9. Write a Python Program to draw 3D plots using Plotly library.

1. Introduction.

- Plotly is a data visualization library that provides a wide range of tools for creating interactive plots in Python, as well as in other programming languages like R and JavaScript. It is designed to enable the creation of visually appealing and interactive charts, graphs, and dashboards for data analysis and presentation.
- This Python program uses the Plotly library to construct an interactive 3D scatter plot, illustrating student data. The primary objective is to depict the correlation between students' height, weight, and age within a three-dimensional space. The Plotly graph_objects module is employed to create the plot, and the offline module is used to generate the resulting interactive HTML file locally.
- The program also generates hover text for each data point, giving detailed information about each student. The Scatter3d trace from Plotly is utilized to craft the 3D scatter plot, and the layout is enhanced by incorporating axis titles and a main title. The final plot is then saved as an HTML file, offering an engaging and informative visualization of the intricate relationships among students' height, weight, and age.

2. Program code.

```
# Import necessary libraries
import plotly.graph_objects as go
from plotly.offline import plot

# Dataset
students_data = {
    'Name': ['Anil', 'Sunil', 'Kumar', 'Rajesh', 'Naveen'],
    'Height': [160, 155, 170, 165, 175],
    'Weight': [55, 50, 65, 60, 70],
    'Age': [20, 19, 21, 20, 22]
}
```

```
# Create hover text with student names
hover_text = [f'Name: {name}<br>Height: {height}<br>Weight:
{weight}<br>Age: {age}'
for name, height, weight, age in zip(students_data['Name'],
students_data['Height'],
students_data['Weight'], students_data['Age'])]

# Create a 3D scatter plot
fig = go.Figure(data=[
    go.Scatter3d(
        x=students_data['Height'],
        y=students_data['Weight'],
        z=students_data['Age'],
        mode='markers',
        marker=dict(
            size=10,
            color='blue',
            opacity=0.8
        ),
        text=hover_text  # Assign hover text to each marker
    )
])

# Add labels and title
fig.update_layout(scene=dict(xaxis_title='Height', yaxis_title='Weight',
zaxis_title='Age'),title='3D Scatter Plot of Student Data')

# Show the plot
plot(fig, filename='Student_plot.html')
```

3. Explanation of the code

3a. Importing the necessary libraries for creating 3D plots using Plotly.

import plotly.graph_objects as go:

- This line imports the graph_objects module from the Plotly library and assigns it the alias go for convenience. The graph_objects module provides a high-level interface for creating complex visualizations, including 3D plots. By using the as go alias, it simplifies the code when referencing functions and classes from this module.

from plotly.offline import plot:

- Plotly supports both online and offline modes for generating plots. In the offline mode, the offline module provides functions to create and save plots without needing an active internet connection. The plot function, in particular, is employed to visualize the plot interactively and to save it as an HTML file locally. This HTML file can then be opened in a web browser, allowing users to interact with and explore the plot without the need for an internet connection.

3b. Create a dictionary with student data set

```
students_data = {
    'Name': ['Anil', 'Sunil', 'Kumar', 'Rajesh', 'Naveen'],
    'Height': [160, 155, 170, 165, 175],
    'Weight': [55, 50, 65, 60, 70],
    'Age': [20, 19, 21, 20, 22]
}
```

- The variable students_data is a dictionary with keys representing different attributes of students, and the corresponding values are lists containing the respective data for each student. Each list is ordered such that the information for the same student is at the same index in each list.

3c. Creating hover text for each data point in the student dataset.

hover_text = [...]:

- Hover text is additional information that appears when a user hovers over a data point in a plot, providing more details about that specific point.
- This line initializes a list comprehension to create a list of formatted strings, where each string represents the hover text for a corresponding student. The hover text includes the student's name, height, weight, and age.

*for name, height, weight, age in **zip**(students_data['Name'], students_data['Height'], students_data['Weight'], students_data['Age']):*

- This part of the code uses the zip function to iterate over the lists of names, heights, weights, and ages simultaneously. In each iteration, it unpacks the values for a specific student.

*f'Name: {name}
Height: {height}
Weight: {weight}
Age: {age}'*

- This part formats a string using an f-string (formatted string literal). It combines the values of name, height, weight, and age for each student, creating a string that includes the student's name, height, weight, and age, separated by line breaks (
).

Example Result (for the first student):

'Name: Anil
Height: 160
Weight: 55
Age: 20'

3d. Creating a 3D scatter plot using the Plotly library in Python.

fig = go.Figure(data=[...]:

- This line initializes a Figure object from Plotly's graph_objects module. The Figure object is the top-level container for all visual elements in a Plotly plot.

data=[go.Scatter3d(...)]:

- The data argument is a list containing trace objects that define the data and appearance of the plot. In this case, the list contains a single trace, which is a Scatter3d trace. The Scatter3d trace is specifically designed for 3D scatter plots.

x=students_data['Height'], y=students_data['Weight'], z=students_data['Age']:

- These parameters specify the x, y, and z coordinates for each data point. In this case, the x-axis represents heights, the y-axis represents weights, and the z-axis represents ages. The data is extracted from the students_data dictionary.

mode='markers':

- The 'markers' mode indicates that each data point should be represented as a marker in the plot.

marker=dict(size=10, color='blue', opacity=0.8):

- This parameter configures the appearance of the markers. They are set to be blue, have a size of 10, and an opacity of 0.8 (80% transparency).

The resulting Figure object (fig) encapsulates the 3D scatter plot configuration.

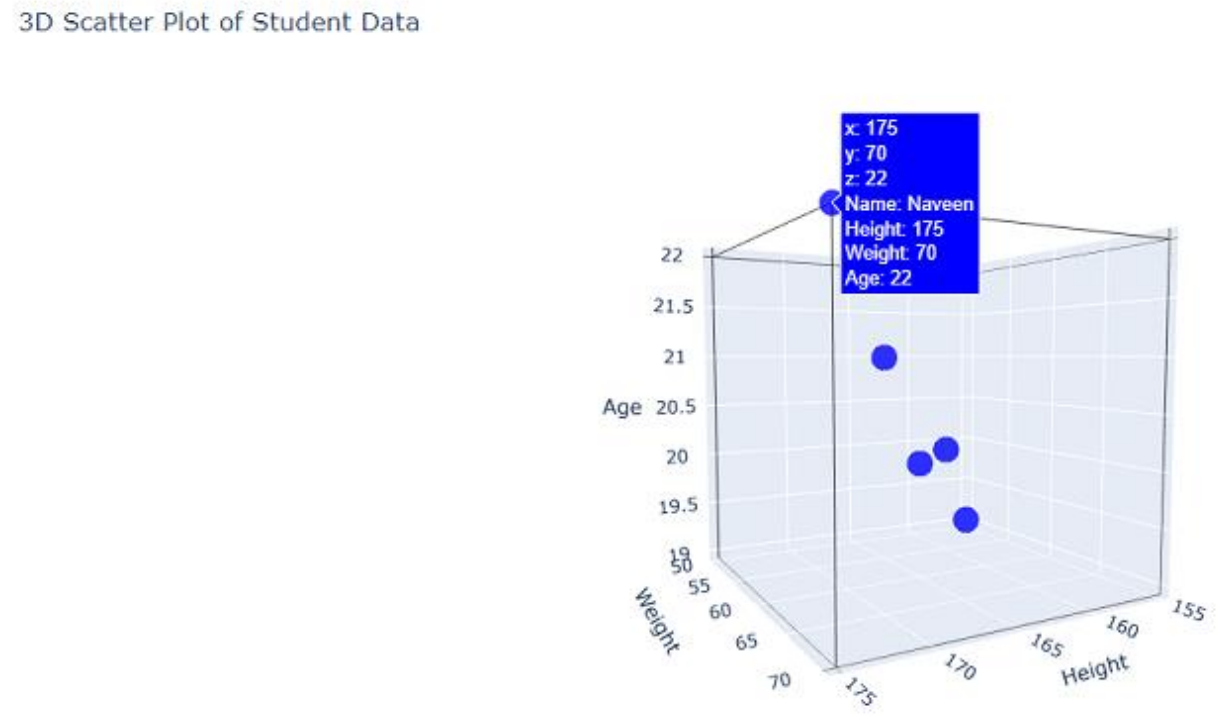
3e. Modify the layout settings of a 3D scatter plot

fig.update_layout(scene=dict(xaxis_title='Height',yaxis_title='Weight',zaxis_title='Age'), title='3D Scatter Plot of Student Data')

- The update_layout method is used to modify the layout settings of the Figure object (fig). It allows customization of various aspects of the plot's appearance.
- scene=dict(xaxis_title='Height', yaxis_title='Weight', zaxis_title='Age'): This part of the code configures the scene settings for the 3D plot. It uses a dictionary (dict) to specify the titles for the x-axis, y-axis, and z-axis. In this case, the x-axis is labeled 'Height,' the y-axis is labeled 'Weight,' and the z-axis is labeled 'Age.'
- title='3D Scatter Plot of Student Data': This parameter sets the main title of the plot. The title is a string that describes the overall content or purpose of the plot. Here, it is set to '3D Scatter Plot of Student Data.'

The above code updates the layout of a 3D scatter plot by configuring axis titles and adding a main title using the update_layout method of the Plotly Figure object. These modifications contribute to the clarity and interpretation of the visualized student data.

4. Output



Output Analysis:

key elements of the output:

1. Type of Chart:

- It's a 3D scatter plot, which visualizes the relationships between three numerical variables in a 3D space.

2. Variables Mapped:

The plot maps three variables:

Height (x-axis)

Weight (y-axis)

Age (z-axis)

3. Data Points:

Each blue marker represents a student.

The position of each marker indicates their height, weight, and age values.

4. Hover Text:

- When you hover over a marker, it displays the student's name, height, weight, and age details.

5. Axes and Title:

- The axes are labeled as Height, Weight, and Age.
- The title of the plot is "3D Scatter Plot of Student Data.

Insights from the Chart:

Visualizing Relationships: The plot allows you to explore potential relationships between the variables. For example, you might observe trends like taller students tending to be heavier.

5. Use Cases of Plotly

Plotly is a versatile Python library for creating interactive visualizations. Here are few use cases for Plotly.

1. **Financial Dashboards:**

- Plotly can be used to create interactive financial dashboards that allow users to explore and analyze stock prices, portfolio performance, and economic indicators. You can use candlestick charts, line plots, and scatter plots to visualize trends, and incorporate sliders or dropdowns for dynamic date range selection.

2. **Geospatial Data Visualization:**

- Plotly is well-suited for geospatial data visualization. You can create interactive maps with features like choropleth maps, bubble maps, and scatter plots on geographical data. This is useful for visualizing spatial patterns, demographics, or any other location-based information.

3. **Machine Learning Model Evaluation:**

- When working with machine learning models, Plotly can be employed to create visualizations for model evaluation. You can generate ROC curves, precision-recall curves, confusion matrices, and other metrics to assess the performance of your models. Interactive features can provide a deeper understanding of model behavior.

4. Scientific Data Exploration:

- Plotly is useful for visualizing scientific data in fields such as biology, physics, and environmental science. You can create interactive 3D plots, surface plots, and contour plots to explore complex datasets. Features like hover tooltips and zooming help researchers interactively analyze and understand their data.

5. Real-time Data Monitoring:

- Plotly is great for building real-time monitoring dashboards. You can use it to display live streaming data, such as IoT sensor readings, server performance metrics, or social media mentions. The interactivity features allow users to zoom in on specific time periods and get detailed information on the fly.