Name: Girish Nandanwar	Course: 22CT744 – Lab. Machine Learning
Roll No: A-56	Department: Computer Technology

Practical No.1

Aim: Introduction to Python Libraries, Datasets and Data cleaning techniques along with data visualization techniques.

Theory:

Python Libraries

1. NumPy (Numerical Python)
NumPy is the fundamental library for numerical and scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices through its N-dimensional array object (**ndarray**), enabling efficient storage and manipulation of large-scale data.
Key features include vectorized operations, broadcasting capabilities, and mathematical functions for linear algebra, statistics, and random number generation.

2. Pandas
Pandas is the core library for data manipulation and analysis, built on top of NumPy. It offers two powerful data structures: **Series (1D)** and **DataFrame (2D)**, which make data preprocessing, cleaning, and manipulation intuitive.
Pandas supports importing and exporting data in various formats such as **CSV, Excel, SQL, and JSON**, and provides functions for **filtering, grouping, merging, and aggregating** data.

3. Matplotlib
Matplotlib is the foundational plotting library in Python that enables users to generate visualizations such as **histograms, scatter plots, bar charts, line graphs, and pie charts**. It provides fine-grained control over every aspect of a figure and serves as the base for other visualization libraries.

4. Seaborn
Seaborn is a **statistical visualization** library built on top of Matplotlib that enables more sophisticated visualizations with better aesthetic formatting.
It simplifies the creation of complex plots and provides built-in themes for improved visual appeal.

5. Scikit-learn
Scikit-learn is the most widely used **machine learning** library in Python, built on **NumPy, SciPy, and Matplotlib**.

It provides tools for **data preprocessing, model selection**, and implementation of various **machine learning algorithms**.

Data Cleaning Techniques

1. Handling Missing Data
Missing data can be categorized as:

- **MCAR (Missing Completely at Random)**
- **MAR (Missing at Random)**
- **MNAR (Missing Not at Random)**

Common strategies include:

- **Deletion:** Removing missing values either completely (*listwise deletion*) or partially (*pairwise deletion*).
- **Imputation:** Filling missing values using **mean**, **median**, or **mode**, or more advanced methods like **regression** or **multiple imputation**.
- **Algorithm-based handling:** Using algorithms like **decision trees** that can handle missing data automatically.

2. Removing Duplicate Data

Duplicate records can distort analytical results and must be identified and removed.
This can be done using functions such as drop_duplicates() in Pandas.

3. Handling Outliers
Outliers can be detected using:

- **Visual methods:** Box plots, scatter plots
- **Statistical methods:** Z-scores, Interquartile Range (IQR)

Ways to handle outliers include:

- Removing outlier records
- Transforming data (e.g., log or square root transformation)
- Capping or flooring extreme values
- Using robust algorithms less sensitive to outliers

4. Data Normalization
Normalization ensures all features are on a **comparable scale**, improving algorithm performance. Common techniques:

- **Min-Max Scaling:** Scales data to a specific range, usually [0, 1]
- **Z-Score Standardization:** Rescales data based on mean and standard deviation

- **Robust Scaler:** Uses median and IQR, making it less sensitive to outliers

5. Fixing Structural Errors

Structural errors occur due to typos, inconsistent naming, or incorrect data types.
These are fixed by checking column formats, correcting labels, and ensuring consistent units.

Data Visualization Techniques

1. Univariate Analysis
Used to analyze a single variable at a time.
- **Histograms:** Show frequency distribution of one variable
- **Box Plots:** Display spread and detect outliers
- **Density Plots:** Show probability distribution of data

2. Bivariate Analysis
Used to study relationships between two variables.
- **Scatter Plots:** Show correlation between continuous variables
- **Line Plots:** Display trends or changes over time
- **Bar Charts:** Compare data across categories

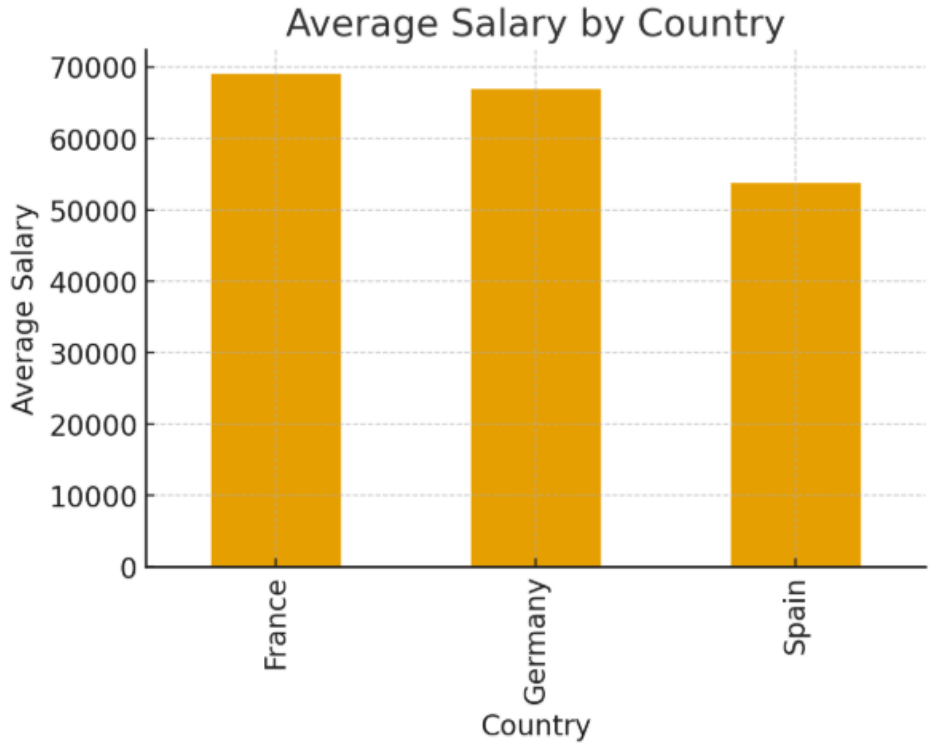
3. Multivariate Analysis
Used to explore relationships among three or more variables.
- **Heatmaps:** Visualize correlation matrices
- **Pair Plots:** Show pairwise relationships in datasets
- **Violin Plots:** Combine box plot and density plot information

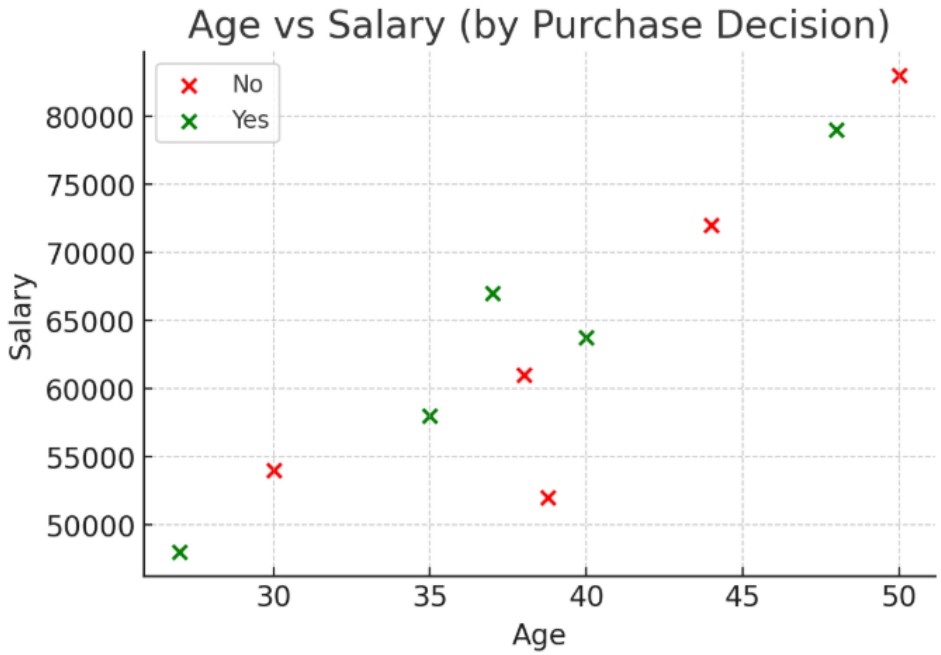
Code:

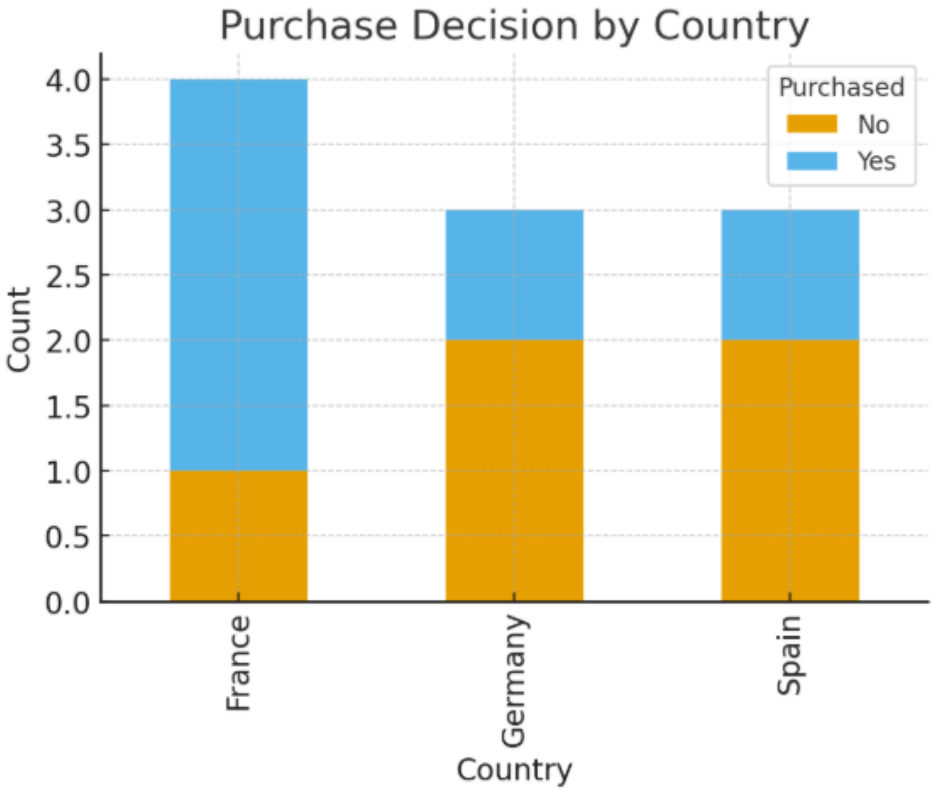
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("Data.csv").values

df["Age"].fillna(df["Age"].mean(), inplace=True)
df["Salary"].fillna(df["Salary"].mean(), inplace=True)

plt.figure(figsize=(6,4))
avg_salary = df.groupby("Country")["Salary"].mean()
```

```
avg_salary.plot(kind="bar", color="skyblue", edgecolor="black")
plt.title("Average Salary by Country")
plt.ylabel("Average Salary")
plt.xlabel("Country")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()

plt.figure(figsize=(6,4))
colors = {"Yes": "green", "No": "red"}
for purchase_status in df["Purchased"].unique():
   subset = df[df["Purchased"] == purchase_status]
   plt.scatter(subset["Age"], subset["Salary"],
            label=purchase_status,
            color=colors[purchase_status],
            s=70, alpha=0.7, edgecolors="k")
plt.title("Age vs Salary (by Purchase Decision)")
plt.xlabel("Age")
plt.ylabel("Salary")
plt.legend(title="Purchased")
plt.grid(True, linestyle="--", alpha=0.6)
plt.show()

plt.figure(figsize=(6,4))
purchase_counts = df.groupby(["Country", "Purchased"]).size().unstack(fill_value=0)
purchase_counts.plot(kind="bar", stacked=True, color=["red", "green"], edgecolor="black")
plt.title("Purchase Decision by Country")
plt.ylabel("Count")
plt.xlabel("Country")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```

Result:


Average Salary by Country
Average Salary
70000
60000
50000
40000
30000
20000
10000
0
France
Germany
Spain
Country


Age vs Salary (by Purchase Decision)
No
Yes
Salary
80000
75000
70000
65000
60000
55000
50000
30
35
40
45
50
Age



Conclusion:

This practical helped us learn how to clean, analyze, and visualize data using Python libraries. These steps are important for building accurate machine learning mode.