**EXPERIMENT 1**

**Aim:** Select a dataset and perform exploratory data analysis using Python (data

preprocessing, transformation, discretization and visualisation)

**Theory:**

**Exploratory Data Analysis (EDA):**

Exploratory Data Analysis (EDA) is a critical step in data analysis, where the primary goal is to summarize and visualize the main characteristics of a dataset, often with visual methods. It provides a better understanding of the data, helps detect anomalies, test assumptions, and suggest hypotheses. EDA typically involves:

**1. Data Preprocessing:**

Data preprocessing involves cleaning and preparing raw data for analysis. This step includes:

* **Handling Missing Values:** Techniques include removing or imputing missing values.
* **Removing Duplicates:** Identifying and removing duplicate records to avoid data redundancy.
* **Data Integration:** Combining data from different sources.
* **Data Transformation:** Normalizing or standardizing data to bring it to a common scale.
* **Data Reduction:** Reducing dimensionality through techniques like PCA.

**2. Data Transformation:**

Data transformation involves converting data into a suitable format or structure for analysis. Key methods include:

* **Normalization:** Scaling data to a range, typically [0,1].
* **Standardization:** Transforming data to have zero mean and unit variance.
* **Encoding:** Converting categorical data into numerical format using techniques like one-hot encoding.
* **Log Transformation:** Applying logarithm to reduce skewness.

**3. Data Discretization:**

Data discretization involves converting continuous data into discrete buckets or intervals. This is useful for simplifying models and handling non-linear relationships. Common methods are:

* **Binning:** Dividing data into bins or intervals, either equally spaced (equal-width binning) or containing an equal number of points (equal-frequency binning).
* **Cluster Analysis:** Using clustering algorithms (e.g., k-means) to group data.
* **Decision Tree Binning:** Utilizing decision trees to determine optimal splits for discretization.

**4. Data Visualization:**

Data visualization involves creating graphical representations of data to uncover patterns, trends, and insights. Common visualization techniques include:

* **Histograms:** Displaying the distribution of a single variable.
* **Box Plots:** Summarizing the distribution of data based on five-number summary (minimum, first quartile, median, third quartile, and maximum).
* **Scatter Plots:** Visualizing relationships between two continuous variables.
* **Bar Charts:** Comparing categorical data.
* **Heatmaps:** Showing correlations between variables.
* **Line Graphs:** Displaying trends over time.
* **Pie Charts:** Showing proportions of categorical data.

Combining these EDA techniques provides a comprehensive understanding of the data, facilitating informed decisions on subsequent analysis steps, model selection, and hypothesis testing.

**Code:**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler, StandardScaler

from scipy.stats import zscore

# Load the dataset

file\_path = "./CSV\_files/Student\_Performance\_Data\_Wide\_Version.xlsx"

df = pd.read\_excel(file\_path, "Sheet1")

# Data Cleaning

# Handling missing values

df = df.head(15)

df = df.copy() # Work on a copy of the DataFrame to avoid SettingWithCopyWarning

df.fillna(0, inplace=True) # Fill missing values with 0

# Removing outliers using Z-score

df\_numeric = df.select\_dtypes(include=[np.number])

df = df[(np.abs(zscore(df\_numeric)) < 3).all(axis=1)]

# Data Transformation

# Min-max normalization

scaler = MinMaxScaler()

df[df\_numeric.columns] = scaler.fit\_transform(df[df\_numeric.columns])

# Z-score normalization

scaler = StandardScaler()

df[df\_numeric.columns] = scaler.fit\_transform(df[df\_numeric.columns])

# Decimal scaling

df[df\_numeric.columns] = df[df\_numeric.columns] / 10\*\*np.ceil(np.log10(df[df\_numeric.columns].abs().max()))

# Data Discretization - Binning

# Binning continuous data into 4 bins

for col in df\_numeric.columns:

df[col + '\_binned'] = pd.cut(df[col], bins=4, labels=False)

# Data Analysis and Visualization

# Line Chart

plt.figure(figsize=(10, 6))

sns.lineplot(data=df.drop(columns=[col for col in df.columns if 'binned' in col]))

plt.title('Line Chart')

plt.savefig('line\_chart.png')

plt.close()

# Example Bar Graph (needs specific categorical and numerical columns)

plt.figure(figsize=(10, 6))

sns.barplot(x='Student\_ID', y='Paper 1', data=df)

plt.title('Bar Graph of Paper 1 Scores by Student ID')

plt.xticks(rotation=90)

plt.savefig('bar\_graph.png')

plt.close()

# Example Histogram

plt.figure(figsize=(10, 6))

sns.histplot(df['Paper 1'], bins=30)

plt.title('Histogram of Paper 1 Scores')

plt.savefig('histogram.png')

plt.close()

# Example Box Plot

plt.figure(figsize=(10, 6))

sns.boxplot(x='Semster\_Name', y='Paper 1', data=df)

plt.title('Box Plot of Paper 1 Scores by Semester')

plt.xticks(rotation=90)

plt.savefig('box\_plot.png')

plt.close()

# Example Scatter Plot

plt.figure(figsize=(10, 6))

sns.scatterplot(x='Paper 1', y='Paper 2', data=df)

plt.title('Scatter Plot of Paper 1 vs Paper 2 Scores')

plt.savefig('scatter\_plot.png')

plt.close()

# Heat Map

numeric\_df = df.select\_dtypes(include=[np.number]) # Ensure only numeric columns are used

plt.figure(figsize=(10, 6))

sns.heatmap(numeric\_df.corr(), annot=True, cmap='coolwarm')

plt.title('Heat Map of Correlations')

plt.savefig('heat\_map.png')

plt.close()

**Output:**











