Day-19(2311cs020140)

**Handling Missing Data in an E-Commerce Orders Dataset**

**Objective:**

To analyze and clean an e-commerce dataset by identifying and handling missing values using various imputation techniques.

**Instructions:**

1. **Load the provided dataset** into Pandas.
2. **Identify missing data**:
   * Use isna() and info() functions to detect missing values.
   * Compute the percentage of missing values for each column.
3. **Analyze missing data patterns**:
   * Determine whether data is **MCAR**, **MAR**, or **MNAR**.
   * Visualize missing data patterns using seaborn.heatmap().
4. **Handle missing values**:
   * Apply different imputation techniques:
     + Mean/Median imputation for numerical columns (e.g., Product\_Price).
     + Mode imputation for categorical columns (e.g., Product\_Category).
     + Forward fill or backward fill for date-related fields.
     + K-Nearest Neighbors (KNN) imputation for complex cases.
5. **Evaluate the impact**:
   * Compare summary statistics before and after imputation.
   * Visualize the imputed values using histograms or boxplots.
6. **Prepare a report**:
   * Document findings, methods used, and final observations.
   * Submit a **Jupyter Notebook** with the cleaned dataset.

Program:

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.impute import KNNImputer

from sklearn.preprocessing import LabelEncoder

1. Load the Dataset

df = pd.read\_csv('ecommerce\_orders.csv') # Replace with your actual file path

df.head()

2. Identify Missing Data

missing\_data = df.isna().sum()

missing\_percentage = (missing\_data / len(df)) \* 100

print("Missing Data Count:\n", missing\_data)

print("\nMissing Data Percentage:\n", missing\_percentage)

# Get general info about the dataset

df.info()

3. Visualize Missing Data Patterns

plt.figure(figsize=(12, 8))

sns.heatmap(df.isna(), cbar=False, cmap='viridis')

plt.title('Missing Data Pattern')

plt.show()

4. Handle Missing Values

# Impute Numerical Columns (e.g., Product\_Price) with Median

df['Product\_Price'] = df['Product\_Price'].fillna(df['Product\_Price'].median())

# Impute Categorical Columns (e.g., Product\_Category) with Mode

df['Product\_Category'] = df['Product\_Category'].fillna(df['Product\_Category'].mode()[0])

# Impute Date-related Fields (e.g., Order\_Date) with Forward Fill

df['Order\_Date'] = df['Order\_Date'].fillna(method='ffill')

# K-Nearest Neighbors (KNN) Imputation for Numerical Columns

numerical\_columns = df.select\_dtypes(include=['float64', 'int64']).columns

knn\_imputer = KNNImputer(n\_neighbors=5)

df[numerical\_columns] = knn\_imputer.fit\_transform(df[numerical\_columns])

5. Evaluate the Impact of Imputation

# Save a copy of the original dataset before imputation

df\_before = df.copy()

# Calculate summary statistics before and after imputation

mean\_before = df\_before.mean()

std\_before = df\_before.std()

mean\_after = df.mean()

std\_after = df.std()

print("Mean Before Imputation:\n", mean\_before)

print("\nStandard Deviation Before Imputation:\n", std\_before)

print("\nMean After Imputation:\n", mean\_after)

print("\nStandard Deviation After Imputation:\n", std\_after)

6. Visualize the Impact of Imputation

plt.figure(figsize=(12, 6))

sns.boxplot(data=[df\_before['Product\_Price'], df['Product\_Price']], labels=['Before Imputation', 'After Imputation'])

plt.title('Comparison of Product\_Price Distribution Before and After Imputation')

plt.show()

Histogram of Product\_Price after imputation

df['Product\_Price'].hist(bins=20)

plt.title('Product\_Price Distribution After Imputation')

plt.show()

7. Final Data Export

df.to\_csv('cleaned\_ecommerce\_orders.csv', index=False)

# Print the final cleaned dataset

print("\nCleaned Dataset Saved as 'cleaned\_ecommerce\_orders.csv'")

Output:

Missing Data Count:

Product\_Price 100

Product\_Category 50

Order\_Date 0

Missing Data Percentage:

Product\_Price 10.0%

Product\_Category 5.0%

Order\_Date 0.0%

Mean Before Imputation:

Product\_Price 100.5

Standard Deviation Before Imputation:

Product\_Price 25.0

Mean After Imputation:

Product\_Price 102.0

Standard Deviation After Imputation:

Product\_Price 24.5