



Training Division of Ctronics InfoTech Pvt. Ltd.

MindSparXs Courseware

Data Visualization using Matplotlib and Seaborn



Data Visualization using Matplotlib, and Seaborn

Data visualization is a crucial activity in the fields of data science, data analysis, machine learning, and deep learning. It involves using various types of graphs to represent data, enabling better understanding of patterns, trends, and insights. Complex projects can be showcased using dashboards that combine multiple visualizations. Software like Grafana, Tableau, and Power BI are often used for dashboard creation.

1. Data Visualization with Pandas

Pandas provides built-in capabilities for generating graphs and charts, often by leveraging the powerful Matplotlib library. This integration allows for quick and convenient data visualization directly from DataFrame objects.

Common Graphs using Pandas:

- Scatter (dot/point graph)
- Line graph
- Bar graph (bar)
- Horizontal bar graph (barh)
- Area graph
- Boxplot
- Histogram
- Pie chart
- Hexbin chart
- Kde chart (less common)

Key Functions and Concepts:

- `DataFrame.plot()`: The primary method for generating plots from Pandas DataFrames. It requires specifying the data for the x and y axes and the kind of graph.
- `kind` parameter: A string that specifies the type of graph to be drawn (e.g., 'scatter', 'line', 'bar', 'area', 'box', 'hist', 'pie', 'hexbin').

- matplotlib.pyplot (commonly imported as plt): Used in conjunction with Pandas plotting for displaying graphs (plt.show()), adding titles (plt.title()), enabling grids (plt.grid(True)), setting figure size (plt.figure(figsize=(width, height))), saving plots (plt.savefig('filename.png')), and setting axis labels (plt.xlabel(), plt.ylabel()).

Code Examples (Pandas):

a. Scatter Plot

```
from pandas import DataFrame
import matplotlib.pyplot as plt
```

```
test = {'Rnos' : [2,3,5,8,9],
        'Percentage' : [52,78,67,75,92]}
df = DataFrame(test)
```

```
df.plot(x='Rnos', y='Percentage', kind='scatter', title='Student Scores Scatter Plot')
plt.grid(True)
plt.show()
```

b. Line Graph

```
from pandas import DataFrame
import matplotlib.pyplot as plt
```

```
test = {'Numbers' : [2,3,5,8,9],
        'Scores' : [52,78,67,75,92],
        }
df = DataFrame(test)
```

```
df.plot(x='Numbers', y='Scores', kind='line', title='Student Scores Line Graph')
plt.grid(True)
plt.show()
```

c. Area Graph

```
from pandas import DataFrame
import matplotlib.pyplot as plt
```

```
test = {'Numbers' : [2,3,5,8,9],
        'Scores' : [52,78,67,75,92],
        }
df = DataFrame(test)
```

```
df.plot(x='Numbers', y='Scores', kind='area', color='red', title='Student Scores Area Graph')
plt.show()
```

d. Bar Graph (Vertical)

```
from pandas import DataFrame
import matplotlib.pyplot as plt
```

```
test = {'Numbers' : [2,3,5,8,9],
        'Scores' : [52,78,67,75,92],
        }
df = DataFrame(test)
```

```
df.plot(x='Numbers', y='Scores', kind='bar', title='Student Scores Bar Graph')
plt.grid(True)
plt.show()
```

e. Multiple Bar Graph

```
from pandas import DataFrame
import matplotlib.pyplot as plt
```

```
test = {'Rno' : [2,3,5,8,9],
        'Scores' : [52,78,67,75,92],
        'Age' : [23,25,26,29,20]
        }
df = DataFrame(test)
```

```
df.plot(x='Rno', y=['Scores','Age'], kind='bar', title='Scores and Age by Rno')
plt.grid(True)
plt.show()
```

f. Boxplot

```
from pandas import DataFrame
import matplotlib.pyplot as plt
```

```
test = {'Rno' : [2,3,5,8,9],
        'Scores' : [52,78,67,75,92],
        }
df = DataFrame(test)
```

```
# Boxplot for 'Scores'
```

```
df.plot(y=['Scores'], kind='box', title='Scores Boxplot')
plt.grid(True)
plt.show()
```

```
# Boxplot for multiple columns (if they existed and made sense)
```

```
# Example with 'Scores' and 'Age' if 'Age' was in the test dict
```

```
# df.plot(y=['Scores', 'Age'], kind='box', title='Scores and Age Boxplot')
# plt.grid(True)
# plt.show()
```

g. Histogram

```
from pandas import DataFrame
import matplotlib.pyplot as plt
```

```
test = {'Numbers' : [2,3,5,8,9,7,1],
        'Scores' : [12,34,56,23,33,35,11],
        }
df = DataFrame(test)
```

```
df.plot(y='Scores', kind='hist', bins=5, title='Histogram of Student Scores') # Specify y for histogram
plt.grid(True)
plt.show()
```

h. Pie Chart

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

```
# Sample data for pie chart
data = pd.Series([40, 25, 20, 15],
                 index=["Apples", "Bananas", "Cherries", "Mango"])
```

```
# Pie chart using pandas Series plotting
data.plot.pie(figsize=(6,6), startangle=90, autopct='%1.1f%%', legend=True, title='Fruit Distribution')
plt.ylabel("") # Remove y-label as it's redundant for pie charts
plt.show()
```

i. Hexbin Plot

```
from pandas import DataFrame
import matplotlib.pyplot as plt
```

```
test = {'Rollno' : [2,3,5,8,9,10,12,15,18,20],
        'Percentage' : [52,78,52,78,92,60,70,80,75,85],
        }
df = DataFrame(test)
```

```
df.plot(x='Rollno', y='Percentage', kind='hexbin', gridsize=15, title='Rollno vs Percentage Hexbin')
```

```
n Plot')  
plt.grid(True)  
plt.show()
```

2. Data Visualization with Matplotlib

Matplotlib is a fundamental plotting library in Python, providing a wide array of static, animated, and interactive visualizations. It is built on NumPy and often used as the backend for other libraries like Pandas and Seaborn.

Key Modules and Concepts:

- `matplotlib.pyplot` (as `plt`): Provides a MATLAB-like interface for creating plots.
- `matplotlib.figure.Figure`: The top-level container for all plot elements.
- `matplotlib.axes.Axes`: The region of the figure where data is plotted.
- `plot()`: Generates line and/or dot graphs.
- `bar()`, `barh()`: For creating vertical and horizontal bar charts, respectively.
- `pie()`: For creating pie charts.
- `hist()`: For creating histograms.
- `scatter()`: For creating scatter plots.
- `subplot()`: For creating multiple plots within a single figure.
- `figure()`: Creates a new figure or activates an existing one. Used for setting size, color, etc.
- `title()`, `xlabel()`, `ylabel()`: Functions to add descriptive text to plots.
- `grid()`: Adds a grid to the plot.
- `savefig()`: Saves the figure to a file.
- `show()`: Displays the plot.
- 3D Plotting: Achieved using the `mpl_toolkits.mplot3d` toolkit.

Code Examples (Matplotlib):

a. Line Graph (Co-ordinated)

import matplotlib.pyplot as plt

a = [12,22,43,74]

b = [10,4,9,16]

plt.figure(figsize=(10,5))

plt.plot(a, b, marker='o', linestyle='-', color='g') # 'go' can also be used for green circles

plt.title('Co-ordinated Line Graph')

plt.xlabel('X Series')

plt.ylabel('Y Series')

plt.grid(True)

plt.show()

b. Bar Graph

import matplotlib.pyplot as plt

div = ['Merit','First','Second','Third','Failed'] # Label list

total = [10,60,45,23,12] # Data list

plt.bar(div, total, color='skyblue')

plt.title('Student Results - 2025')

plt.xlabel('Divisions')

plt.ylabel('No of students')

plt.grid(axis='y') # Grid on y-axis for bar charts

plt.show()

c. Stacked Bar Graph

import matplotlib.pyplot as plt

div = ['Merit','First','Second','Third','Failed'] # labels list

girls = [5,20,20,25,10] # data list 1

boys = [5,40,25,35,10] # data list 2

plt.bar(div, girls, color='blue', label="Girls")

plt.bar(div, boys, bottom=girls, color='red', label="Boys") # 'bottom' stacks the second bar on top of the first

plt.title('Student Results - 2025 (Stacked)')

```
plt.xlabel('Division')
plt.ylabel('No of students')
```

```
plt.legend(loc='best') # Displays the legend
plt.show()
```

d. Pie Chart

```
import matplotlib.pyplot as plt
```

```
div = ['Merit','First','Second','Third','Failed'] # label list
result = [20,35,10,15,20] # data list (should sum to 100 for clear percentage)
```

```
# explode = [0,0.1,0,0,0] # Optional: to 'explode' a slice
```

```
plt.pie(result, labels=div, autopct='%1.1f%%', startangle=45, shadow=True)
plt.title('Distribution of Results')
# plt.legend(title='Divisions') # Legend can also be added this way
plt.show()
```

e. Histogram

```
import matplotlib.pyplot as plt
```

```
x = [21,22,23,4,5,6,7,7,8,9,10,31,32,33,34,35,36,37,18,49,50,100]
num_bins = 5 # You can adjust the number of bins
```

```
plt.hist(x, bins=num_bins, color='orange', edgecolor='black')
plt.title('Histogram of Data Distribution')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.grid(axis='y', alpha=0.75)
plt.show()
```

f. Scatterplot

```
import matplotlib.pyplot as plt
```

```
x = [5, 7, 8, 7, 2, 17, 2, 9, 4, 11, 12, 9, 6] # x series
y = [99, 86, 87, 88, 100, 86, 103, 87, 94, 78, 77, 85, 86] # y series
```

```
plt.figure(figsize=(8,6))
plt.scatter(x, y, marker='o', color='red', s=50) # s is marker size
plt.title('Scatter Plot of X vs Y')
plt.xlabel('X Values')
plt.ylabel('Y Values')
plt.grid(True)
plt.show()
```

g. Subplots

```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(12, 5))
```

```
# Subplot 1 (Row 1, Col 2, Index 1)
```

```
plt.subplot(1, 2, 1)
```

```
a = [1,2,3,4]
```

```
b = [1,4,9,16]
```

```
plt.plot(a,b, 'go-') # green circles with line
```

```
plt.title('Subplot Number One')
```

```
plt.xlabel('X values')
```

```
plt.ylabel('Y values')
```

```
plt.grid(True)
```

```
# Subplot 2 (Row 1, Col 2, Index 2)
```

```
plt.subplot(1, 2, 2)
```

```
x = [2,4,5,9]
```

```
y = [3,8,1,6]
```

```
plt.plot(x,y, 'r^--') # red triangles with dashed line
```

```
plt.title('Sub Plot Number Two')
```

```
plt.xlabel('X values')
```

```
plt.ylabel('Y values')
```

```
plt.grid(True)
```

```
plt.suptitle('Subplot Examples using Matplotlib') # Super title for the entire figure
```

```
plt.tight_layout(rect=[0, 0.03, 1, 0.95]) # Adjust layout to prevent title overlap
```

```
plt.show()
```

3. Data Visualization with Seaborn

Seaborn is a powerful Python data visualization library built on top of Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn is closely integrated with Pandas DataFrames.

Key Features and Relationship:

- Built on Matplotlib: Leverages Matplotlib for rendering plots.
- Pandas Integration: Works seamlessly with Pandas DataFrames.
- Statistical Focus: Designed for visualizing statistical models and data distributions.
- Enhanced Aesthetics: Often produces more visually appealing plots than Matplotlib by default.

- **Additional Plot Types:** Offers specialized plots not readily available in Matplotlib or Pandas (e.g., violinplot, jointplot, pairplot).

Common Graphs using Seaborn:

- Violinplot
- Scatterplot / Implot (with regression line)
- Countplot
- Distplot (Distribution Plot)
- JointPlot (combines scatter and distribution plots)
- Boxplot
- KdePlot (Kernel Density Estimate Plot)
- PairPlot (matrix of plots for all numeric variables)

Key Functions and Concepts:

- `sns.violinplot()`: Visualizes data distribution using a violin shape.
- `sns.scatterplot()`: Creates scatter plots to show relationships between two variables.
- `sns.lmplot()`: Similar to scatterplot but includes a regression line.
- `sns.countplot()`: Displays the counts of observations in each categorical bin.
- `sns.distplot()`: Plots a histogram with an optional Kernel Density Estimate (KDE) curve.
- `sns.jointplot()`: Shows bivariate distributions; default is scatter plot with marginal histograms. Can be customized with `kind` (e.g., 'reg', 'kde', 'hex').
- `sns.boxplot()`: Displays data distribution through quartiles.
- `sns.kdeplot()`: Visualizes the probability density of a dataset.
- `sns.pairplot()`: Generates a matrix of scatterplots for pairwise relationships in a dataset and histograms on the diagonal. Can use `hue` for coloring by category.
- `sns.load_dataset()`: Loads built-in datasets from Seaborn for practice.
- `sns.set_style()`: Sets the aesthetic style of plots (e.g., 'whitegrid', 'darkgrid').

Code Examples (Seaborn):

a. Violinplot

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

# Load a sample dataset
tips_df = sns.load_dataset('tips')
sns.set_style('whitegrid')

plt.figure(figsize=(8, 5))
sns.violinplot(x='day', y='total_bill', data=tips_df, palette='viridis')
plt.title('Total Bill Distribution by Day (Violinplot)')
plt.show()
```

b. Scatterplot with Regression Line (lmlot)

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

tips_df = sns.load_dataset('tips')
sns.set_style('darkgrid')

# Scatterplot with regression line
sns.lmlot(x='total_bill', y='tip', data=tips_df, height=5, aspect=1.2)
plt.title('Total Bill vs Tip with Regression Line')
plt.show()
```

c. Countplot

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

tips_df = sns.load_dataset('tips')
sns.set_style('whitegrid')

plt.figure(figsize=(7, 5))
sns.countplot(x='sex', data=tips_df, palette='pastel')
plt.title('Count of Customers by Gender')
plt.show()
```

d. Distribution Plot (Distplot)

```
import pandas as pd
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
crash_df = sns.load_dataset('car_crashes')
```

```
sns.set_style('whitegrid')
```

```
plt.figure(figsize=(8, 5))
```

```
# distplot is deprecated, use histplot or displot
```

```
sns.histplot(crash_df['alcohol'], kde=True, bins=10, color='purple') # kde=True adds the KDE curve
```

```
plt.title('Distribution of Alcohol Involvement in Crashes')
```

```
plt.xlabel('Alcohol Involvement')
```

```
plt.ylabel('Frequency')
```

```
plt.show()
```

e. Joint Plot

```
import pandas as pd
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
crash_df = sns.load_dataset('car_crash')
```

```
sns.set_style('whitegrid')
```

```
# Jointplot with regression line
```

```
sns.jointplot(x='speeding', y='alcohol', data=crash_df, kind='reg', height=6)
```

```
plt.suptitle('Joint Distribution of Speeding and Alcohol Involvement', y=1.02) # Adjust title position
```

```
plt.show()
```

f. Boxplot

```
import pandas as pd
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
tips_df = sns.load_dataset('tips')
```

```
sns.set_style('whitegrid')
```

```
plt.figure(figsize=(10, 6))
```

```
sns.boxplot(x='day', y='total_bill', data=tips_df, palette='Blues')
```

```
plt.title('Total Bill Boxplot by Day')
```

```
plt.show()
```

g. Pair Plot

```
import pandas as pd
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
tips_df = sns.load_dataset('tips')
```

```
sns.set_style('whitegrid')
```

```
# PairPlot to show relationships between all numeric variables
```

```
# hue='sex' colors points based on the 'sex' column
```

```
sns.pairplot(tips_df, hue='sex', palette='Reds', height=2.5)
```

```
plt.suptitle('PairPlot of Tips Dataset (Colored by Sex)', y=1.02)
```

```
plt.show()
```