**Fall 2023 DATA 230 – 11**

**Data Visualization**

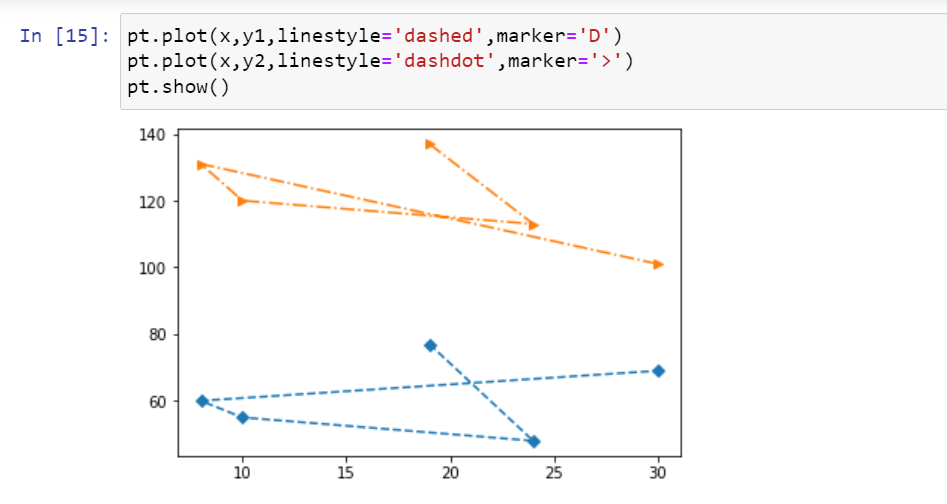
**Homework – 2**

**Name :- Prayag Nikul Purani**

**SJSU Id :- 017416737**

**Question 1:**

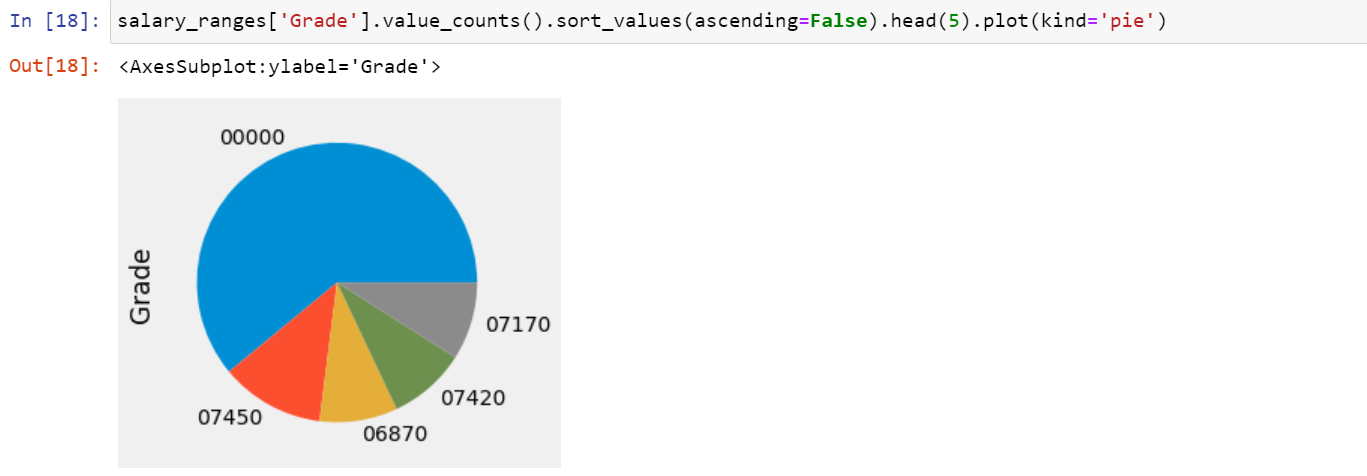
**Visualization 1: Mapping data onto aesthetics (Linetypes and markdown)**



In line graph it has dotted and dotted dash line with different types marker to separate the data. Line graphs are commonly used in various fields, including economics, science, engineering, and social sciences, to visualize and analyze data. They can help you observe trends, compare data sets, and make predictions based on the data. Line graphs are particularly suitable for showing data with continuous variables, such as temperature fluctuations over time, stock price changes, or population growth.

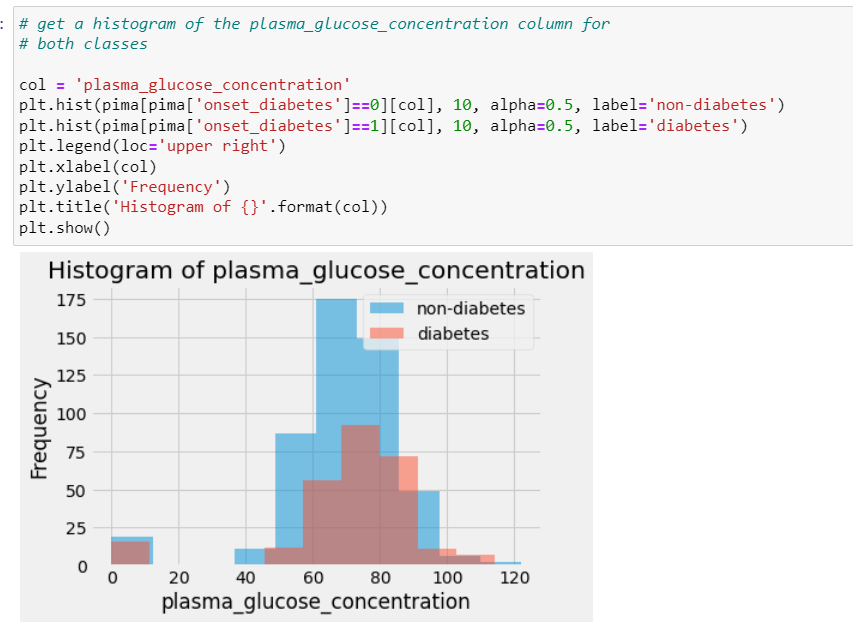
When creating a line graph, it's essential to choose the appropriate scale for both the X and Y axes, label your axes clearly, and provide a title that summarizes the graph's purpose. This makes it easier for the audience to understand the data and the insights it conveys.

**Visualization 2: Mapping data onto aesthetics (Color in pie chart)**



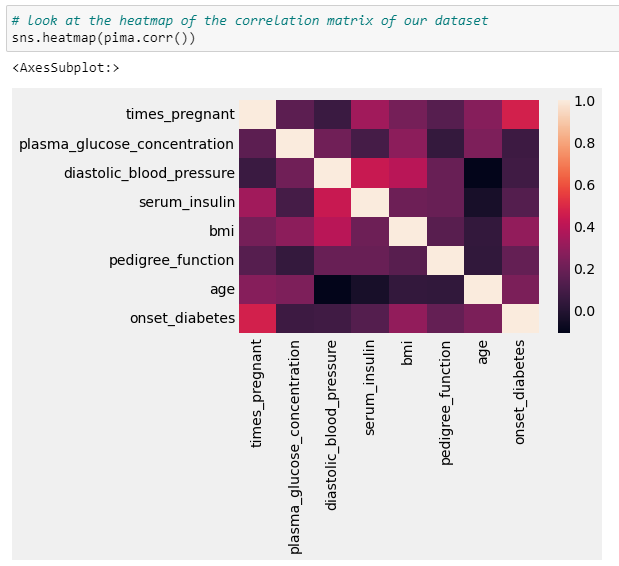
A pie chart is a circular chart that is divided into sectors, much like slices of a pie. Each sector represents a proportion of the whole, and the size of each sector corresponds to the quantity it represents. Pie charts are used to display data as a part-to-whole relationship and are especially useful for showing the composition or distribution of a categorical data set.

**Visualization 3: Mapping data onto aesthetics (Color in histogram Continuous data)**



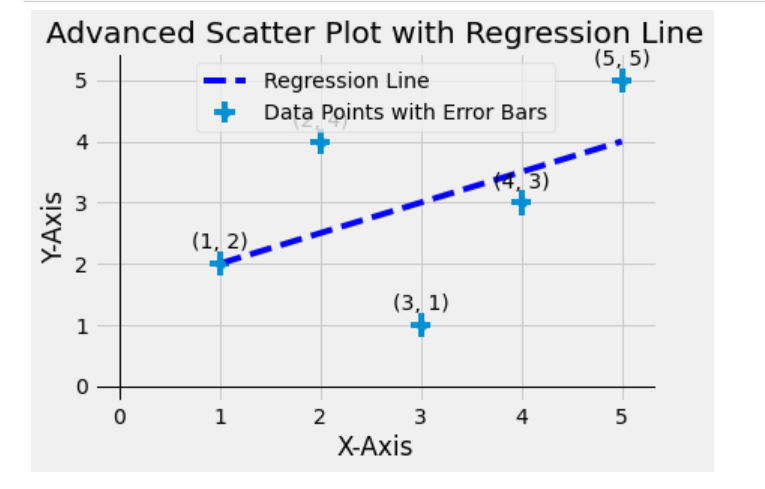
A density histogram is just a modified relative frequency histogram. That is, a density histogram is defined so that: the area of each rectangle equals the relative frequency of the corresponding class, and. the area of the entire histogram equals 1.

**Visualization 4: Scales map data values onto aesthetics (Heatmap)**



**Question 2:**

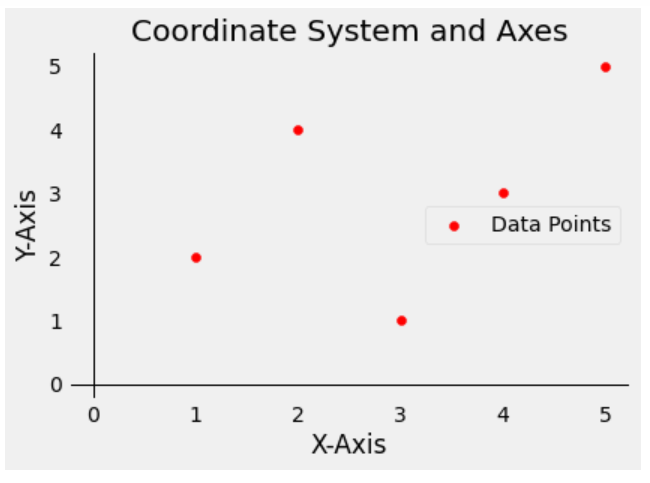
**Visualization 1: Cartesian coordinates**



In this code, we added the following features to the plot:

1. Error bars using the errorbar function to represent the uncertainty in data points.
2. A linear regression line calculated using numpy.polyfit and numpy.poly1d.
3. Annotated labels for data points, which show their coordinates.
4. A legend to distinguish between data points and the regression line.
5. Grid lines for better visualization.

**Visualization 2: Cartesian coordinates (X-axis and Y-axis)**

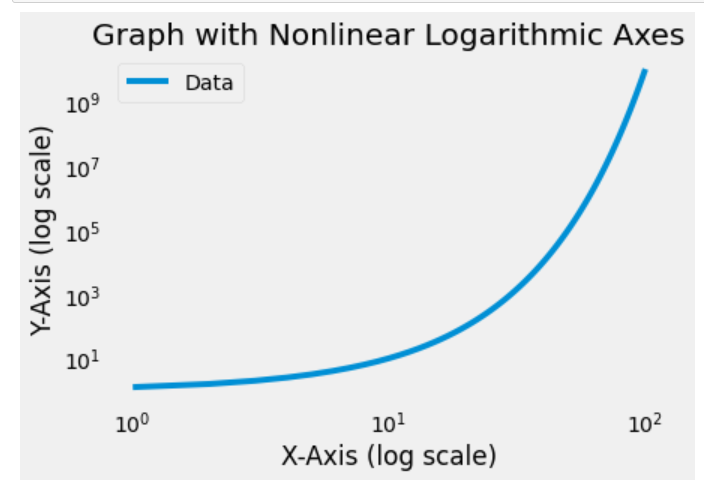


This code creates a basic 2D coordinate system with X and Y axes. It plots data points represented by x\_values and y\_values. You can replace these values with your own data.

After running the code, you will see a visualization of the coordinate system with axes and data points.

Feel free to modify the code to suit your specific data and visualization requirements. You can also incorporate more advanced features and annotations to enhance the visualization, depending on your needs.

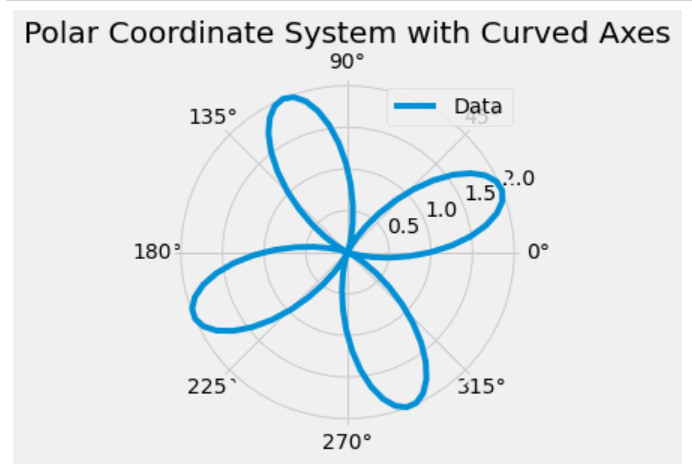
**Visualization 3: Nonlinear axes (Log axis)**



In this code, we used a logarithmic scale for both the X and Y axes to create a graph that visualizes data with a nonlinear relationship. The set\_xscale and set\_yscale functions are used to set the axes to a logarithmic scale. The data points are generated with x\_values and y\_values, where y\_values are calculated as 10 raised to the power of 0.1 times x\_values to create an exponential relationship.

You can adapt this code to other types of nonlinear transformations, such as power-law or exponential scaling, based on your specific needs and data.

**Visualization 4: Coordinate systems with curved axes**

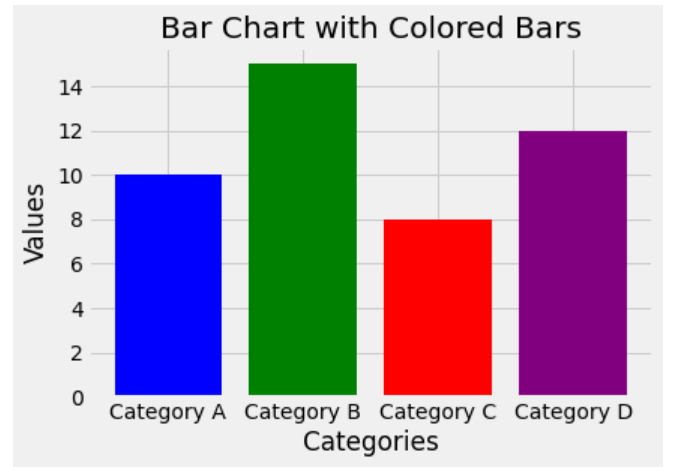


In this code, we use a polar coordinate system, which has curved radial axes. We define custom functions to create the theta and r values for the plot, and we use ax = fig.add\_subplot(111, polar=True) to create a polar plot. We then set radial and angular grid lines and labels.

You can modify this example to create other types of curved axes, such as a logarithmic or custom curve, based on your specific requirements. Keep in mind that creating custom curved axes may involve more complex mathematical transformations and adjustments to the plot.

**Question 3:**

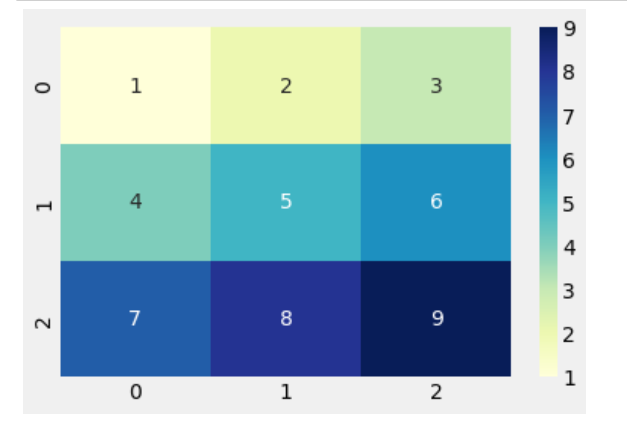
**Visualization 1: Color as a tool to distinguish (bar chart)**

****

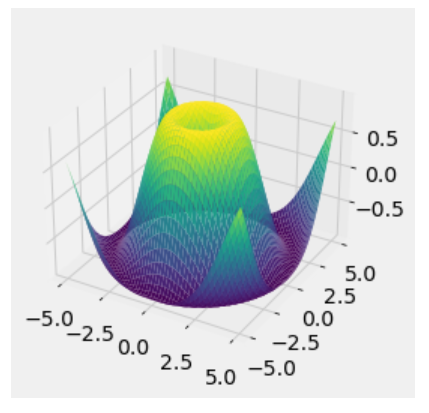
Purpose: A box plot (or box-and-whisker plot) is used to visualize the distribution of a dataset, showing the median, quartiles, and potential outliers.

Explanation: In this example, we are using the Seaborn library to create a box plot that visualizes the distribution of restaurant bill amounts (total\_bill) by day of the week, differentiating by the sex of the patrons. The box represents the interquartile range (IQR), the line inside the box is the median, and the whiskers extend to the minimum and maximum non-outlier values. Outliers are displayed as individual points.

**Visualization 2: Color as a tool to distinguish (Heatmap)**

****

**Visualization 3: Color to represent data values (Meshgrid)**

****

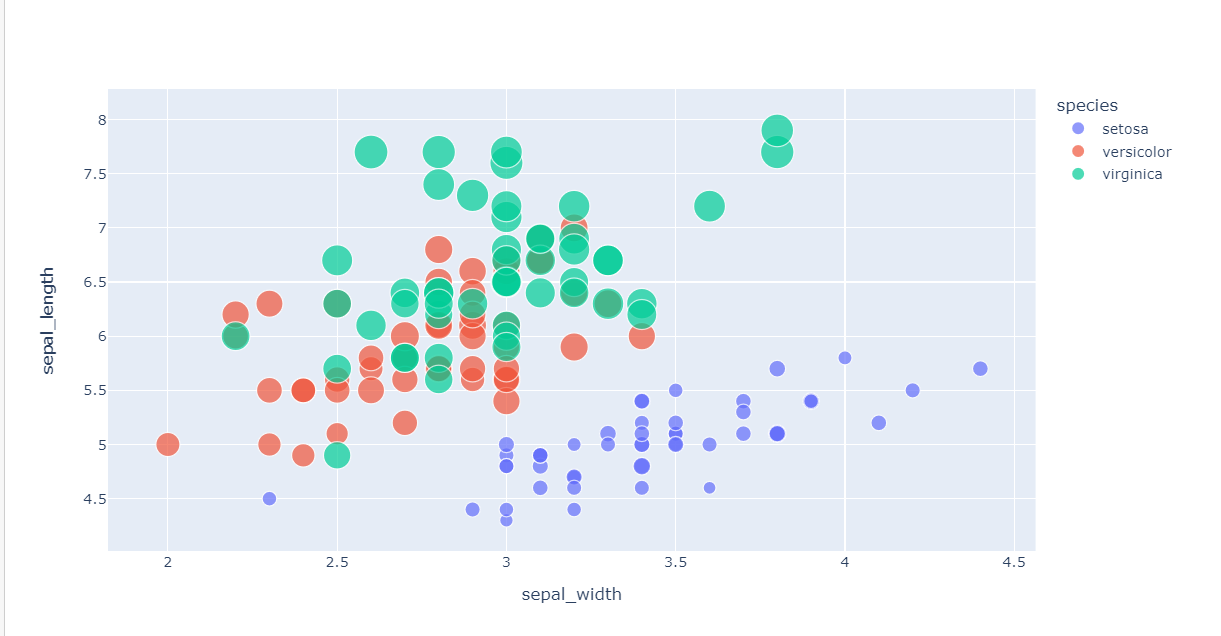
Purpose: 3D surface plots are used to visualize functions of two variables and understand their behavior in three dimensions.

Explanation: In this example, we create a 3D surface plot using Matplotlib. The function Z = sin(sqrt(X^2 + Y^2)) is visualized on a grid in three dimensions. The X and Y coordinates define the grid, while Z represents the height of the surface. The color mapping is done using the 'viridis' colormap, with colors representing the surface height.

**Visualization 4: Color to represent data values (Scatter Plot)**

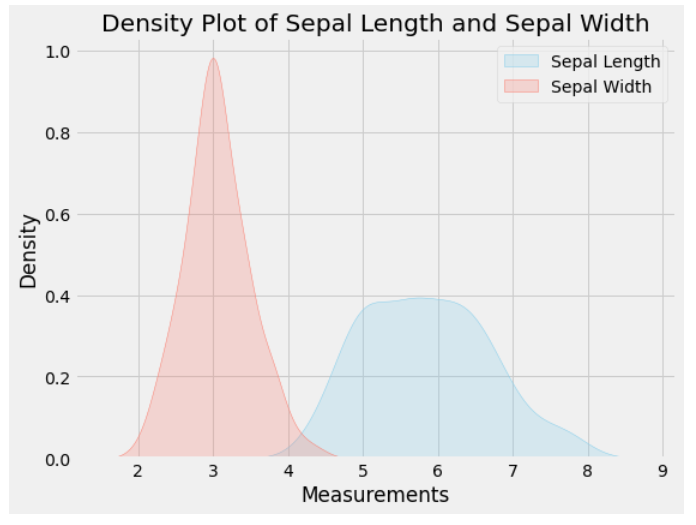
Plotly is a library that allows you to create interactive graphs. Here's an example of an interactive scatter plot. Purpose: Interactive plots provide an engaging way to explore data, allowing users to interact with the visualization.

Explanation: This example uses Plotly Express to create an interactive scatter plot. It displays the relationships between sepal width and sepal length for different species of iris flowers. The color represents the species, and the size of the points corresponds to the petal length. You can hover over points to view additional data (petal width) and interactively pan and zoom in the plot.

****

**Question 4:**

**Visualization 1: Density plot**

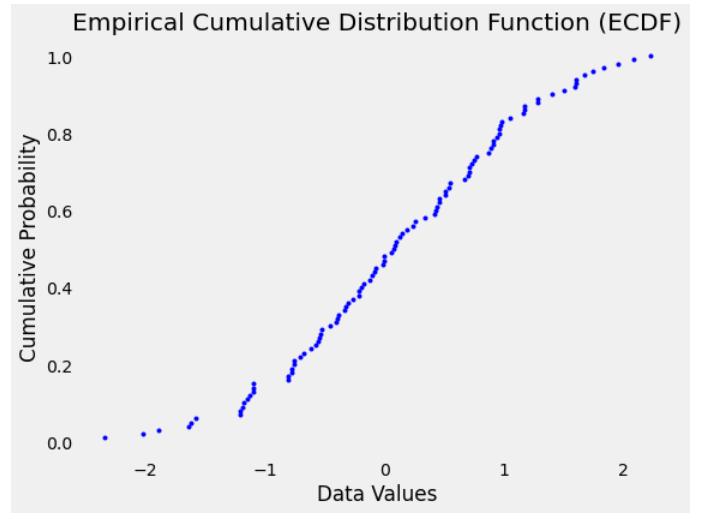
****

In this example, we're using the Iris dataset from Seaborn as a sample dataset. You can replace it with your own dataset.

The sns.kdeplot function is used to create the density plot. We specify the data to plot, set shade=True to fill the area under the curve, and specify colors and labels for different variables. You can create density plots for other variables as needed.

Customize the plot by changing colors, labels, and other visual elements to fit your specific data and presentation requirements.

**Visualization 2: Cumulative plot**

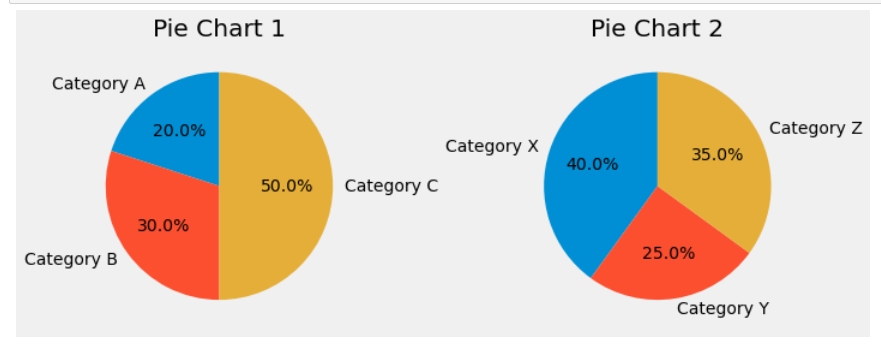
****

In this example, we generate some random data using NumPy. You should replace the data variable with your own dataset. The np.sort function sorts the data in ascending order, and the y variable represents the cumulative probability for each data point. The ECDF is plotted using plt.step, which creates a step function representing the cumulative distribution.

Customize the plot by changing colors, labels, and other visual elements to suit your specific data and presentation needs. This type of plot is especially useful for understanding the distribution and quantiles of a dataset.

**Top of Form**

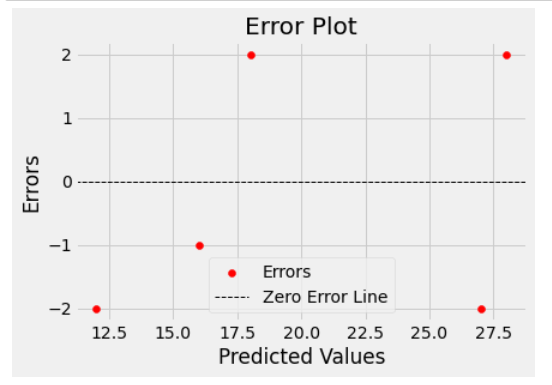
**Visualization 3: Pie chart**

****

In this example, we create two pie charts (Pie Chart 1 and Pie Chart 2) side by side in a single row of subplots. You can customize the data and labels for each pie chart according to your needs. You can also adjust the number of rows and columns in the subplot grid by changing the parameters in plt.subplots.

Make sure to replace the data and labels with your own values to create multiple pie charts that represent different datasets or categories.

**Visualization 4: Error plot**

****

In this example, the scatter plot displays the errors as points, and the "Zero Error Line" is a reference line to help visualize the errors in relation to zero error.

The specific details of your error plot will depend on your dataset and the context of your analysis. If you have a specific error plot or dataset in mind, please provide more details, and I can offer more tailored guidance.