MACHINE LEARNING OPERATIONS

MASTER'S DEGREE IN DATA SCIENCE AND ADVANCED ANALYTICS

Predicting Resort Hotel Booking Cancellations

Group F:

Beatriz Sousa, nº 20220674
David Castro, nº 20220688
Miguel Cruz, nº 20221391

June 2023

TABLE OF CONTENTS

1. INTRODUCTION

This Machine Learning Operations project aims to simulate the real process of deploying machine learning models. In this project, we will work with a resort hotel dataset to predict which reservations are likely to be cancelled. By developing an accurate predictive model, we can provide hotel management with valuable information, allowing them to optimize their operations, make informed decisions, and improve guest satisfaction.

For the implementation of the project, we will adhere to the Kedro organization and modular code structure, which promotes maintainable and scalable codebases. This approach allows us to organize our project into distinct modules and pipelines, facilitating code reusability, and making it easier to maintain and extend our solution in the future. Within our Kedro project structure, we will make use of exploration and testing notebooks as a record of our data analysis, visualization, and initial insights, as well as other methods tested for feature selection or modelling. To build a robust pipeline, we aim to include unit data tests to ensure data quality, use MLflow to facilitate experimentation and model versioning, evaluate data drift to help monitor model performance, and comprehensive testing to guarantee the reliability of our code. This holistic approach ensures that our solution is scalable and can be reliably reproduced by others.

This report will explain the resort hotel dataset used in our project in-depth. Additionally, we will define the success metrics that we will use to evaluate the predictive model's performance. Then, we will outline our project planning approach. Afterwards, we will present the results obtained from our data exploration, modelling, and evaluation processes and point out significant conclusions. Finally, we will discuss the advantages, risks, and possible mitigations of the technologies we employed throughout the project.

2. DATASET EXPLANATION AND SUCCESS METRICS

In the hotel industry, managing demand is done through advanced bookings or reservations, with the option to cancel, which poses risks for hotels as they must honour existing bookings while facing the opportunity costs of vacant rooms when cancellations occur, leaving them little time to resell the room or offer it a discounted price. Over the years, the cancellation rate in Europe has risen from 33% to 40% (Hertzfeld, 2019). Since this is a big problem for hotels, and there is room for a data-driven solution, a project was developed within the scope of the Business Cases with Data Science curricular unit, to predict what reservations are going to be cancelled. The solution developed previously used a Jupyter Notebook to do all the exploration and preprocessing steps, as well as the modelling part, with the data imported as a Pandas DataFrame. So, we decided to transition from a machine learning project developed in Jupyter Notebooks to a Machine Learning Operations project, driven by the need for scalability, efficiency, reproducibility, and long-term support for a solution to the hotel. This transition enables us to deliver a tool that can handle larger datasets, accommodate increased computational requirements, and ensure reliable and efficient machine learning workflows. The advantages of the technologies used as well as other concerns will be mentioned below in this report.

This data set contains booking information for a city hotel and a resort hotel and can be found on Kaggle (Mostipak, 2020). However, for our specific project, we will only consider data from the resort hotel, as we do not want to mix data from different hotels, bearing in mind that customer behaviour can vary depending on the hotel in question and the types of vacation each hotel accommodates for. Still, the resort hotel dataset contains 40,060 rows, which is enough to perform a good analysis.

The used dataset includes customer information, like country, customer segment, and whether it is a repeated guest or not, as well as reservation information, such as date of arrival, number of weeknights or weekend nights, ADR (average daily rate), number of adult guests, child guests, and baby guests, among

others, making a total of 30 features in addition to the target variable we want to predict, *is_canceled*, which is a binary variable, labelled 0 when the reservation was not cancelled and 1 when it was. Also, to these data were added Portuguese holidays and weather data in Algarve between 2015 and 2017, because we believe that these can influence the reservations made and the behaviour of the customers.

In terms of metrics used to evaluate the quality of our model, we have multiple options, such as precision, accuracy, f1-score, and recall, since we are solving a classification problem. However, in the context of hotel churn prediction, recall is often the most important metric. It measures the model's ability to correctly identify customers who are likely to churn (it is a measure of the proportion of actual positive cases that are correctly identified as positive by the model). Maximizing recall (true positive rate) helps capture as many potential churners as possible, reducing the number of false negatives and ensuring that intervention strategies can be applied effectively to retain those customers. Thus, our success metric is reaching at least 85% Recall in the Test Set, knowing that values closer to 1 indicate better performance.

3. PROJECT PLANNING

Sprint 1: Data Exploration (2 days). Goal: Explore the dataset and comprehensively understand its structure, features, and potential challenges. Perform basic statistical analysis and visualization to identify patterns, anomalies, and potential areas of improvement.

Sprint 2: Data Preprocessing (2 days). Goal: Build a robust feature engineering pipeline that preprocesses and transforms the data to make it suitable for machine learning models. This will involve steps such as data cleaning, handling missing values, feature scaling, encoding categorical variables, and creating new features.

Sprint 3: Model Development (3 days). Goal: Develop and evaluate machine learning models using pre-processed data. Experiment with various algorithms and techniques, such as CatBoost and ExtraTrees. Perform model evaluation and select the best-performing model based on a predefined success metric (Recall > 0.85).

Sprint 4: Deployment and Monitoring (3 days). Goal: Prepare the selected model (CatBoost with the defined parameters) for deployment in a production environment. This involves registering the model and implementing monitoring mechanisms to track the model's performance, such as data drift detection and model versioning using MLflow.

Sprint 5: Testing and Documentation (1 day). Goal: Develop unit tests for the data, relevant functions, and pipelines to ensure the correctness and reliability of your code. Write comprehensive documentation that provides clear instructions on running the pipeline, reproducing the results, and understanding the code structure. Ensure that the code and documentation adhere to the Kedro organization and modular code principles.

4. RESULTS AND CONCLUSIONS

Before diving into modelling, it is crucial to explore and understand the dataset in detail, to know what it contains, how it is structured, and if there are issues with it needed to be addressed in data preprocessing. The dataset contained reservations from July 2015 to August 2017. The data types and missing values were checked, and the ones found in *company* and *agent* variables "should not be considered a missing value, but rather as "not applicable". For example, if a booking "Agent" is defined as "NULL" it means that the booking did not come from a travel agent" (António, 2019). Several incoherences were found and later removed on the preprocessing pipeline, such as reservations without people associated with them and reservations where

both the number of days booked for the weekend and the number of days booked for the week were 0. The main descriptive statistics were checked for the metric variables, and the mode as well as unique values for the non-metric variables. From the plots, it is possible to conclude that: most of the bookings include breakfast; most of the guests are Portuguese and come to the hotel through online travel agents; the most requested room type is A (which is considered to be the standard one); the busiest months are the summer ones (both August and July), keeping in mind that the data referring to those months includes 3 years when compared to the other months, that only includes 2 years. For the *deposit_type* variable, a misclassification problem can be found when plotting bar charts separating the observations according to the target value. Almost all reservations where the *deposit_type* is "Non Refund" were cancelled, which was not expected, therefore this variable cannot be used for business understanding. Regarding the *country* variable, we know that the information about the customers' nationality can only be confirmed after check-in (when the hotel's employees see customers' identification), so reservations that were cancelled with guests that never checked in at the hotel can have wrong nationalities in the dataset. Therefore, this feature is not trustable, and should not be used in the final model.

Including irrelevant or redundant features in the models can result in poor performance on unseen data (overfitting), and feature selection helps address this issue by focusing on the most informative and relevant features while eliminating noise, thereby improving the model's ability to identify meaningful patterns in the data. During the feature selection process (which was performed using several different methods) a problem with data leakage was identified. Data leakage occurs when information from the target variable, inappropriately influences the predictor variables. In this case, including the *reservation_status* variable in the model would lead to data leakage as it directly reflects the outcome of interest (whether the booking was cancelled or not). So, including this variable would give the model access to information available only after the booking has been cancelled or confirmed, which could cause optimistic performance metrics and incorrect assessment of the model's effectiveness, as well as the inability for the model to make predictions on new reservations made (that do not have this information). Similarly, other variables, such as *assigned_room_type* (known only upon customer check-in and being equal to *reserved_room_type* if the reservation is cancelled) and country, were removed from the dataset, along with the variables representing dates (after the proper data extraction). Further detailed information about the data understanding steps can be found in the "Data Exploration" notebook, provided inside the folder delivered.

When it comes to modelling, 9 different algorithms, with their parameters optimized using Bayesian Optimization, were tested and 3 algorithms were found to provide the best results: CatBoost, LGBM and ExtraTrees. Then, after further testing with feature selection methods, CatBoost was selected as the best algorithm (using recall). The parameters and techniques used, such as different feature selection methods, were further explored in mlflow to optimize the evaluation metrics and, especially, the recall. The final model, despite achieving a strong accuracy cross-validation score (84%), was unable to meet the success metric. The final recall, determined through cross-validation, was 55%.

Using SHAP, the most important features were found along with how their specific values impact the output of the model. *lead_time* had the strongest impact by far while *required_car_parking_spaces*, *arrival_date_year*, *booking_changes* and *adr* followed as the next most important predictors (figure 1). In Figure 2, it is possible to see that when the lead time or the year of the arrival date is higher, the client tends to cancel more. The same happens for the average daily rate of the rooms. When it comes to the number of booking changes and the number of car parking spaces required, as they are higher the bookings tend to be less cancelled. By using these insights, policies to reduce the cancellation rate or target the customers predicted to cancel can be created and improved.

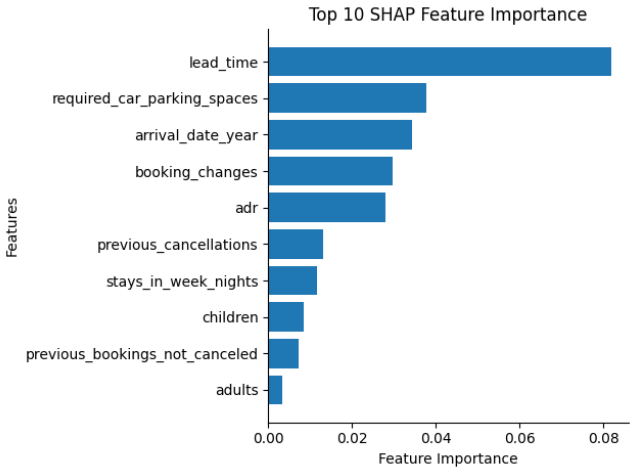

Figure 1 – SHAP Summary Plot 1

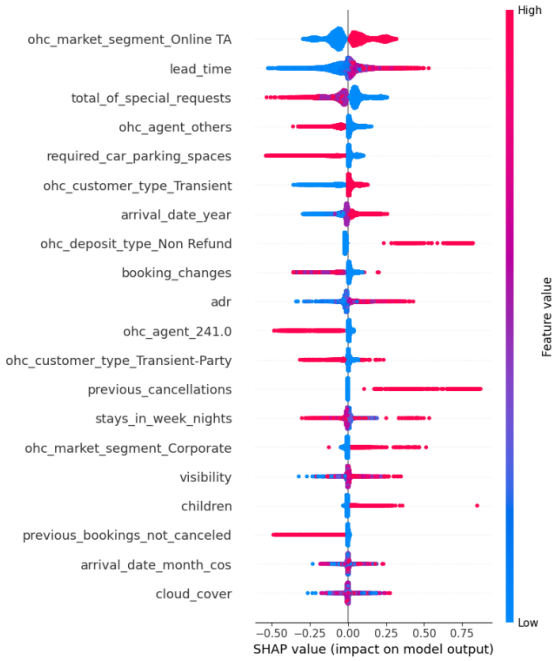

Figure 2 – SHAP Summary Plot 2

Data drift has a potential impact on the performance and reliability of machine learning models in production. When a model is trained on a specific data distribution (train data), it assumes that the future data it encounters during inference (test data) will follow a similar distribution. However, in real-world scenarios, the data distribution can change over time due to various factors such as shifts in user behaviour, evolving trends, or external influences. Comparing the train data with the test data allows us to assess whether the underlying data distribution has changed, which is crucial for maintaining the model's effectiveness. By detecting data drift and understanding its causes, we can identify when the model may be operating in an environment different from what it was trained on. The decision to use both univariate and multivariate drift detection techniques, along with an estimation of performance, was driven by the need to capture a holistic view of data drift. In all three techniques, we used a threshold with a std multiplier equal to 2 to the upper and lower limit and the chunk size set as default to split the data into 10 equal groups.

Univariate drift detection involves analysing individual input features or variables independently to detect any significant changes in their distributions over time. This allows us to pinpoint specific variables that may be contributing to the data drift. To perform the univariate drift detection, we used the

“jensen_shannon” method for the continuous variables and the “chi2” method to compare the categorical variables between the “X_test_data” with “X_train_data”. Thanks to this technique we could visualize that 5 columns out of 22 have some chunks out of the limit representing univariate drift.

On the other hand, multivariate drift detection considers the relationships and dependencies between multiple input features simultaneously, providing a more comprehensive understanding of the overall data drift patterns. For this case, we also compare the same data frames as before but now we applied the Data Reconstruction with the PCA method. Thanks to this technique, and besides the univariate drift perceived, we can state that globally there is no multivariate drift on the test data.

For the estimation of performance, we added 3 columns to the “X_test_data”: the prediction probabilities column “predicted_proba”, the averaged “prediction” column and the target “is_canceled”. Then we performed the Confidence-based Performance Estimation (CBPE) to estimate the performance of our best model. CBPE leverages the confidence scores associated with the model's predictions to estimate metrics without relying on the ground truth. Thanks to this technique we could visualize that both recall and accuracy had a significant decrease in its scores, meaning that our model's performance is below the desired level. This comprehensive approach enables us to detect, diagnose, and mitigate data drift effectively, ensuring the ongoing reliability and performance of the machine learning model in production.

5. ADVANTAGES OF THE TECHNOLOGIES USED, RISKS AND POSSIBLE MITIGATIONS

The first thing that we think about when working on a machine-learning model is its scalability and efficiency. As the amount of data increases, relying solely on Pandas for data storing and processing might become a problem, as Pandas is an in-memory data processing library, meaning it loads the entire dataset into memory for preprocessing, which can become problematic when dealing with large datasets that exceed the available memory capacity. Also, Pandas does not support distributed computing or parallel processing across multiple nodes, so, with large datasets, the computational workload cannot be easily distributed across multiple machines or clusters to scale horizontally. One possible solution would be to incorporate Spark, for example, allowing to take advantage of its ability to handle large-scale data processing by distributing the work through multiple machines. This way, the pipeline would be more efficient and scalable, even when the size of the data being used increases.

When deploying the pipelines and the models into a production environment, data quality and data governance policies are critical concepts that developers must consider. Data validation and cleansing techniques were incorporated to mitigate the risks of using raw data (data inconsistencies, missing values, potential privacy concerns, etc.). Some data unit tests were performed, to make sure that the data types of the variables collected were correct, the necessary columns were being extracted, the values collected were not outliers, etc. By defining expectations and constraints for the data (using the Great Expectation package), we can verify that it meets the desired criteria (missing values, data types, range validations, etc.), assuring automatic and regular validation of the pipelines, maintaining the reliability and robustness of the pipeline during ongoing development and deployment. By identifying data issues early, we can prevent them from being propagated throughout the pipeline, potentially saving time and resources, and minimizing the risk of making decisions based on flawed data.

Also, the usage of Git as a version control system to manage and track changes in the machine learning pipeline should be considered. Git allows us to maintain a complete history of code changes and provides a detailed timeline of modifications made, so we can track the evolutions of the pipeline and its inception to the final product. Since Git provides a collaborative environment, allowing multiple team members to work simultaneously on different aspects of the project (through branching and merging capabilities), it should

have been used during the development of this project to ensure that the work of each member was integrated and to prevent code conflicts.

Lastly, in a production environment, monitoring the performance of a deployed model and assessing the necessity to retrain it are essential practices. The models must remain accurate (in this case, predicting as many cancellations as possible, to minimize the false negatives, and maximize recall), aligned with the data patterns and business requirements, as well as updated. Knowing that models deployed in production usually suffer from degradation due to factors such as concept drift, data drift, or changes in user behaviour, monitoring the models is essential to identify significant drops in performance or anomalies. Concept drift occurs when the underlying data distribution shifts over time (making the model less effective), while data drift occurs when the input data's statistical properties change. So, when at least one of the previous concepts happens, the model's assumptions are no longer valid, and action needs to be done, for example, retraining the model periodically or based on performance degradation triggers.

6. LIST OF THE PACKAGES AND VERSIONS

Along with this report and the folder containing all the project's code, a *txt* file was provided. In this file, there is a list containing all the requested packages and virtual environment definitions to execute the project. Below, we provide some lines of code to replicate the project:

1. Create two virtual environments: one to execute all code, and another to evaluate data drift, since there is a package conflict for this step.
2. Activate the virtual environment to execute all the code and set the directory to the root folder of the project, using `cd "path"`.
3. To run all the pipelines, merging, data_split, preprocessing, target_split, feature_selection, hyperparameter_search, train, predict, use `kedro run`.
4. To run the data drift pipeline on Anaconda Prompt: `kedro run –pipeline drift_test`.

7. REFERENCES

Antonio, N., de Almeida, A., & Nunes, L. (2019). Hotel booking demand datasets. *Data in Brief*, 22, 41–49. https://doi.org/10.1016/j.dib.2018.11.126

Choosing univariate drift detection methods. Choosing Univariate Drift Detection Methods - NannyML 0.8.6 documentation. (2022). https://nannyml.readthedocs.io/en/stable/how_it_works/univariate_drift_comparison.html

Detecting data drift. Detecting Data Drift - NannyML 0.8.6 documentation. (2022). https://nannyml.readthedocs.io/en/stable/tutorials/detecting_data_drift.html

Estimation of performance of the monitored model. Estimation of Performance of the Monitored Model - NannyML 0.8.6 documentation. (2022). https://nannyml.readthedocs.io/en/stable/how_it_works/performance_estimation.html

Hertzfeld, E. (2019, April 23). *Study: Cancellation rate at 40% as otas push free change policy*. Hotel Management. https://www.hotelmanagement.net/tech/study-cancelation-rate-at-40-as-otas-push-free-change-policy

Mostipak, J. (2020, February 13). Hotel Booking Demand. Kaggle. https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand?resource=download