Hotel Bookings Dataset Walkthrough

GTC ML Project 1 — Data Cleaning and Preprocessing

Mohamed Fouad Rabahi

Introduction

In this guide we explore how to take a real-world dataset (hotel bookings) and prepare it for predictive modeling. We focus on:

- Exploratory Data Analysis (EDA)
- Data Cleaning (handling missing values, duplicates, outliers)
- Preventing Data Leakage
- Feature Engineering
- Train/Test Split for modeling

Note: This workflow is typical in many machine learning projects. Cleaning and preparing data often takes much more time than model building itself!

Step 1: Exploratory Data Analysis (EDA)

Loading the Data

```
import pandas as pd

df = pd.read_csv("hotel_bookings.csv")
print(df.shape)
df.head()
```

Why? — To understand the dataset's structure: number of rows, columns, and first impressions.

Inspecting Missing Values

```
df.isna().sum().sort_values(ascending=False)
```

Concept: Missing values can bias models if not treated.

Tip: Visualize missingness. In Python, we can use `matplotlib` or libraries like `missingno`.

Checking Duplicates and Outliers

```
# Duplicates
df.duplicated().sum()

# Outliers with IQR
q1, q3 = df['adr'].quantile([0.25,0.75])
iqr = q3 - q1
lower, upper = q1 - 1.5*iqr, q3 + 1.5*iqr
outliers = ((df['adr'] < lower) | (df['adr'] > upper)).sum()
```

Concept: Outliers (extreme values) can distort models, especially in price-related columns like ADR (Average Daily Rate).

Step 2: Data Cleaning

Drop Leakage Columns

We immediately drop `reservation_status` and `reservation_status_date`.

```
df = df.drop(columns=["reservation_status", "reservation_status_date"])
```

Note: These contain future information (whether a booking was cancelled and the date), which would not be known at prediction time.

Handle Missing Values

- Company → replace with "None"
- Agent → replace with 0
- Country → replace with most frequent (mode)
- Children → replace with median

```
df['company'] = df['company'].fillna('None')
df['agent']   = df['agent'].fillna(0).astype(int)
df['country'] = df['country'].fillna(df['country'].mode()[0])
df['children']= df['children'].fillna(int(df['children'].median()))
```

Remove Duplicates

```
df = df.drop_duplicates()
```

Cap Outliers

```
df.loc[df['adr'] > 1000, 'adr'] = 1000
```

Step 3: Feature Engineering

Create Useful Features

```
df['total_guests'] = df['adults'] + df['children'] + df['babies']
df['total_nights'] = df['stays_in_weekend_nights'] + df['stays_in_week_nights']
df['is_family']    = ((df['children']>0) | (df['babies']>0)).astype(int)
```

Concept: Derived features often reveal hidden patterns (families may have different cancellation behavior than solo travelers).

Encoding High-Cardinality Columns

The `country` column has hundreds of values. We group rare ones and apply frequency encoding.

```
freq = df['country'].value_counts(normalize=True)
rare_countries = freq[freq < 0.01].index
df['country_grouped'] = df['country'].apply(
    lambda x: 'Other' if x in rare_countries else x)
country_freq = df['country_grouped'].value_counts(normalize=True).to_dict()
df['country_freq_enc'] = df['country_grouped'].map(country_freq)
```

Step 4: Preprocessing Pipeline

We automate imputations and encodings with a Scikit-learn pipeline.

```
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer

numeric_cols = df.select_dtypes(include=['int64','float64']).columns
low_card = ['meal','market_segment','distribution_channel',
            'deposit_type','reserved_room_type']

num_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='median'))
])

cat_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='constant', fill_value='Missing')),
    ('ohe', OneHotEncoder(handle_unknown='ignore', sparse_output=False))
])

preprocessor = ColumnTransformer([
    ('num', num_pipeline, numeric_cols),
    ('cat', cat_pipeline, low_card)
], remainder='passthrough')
```

Step 5: Train/Test Split

Finally, we prepare for modeling.

```
from sklearn.model_selection import train_test_split

X = df.drop(columns=['is_canceled'])
y = df['is_canceled']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

Key point: We use `stratify=y` so the train and test sets maintain the same proportion of cancellations.

Summary

- We performed EDA to understand the dataset.

- Cleaned data by handling missing values, duplicates, and outliers.
- Prevented leakage by dropping `reservation_status` and date.
- Engineered new features: `total_guests`, `total_nights`, `is_family`.
- Used encoding strategies for categorical data.
- Built a preprocessing pipeline and split into train/test sets.

Remember: Clean data ⇒ Better models. This workflow is a template you can apply to many ML problems.