**Batch: B - 1**

**Roll No.: 16014022050**

**Experiment No.: 1**

|  |
| --- |
| 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.geeksforgeeks.org/data-preprocessing-machine-learning-python/>

# Resources used:

# Kaggle

# Theory (About Data Preprocessing):

# Data preprocessing is a crucial step in data analysis and machine learning. It involves preparing raw data to ensure that models and algorithms can analyze it effectively. Preprocessing can significantly impact the performance and accuracy of the model. The key steps involved in data preprocessing include handling missing values, encoding categorical data, normalization, and discretization. Here's a detailed explanation of these concepts:

# Finding missing and null values:

# In any dataset, missing or null values can cause significant issues during analysis or model training. Identifying these values is the first step in preprocessing. Missing values might occur due to various reasons, such as incomplete data entry or data corruption. It's essential to identify these gaps and take corrective actions.

# Method – The presence of missing values can be detected using functions such as isnull() in pandas, which returns a DataFrame of the same shape, where each cell is True if the original cell is null, otherwise False. By summing this DataFrame, we can get the count of missing values in each column.

# Replacing Missing and Null Values with Statistical Parameters:

# Once missing values are identified, the next step is to handle them appropriately. One common approach is to replace missing values with statistical parameters like the mean, median, or mode of the respective columns.

# Method –

# Mean: Used when data distribution is normal.

# Median: Used when data has outliers or is skewed.

# Mode: Used for categorical data where the most frequent value is preferred.

# Replacing missing values with these statistics helps retain the integrity of the dataset while minimizing bias introduced by missing data.

# Encoding Categorical Data:

# Categorical data represents variables that contain label values rather than numerical values. Most machine learning algorithms require input data to be numeric. Thus, categorical data needs to be converted into a numerical format. This process is known as encoding.

# Normalization

# Normalization is the process of scaling numeric data to a standard range, typically [0, 1]. This ensures that all features contribute equally to the model's learning process and prevents features with larger ranges from dominating those with smaller ranges. It’s particularly important in algorithms like k-NN, SVM, and neural networks.

# Discretization

# Discretization involves converting continuous data into discrete buckets or bins. This is particularly useful when the relationship between the target variable and a feature is non-linear or when a continuous variable needs to be categorized for better interpretability.

# ---------------------------------------------------------------------------------------------------------------

# 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): [Titanic - Machine Learning from Disaster | Kaggle](https://www.kaggle.com/competitions/titanic/overview)

# Platform used by the student: Kaggle

# 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)

# Working (Paste the code and Output for each Data Preprocessing task):

# Students need to write comments wherever needed.

# Program –

# import pandas as pd

# import numpy as np

# # Load the dataset

# titanic = pd.read\_csv('/content/titanic.csv', header=0, dtype={'Age': np.float64})

# # Deleting less useful columns

# titanic.drop(columns=["Name", "Ticket", "Fare", "Cabin"], inplace=True)

# # Finding and filling missing values in the Age column based on Survived status

# mean\_age = titanic.groupby('Survived')['Age'].transform('mean')

# titanic['Age'] = titanic['Age'].fillna(mean\_age)

# # Removing rows containing NaN

# titanic.dropna(inplace=True)

# # Encoding categorical data for the Sex column

# titanic['Gender'] = titanic['Sex'].map({'male': 1, 'female': 2})

# # Deleting Sex column, since no use of it now

# titanic.drop(columns=['Sex'], inplace=True)

# # Encoding the Embark column

# titanic['Embark'] = titanic['Embarked'].map({'S': 1, 'Q': 2, 'C': 3})

# # Normalizing the Age column

# titanic['Age'] = (titanic['Age'] - titanic['Age'].min()) / (titanic['Age'].max() - titanic['Age'].min())

# # Discretization

# titanic['Age\_Group'] = pd.cut(titanic['Age'], bins=[0, 0.25, 0.5, 0.75, 1.0], labels=['Child', 'Young Adult', 'Adult', 'Senior'])

# # Final preprocessed dataset

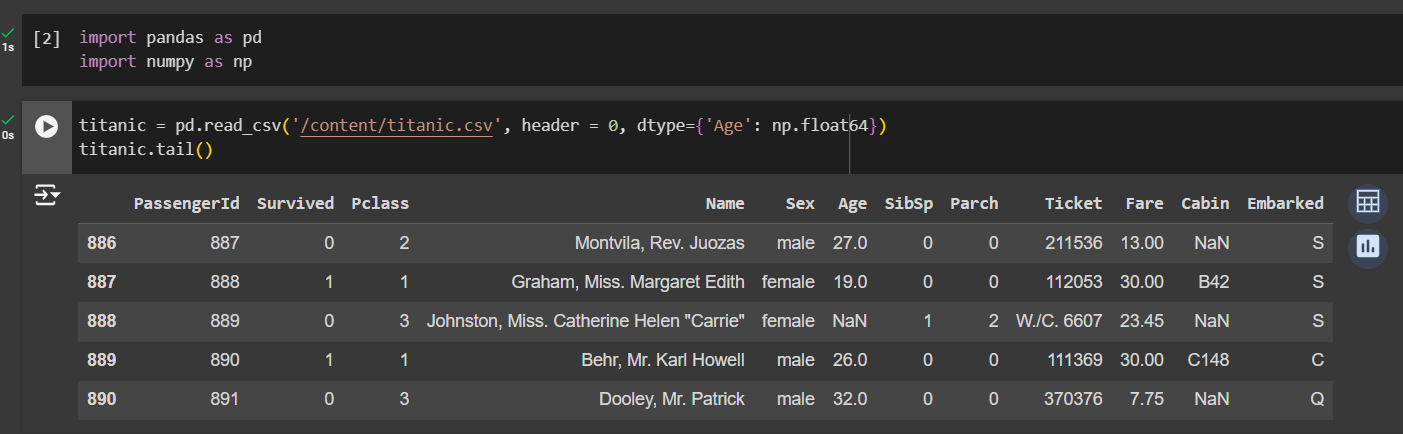
# titanic.head()

# # Saving the preprocessed dataset

# from google.colab import files

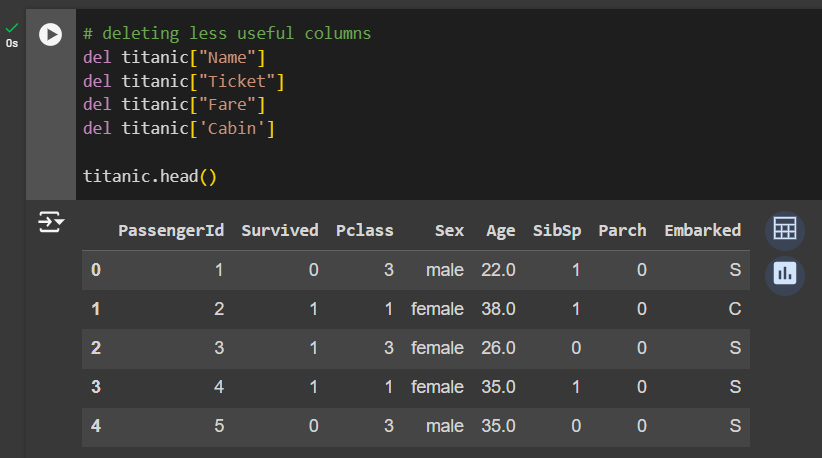
# titanic.to\_csv('titanic\_modified.csv', index=False)

# files.download('titanic\_modified.csv')

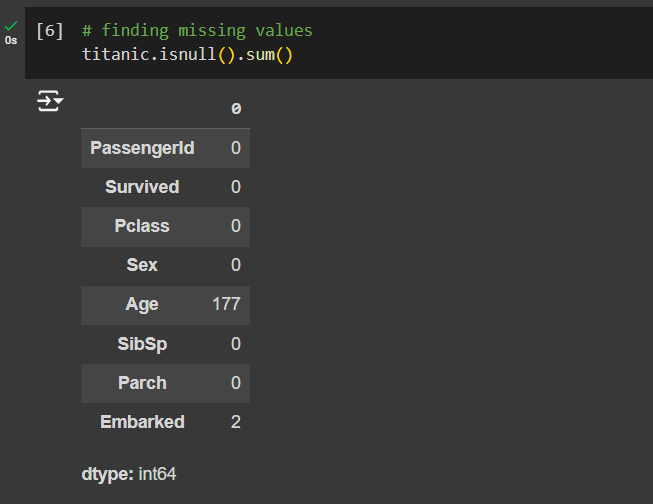
1. Uploading dataset –  
   
2. Describing dataset –



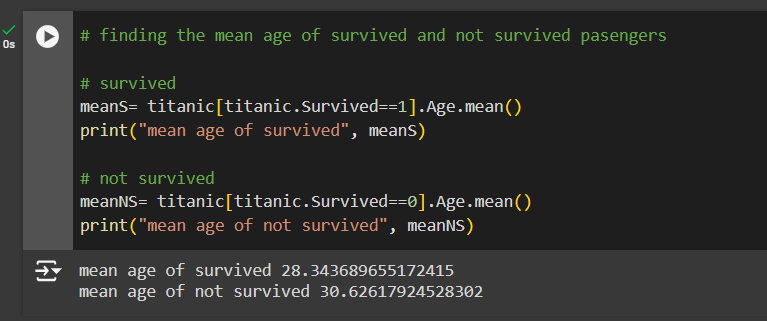
1. Deleting less useful columns –



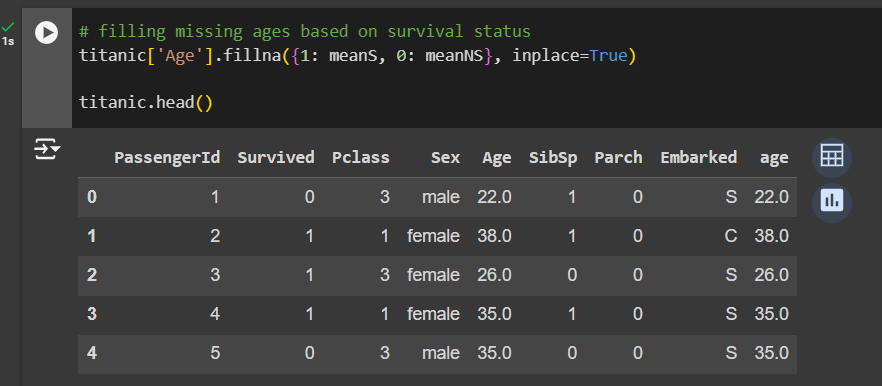
1. Finding number of missing values in each column –



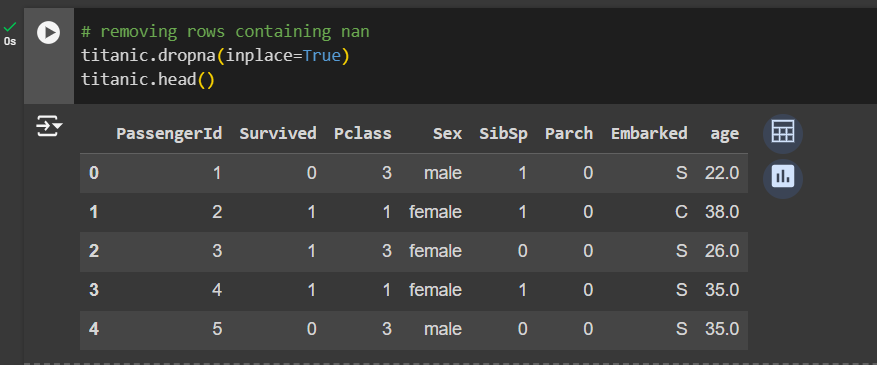
1. Finding mean age of survived and non-surviving passengers –



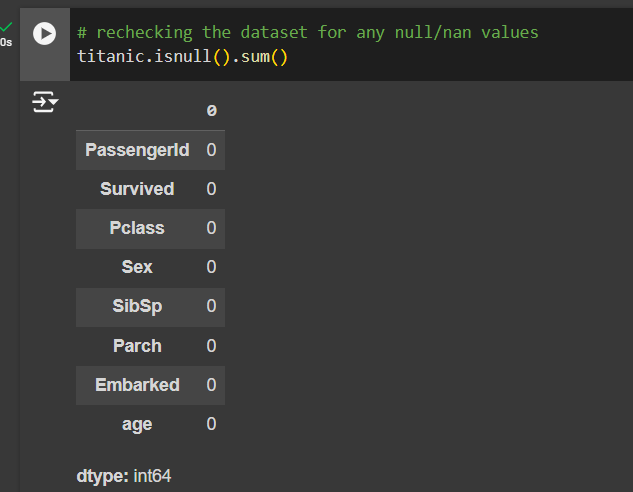
1. Replacing the null values with mean ages calculated in previous step in age column –



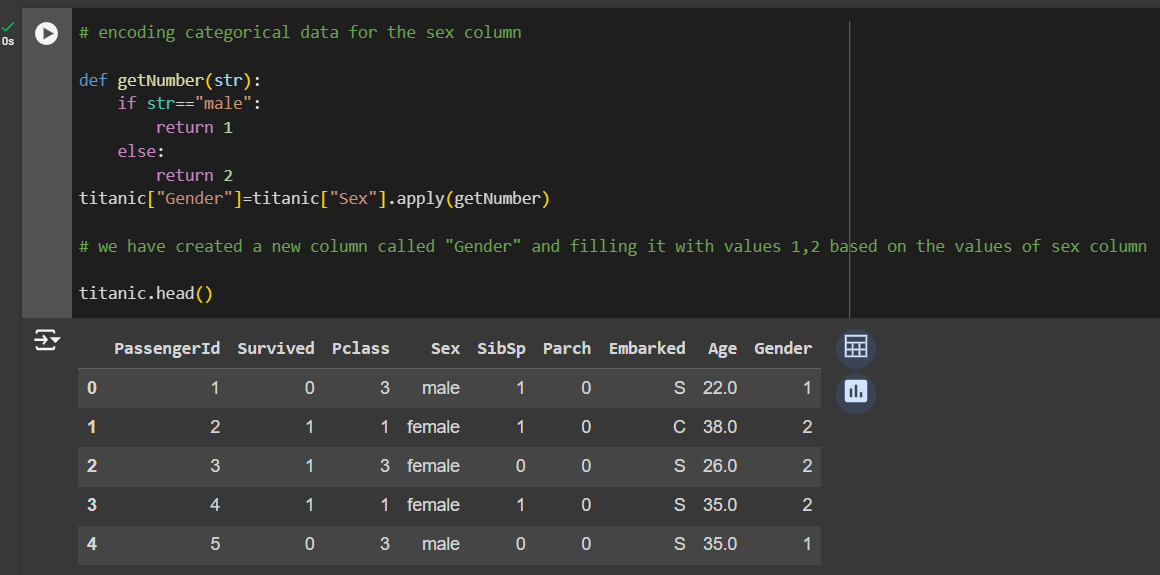
1. Removing any rows containing nan –



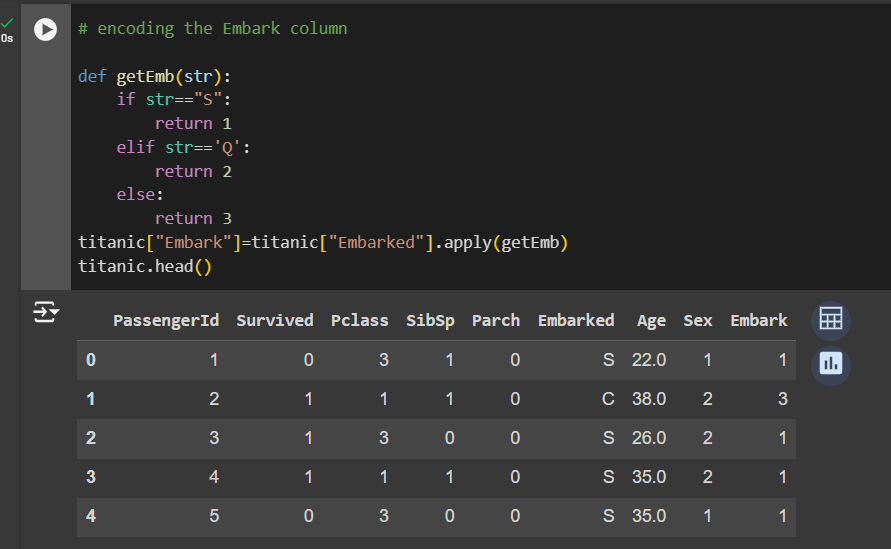
1. Rechecking if any null/nan values present in dataset –



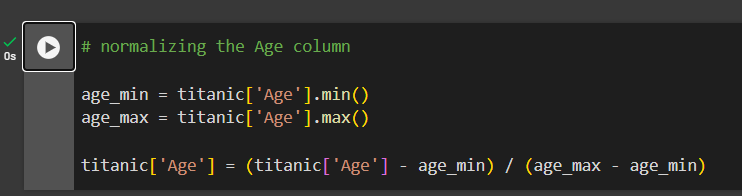
1. Encoding categorical data for Sex column, Male=1, Female=2 –



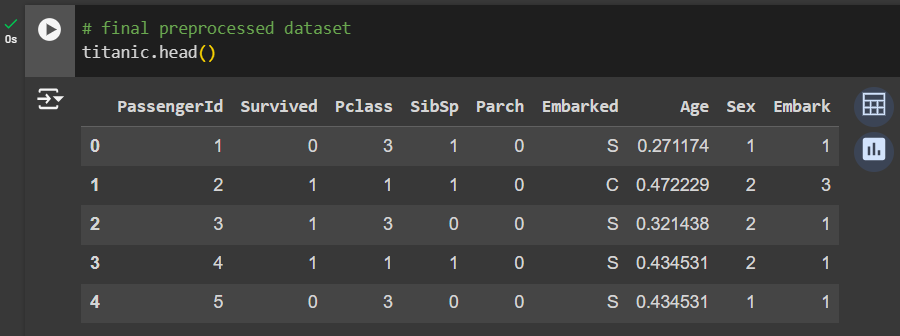
1. Encoding categorical data from the Embark column, S=1, Q=2, anything else=3 –



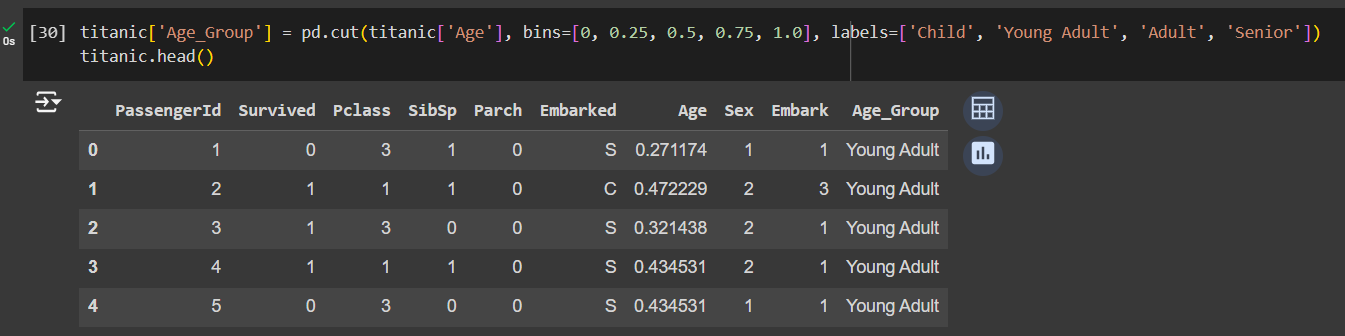
1. Normalizing the age column –



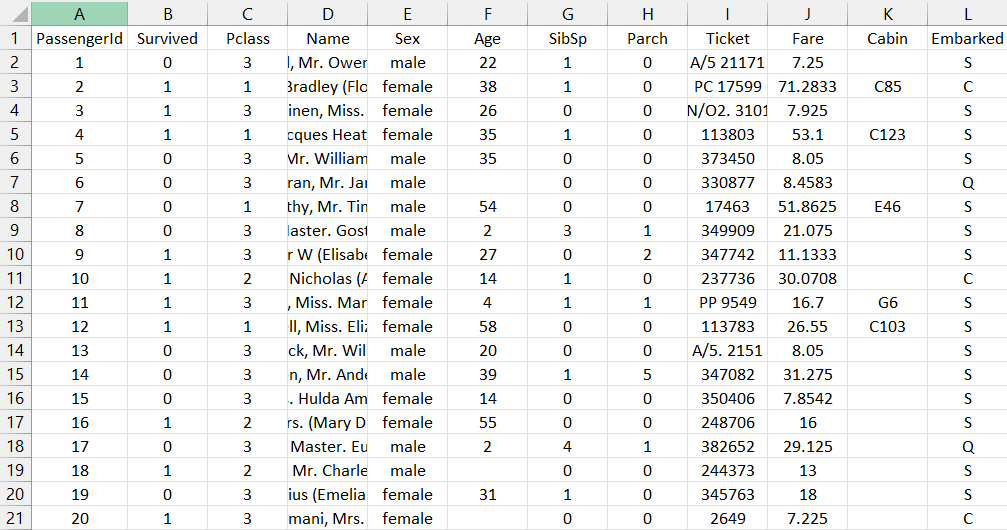
1. Final pre-processed dataset –

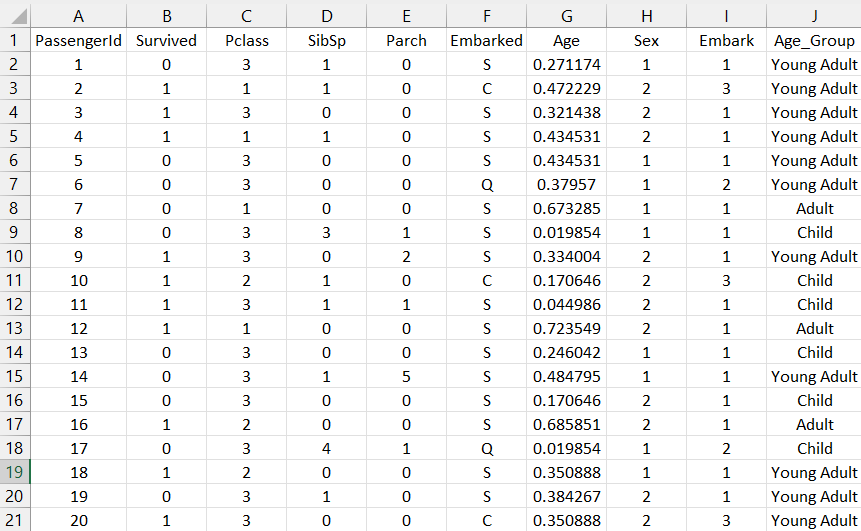


1. Dividing the different age groups into different bins such as Child, Young Adult etc. –



1. Comparison of Dataset Before and After Preprocessing–





# Conclusion:

# Data preprocessing is essential for building robust and effective machine learning models. By systematically addressing issues like missing values, encoding categorical data, normalizing features, and discretizing continuous variables, we can ensure that the dataset is in the best possible shape for analysis. Proper preprocessing not only improves model performance but also helps in making more accurate and reliable predictions.

# Post Lab Questions:

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

# Data cleaning is a crucial step in data preprocessing, but it comes with several challenges:

# Missing Data: Missing values can lead to biased estimates and reduce the statistical power of the dataset. Handling them appropriately is vital to maintain data integrity.

# Inconsistent Data: Data might be recorded in different formats, making it difficult to analyze (e.g., date formats, inconsistent naming conventions).

# Outliers: Extreme values that do not align with the majority of the data can distort analysis and models.

# Duplicate Records: Duplicates can inflate the dataset and lead to misleading results.

# Incorrect Data: Errors in data entry or measurement can introduce inaccuracies.

# Handling Missing Values in the Provided Dataset

# In the provided Titanic dataset, missing values in the 'Age' column were handled by replacing them with the mean age of passengers based on their survival status (Survived column). Specifically:

# For passengers who survived (Survived == 1), missing 'Age' values were replaced with the average age of those who survived.

# For passengers who did not survive (Survived == 0), missing 'Age' values were replaced with the average age of those who did not survive.

# This approach ensures that the imputed values are contextually appropriate and do not introduce bias based on survival status.

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

# Normalization is the process of scaling data to a standard range, typically [0, 1]. It is essential for several reasons:

# Equal Contribution: Features with different scales can disproportionately influence the model's learning process. Normalization ensures that each feature contributes equally, preventing larger-scale features from dominating smaller-scale ones.

# Improved Convergence: Many machine learning algorithms, especially gradient-based ones like neural networks and logistic regression, perform better and converge faster when the data is normalized. Without normalization, the optimization algorithm might struggle to find the minimum, leading to longer training times and suboptimal models.

# Distance-Based Algorithms: For algorithms that rely on distance calculations, such as k-Nearest Neighbors (k-NN) and Support Vector Machines (SVM), normalization is crucial. If the features are not normalized, the distance metrics might be skewed, leading to incorrect predictions or classifications.

# Benefits of Normalizing in the Provided Dataset

# In the Titanic dataset, normalizing the 'Age' column ensures that age-related features do not overshadow others. This is particularly important if the model considers multiple features with different units (e.g., Age vs. Embark). Normalizing helps in maintaining a balanced contribution of all features during the analysis.

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

# Categorical variables like 'gender' often carry significant information that needs to be used in machine learning models. However, most machine learning algorithms cannot process non-numeric data directly. Therefore, converting categorical variables into numerical representations is essential. Here’s why:

# Compatibility with Algorithms: Most machine learning models require numerical input. For instance, regression models, neural networks, and clustering algorithms work with numerical data, so categorical variables must be converted to numbers.

# Capturing Relationships: Converting categorical variables to numbers allows the model to capture the relationship between the categories and the target variable. For instance, in the Titanic dataset, 'Gender' is a critical feature affecting survival rates. Converting 'Gender' into numerical values (e.g., male = 1, female = 2) enables the model to learn this relationship effectively.

# Simplifying Calculations: Numerical representations simplify mathematical operations during training. For example, in a decision tree, splits based on numerical values are straightforward, whereas handling categorical variables directly would be more complex.

# Handling High Cardinality: Some categorical variables have many unique values. Techniques like one-hot encoding or label encoding (simplified using a mapping function) convert these variables into a format that models can easily digest, without losing valuable information.