HOTEL BOOKING CANCELLATION CLASSIFICATION PROJECT

Artificial Intelligence Course Final Project

Ahmed Osama 2023/06328	Yehia Hany 2023/07891	Omar Alaa 2023/01277	Belal Ahmed 2023/05852	Malak Mowaffak 2023/03843

Part A) Data preprocessing and cleaning

- ➔ The objective of this step is to prepare the dataset for what is coming next. The raw dataset we initially have is huge with a lot of issues such as outliers , missing values and more.
- ➔ At the end of this step, our dataset is:
    - Free of missing values and duplicates
    - Cleaned from extreme outliers
    - Properly encoded and numerically formatted
- ➔ We will work on the raw data to prepare it and improve for it to be ready for the anaylsis and machine learning by performing the following tasks:

Step 1) Handle missing values.

Some columns in the dataset contain missing values, To fix this we did the following:

- Missing values in **children** were replaced with 0, assuming no children were present.
- Missing values in **country** were filled using the most frequent value (mode).
- Missing values in **agent** and **company** were filled with -1, indicating the absence of an agent or company.

Step 2) Fix Data Types

Certain columns, especially date and time fields, were stored as strings.
The reservation_status_date column was converted into a proper time format (Int) to allow the model to work with values it can interprete.

Step 3) Remove Duplicate Records

Duplicate rows were identified and removed to prevent redundant information from affecting the analysis or biasing the model.

Step 4) Handle Outliers

Extreme values in numerical columns such as **lead_time** and **average daily rate (adr)** can distort patterns and mislead the model.
The **Interquartile Range (IQR)** method was applied to detect and remove these extreme values, ensuring a more stable and representative dataset.

Step 5) Encode Categorical Variables

Since machine learning models require numerical inputs:

- Selected categorical variables were converted using **Label Encoding**.
- Remaining categorical variables were transformed using **One-Hot Encoding** to avoid introducing false ordinal relationships.

Part B) Exploratory Data Analysis (EDA)

- ➔ **The objective of this step is to explore and understand the dataset before building machine learning models.**
- ➔ To do this we plotted several diagrams and graphs that will help us better understand booking behavior and cancellation patterns.
- ➔ The following diagrams were plotted:

1. Cancellation Distribution
The distribution of canceled and non-canceled bookings was analyzed. The results show a moderate cancellation rate, And an imbalance between canceled and not canceled (64/36) which highlights a class imbalance problem.

2. Seasonal Bookings
Monthly booking trends revealed strong seasonality, with peak demand occurring during summer months and significantly lower activity during off-peak periods.

3. Weekly Booking Patterns
Weekly analysis showed variation in booking volume across different weeks.

4. Bookings per city
Booking activity is geographically concentrated, with a small number of cities accounting for a large portion of reservations and with the top city being Bhopal (7,632 bookings, 6.8% of total)

5. Average Daily Rate (ADR) and Cancellations
Canceled bookings were found to have a higher average daily rate compared to non-canceled bookings ($6.29 more on average)which says that customers cancel expensive bookings more often

6. Correlation Analysis
Correlation analysis of key numerical features showed that **lead time** has the strongest positive correlation with cancellations. Customers who book far in advance are more likely to cancel, making lead time a crucial predictive feature.

7. Lead Time Distribution
Lead time analysis revealed a right-skewed distribution, with most bookings made within a limited number of days before arrival, and fewer customers booking far in advance.

8. Categorical Feature Analysis
Cancellation rates vary significantly across hotel types, market segments, and customer types. These differences highlight distinct customer behaviors and confirm the importance of categorical features in predicting cancellations.

Part C) Feature engineering

➔ This is actually part D in the requirements but we chose to do it before so when we split the data in the next step the new features are present.
➔ In this step we ennhanced the dataset with meaningful new features while removing irrelevant or redundant ones (Highly correlated data)
➔ This step continues the preparation of the data for machine learning, improving predictive power and reducing noise or leakage.

Step 1) Creation of new features

Feature Name	Type	Description / How Calculated	Purpose / Insight
arrival_weekday	Integer	Day of the week of arrival (0=Monday, 6=Sunday)	Capture weekly patterns in bookings
arrival_week	Integer	Week number of the year for arrival	Identify seasonal weekly trends
arrival_season	Integer	Season of arrival (1=Winter, 2=Spring, 3=Summer , 4=Fall)	Detect seasonal booking patterns
total_stay	Integer	stays_in_weekend_nights + stays_in_week_nights	Total nights booked per reservation
total_guests	Integer	Adults + children + babies	Total number of people per booking
is_family	Binary	1 if children or babies > 0, else 0	Identify family bookings
adr_per_person	Float	adr / (total_guests + 1)	Normalized rate per guest, avoids bias for large groups
long_lead_time	Binary	1 if lead_time > median(lead_time), else 0	Flag bookings made far in advance

Step 2) Drop irrelevant and highly correlated features

➔ The following columns were removed to prevent **data leakage or redundancy**:

- arrival_date (raw date no longer needed)
- arrival_date_month (converted to numeric)
- reservation_status_date (potential leakage)
- company, agent, country (missing values, irrelevant for modeling)

➔ **Highly correlated features** that could inflate multicollinearity were dropped:

- distribution_channel_Direct
- assigned_room_type_H
- assigned_room_type_P
- reservation_status_Check-Out
- total_stay
- is_family

Part D) Checking data imbalance

➔ This step ensures the **target variable (`is_canceled`)** is not heavily imbalanced, which is crucial for machine learning models to avoid biasing the model toward the majority class.

Step 1: Train–Test Split

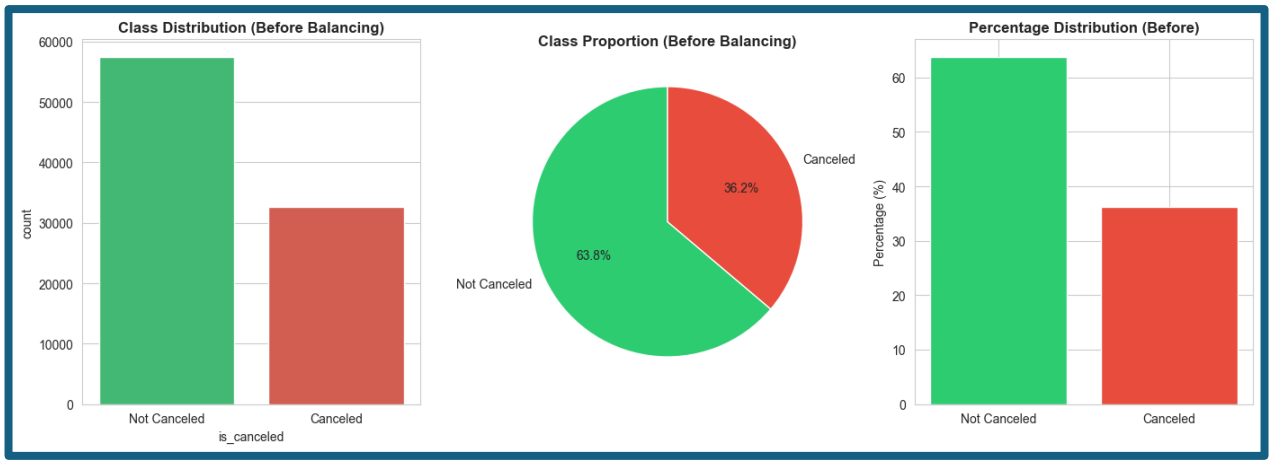
- The dataset is split **before balancing** into train and test sets.
- Only the **training set** will be balanced to **avoid data leakage**.

Step 2: Analyze Class Distribution (Training Only)

- Count how many bookings were canceled vs not canceled in the training set.
- Compute **imbalance ratio**: If the ratio > 1.5, the dataset is considered **significantly imbalanced**.

Step 3: Visualize Class Distribution

- Three visualizations help **understand imbalance**:
    1. **Count plot** (bar for each class)
    2. **Pie chart** (proportion of classes)



Step 4: Apply Class Balancing

- **Resampling (oversampling)** of the minority class (Canceled bookings) is applied **only to the training set**.
- The minority class is **duplicated randomly** until it matches the majority class size.
- After balancing, the classes are **perfectly 50/50**, preventing the model from being biased toward non-canceled bookings.

Part E) feature selection using genetic algorithm

Preventing Data Leaks and Verifying Correlation Sanity To prevent data leaking, a thorough cleaning phase was carried out prior to starting the feature selection process. Because they contain information about the booking outcome that would not be available at the time of prediction, variables like reservation_status (and its one-hot encoded derivatives) were specifically eliminated. A correlation study between the feature set and the goal variable (is_canceled) was carried out in order to further guarantee model integrity. Features that showed an absolute correlation higher than 0.9 were deemed "leakage-like" and eliminated. By doing this, the model is kept from depending on variables that cause significant multicollinearity or falsely expose the target.

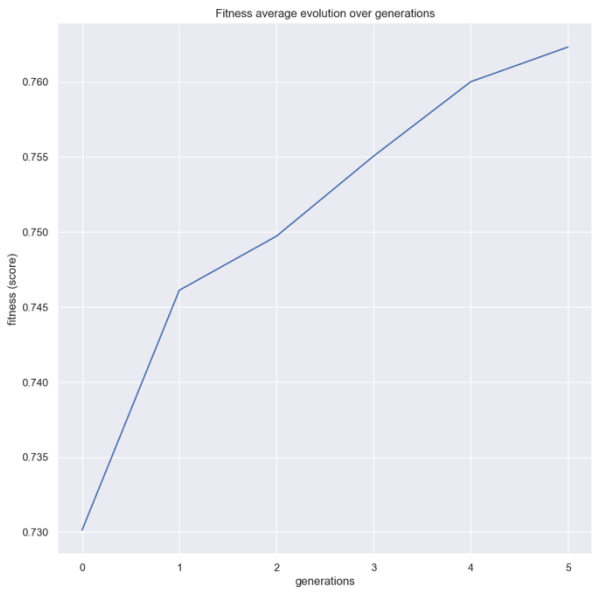
Methodology of Genetic Algorithms The project used the GAFeatureSelectionCV module to create a Genetic Algorithm (GA) to optimize the feature subset. The GA approaches feature selection as an evolutionary optimization issue, in contrast to conventional filter techniques (like Chi-Square) or wrapper techniques (like RFE). In order to enhance predictive accuracy, a population of candidate feature subsets is evolved across several generations, simulating natural selection.

Configuration and Parameters of the Algorithm Because of its computational efficiency and stability during repeated fitting, a Logistic Regression classifier (solver='liblinear', max_iter=500) was chosen as the basic estimator for the GA.

The following hyperparameters were used to set up the evolutionary process:

- Population and Generations: To balance computational expense with search space exploration, a population size of ten and a generation count of five were used.
- Evolutionary Operations: A mutation probability of 0.2 and a crossover probability of 0.5 were used. By combining useful features from several subsets and introducing new variables at random, these parameters help the algorithm avoid becoming trapped in local optima.
- Evaluation Metric: 5-fold cross-validation was used to assess each feature subset's fitness, and accuracy was used as the scoring metric.

Implementation and Visualization The feature set was iteratively improved by fitting the algorithm to the training data. After that, the support_ property was used to extract the ideal subset of features (those in the "best individual"). In order to confirm that the algorithm effectively converged toward a higher-accuracy solution, the method also produced a fitness evolution plot, which shows the trajectory of the best fitness scores over generations.



Part F) Model building

1. KNN: K-Nearest Neighbors The K-Nearest Neighbors classifier is the first algorithm used. The ideal number of neighbors (k) is found using a hyperparameter tuning loop.

   The model iterates over k values of [3, 5, 7, 9] during the tuning process.

   Selection: Every configuration is assessed on the validation set after being trained on the training set. The ideal parameter is chosen to be the k value that produces the best validation accuracy.

   Final Model: Using this optimal k value, a final KNN model is created and trained.

```
Best k: 9
Accuracy : 0.6552418161368614
Precision: 0.532061238074107
Recall   : 0.3922787502044823
F1 Score : 0.45160075329566857
Confusion Matrix:
 [[8671 2109]
 [3715 2398]]
```

2. Decision Trees :

   The second method captures non-linear correlations in the data by using a Decision Tree.

   Tuning Process: To manage complexity and avoid overfitting, the model verifies several maximum depths. The tested depths are [3, 5, 7, 9, None], with None permitting complete tree development.

   Selection: best_depth is the depth that yields the highest accuracy on the validation set.

   Final Model: This ideal depth plus a fixed random state for repeatability are used to train
   the final Decision Tree.

```
Best max_depth: 9
Accuracy : 0.8207541585272006
Precision: 0.7777777777777778
Recall   : 0.7065270734500245
F1 Score : 0.7404423109891994
Confusion Matrix:
 [[9546 1234]
 [1794 4319]]
```

3. Multi-Layer Perceptron (MLP):

Tuning Procedure: The hidden_layer_sizes are changed to adjust the architecture. A single layer of 50 neurons, a single layer of 100 neurons, and two layers of 50 neurons each (50, 50) are among the designs that were explored.

Parameters: The network employs the Adam solver with a maximum of 300 iterations and the ReLU activation function.

Selection: The architecture with the highest validation accuracy is selected.

Final Model: The entire training set is used to train an MLP classifier with the optimal hidden layer configuration.

```
Best hidden_layer_sizes: (100,)
Accuracy : 0.8202805895933227
Precision: 0.7627668659265585
Recall   : 0.7305741861606413
F1 Score : 0.7463235294117647
Confusion Matrix:
 [[9391 1389]
 [1647 4466]]
```

Part G) Performance evaluation

The main goal of evaluation is to compare how well the various Algorithm types (KNN, Decision Tree and MLP) performed on unseen data. The goal is to find out which technique has the highest generalization capacity for new/unseen data.

In addition, it provides a systematic way of generating a comprehensive set of classification metrics for each model such as Accuracy, Precision, Recall and F1 Score. Once all metrics for every model have been computed they are collated into a table for easy reference, with each metric being sorted in order of F1 Score. The result will provide users with an objective and evidence-based ranking of model performance, which is especially important when utilizing accuracy alone on the imbalanced dataset that can frequently skew results.

The confusion matrix created by the program for every algorithm gives you an in-depth description of how well or poorly a model performs. A confusion matrix includes the following information about all the events in a single instance: true positives, true negatives, false positives, and false negatives. This level of detail is important for indicating a model's weaknesses like if it tends to misclassify a certain class often.

The program then automatically identifies and outputs the 'best model' as defined by the F1 score. The use of the F1 score allows the program to balance precision and recall across all of the models and provide the ability to select the most reliable final prediction model.

```
...  ======================================================================
     PART G: PERFORMANCE EVALUATION
     ======================================================================


     Performance summary (sorted by F1):
                   accuracy  precision  recall      f1
     model
     MLP             0.8203     0.7628  0.7306  0.7463
     Decision Tree   0.8208     0.7778  0.7065  0.7404
     KNN             0.6552     0.5321  0.3923  0.4516

     Confusion Matrix for KNN:
     [[8671 2109]
      [3715 2398]]

     Confusion Matrix for Decision Tree:
     [[9546 1234]
      [1794 4319]]

     Confusion Matrix for MLP:
     [[9391 1389]
      [1647 4466]]

     Best model by F1 score: MLP (F1=0.7463)
```