Forecasting Cancellation Flags: A Data-driven Approach to Hotel Reservation Cancellation Prediction

Naveen Mathews Renji - 20016323

EDA -

I would like to share with you some key findings from our Exploratory Data Analysis (EDA) on hotel booking data. Our analysis has uncovered interesting insights that can help hoteliers better understand their customers and improve their services.

First, we discovered that the majority of rooms were booked by couples (72%), followed by singles (22%), and families (7%). Interestingly, most people booked for weekends, with 17k bookings for 2 nights, while weekday bookings still accounted for a significant 11k bookings for 2 days.

Parking was not a priority for our guests, as 97% did not require it, indicating that most of them were likely from out of town. The booking patterns throughout the year showed that February was the peak season, accounting for 15% of bookings, probably due to Valentine's Day celebrations. The beginning of fall in September saw a surprising 12% of bookings, while January had the lowest bookings, possibly due to people spending time at home with family.

Arrival dates were evenly distributed throughout the month, but weekends generally saw more bookings. Couples also had fewer cancellations than other groups. In terms of booking channels, 63% were online, 30% offline, 5% corporate, 1% complimentary, and the remainder was for aviation.

Notably, only 3% of the bookings were from repeat customers, suggesting that guests prefer to try different hotels. As a result, precedence for cancellation was not a significant factor. Moreover, the majority of bookings (90%) were within the 50-100 dollar price range.

Lastly, 32% of bookings were cancelled, with most cancellations coming from online bookings.

In conclusion, our EDA has provided valuable insights into customer preferences, booking patterns, and pricing trends. These findings can inform hoteliers' strategies to enhance their offerings, improve customer satisfaction, and ultimately, increase revenue. Thank you for your attention, and I hope this information proves valuable for your businesses.

Pre - processing

Z score -

The efficient method to improve the quality of your datasets using the Z-score method for outlier removal in Python.

We begin by understanding what a Z-score is. It is a measure that indicates how far a data point is from the mean, in terms of standard deviations. A higher Z-score means that the data point is further away from the mean. This information is valuable when identifying and removing outliers in our datasets, as they can negatively affect the accuracy of statistical analysis or machine learning models.

In our method, we choose a threshold of 3 for identifying outliers in a normal distribution. This choice is based on the fact that approximately 99.7% of the data falls within three standard deviations from the mean. By selecting a threshold of 3, we aim to remove the data points in the extreme 0.3% of the distribution, which are likely to be outliers. This approach strikes a balance between preserving most of the data and removing potential outliers.

Step-by-step process
Set target column
Create dataframe copy
Separate target and feature columns
Calculate Z-scores
Calculate absolute values of Z-scores
Apply threshold of 3
Store cleaned feature indices
Combine target and cleaned feature columns
Reset index of dataframe
Print row counts before and after removal

KNN

In this code snippet, a k-Nearest Neighbors (KNN) classification algorithm is applied with different values of k (3, 5, and 10) to find the optimal value for the specific dataset. KNN is a machine learning algorithm used for classification and regression tasks. It determines the class of a new data point based on the majority class of its k nearest neighbors.
Here's what happens in the code:

1. For each value of k (3, 5, and 10), a KNN classification model is trained and evaluated.
2. The score, confusion matrix, and classification report are printed for each k value.
3. The accuracy, precision, recall, and f1-score are calculated to measure the performance of each model.

The results show that k=3 provides the highest accuracy, which means that considering the 3 nearest neighbors provides the best classification results for this specific dataset.
Using 3 neighbors in this case is useful because it leads to better performance compared to using more neighbors (k=5 and k=10). The KNN algorithm relies on the assumption that similar data points are close to each other in the feature space. By using fewer neighbors, the algorithm is more sensitive to local patterns in the data, which might be beneficial for the dataset at hand. It is important to note that the optimal value of k depends on the specific problem and dataset. The choice of k balances between overfitting (smaller k) and underfitting (larger k). In this example, k=3 showed the highest accuracy, indicating that considering only the three nearest neighbors provides the best balance between overfitting and underfitting for this dataset.

DT

I am excited to present an efficient approach to optimizing a Decision Tree Classifier using grid search and cross-validation in Python. This method allows us to find the best hyperparameters for our model, resulting in improved performance and generalization.
First, let's understand the importance of the two splitting criteria, Gini and entropy, in the context of decision trees. Both are used to measure the impurity or homogeneity of a node while constructing the tree, guiding the selection of the best feature to split on at each node. Gini is computationally faster, while entropy often results in more balanced trees.
Another crucial aspect of decision trees is their depth, which represents the length of the longest path from the root node to a leaf node. The depth of the tree influences the model's complexity and its propensity to overfit or underfit the data. Therefore, finding the optimal depth is essential for achieving the best performance.
In our code snippet, we perform a grid search to find the best combination of the splitting criterion and maximum depth for a Decision Tree Classifier. To ensure a reliable estimate of the model's performance and to prevent overfitting, we employ 5-fold cross-validation during the grid search process. This technique involves dividing the training data into five equal parts and iteratively training and validating the model on different subsets of the data.