Lesson: Data Understanding and Preprocessing (Part 1)

In this session, we will focus on understanding and cleaning datasets to prepare them for machine learning. We'll use simple datasets like car prices and student scores for practice.

Topics Overview

1. **Exploring Datasets using Pandas and NumPy**
   Learn how to load, inspect, and explore datasets.
2. **Handling Missing Values**
   Fill missing data using techniques like replacing with the mean, median, or mode.
3. **Identifying and Removing Duplicates**
   Ensure data consistency by identifying and removing duplicate rows.

Example 1: Exploring and Cleaning a Dataset (Car Prices)

Dataset (car_prices.csv)

Model	Price	Year	Mileage	Fuel_Type
Toyota	20000	2018	30000	Petrol
Honda	NaN	2017	40000	Diesel
BMW	30000	NaN	20000	Petrol
Toyota	20000	2018	30000	Petrol

Step-by-Step Code and Explanation

```
import pandas as pd

# Load the car prices dataset
car_data = pd.read_csv('car_prices.csv')

# Display the original dataset
print("Original Dataset:")
print(car_data)

# Step 1: Check for missing values
print("\nMissing Values:")
print(car_data.isnull().sum())

# Step 2: Fill missing values
# Fill missing 'Price' with the mean
car_data['Price'] = car_data['Price'].fillna(car_data['Price'].mean())

# Fill missing 'Year' with the median
car_data['Year'] = car_data['Year'].fillna(car_data['Year'].median())

# Step 3: Check for duplicates
print("\nDuplicate Rows:")
```

```
print(car_data.duplicated().sum())

# Step 4: Remove duplicates
car_data = car_data.drop_duplicates()

# Display the cleaned dataset
print("\nCleaned Dataset:")
print(car_data)
```

Explanation of Steps

1. **Checking for Missing Values**
   - **`isnull().sum()`** counts missing values in each column.
2. **Filling Missing Values**
   - Replacing missing `Price` with the **mean** using `fillna()`.
   - Replacing missing `Year` with the **median** using `fillna()`.
3. **Removing Duplicates**
   - **`duplicated()`** checks for duplicate rows.
   - **`drop_duplicates()`** removes duplicate rows.

Example 2: Handling Missing Values (Student Scores)

Dataset (student_scores.csv)

Name	Math	Science	English
Alice	85	90	88
Bob	NaN	85	78
Charlie	80	NaN	92
Alice	85	90	88

Step-by-Step Code and Explanation

```
# Load the student scores dataset
student_scores = pd.read_csv('student_scores.csv')

# Display the original dataset
print("Original Dataset:")
print(student_scores)

# Step 1: Check for missing values
print("\nMissing Values:")
print(student_scores.isnull().sum())

# Step 2: Fill missing values
# Fill missing 'Math' scores with the mean
student_scores['Math'] =
student_scores['Math'].fillna(student_scores['Math'].mean())

# Fill missing 'Science' scores with the median
```

```
student_scores['Science'] =
student_scores['Science'].fillna(student_scores['Science'].median())

# Step 3: Check for duplicates
print("\nDuplicate Rows:")
print(student_scores.duplicated().sum())

# Step 4: Remove duplicates
student_scores = student_scores.drop_duplicates()

# Display the cleaned dataset
print("\nCleaned Dataset:")
print(student_scores)
```

Example 3: Visualization of Missing Data

We can visually inspect missing data using a heatmap.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Visualize missing data in the student_scores dataset
plt.figure(figsize=(8, 6))
sns.heatmap(student_scores.isnull(), cbar=False, cmap="YlGnBu")
plt.title("Missing Data Heatmap")
plt.show()
```

Explanation

- **`sns.heatmap()`** highlights missing values in the dataset for easy visualization.

Example 4: Advanced Example (E-Commerce Data)

Dataset (ecommerce_data.csv)

Product	Price	Quantity	Rating
Laptop	1000	5	4.5
Phone	NaN	10	4.0
Tablet	500	NaN	3.5
Laptop	1000	5	4.5

Code to Clean the Dataset

```
# Load the e-commerce dataset
ecommerce_data = pd.read_csv('ecommerce_data.csv')

# Display the original dataset
print("Original Dataset:")
print(ecommerce_data)
```

```
# Step 1: Check for missing values
print("\nMissing Values:")
print(ecommerce_data.isnull().sum())

# Step 2: Fill missing values
# Fill missing 'Price' with the mean
ecommerce_data['Price'] =
ecommerce_data['Price'].fillna(ecommerce_data['Price'].mean())

# Fill missing 'Quantity' with the median
ecommerce_data['Quantity'] =
ecommerce_data['Quantity'].fillna(ecommerce_data['Quantity'].median())

# Step 3: Check for duplicates
print("\nDuplicate Rows:")
print(ecommerce_data.duplicated().sum())

# Step 4: Remove duplicates
ecommerce_data = ecommerce_data.drop_duplicates()

# Display the cleaned dataset
print("\nCleaned Dataset:")
print(ecommerce_data)
```

Penguins Dataset (Classification of Penguin Species)

The Penguins dataset contains data on penguin species along with their measurements like bill length and depth.

```
import seaborn as sns
from sklearn.model_selection import train_test_split
import pandas as pd

# Step 1: Load the Penguins dataset from seaborn
penguins = sns.load_dataset('penguins')

# Step 2: Drop rows with missing values
penguins = penguins.dropna()

# Step 3: Convert categorical labels to numeric values
penguins['species'] = penguins['species'].astype('category').cat.codes

# Step 4: Display the first few rows of the dataset
print("Penguins Dataset Preview:")
print(penguins.head())

# Step 5: Split the dataset into features and labels
# Features: All columns except 'species'
features = penguins.drop(columns=['species'])
labels = penguins['species']

# Step 6: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, labels,
test_size=0.3, random_state=42)
```

```
# Step 7: Display the sizes of the split data
print("\nTraining Features Size:", X_train.shape)
print("Testing Features Size:", X_test.shape)
print("Training Labels Size:", y_train.shape)
print("Testing Labels Size:", y_test.shape)

# Step 8: Display sample training data
print("\nTraining Features Sample:")
print(X_train.head())
print("\nTraining Labels Sample:")
print(y_train.head())
```

Explanation

1. **Seaborn's `load_dataset`** provides the Penguins dataset.
2. Rows with missing data are removed using `dropna`.
3. The species column, which is categorical, is converted to numeric codes (e.g., Adelie -> 0, Chinstrap -> 1, Gentoo -> 2) for classification.
4. Data is split into 70% training and 30% testing.

Titanic Dataset (Survival Prediction)

This dataset predicts whether passengers survived the Titanic disaster based on their demographic and travel information.

```
from sklearn.model_selection import train_test_split
import pandas as pd

# Step 1: Load the Titanic dataset
titanic = sns.load_dataset('titanic')

# Step 2: Drop unnecessary columns and rows with missing values
titanic = titanic.drop(columns=['deck', 'embark_town', 'alive', 'class', 'who',
'adult_male', 'alone'])
titanic = titanic.dropna()

# Step 3: Convert categorical columns to numeric values
titanic['sex'] = titanic['sex'].astype('category').cat.codes
titanic['embarked'] = titanic['embarked'].astype('category').cat.codes

# Step 4: Display the first few rows of the dataset
print("Titanic Dataset Preview:")
print(titanic.head())

# Step 5: Split the dataset into features and labels
# Features: All columns except 'survived'
features = titanic.drop(columns=['survived'])
labels = titanic['survived']

# Step 6: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, labels,
test_size=0.25, random_state=42)

# Step 7: Display the sizes of the split data
```

```
print("\nTraining Features Size:", X_train.shape)
print("Testing Features Size:", X_test.shape)
print("Training Labels Size:", y_train.shape)
print("Testing Labels Size:", y_test.shape)

# Step 8: Display sample training data
print("\nTraining Features Sample:")
print(X_train.head())
print("\nTraining Labels Sample:")
print(y_train.head())
```

Explanation

1. The Titanic dataset is loaded via `seaborn` and cleaned by dropping irrelevant columns and rows with missing values.
2. Categorical columns (`sex` and `embarked`) are converted into numeric codes.
3. Features include information like age, sex, fare, and Pclass, while the label is `survived` (1 = survived, 0 = did not survive).
4. Data is split into 75% training and 25% testing.

Summary of What We Learned

1. **Exploring Datasets**
   - Methods like `head()`, `info()`, and `describe()` help us understand datasets.
2. **Handling Missing Values**
   - Replace missing values with the **mean**, **median**, or **mode** depending on the context.
3. **Removing Duplicates**
   - Identify duplicates with `duplicated()` and remove them with `drop_duplicates()`.
4. **Visualizing Missing Data**
   - Use Seaborn heatmaps to quickly identify where data is missing.

By practicing with different datasets, we now have a clear understanding of how to clean and preprocess data for further analysis.