**Name**: Vaibhav Kumar Singh  
**Roll No**: 281010  
**Batch**: A1

**Assignment 3**

**Statement**

**Q.** Visualize the data using R/Python by plotting the graphs for Assignment No. 1 and 2. Consider a suitable dataset. Use the following types of plots:  
a) **Scatter Plot**  
b) **Bar Plot**  
c) **Box Plot**  
d) **Pie Chart**  
e) **Line Chart**

**Objective**

1. To introduce data visualization techniques using Python’s **Matplotlib** and **Seaborn** libraries.
2. To understand various types of plots and their application in analyzing data patterns and distributions.
3. To enhance the ability to interpret and communicate data insights effectively using graphical representations.

**Resources Used**

* **Software**: Google Colab
* **Libraries**: Matplotlib, Seaborn, Pandas

**Introduction to Data Visualization**

Data visualization plays a vital role in data analysis by enabling users to detect patterns, spot anomalies, and gain insights through visual formats. Python provides versatile libraries like **Matplotlib** and **Seaborn**, which are extensively used for generating insightful and attractive visualizations.

Key benefits include:

* Better understanding of data characteristics
* Ability to communicate insights effectively
* Facilitating faster decision-making

**Types of Graphs Used**

1. **Scatter Plot**
   * **Purpose**: To show relationships between two numerical variables.
   * **Functions**:
     + plt.scatter(x, y) (Matplotlib)
     + sns.scatterplot(x, y, data=df) (Seaborn)
2. **Bar Plot**
   * **Purpose**: To represent categorical data with rectangular bars.
   * **Functions**:
     + plt.bar(x, y)
     + sns.barplot(x, y, data=df)
3. **Box Plot**
   * **Purpose**: To display data distribution and detect outliers.
   * **Function**:
     + sns.boxplot(x, y, data=df)
4. **Pie Chart**
   * **Purpose**: To visualize proportions of categorical data.
   * **Function**:
     + plt.pie(sizes, labels=labels, autopct='%1.1f%%')
5. **Line Chart**
   * **Purpose**: To display trends over time or continuous data changes.
   * **Function**:
     + plt.plot(x, y)

**Methodology**

1. **Data Collection and Preparation**
   * Loaded a relevant dataset aligned with Assignments 1 and 2 (e.g., maternal health dataset).
   * Performed cleaning and preprocessing to ensure data readiness.
2. **Data Visualization**
   * Selected suitable plot types based on feature types and analysis goals.
   * Generated visualizations using **Matplotlib** and **Seaborn**.
   * Enhanced plots with **titles**, **labels**, **legends**, and **color schemes** for clarity and aesthetics.

**Advantages**

1. Makes hidden patterns and relationships easily visible.
2. Helps in quickly identifying trends, clusters, and outliers.
3. Aids in better communication and presentation of data-driven findings.

**Disadvantages**

1. Poor design or misleading graphs can result in incorrect interpretations.
2. For very large or complex datasets, basic plots might be insufficient, requiring more advanced visualization techniques.

**Graphs**

*(Graphs generated using Google Colab based on cleaned and preprocessed dataset. Include screenshots or embed code snippets and outputs here if submitting digitally.)*

* **Scatter Plot**: Visualized heart rate vs. glucose levels.
* **Bar Plot**: Compared counts of each risk category.
* **Box Plot**: Showed distribution of systolic blood pressure.
* **Pie Chart**: Illustrated the percentage of patients in each risk category.
* **Line Chart**: Tracked average diastolic pressure across observations.

**Conclusion**

In conclusion, this assignment provided a comprehensive introduction to **data visualization techniques using Python**. By plotting various types of graphs—scatter, bar, box, pie, and line—we explored different ways of representing data for insightful analysis. Through hands-on implementation, we learned how to choose the right visualization for different types of data, enabling more informed and effective decision-making. Mastering these techniques is a fundamental skill for any data analyst or data scientist.