**PLOTLY (GRAPHIC LIBRARY)**

**PLOTLY**

* Plotly is a versatile data visualization library developed by the Canadian company Plotly.
* It supports various languages, including Python, JavaScript, and Julia.
* Built on the plotly.js JavaScript library, Plotly empowers Python users to create captivating, interactive visualizations for Jupyter notebooks, standalone HTML files, or as components in web applications developed using Dash.

**ADVANTAGES**

**Multi-language Support:**

Plotly is accessible across different programming languages.

**Diverse Graphs:**

Offers a wide variety of graph types for comprehensive data representation.

**Interactive Plots:**

Enables the creation of interactive and engaging plots.

**Aesthetic Appeal:**

Generates visually appealing plots.

**NOTE**

**Not Ideal for Live Data Streams:**

Plotly is not designed for handling live data streams.

For real-time data, Dash, another product from Plotly, is recommended.

**THE PLOTLY ROADMAP**

**Plotly Go:**

A part of the Plotly ecosystem, Plotly Go likely signifies a user-friendly interface or functionality.

**Plotly Express:**

An expressive layer for creating visualizations quickly and easily.

**Dash:**

A framework for building analytical web applications, extending the capabilities of Plotly for interactive data-driven applications.

**INSTALLATION**

Plotly can be installed using pip:

**$ pip install plotly==5.18.0**

**IMPORTING THE PLOTY LIBRARY**

To use Plotly in your Python code, import the Plotly Express module as follows:

**import** plotly.express **as** px

**SCATTER PLOT**

The px.scatter() function in Plotly Express comes with a variety of parameters that allow you to customize the appearance and behavior of the scatter plot

P**arameters**

|  |  |
| --- | --- |
| **data\_frame**: | (DataFrame or array-like or dict)  input data for the plot. |
| **x, y**:  (str or array-like) | columns in the DataFrame or array-like object to be used as the x and y coordinates of the scatter plot. |
| **color**:  (str or array-like) | column in the DataFrame or array-like object to be used as the color of the points. |
| **size**:  (str or array-like) | column in the DataFrame or array-like object to be used as the size of the points |
| **symbol**:  (str or array-like) | column in the DataFrame or array-like object to be used as the symbol of the points. |
| **hover\_name, hover\_data**:  (str or array-like) | columns in the DataFrame or array-like object to be used as the label and additional data for hover information. |
| **animation\_frame, animation\_group**:  (str) | columns in the DataFrame to be used for animation. animation\_frame represents the time-based animation, and animation\_group is used to identify data points that should be treated as the same in each frame. |
| **text**:  (str or array-like) | column in the DataFrame or array-like object to be used as the text labels for the points. |
| **title**: (str) | title of the plot. |
| **labels**: (dict) | a dictionary specifying axis labels. |
| **log\_x, log\_y**: (bool) | logarithmic scale for the x and y axes. |
| **range\_x, range\_y**:  (list or tuple) | specifies the range of values to be displayed on the x and y axes. |
| **facet\_col, facet\_row**: (str) | columns in the DataFrame to be used for faceting the plot into subplots. |
| **marginal\_x, marginal\_y**: | {'rug', 'box', 'violin', 'histogram'}  Adds marginal plots along the x and y axes. |
| **trendline**: (str) | type of trendline to be added to the plot ('ols', 'lowess', None). |
| **template**:  (str or dict) | name of the template or a dictionary specifying the attributes of the template for the plot. |

**go.Figure()**

In Plotly, go.Figure() is a function used to create a new figure or plot. It's a part of Plotly's Graph Objects module (go).

You use it to initialize a figure that can be customized and extended with various traces and layout options.

**BAR CHART**

px.bar(data\_frame, x, y, color, hover\_name, hover\_data, text,

facet\_col, facet\_row, animation\_frame, animation\_group, barnorm ,category\_orders, animation\_duration, animation\_easing, labels, opacity, barmode='relative', log\_x=False, log\_y=False, range\_x, range\_y, text\_auto=False, title, template, width, height)

**Parameters**

|  |  |
| --- | --- |
| **color**:  (str or array-like) | column in the DataFrame or array-like object to be used as the color of the bars. |
| **facet\_col, facet\_row**: (str) | columns in the DataFrame to be used for faceting the plot into subplots. |
| **barmode**: | {'stack', 'group', 'overlay', 'relative'}  Specifies the mode of the bars. |
| **hover\_name, hover\_data**: | columns in the DataFrame to be used for hover information. |
| **width, height**: (int) | width and height of the plot in pixels. |
| **category\_orders**: (dict) | order of categorical values on the axes. |
| **range\_x, range\_y**:  (list or tuple) | range of values to be displayed on the x and y axes. |
| **marginal**: (str) | Adds marginal plots along the x and/or y axes. |
| **barnorm**: | normalization mode for the colorscale. |
| **animation\_duration**: (int) | duration of the animation in milliseconds. |
| **animation\_easing**: (str) | The easing function for the animation. |

**PIE CHART**

**Syntax: px.pie( values, names, title, labels, color, hover\_data, hole, color\_discrete\_sequence, template)**

**Parameters**

**values**: A list or array-like object representing the values for each slice of the pie chart.

**names**: A list or array-like object representing the labels for each slice.

**title**: Title of the pie chart.

**labels**: Custom labels for each slice.

**color\_discrete\_sequence**: Set the color sequence for the slices.

**color**: Allowing you to set a single color for all slices.

**hover\_data**: An optional list or array-like object specifying additional data to be displayed when hovering over the slices.

**hole**: Specifying the size of the center hole in the pie chart (useful for creating a donut chart).

**template**: Specifying a template to be used for the chart layout.

**SUNBURST PLOT**

Sunburst plots visualize hierarchical data spanning outwards radially from root to leaves.

**Syntax: px.sunburst(data\_frame, path, values, color,hover\_data, title, width, height)**

**Parameters**

**data\_frame**: The DataFrame containing the data to be plotted.

**path**: A list representing the hierarchical data columns. The values in these columns define the hierarchy of the sunburst plot.

**values**: The column in the DataFrame that contains the numerical values associated with each node in the hierarchy.

**color**: The column in the DataFrame that provides color information for each segment of the sunburst plot.

**hover\_data**: Additional columns from the DataFrame to be included in the hover tooltip.

**title**: The title of the plot.

**width and height**: The width and height of the plots.

**TREEMAP CHARTS**

Treemap charts visualize hierarchical data using nested rectangles.

**px.Constant()**

In Plotly, px.Constant() is a function used to create a constant trace.

This trace represents a constant value on the plot, which can be useful for highlighting a specific point or line in your data visualization.

**HEATMAP**

Combination of two catogorical column on the top of the numerical column

**px.imshow()**

The px.imshow() function in Plotly Express is used to create a heatmap from a 2D array-like object or a 2D DataFrame.

**Parameters for px.imshow():**

* **z:** The 2D array or DataFrame containing the data for the heatmap.
* **labels:** A dictionary specifying the labels for the x and y axes. For example, labels=dict(x="X-axis label", y="Y-axis label").
* **x and y:** Lists or arrays providing the labels for the x and y axes, respectively. These are optional parameters, and if not provided, integer indices will be used as labels.
* **color\_continuous\_scale:** A string representing the color scale for continuous data. You can use predefined scales like 'Viridis', 'Plasma', 'Jet', etc.
* **title:** A string representing the title of the plot.
* **width and height:** Integers specifying the width and height of the plot in pixels.
* **aspect:** The aspect ratio of the plot. Can be 'equal', 'auto', or a numerical value.
* **template:** The Plotly template to use. You can use predefined templates like 'plotly', 'plotly\_dark', 'ggplot2', etc.
* **coloraxis\_colorbar:** A dictionary to customize the color bar. For example, coloraxis\_colorbar=dict(title="Colorbar Title").

## **3D SCATTER PLOT**

Like the 2D scatter plot px.scatter, the 3D function px.scatter\_3d plots individual data in three-dimensional space

**Parameters**e.

**x, y, z:** These parameters represent the features or dimensions of the data that you want to visualize. In machine learning, you often visualize relationships between input features or the distribution of feature values.

**hue or color:** This parameter is often used to represent different classes or categories in your data. It can be helpful for visualizing how different classes are distributed or how a model's predictions compare to the true labels.

**size:** Used to represent the size of data points, which can be useful for visualizing the importance or magnitude of a particular aspect of the data.

**style or symbol:** This parameter can be used to differentiate between different groups or subsets of data points. For example, you might use different symbols or styles for data points belonging to different classes.

**animation\_frame, animation\_group:** These parameters are used for creating animated visualizations, which can be helpful for understanding how data evolves over time or iterations, especially in the context of iterative machine learning algorithms.

**facet\_col, facet\_row:** These parameters are used for creating facet plots, which allow you to break down the data into subsets based on certain conditions. This can be useful for visualizing how relationships differ across different subsets.

**hover\_data:** Used to specify additional information that will be displayed when you hover over a data point. This can include details such as feature values, class labels, or model predictions.

**marginal\_x, marginal\_y:** These parameters are used to add marginal plots (histograms, box plots, etc.) along the x and y axes, providing additional insights into the distribution of individual features. 7

**SCATTER MATRIX WITH PLOTLY EXPRESS**

**PLOTLY GRAPH OBJECT**

**Introduction to Plotly**

Matplotlib, pandas all are produce static image files.

Plotly is company based out in canada and it is famous for it's products like Plotly and Dash

Plotly creates interactive visulisations in the form of HTML files

Drawback : can't work with live data source

Dash is used to create live data based dashboards.

**What is Graph objects ?**

Graph Objects in Plotly are Python class instances found in **plotly.graph\_objects** module (imported as go).

They represent figures in Plotly, follow the Plotly.js schema, and are serialized to JSON for rendering by Plotly.js.

**Installing the plotly:**

pip install Plotly

**Importing the plotly graph object:**

import plotly.graph\_objects as go

**When to use Graph Objects vs Plotly Express**

Plotly Express is quick and intuitive for simple visualizations, while Graph Objects offer fine-grained control for advanced features.

Choose based on your preference, and consider using both in a project depending on the complexity of your visualizations.

**Comparing Graph Objects and Plotly Express:**

While figures from Plotly Express can be constructed using graph objects, the latter typically requires 5-100 lines of code compared to the more concise 1 line in Plotly Express.

**Note:**

Plotly Express functions, recommended for entry into the Plotly library, are built on top of Graph Objects. All functions in Plotly Express return instances of **plotly.graph\_objects. Figure**.

**go.Figure() function**

This is the top-level object that represents the entire figure or visualization.

**Syntax: fig = go.Figure(data, layout)**

**data** (list): A list containing trace objects representing the data visualizations.

**layout** (go.Layout or dict): An instance of go.Layout or a dictionary specifying layout settings for the figure.

**go.Scatter() function**

* This is a trace object representing a scatter plot, which is a set of data points displayed as markers on a 2D plane.
* **Syntax: e trace = go.Scatter(x, y, mode, ...additional\_paramete**rs)
* **x** (list/array): X-axis data.
* **y** (list/array): Y-axis data.
* **mode** (str): Specifies the mode of the scatter plot (e.g., 'markers', 'lines', 'lines+markers').
* **marker** (dict): Customize marker properties.
  + **color** (str): Marker color.
  + **size** (int): Marker size.
  + ... (other marker properties).
* **line** (dict): Customize line properties.
  + **color** (str): Line color.
  + **width** (int): Line width.
  + ... (other line properties).
* ... (other optional parameters).

**go.Layout() function:** : This is used to customize the layout or appearance of the entire figure.

**Syntax**: **layout = go.Layout(title, xaxis, yaxis, …. )**

* **title** (str): Title of the figure.
* **xaxis** (dict): Dictionary specifying settings for the x-axis.
  + **title** (str): X-axis title.
  + **range** (list/tuple): X-axis range.
  + ... (other x-axis properties).
* **yaxis** (dict): Dictionary specifying settings for the y-axis.
  + **title** (str): Y-axis title.
  + **range** (list/tuple): Y-axis range.
  + ... (other y-axis properties).
* **width** (int): Width of the figure.
* **height** (int): Height of the figure.
* **margin** (dict): Margin settings.
  + **l** (int): Left margin.
  + **r** (int): Right margin.
  + ... (other margin properties).
* **annotations** (list): List of annotations to be added to the figure.
  + **text** (str): Annotation text.
  + **x** (float): X-coordinate of the annotation.
  + **y** (float): Y-coordinate of the annotation.
  + ... (other annotation properties).

**Histogram:**

In statistics, a histogram is a representation of the distribution of numerical data, where the data are binned and the count for each bin is represented.

**Syntax: go.Histogram(x, y, xbins, ybins, nbinsx, nbinsy, histnorm, opacity, marker, legend, showlegend, name, hoverinfo)**

**Parameters:**

x: Data or a list of data arrays for the x-axis

y: Data or a list of data arrays for the y-axis

xbins: Customize binning on the x-axis with properties like start, end, and size.

ybins: Customize binning on the y-axis with properties like start, end, and size.

nbinsx: The number of bins along the x-axis

nbinsy: The number of bins along the y-axis

histnorm: Normalization applied to the histogram ('probability', 'percent', etc.)

opacity: Sets the opacity of the bars

marker: Specifies the marker style for the bars

legend: Determines whether an item appears in the legend

showlegend: Determines whether the legend is displayed

name: Sets the trace name

hoverinfo: Specifies the information that appears on hover