Hotel Booking: From DataOps to Predictive Models & Business KPIs

Part 1: Exploratory Machine Learning for Booking Cancellations & Stakeholders Reporting

Project Management

This is a complementary project to '**Part 2: KPI Business Dashboard in Looker Studio for Strategic Planning Monitoring**'.

For detailed project management and automation information, please refer to the ***Full_Project_Explained.pdf***.

This file is common to both projects and solves all the questions.

<u>The reason for creating this extra file is to keep both projects' documentation short.</u>

Tasks

Task 1: Data Preprocessing

- Script file: ***preprocessing.py*** (extracts ***hotel_booking*** raw data from PostgreSQL, transforms it and finally loads cleaned ***logreg_rf_data*** and ***dashboard_data*** to PostgreSQL).

Task 2: Machine Learning (ML) for Cancellation Prediction

- Script file: ***logistic_regression.ipynb*** & ***random_forest.ipynb*** (extracts ***logreg_rf_data*** from PostgreSQL and generates machine learning outputs, such as evaluation information, odds and feature importance)

Task 3: Short Report to Stakeholders

- Document: ***Report_to_Stakeholders.pdf*** (this is based on exploratory ML results that indicate the most important features for booking cancellations).

Other Documents

Classification_Metrics_Interpretation.pdf → A brief interpretation file of machine learning classification metrics.

Logistic_Regression_Interpretation.pdf → A detailed logistic regression interpretation file.

Doc_ML_for_Cancel_Prediction.pdf → This is the project's general documentation file. Please also read ***prj_structure_and_instructions.pdf***.

prj_structure_and_instructions.pdf → See section's introduction above.

Goals

- To work hands-on with real-life, 'dirty' data.
- To develop an advanced traditional machine learning model (Random Forest) that beats the baseline model (Logistic Regression) for booking cancellations prediction.
- To prepare a thorough logistic regression interpretation PDF file. Since logistic regression can be used as a baseline model for most binary classification challenges, I considered it important not to overestimate my understanding and to keep detailed notes.
- To prepare a brief machine learning classification metrics interpretation file since I use them for model evaluation purposes.
- To present a real-life PDF report of what might go wrong with the booking cancellations to the owner of these two hotels.
- To fully prepare the ground for my second project over the same dataset.

Data Preprocessing

Handling Date-Related Columns

Months and days of month have a cyclical nature which is not profound in the 1 to 12-month format and the corresponding 1 to 31-day format. The model will treat them as raw values where 12 is bigger than 1 and so on. To fix this, I created a custom function which calculates the cosine and sine components of months and days. The function can properly treat all possible number of days a month has, as well as February days even if it is a leap year.

Data Leakage, High Cardinality, Data Anomalies & Strong Corellations

I dropped high cardinality columns which are unimportant for predicting cancellations. Additionally, columns associated with reservations must be dropped since they carry future information. For instance, if the reservation status date is earlier than the arrival date, the booking had already been canceled. The assigned room type was also dropped for the same reason.

There are some data anomalies in the deposit type and car parking columns. Specifically, 99% of people who chose a non-refundable deposit type canceled. Additionally, 100% of people who required a parking space didn’t cancel. These are very strange, and since I can't reach the stakeholders for further clarification, they can’t be relied upon. For this reason, I dropped them from the dataset.

Finally, the week number is highly correlated with the month of arrival, which makes sense. I decided to drop the week number and keep the month of arrival.

Remaining Preprocessing Steps

1) I checked for duplicates. Fortunately, there are no duplicates at this dataset.

2) I handled NaNs without dropping any feature, even the ones which have a great number of NaNs.

3) I applied some feature engineering creating total_kids feature combining ‘children’ and ‘babies’ features. Thus, reducing dimensionality by 1 without losing existing information.

4) For the rest of the columns, I merged feature categories either using custom functions (‘country’) or applying lambda functions (‘agent’, ‘company’ etc.), applied some mapping to turn string dtypes to reasonable integers (‘meal’) to name but a few.

5) I checked for outliers creating a custom function which returns a histogram but also showing the lower and upper whiskers.

6) I encoded categorical columns.

7) I leveraged statsmodels’s variance inflation factor (VIF) to make a quick check to my remaining features multicollinearity.

8) I uploaded the cleaned datasets to PostgreSQL local server.

Baseline Model (Logistic Regression)

The Reason of Modelling

The goal is to predict the is_canceled feature following the owner's request to understand the reasons behind the high cancellation rates. Cancellations are a significant issue in the hospitality industry, as a room is reserved until the cancellation date, potentially resulting in lost revenue. If the cancellation occurs late, the room may not be rebooked in time. Furthermore, if the accommodation facility lacks a property management system (PMS), a high volume of cancellations can create additional complications and administrative overload. This increases operational costs by adding extra work for employees who do not contribute to the hotel's revenue-generating activities. Such inefficiencies are prone to errors, which could lead to double bookings or false reservations, further disrupting hotel operations and customer experience.

Model Information

Logistic Regression (from now on: ‘logreg’) ran smoothly without encountering any challenges. I decided not to use GridSearchCV because I didn’t want to increase the runtime too much. I prefer to tune the hyperparameter values manually using trial-and-error techniques, reducing automation but conducting many trials in a short amount of time.

The best logreg model occurred with the ‘newton-cg’ solver, which is efficient for large datasets with a lot of features. Additionally, I changed class_weight to ‘balanced’ because my classes are slightly imbalanced and I don’t want to affect the dataset using either undersampling or oversampling techniques. Of course, I ran the logreg model with random_state equal to 42 to compare its results with the Random Forest (from now on: ‘RF’) model.

Other Issues to Note

1) RF will run using exactly the same cleaned dataset, as well as the same train-test split and scaling scripts. Stratification was used in the train-test split to account for imbalanced classes during separation, keeping the original percentages of each class the same in both sets.

2) The validation process will also be exactly the same for both models, although it doesn’t affect the model results. Imbalanced classes were also considered during validation using the StratifiedKFold instance with shuffle=True. I used 'recall' as the scoring metric because I consider it important to track false negatives (true cancellations that the model couldn’t predict).

3) I like to have control over evaluation results and interpretation. For this reason, I created two custom functions: one for model evaluation and one for interpretation.

Random Forest

Random Forest (RF) is an advanced traditional machine learning technique capable of both regression and classification tasks, making it a suitable choice for comparison against the baseline logistic regression model. The goal is to outperform the logreg model by generating better predictions for cancellations.

Model Information

RF ran smoothly without encountering any challenges. I did not use GridSearchCV for the same reasons mentioned earlier—specifically, to avoid increasing the runtime unnecessarily.

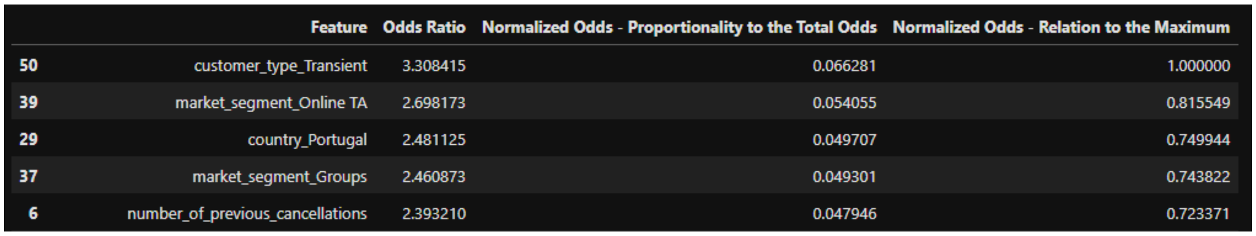
The best RF model was achieved with the criterion set to ‘entropy’, as it provided slightly better results compared to ‘gini’, while the runtimes were nearly identical. Additionally, I set max_features=None after noticing that this setting improved the results without leading to overfitting. Of course, I applied all the considerations from the previous section to ensure the comparison with the logreg model remained valid.

Models Evaluation Results

Metric	Logistic Regression	Random Forest
Accuracy	0.7866	0.8780
Balanced Accuracy	0.7825	0.8638
Recall	0.7663	0.8073
Precision	0.6954	0.8587
F1 Score	0.7291	0.8322
Φ Coefficient (MCC)	0.5555	0.7373
Runtime (Minutes)	0.06	0.36

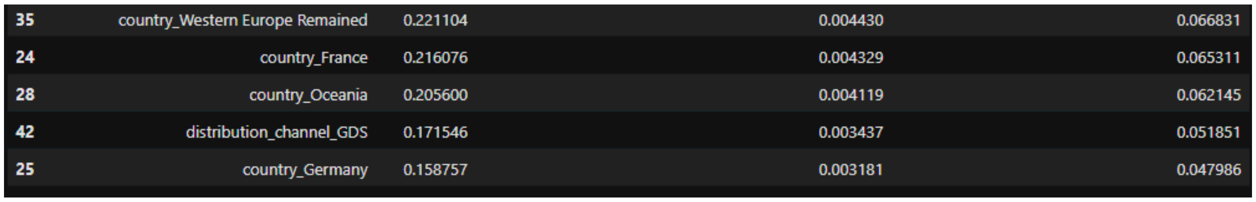
Logistic Regression Odds

The five most influential positive odds:



	Feature	Odds Ratio	Normalized Odds - Proportionality to the Total Odds	Normalized Odds - Relation to the Maximum
50	customer_type_Transient	3.308415	0.066281	1.000000
39	market_segment_Online TA	2.698173	0.054055	0.815549
29	country_Portugal	2.481125	0.049707	0.749944
37	market_segment_Groups	2.460873	0.049301	0.743822
6	number_of_previous_cancellations	2.393210	0.047946	0.723371

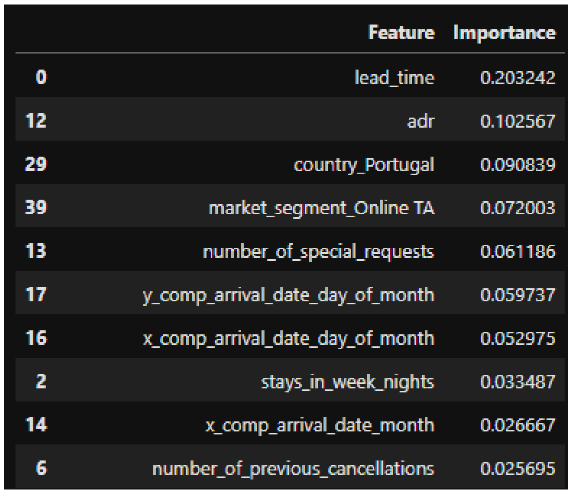
The five most influential negative odds:



35	country_Western Europe Remained	0.221104	0.004430	0.066831
24	country_France	0.216076	0.004329	0.065311
28	country_Oceania	0.205600	0.004119	0.062145
42	distribution_channel_GDS	0.171546	0.003437	0.051851
25	country_Germany	0.158757	0.003181	0.047986

Random Forest Feature Importance

The 10 most important features:



	Feature	Importance
0	lead_time	0.203242
12	adr	0.102567
29	country_Portugal	0.090839
39	market_segment_Online TA	0.072003
13	number_of_special_requests	0.061186
17	y_comp_arrival_date_day_of_month	0.059737
16	x_comp_arrival_date_day_of_month	0.052975
2	stays_in_week_nights	0.033487
14	x_comp_arrival_date_month	0.026667
6	number_of_previous_cancellations	0.025695

Results Discussion

Please refer to the ***Report_to_Stakeholders.pdf file***.