  - `weights`: ['uniform', 'distance']
  - `p`: [1, 2] (1 = Manhattan, 2 = Euclidean)
- **XGBoost**
  - `n_estimators`: [100, 200]
  - `max_depth`: [3, 5, 7]
  - `learning_rate`: [0.01, 0.1, 0.2]
  - `subsample`: [0.8, 1.0]
  - `colsample_bytree`: [0.8, 1.0]
- **LightGBM**
  - `n_estimators`: [100, 200]
  - `max_depth`: [-1, 10, 20]
  - `learning_rate`: [0.01, 0.1]

  - `num_leaves`: [31, 50, 100]
  - `subsample`: [0.8, 1.0]

- **CatBoost**
  - `iterations`: [100, 200]
  - `depth`: [4, 6, 8]
  - `learning_rate`: [0.01, 0.1]
  - `l2_leaf_reg`: [1, 3, 5]

This careful tuning procedure helped ensure fair comparison among models and allowed each algorithm to operate under its most favorable settings.

3.1.3.1 Default-Optimized Comparison

To assess the actual benefit of hyperparameter tuning, each model was evaluated both in its default configuration and in its optimized version—resulting from grid search on the training set. The evaluation metric adopted for comparison was the F1 score, given its suitability in capturing the balance between precision and recall, particularly relevant in the context of cancellation prediction. Table 3.4 summarizes the F1 scores obtained with both configurations, as well as the improvement achieved through tuning. The "Chosen Version" column reflects the final decision based on which configuration delivered superior F1 performance on the validation set.

Table 3.4: Comparison of Default vs. Optimized Models

Model	**F1 (Optimized)**	**F1 (Default)**	**Improvement**	**Chosen Version**
KNN	0.764	0.740	+0.024	Optimized
LightGBM	0.775	0.759	+0.016	Optimized
Decision Tree	0.747	0.735	+0.012	Optimized
XGBoost	0.772	0.765	+0.007	Optimized
Random Forest	0.786	0.782	+0.004	Optimized
AdaBoost	0.720	0.717	+0.003	Optimized
Logistic Regression	0.712	0.712	−0.000	Default
CatBoost	0.763	0.768	−0.005	Default

As shown in the table, six out of the eight models exhibited a positive improvement in F1 score following hyperparameter optimization. The

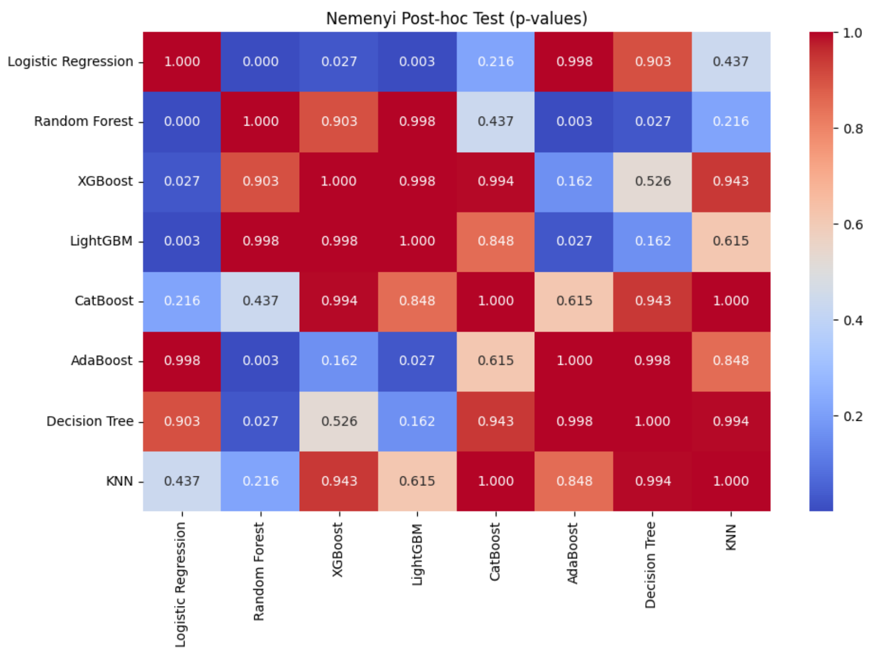
K-Nearest Neighbors model showed the largest relative gain, followed by *LightGBM* and *Decision Tree*. However, for *Logistic Regression* and *CatBoost*, the default configuration outperformed the optimized version, albeit by a narrow margin. Consequently, for these two models, the default version was retained in the final evaluation. This comparison allowed for a more robust model selection, ensuring that hyperparameter tuning was applied only when it yielded a tangible benefit, thereby avoiding overfitting or unnecessary complexity.

3.2 Model Selection

To determine the best-performing model, a rigorous two-phase selection process was adopted.

3.2.1 Statistical Comparison

To assess whether differences in cross-validated performance were statistically significant, a **Friedman test** was conducted on the per-fold F1-scores. When the test revealed significant differences ($p < 0.05$), a **post-hoc Nemenyi test** was performed to identify statistically distinguishable models. Results from this phase were used to narrow down the selection to the top 5 models: **Random Forest**, **LightGBM**, **XGBoost**, **CatBoost**, and **K-Nearest Neighbors**.



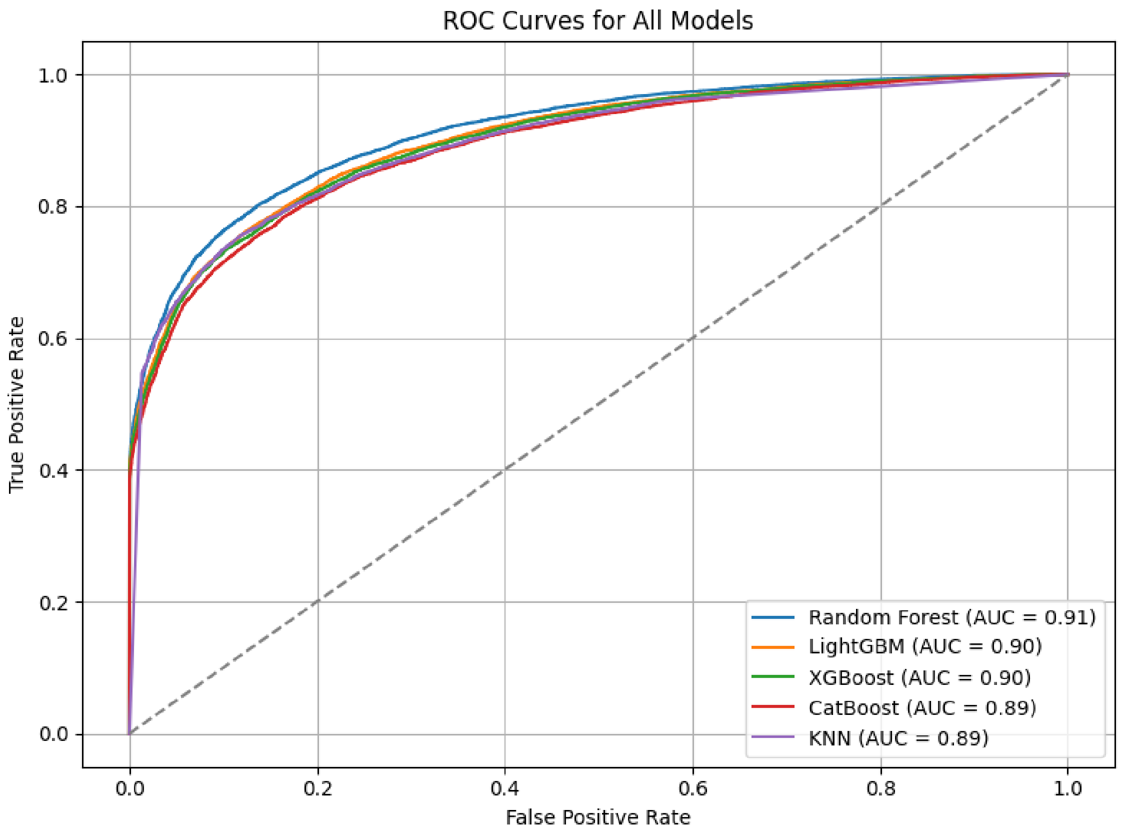
	Logistic Regression	Random Forest	XGBoost	LightGBM	CatBoost	AdaBoost	Decision Tree	KNN
Logistic Regression	1.000	0.000	0.027	0.003	0.216	0.998	0.903	0.437
Random Forest	0.000	1.000	0.903	0.998	0.437	0.003	0.027	0.216
XGBoost	0.027	0.903	1.000	0.998	0.994	0.162	0.526	0.943
LightGBM	0.003	0.998	0.998	1.000	0.848	0.027	0.162	0.615
CatBoost	0.216	0.437	0.994	0.848	1.000	0.615	0.943	1.000
AdaBoost	0.998	0.003	0.162	0.027	0.615	1.000	0.998	0.848
Decision Tree	0.903	0.027	0.526	0.162	0.943	0.998	1.000	0.994
KNN	0.437	0.216	0.943	0.615	1.000	0.848	0.994	1.000

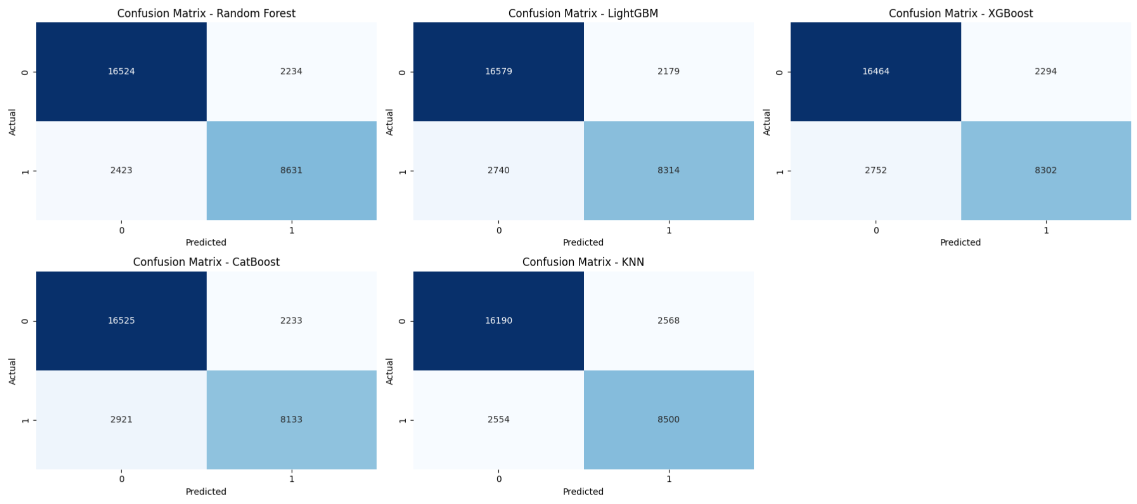
3.2.2 Final Evaluation on Hold-Out Set

In the second phase, the selected models were evaluated on the *hold-out test set* derived from the convenience splitting strategy. Each model was trained on the full training set using its best hyperparameters and evaluated on the test set. Performance metrics such as Accuracy, Precision, Recall, F1-Score, and ROC AUC were computed. Furthermore, confusion matrices and ROC curves were analyzed to visualize the classification behavior and trade-offs of each model. Table 3.5 summarizes the results of the final evaluation. Among the models, **Random Forest** achieved the best performance across all metrics, confirming its robustness in handling the classification task. **LightGBM** and **XGBoost** followed closely, demonstrating competitive performance. Although **KNN** performed slightly worse in terms of precision and AUC, it maintained a high recall.

Table 3.5: Hold-Out Evaluation Metrics for Final Models

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**ROC AUC**
Random Forest	0.844	0.794	0.781	0.788	0.914
LightGBM	0.835	0.792	0.752	0.772	0.902
XGBoost	0.831	0.784	0.751	0.767	0.900
CatBoost	0.827	0.785	0.736	0.759	0.892
KNN	0.828	0.768	0.769	0.768	0.894


ROC Curves for All Models
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Random Forest (AUC = 0.91)
LightGBM (AUC = 0.90)
XGBoost (AUC = 0.90)
CatBoost (AUC = 0.89)
KNN (AUC = 0.89)


Confusion Matrix - Random Forest
16524
2234
2423
8631
Confusion Matrix - LightGBM
16579
2179
2740
8314
Confusion Matrix - XGBoost
16464
2294
2752
8302
Confusion Matrix - CatBoost
16525
2233
2921
8133
Confusion Matrix - KNN
16190
2568
2554
8500
Actual
Predicted
0
1

Chapter 4

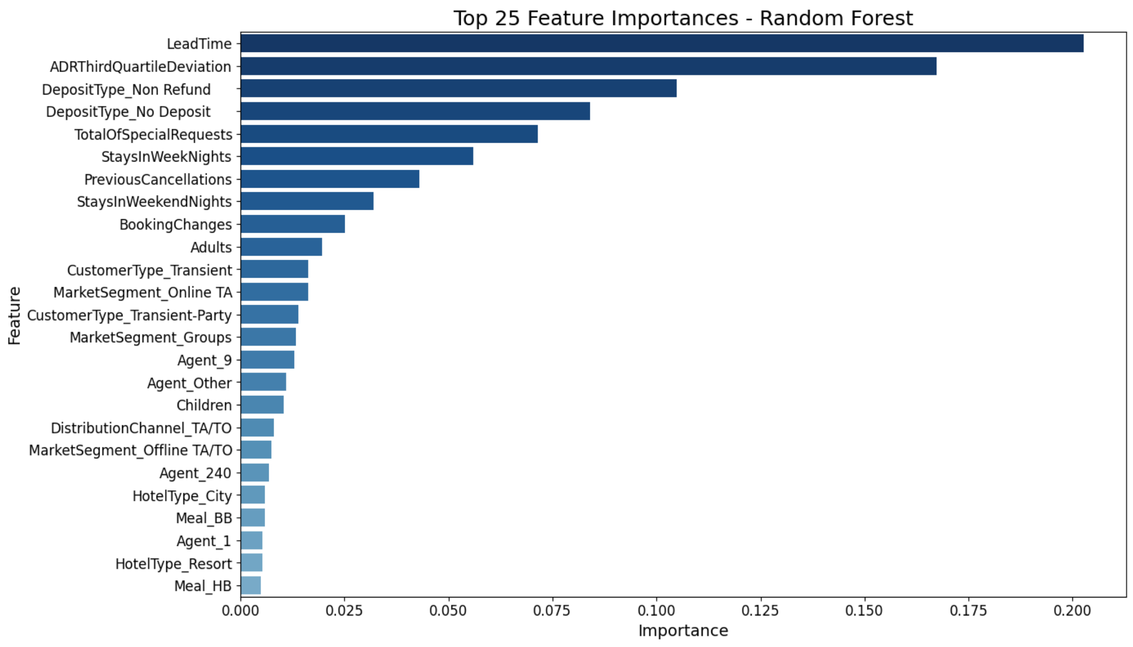
Explainability

To ensure transparency and interpretability of the predictions made by the final model, a **Random Forest** classifier, both global and local explainability analyses were conducted. This allows to better understand the rationale behind the model's decisions, identify the most influential factors.

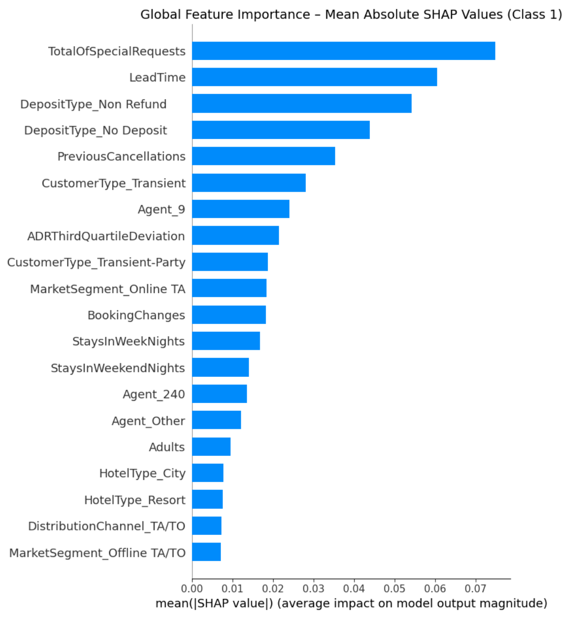
4.1 Global Explainability

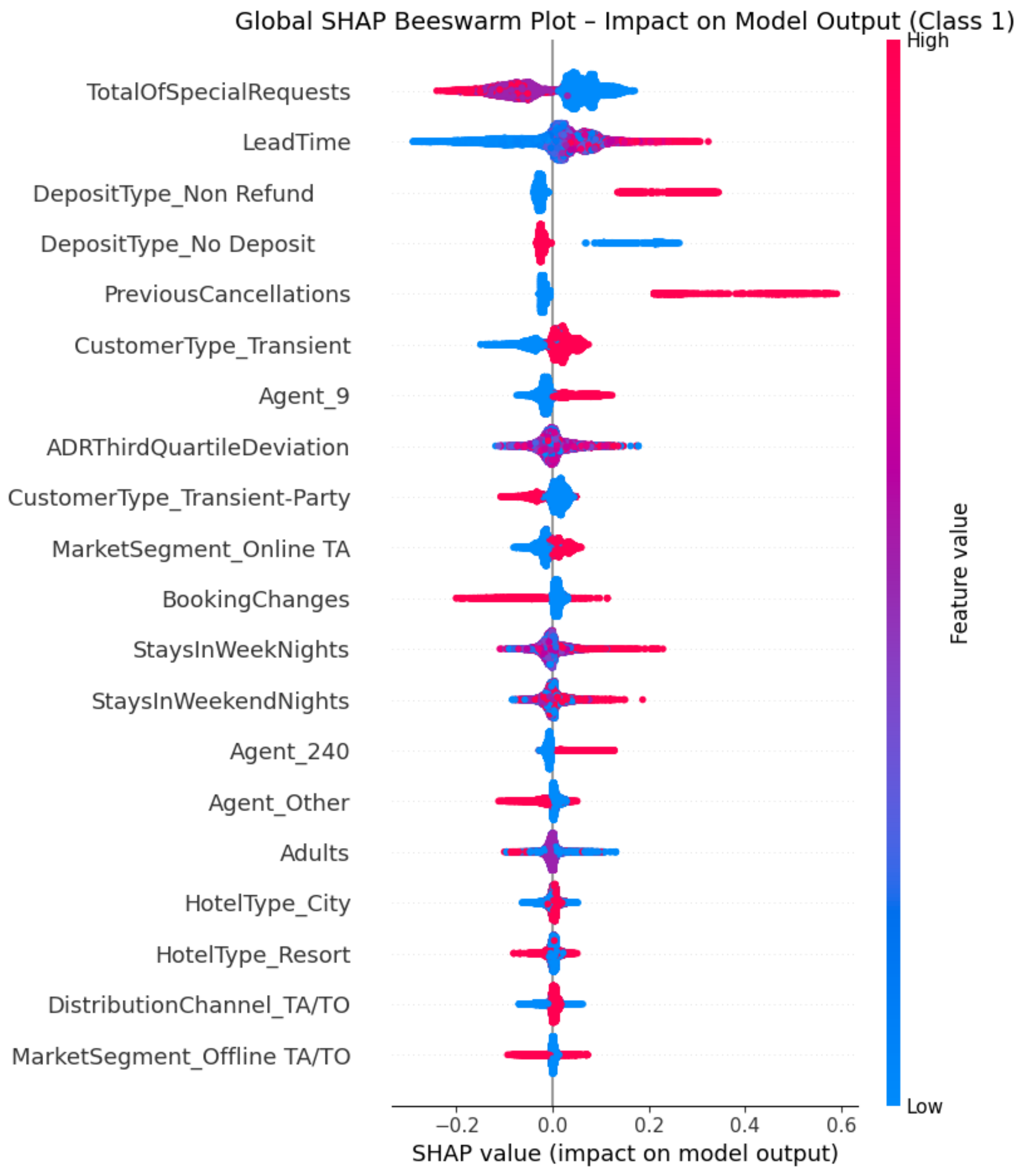
Two techniques were used to assess the overall importance of each feature in determining booking cancellations:

- **Feature Importance from Random Forest:** The first analysis ranks the top 25 features by their mean decrease in impurity, a standard metric used in ensemble tree-based methods. *LeadTime* emerged as the most important feature, followed by the engineered variable *ADRThirdQuartileDeviation*, and *DepositType*.



- **SHAP Global Values:** To obtain a model-agnostic view, we computed the mean absolute SHAP values which quantify each feature's average contribution to the model's output. Here, *TotalOfSpecialRequests* and *LeadTime* ranked highest, reinforcing their strong predictive power. The SHAP summary beeswarm plot also shows the distribution and direction of each feature's impact, where, for instance, a higher number of special requests decreases the likelihood of cancellation.

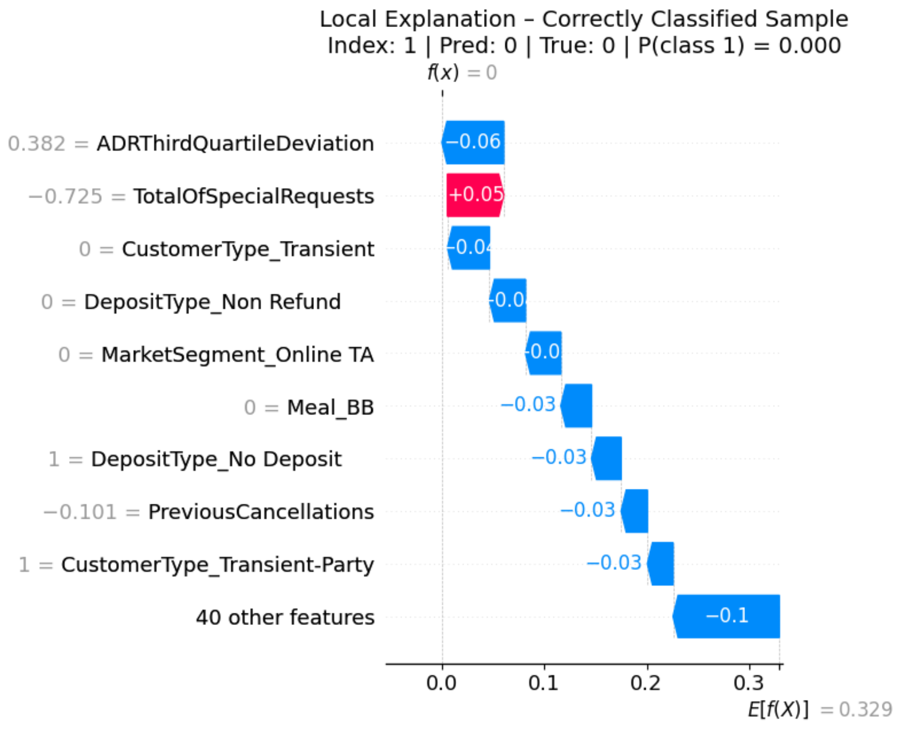



Global SHAP Beeswarm Plot – Impact on Model Output (Class 1)
TotalOfSpecialRequests
LeadTime
DepositType_Non Refund
DepositType_No Deposit
PreviousCancellations
CustomerType_Transient
Agent_9
ADRThirdQuartileDeviation
CustomerType_Transient-Party
MarketSegment_Online TA
BookingChanges
StaysInWeekNights
StaysInWeekendNights
Agent_240
Agent_Other
Adults
HotelType_City
HotelType_Resort
DistributionChannel_TA/TO
MarketSegment_Offline TA/TO
−0.2
0.0
0.2
0.4
0.6
SHAP value (impact on model output)
High
Low
Feature value

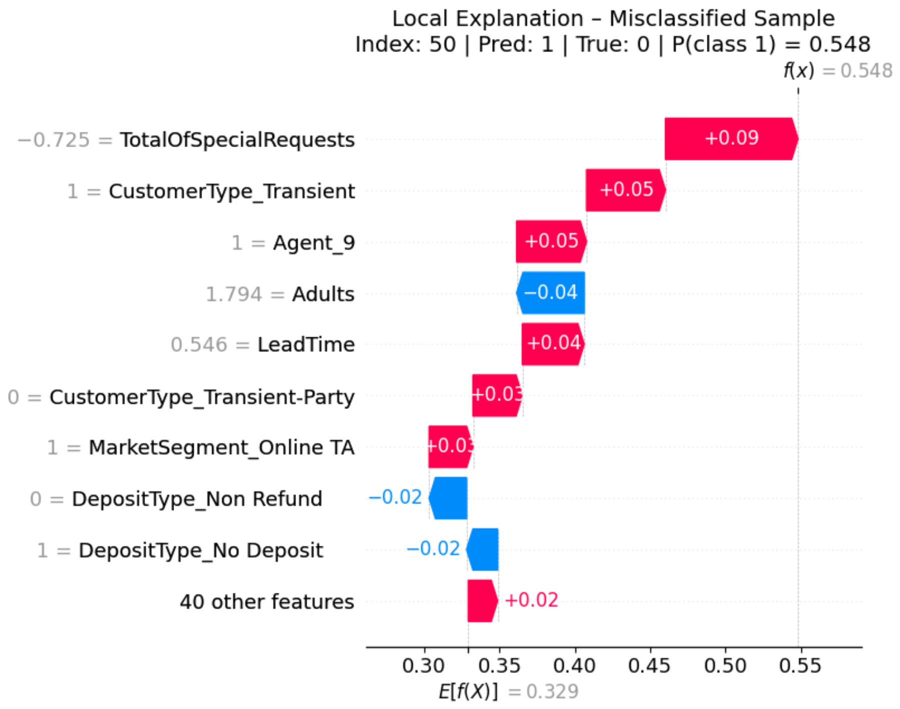
4.2 Local Explainability

In addition to global explanations, we also performed local SHAP analyses to explain individual predictions. The following examples illustrate how the model arrived at its decisions for specific samples:

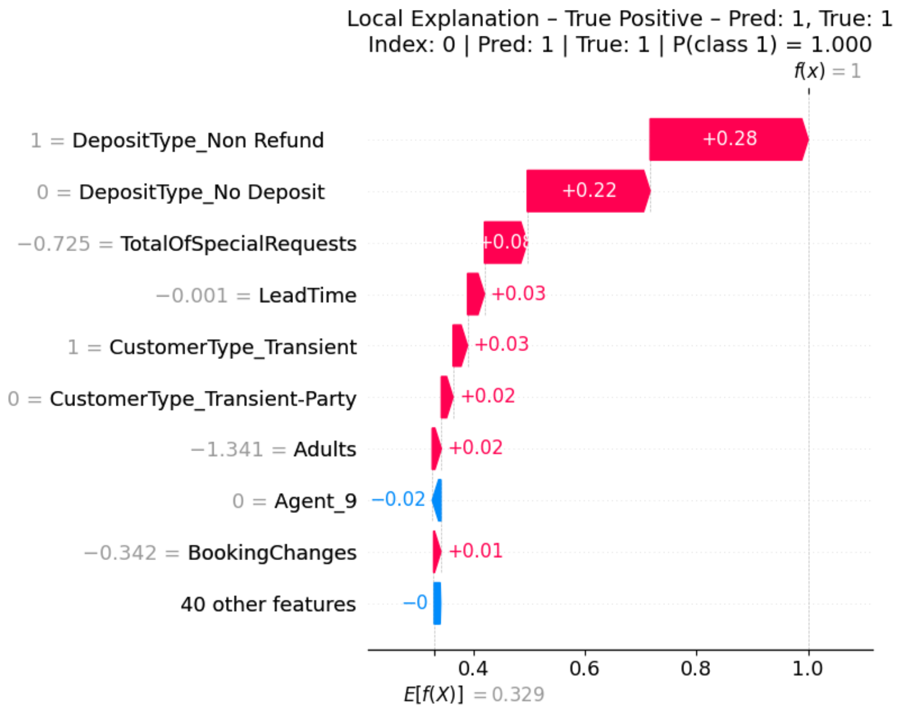
- **Correctly Classified Sample** : The model correctly predicted a non-cancellation. *ADRThirdQuartileDeviation* and *CustomerType* contributed negatively to the cancellation probability.



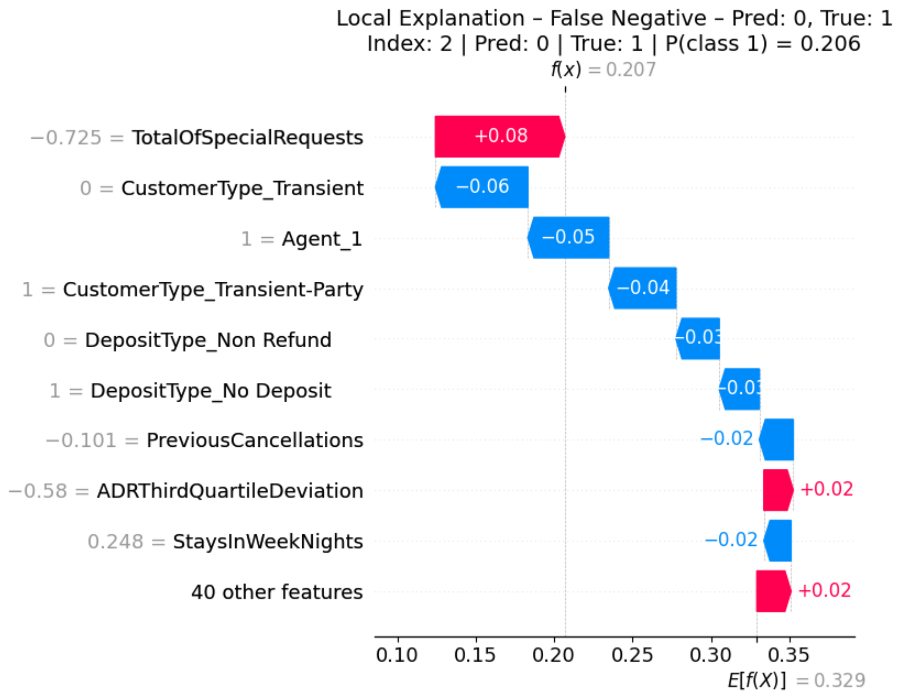
- **Misclassified Sample** : A booking was predicted as canceled, but was actually not canceled. Features such as a number of special requests and *LeadTime* contributed positively to the (wrong) prediction.



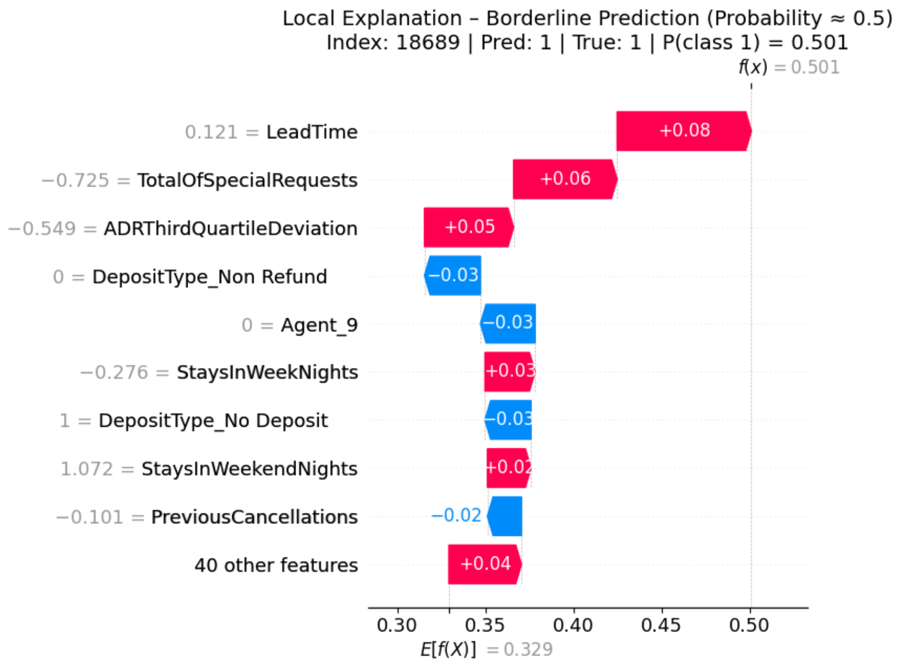
- **True Positive** : The model correctly predicted a cancellation, mainly influenced by *DepositType* and *LeadTime*.



- **False Negative** : The model failed to predict a cancellation.

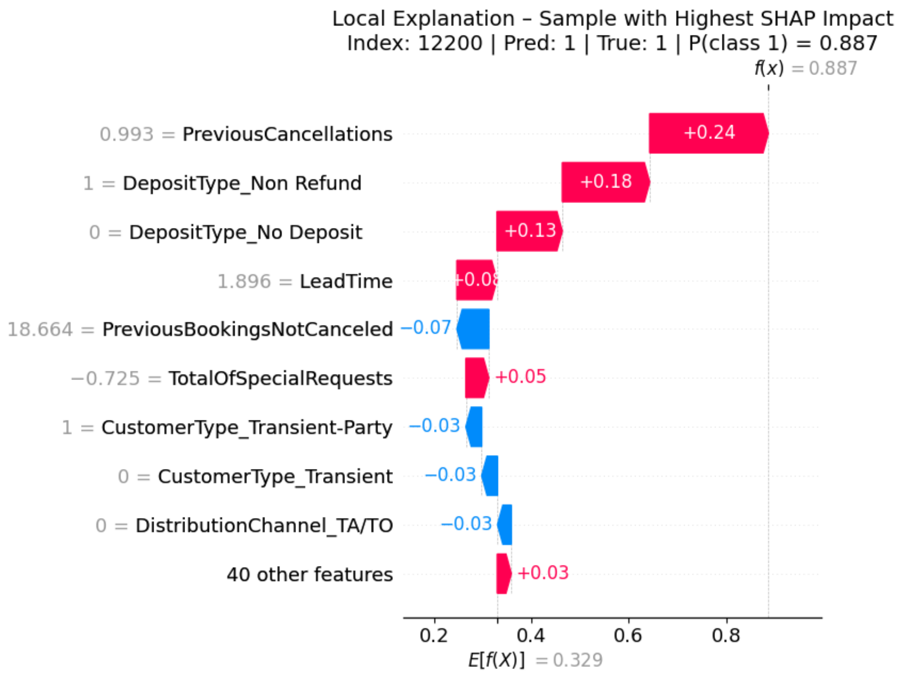


- **Borderline Case :** A booking with a predicted probability close to 0.5 demonstrates how small changes in several features influenced the final decision.



- **Sample with Highest SHAP Impact :** For this observation, the model's prediction was heavily driven by a combination of *Previous-*

Cancellations and *DepositType*.



This multi-level explainability approach highlights the key predictors of cancellations and provides actionable insights for hotel managers and analysts. It also ensures the model's decisions can be trusted and audited.

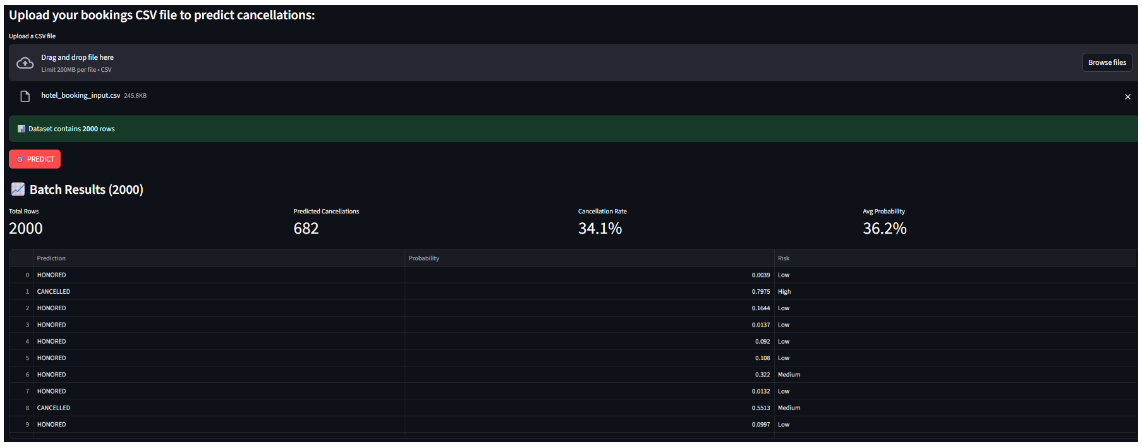
Chapter 5

Graphic User Interface

To facilitate the practical use of the cancellation prediction model by non-technical hotel staff, a user-friendly web-based **Graphical User Interface (GUI)** was developed. The interface allows users to upload a CSV file containing booking data and obtain real-time predictions on the likelihood of cancellation.

- **File Upload and Prediction**: Users can drag and drop or browse a CSV file. Upon clicking the `Predict` button, the system processes the file using the trained Random Forest model.
- **Batch Results Summary**: The GUI displays key statistics such as the number of total bookings, predicted cancellations, the estimated cancellation rate, and the average predicted probability.
- **Per-Booking Output**: Each booking is shown with its predicted outcome (e.g., `CANCELLED` or `HONORED`) and associated probability.
- **Visualization**: A histogram of predicted probabilities is generated to help users visually understand the distribution of cancellation risks across all bookings.

This GUI enhances interpretability and practical applicability by translating complex model outputs into intuitive, actionable insights suitable for operational use.



Upload your bookings CSV file to predict cancellations:

Upload a CSV file

Drag and drop file here
Limit 200MB per file • CSV

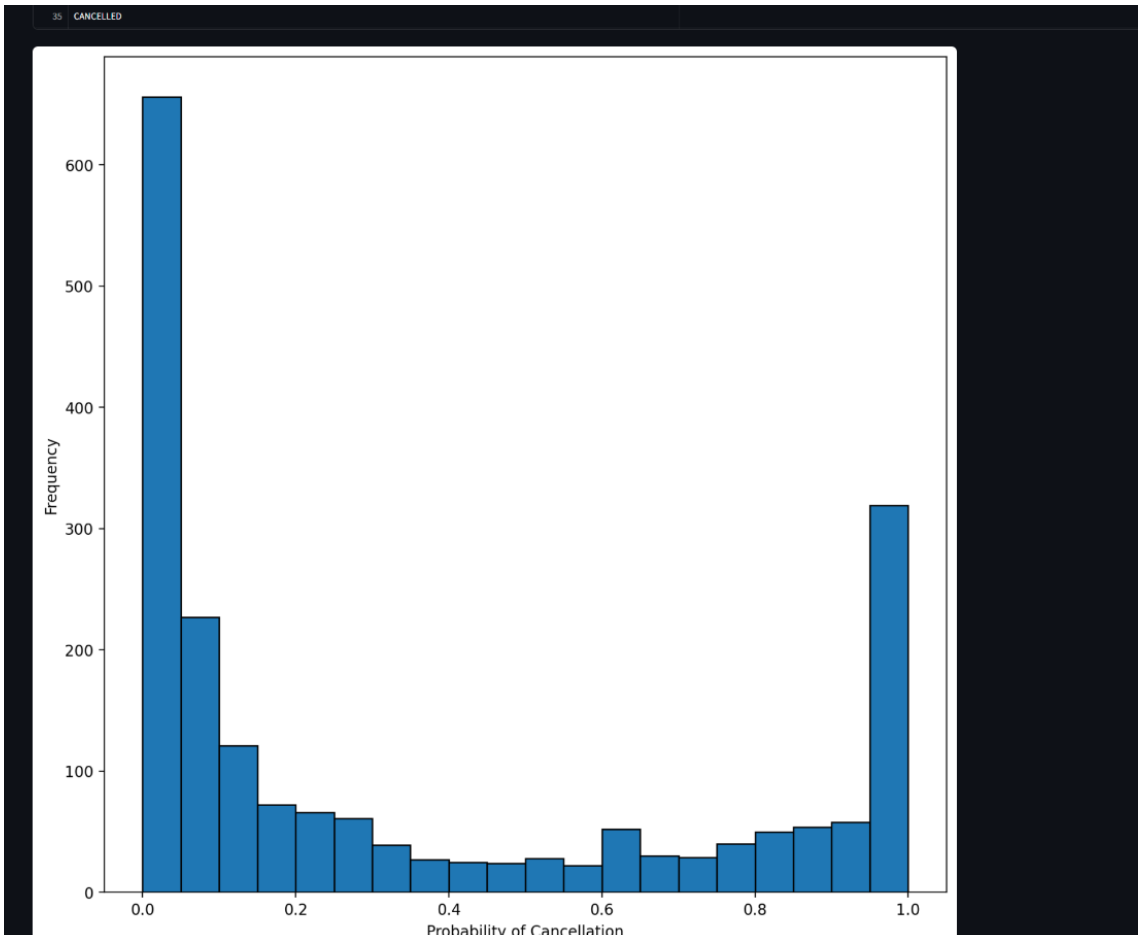
hotel_booking_input.csv 245.6KB

Dataset contains 2000 rows

Batch Results (2000)

Total Rows	Predicted Cancellations	Cancellation Rate	Avg Probability
2000	682	34.1%	36.2%

	Prediction	Probability	Risk
0	HONORED	0.0039	Low
1	CANCELLED	0.7975	High
2	HONORED	0.1644	Low
3	HONORED	0.0137	Low
4	HONORED	0.092	Low
5	HONORED	0.108	Low
6	HONORED	0.322	Medium
7	HONORED	0.0132	Low
8	CANCELLED	0.5513	Medium
9	HONORED	0.0997	Low



5.1 Conclusions

5.1.1 Comparison with Previous Studies

This project builds upon and extends previous work on hotel booking cancellation prediction, notably two recent studies that employed tree-based models such as XGBoost [2] and Random Forest [3]. Compared to the first study, which leveraged XGBoost in a production environment, our optimized Random Forest model achieves higher or comparable predictive performance (e.g., F1 Score of 0.788 vs. 0.741 for H1), while incorporating important improvements in terms of methodology. Unlike the original work, our approach includes a rigorous evaluation framework using cross-validation, statistical testing (Friedman and Nemenyi), and a clear model selection procedure based on both performance and explainability. In relation to the second study, which used a default Random Forest model with limited tuning, our approach introduces a full hyperparameter optimization phase and a detailed pipeline design that includes data balancing via SMOTENC. Moreover, while both studies primarily focus on predictive accuracy, this work distinguishes itself by incorporating global and local explainability techniques to better understand model behavior and decision logic—an essential component for real-world adoption.

5.1.2 Final Considerations

This project addressed the problem of hotel booking cancellation prediction by developing a robust and explainable machine learning pipeline. The main contributions can be summarized as follows:

- **Data Preprocessing and Cleaning:** Careful handling of missing values, encoding of rare categories, removal of leakage-prone features, and creation of a domain-informed feature (*ADRThirdQuartileDeviation*).
- **Pipeline Design and Evaluation:** Two pipelines were compared—with and without SMOTENC—to address class imbalance. The SMOTENC-based pipeline was adopted for its superior recall, a priority in operational settings.

- **Hyperparameter Tuning and Model Selection:** A comprehensive grid search was conducted for eight classifiers, with performance compared against default configurations. Statistical tests (Friedman and Nemenyi) were used to identify the top 5 models for final evaluation.

- **Hold-Out Testing:** The selected models were evaluated on a temporally-aware test set. Random Forest achieved the best overall performance, with an F1 Score of 0.788 and ROC AUC of 0.914.

- **Explainability and Transparency:** Both global and local explainability tools were applied to support model interpretability and trust. SHAP values and feature importance rankings helped identify the key drivers of cancellations.

- **GUI Development:** A web-based interface was developed to facilitate model use by hotel staff, enabling batch predictions and interactive cancellation risk analysis.

Overall, this project demonstrates how a carefully designed machine learning system—grounded in robust preprocessing, balanced data handling, rigorous model selection, and explainability—can efficently handling this type of problem whit a good predictive accuracy and practical utility. By integrating SMOTENC for improved recall, performing exhaustive hyperparameter tuning with statistical validation, and leveraging explainability techniques such as SHAP and feature importances, the developed pipeline ensures not only high performance but also transparency and trust.

Bibliography

[1] N. Antonio, A. de Almeida, and L. Nunes, "Hotel booking demand datasets," https://doi.org/10.1016/j.dib.2018.11.126, Nov. 2019, published in Data in Brief, Volume 22, Pages 41–49.

[2] N. Antonio, A. M. D. Almeida, and L. Nunes, "Predicting hotel bookings cancellation with a machine learning classification model," https://doi.org/10.1109/ICMLA.2017.00-11, Dec. 2017, 2017 IEEE International Conference on Machine Learning and Applications (ICMLA).

[3] Z. A. Andriawan, Ricko, F. Wijayanto, S. R. Purnama, A. Wibowo, A. S. Darmawan, and A. Sugiharto, "Prediction of hotel booking cancellation using crisp-dm," https://doi.org/10.1109/ICICoS51170.2020.9299011, Oct. 2020, 4th International Conference on Informatics and Computational Sciences (ICICoS).