**4. Handling Airport Column Values**

**In the dataset, the airport codes in the ORIGIN\_AIRPORT and DESTINATION\_AIRPORT columns might be represented inconsistently as either numbers or characters. To handle this, we need to ensure that these columns have a uniform data type. The approach taken is to convert all airport codes to strings. This ensures that when we perform operations like merging datasets or encoding categorical variables, there are no type mismatches.**

**Implementation:**

**python**

**# Convert airport codes to string to ensure consistent data type**

**flights\_df['ORIGIN\_AIRPORT'] = flights\_df['ORIGIN\_AIRPORT'].astype(str)**

**flights\_df['DESTINATION\_AIRPORT'] = flights\_df['DESTINATION\_AIRPORT'].astype(str)**

**airports\_df['IATA\_CODE'] = airports\_df['IATA\_CODE'].astype(str)**

**By converting airport codes to strings, we maintain consistency, which is crucial for further data processing steps such as merging and encoding.**

**5. Converting Characters to Numbers or Vice Versa**

**Machine learning models generally require numerical inputs to perform computations. Categorical data, like airport names or airline codes, must be converted into a numerical format to be used effectively by these models. This process is known as encoding.**

**In this notebook, LabelEncoder from sklearn.preprocessing is used to convert categorical variables into numerical values. Each unique category is assigned a unique integer value.**

**Implementation:**

**python**

**# Encode categorical variables**

**label\_encoder = LabelEncoder()**

**flights\_df['AIRLINE'] = label\_encoder.fit\_transform(flights\_df['AIRLINE'])**

**flights\_df['ORIGIN\_AIRPORT\_NAME'] = label\_encoder.fit\_transform(flights\_df['ORIGIN\_AIRPORT\_NAME'])**

**flights\_df['DESTINATION\_AIRPORT\_NAME'] = label\_encoder.fit\_transform(flights\_df['DESTINATION\_AIRPORT\_NAME'])**

**Converting categorical data into numerical format allows the machine learning algorithms to process the data effectively.**

**6. Handling Large Number of Airports**

**When dealing with a large number of unique categorical values (like airport names), it can lead to high-dimensional data, which can adversely affect model performance and computational efficiency. To mitigate this, LabelEncoder is used to convert airport names into numerical labels. This reduces the data dimensionality while preserving the uniqueness of each category.**

**Implementation:**

**python**

**flights\_df['ORIGIN\_AIRPORT\_NAME'] = label\_encoder.fit\_transform(flights\_df['ORIGIN\_AIRPORT\_NAME'])**

**flights\_df['DESTINATION\_AIRPORT\_NAME'] = label\_encoder.fit\_transform(flights\_df['DESTINATION\_AIRPORT\_NAME'])**

**By encoding airport names into integers, we ensure the data remains manageable in size and complexity, improving the efficiency of the machine learning model.**

**7. Predicting Flight Cancellation**

**The notebook outlines a process for predicting whether a flight will be canceled. This involves:**

**Data Preprocessing: Converting categorical variables to numerical values, scaling the features, and preparing the data for modeling.**

**Model Training: Training a machine learning model using the processed data.**

**Prediction: Using the trained model to predict flight cancellations on new data.**

**Implementation of Prediction Function:**

**python**

**def predict\_flight\_cancellation(model, scaler, label\_encoder, new\_data):**

**# Encode new data**

**new\_data\_encoded = new\_data.copy()**

**new\_data\_encoded['AIRLINE'] = new\_data\_encoded['AIRLINE'].apply(lambda x: encode\_new\_category(x, label\_encoder))**

**new\_data\_encoded['ORIGIN\_AIRPORT\_NAME'] = new\_data\_encoded['ORIGIN\_AIRPORT\_NAME'].apply(lambda x: encode\_new\_category(x, label\_encoder))**

**new\_data\_encoded['DESTINATION\_AIRPORT\_NAME'] = new\_data\_encoded['DESTINATION\_AIRPORT\_NAME'].apply(lambda x: encode\_new\_category(x, label\_encoder))**

**# Handle any -1 values in the encoded data (representing unknown categories)**

**new\_data\_encoded.fillna(-1, inplace=True)**

**# Scale the features**

**new\_data\_scaled = scaler.transform(new\_data\_encoded)**

**# Predict using the trained model**

**predictions = model.predict(new\_data\_scaled)**

**return predictions**

**This function encodes new data, handles unknown categories, scales the features, and makes predictions using the trained model.**

**8. Handling New Airports**

**When encountering new airports or categories that were not present in the training data, the model needs a strategy to handle these gracefully. The provided solution is to encode these new airports using a custom function that checks if the airport is in the LabelEncoder's classes. If the airport is not found, it returns -1.**

**Implementation:**

**python**

**# Function to handle new airports during inference**

**def encode\_new\_airport(airport\_name, encoder):**

**if airport\_name not in encoder.classes\_:**

**return -1**

**else:**

**return encoder.transform([airport\_name])[0]**

**# Function to handle new categories during inference**

**def encode\_new\_category(value, encoder):**

**if value not in encoder.classes\_:**

**return -1**

**else:**

**return encoder.transform([value])[0]**

**By returning -1 for unknown values, the model can still process the new data without failing, effectively treating the new category as a separate class. This approach avoids marking the new airport as entirely unknown, allowing for more nuanced handling during prediction.**

**Summary**

**Handling Airport Column Values: Convert all airport codes to strings to ensure consistency in data processing.**

**Converting Characters to Numbers or Vice Versa: Use LabelEncoder to convert categorical variables into numerical values, enabling machine learning models to process the data.**

**Handling Large Number of Airports: Encode airport names to numerical labels to reduce data dimensionality and improve computational efficiency.**

**Predicting Flight Cancellation: Preprocess the data, train a model, and make predictions on new data by encoding and scaling features.**

**Handling New Airports: Use custom functions to encode new categories, returning -1 for unknown values to integrate new data without marking it as unknown.**

**Cell 0: Importing Necessary Libraries**

* **Explanation**: This cell imports essential libraries for data visualization (seaborn, matplotlib), data manipulation (pandas), data splitting (train\_test\_split), and preprocessing (LabelEncoder, StandardScaler).

**Cell 1: Loading Data**

* **Explanation**: This cell loads the datasets airlines.csv, airports.csv, and flights.csv into pandas DataFrames.

**Cell 2 to 4: Displaying DataFrames**

* **Explanation**: These cells display the contents of the loaded DataFrames to understand their structure and contents.

**Cell 5: Ensuring Consistent Data Types**

* **Explanation**: This cell ensures that the airport codes in flights\_df and airports\_df are treated as strings, allowing for consistent merging and processing.

**Cell 6: Merging Origin Airport Names**

* **Explanation**: This cell merges the flights\_df DataFrame with the airports\_df to add the origin airport names, renaming the column and dropping the redundant IATA\_CODE column.

**Cell 7: Merging Destination Airport Names**

* **Explanation**: This cell performs a similar merge operation to add the destination airport names to the flights\_df DataFrame.

**Cell 8: Encoding Categorical Variables**

* **Explanation**: This cell encodes categorical variables (airline, origin airport name, and destination airport name) into numerical values using LabelEncoder.

**Cell 45: Handling New Airports During Inference**

* **Explanation**: This function checks if an airport name is in the LabelEncoder's classes. If not, it returns -1 to handle new or unseen airports.

**Cell 46: Handling New Categories and Making Predictions**

* **Explanation**: This cell defines functions to handle new categories during inference and to predict flight cancellations. It encodes new data, scales features, and makes predictions using a trained model. The example usage demonstrates how to predict cancellations for new data.