

| **Title: Implement data pre-processing using python on real world dataset** |
| --- |

# Course Outcome:

# CO1 Understand basic concepts of data analytics to solve real-world problems

# Books/ Journals/ Websites referred:

https://www.kaggle.com/

# Resources used:

https://www.kaggle.com/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Theory (About Data Preprocessing):

(Students should write)

# Program:

import pandas as pd

import numpy as np

**# Sample data**

data = {

'name': ['Alice', 'Bob', 'Charlie', 'Dave', 'Eve'],

'age': [25, np.nan, 30, 22, 35],

'gender': ['F', 'M', 'M', 'M', 'F'],

'income': [50000, 60000, 75000, np.nan, 80000]

}

df = pd.DataFrame(data)

**# Display the original data**

print("Original DataFrame:")

print(df)

**# User-defined function for discretization**

def discretize\_age(age):

if age < 30:

return 'Young'

elif age >= 30 and age < 40:

return 'Middle-aged'

else:

return 'Old'

**# Handling missing values (NaN)**

**# Fill missing values in 'age' with the mean age**

mean\_age = df['age'].mean()

df['age'].fillna(mean\_age, inplace=True)

**# Apply discretization function to 'age' column**

df['age\_category'] = df['age'].apply(discretize\_age)

**# Drop rows with missing values in any column**

df.dropna(inplace=True)

**# Convert categorical variables (gender) to numerical**

df['gender'] = df['gender'].map({'F': 0, 'M': 1})

**# Data normalization Min -Max**

**# Normalize 'income' column to range [0, 1]**

min\_income = df['income'].min()

max\_income = df['income'].max()

df['income\_normalized'] = (df['income'] - min\_income) / (max\_income - min\_income)

**# Display cleaned, preprocessed, and discretized data**

print("\nCleaned, Preprocessed, and Discretized DataFrame:")

print(df)

—-----------------------------------------------------------------------------------------------------------------------

**Task: Download the real time data set and implement data preprocessing techniques on the real time data set**

# Source of the dataset (URL): [Toyota](https://docs.google.com/spreadsheets/d/1O4vn0aGWA0s2g8Uu2sqrUcD8_rqFfTtsDj2yzqD2SKU/edit?gid=896466594#gid=896466594)

# Platform used by the student: Microsoft Excel, Python

# Following points should be written by students

# Different steps in Data Preprocessing:

# Finding missing, null values

# Replacing missing, null values with statistical parameters

# Encoding categorical data if needed (Write user defined function)

# Normalization (Write user defined function)

# Discretization (Write user defined function)

# Finding missing, null values

# Conclusion (Students should write in their own words):

**Post lab questions:**

**Q.1 What are some common challenges encountered during data cleaning? How did you handle missing values in the provided dataset?**

Common Challenges in Data Cleaning and Handling Missing Values-

Inconsistencies in Data Format: Data might be stored in various formats (e.g., date formats, numerical precision), leading to inconsistencies.

Duplicate Records: Identical records can skew analysis and model performance.

Missing Values: Missing data can occur randomly or systematically and can affect the robustness of analyses.

Outliers: Extreme values might be errors or might represent valid variations; handling them requires careful consideration.

Data Entry Errors: Typos and errors in data entry can lead to incorrect analysis.

Irrelevant or Redundant Features: Some features may not contribute to the analysis and can clutter the dataset.

Handling Missing Values:

Imputation: Filling missing values with mean, median, mode, or using advanced techniques like K-nearest neighbors (KNN) imputation.

Prediction Models: Using machine learning algorithms to predict missing values based on other features.

Deletion: Removing rows or columns with excessive missing values, especially if they are not crucial to the analysis.

Flagging Missing Data: Creating an additional binary feature to indicate whether data was missing.

The choice of method depends on the nature of the data and the extent of the missingness. For instance, if missing values are minimal and randomly distributed, imputation might be appropriate. If they are systematic, more sophisticated methods or model adjustments might be needed.

**Q.2 Explain the importance of data normalization in the context of machine learning models. How does normalizing benefit the analysis?**

Importance:

Consistent Scale: Normalization ensures that all features contribute equally to the model’s learning process, preventing features with larger ranges from dominating.

Improved Convergence: For algorithms that use gradient descent (e.g., neural networks), normalization speeds up convergence and improves the efficiency of the training process.

Enhanced Performance: Algorithms like K-nearest neighbors (KNN) and support vector machines (SVMs) rely on distance metrics, and normalization ensures that distance calculations are not skewed by varying scales of features.

Better Interpretability: Normalized data often makes it easier to interpret the influence of each feature on the outcome.

Benefits:

Prevents Bias: Normalization prevents bias in models that are sensitive to the scale of the input features.

Facilitates Comparisons: Normalized data allows for easier comparison of feature importance and results.

Stabilizes Numerical Computations: It helps in maintaining numerical stability in calculations, reducing the risk of computational errors.

**Q.3 Discuss why it's essential to convert categorical variables like 'gender' into numerical representations.**

Reasons:

Algorithm Requirements: Most machine learning algorithms require numerical input, as they rely on mathematical computations that cannot process categorical data directly.

Distance Calculations: Many algorithms (e.g., KNN, clustering) use distance metrics, which need numerical inputs to compute distances between data points.

Model Interpretability: Numerical encoding of categorical variables can help in assessing the impact of these variables on the outcome of the model.

Common Methods:

One-Hot Encoding: Converts categorical variables into binary vectors where each category is represented as a unique vector. This avoids ordinal relationships between categories but can result in high-dimensional data.

Label Encoding: Assigns a unique integer to each category. While simpler, this method can introduce ordinal relationships that may not be appropriate for nominal categories.

Binary Encoding: Combines the properties of both one-hot and label encoding, offering a compact representation while avoiding some of the issues of high dimensionality.

Choosing the right method depends on the nature of the categorical variable and the machine learning algorithm being used.