01-04-2025

Training Day – 39

Topic: Customizing Visualizations

- Added annotations, adjusted figure sizes, and used advanced legends.
- Example: Annotated key points in a scatter plot.

A scatter plot uses dots to represent values for two different numeric variables. In Python, we have a library matplotlib in which there is a function called scatter that helps us to create Scatter Plots. Here, we will use matplotlib.pyplot.scatter() method to plot.

Syntax : *matplotlib.pyplot.scatter(x,y)*

Parameters:

- *x and y are float values and are the necessary parameters to create a scatter plot*
- *marker : MarkerStyle, default: rcParams["scatter.marker"] (default: 'o')*
- *cmap : cmapstr or Colormap, default: rcParams["image.cmap"] (default: 'viridis')*
- *linewidths : float or array-like, default: rcParams["lines.linewidth"] (default: 1.5)*
- *alpha : float, default: None → represents the transparency*

Annotation of matplotlib means that we want to place a piece of text next to the scatter. There can be two cases depending on the number of the points we have to annotate :

1. Single point annotation
2. All points annotation

Single Point annotation

In single-point annotation we can use matplotlib.pyplot.text and mention the x coordinate of the scatter point and y coordinate + some factor so that text can be distinctly visible from the plot, and then we have to mention the text.

Syntax: *matplotlib.pyplot.text(x, y, s)*

Parameters:

- *x, y : scalars — The position to place the text. By default, this is in data coordinates. The coordinate system can be changed using the transform parameter.*
- *s : str — The text.*
- *fontsize — It is an optional parameter used to set the size of the font to be displayed.*

Approach:

1. Import libraries.
2. Create data.
3. Make scatter plot.
4. Apply plt.text() method.

02-04-2025 # Training Day – 40

Topic: Final Data Analysis

- Conducted descriptive and inferential analyses on the final dataset.
- Example: Analyzed correlations between variables using .corr().

After cleaning and combining datasets, the next critical step in the data analysis process is to conduct both **descriptive** and **inferential analyses** to uncover meaningful insights and relationships within the data. This step helps summarize the data and make predictions or inferences based on it. Below, we'll cover key techniques used in final data analysis.

1. Descriptive Analysis

Descriptive statistics summarize and describe the characteristics of the dataset. This includes measures of central tendency (mean, median, mode), dispersion (variance, standard deviation), and the distribution of variables.

- **Key Metrics:**
  - **Mean:** The average of a dataset.
  - **Median:** The middle value when data is sorted.
  - **Mode:** The most frequently occurring value.
  - **Standard Deviation:** Measures the spread of data points around the mean.
  - **Variance:** The square of the standard deviation.
  - **Skewness:** Measures the asymmetry of the distribution.
  - **Kurtosis:** Measures the "tailedness" of the distribution.
- **Example: Descriptive Statistics in Python**

```
import pandas as pd

# Sample dataset
data = pd.DataFrame({
    'Age': [23, 45, 22, 34, 40],
    'Salary': [45000, 54000, 47000, 58000, 60000]
})

# Descriptive statistics
descriptive_stats = data.describe()
print(descriptive_stats)
```

Output:

```
shell
Copy code
        Age       Salary
count   5.000000      5.000000
mean    32.800000  52800.000000
std     8.460517   5907.926474
min     22.000000  45000.000000
25%     23.000000  47000.000000
50%     34.000000  54000.000000
75%     40.000000  58000.000000
max     45.000000  60000.000000
```

2. **Analyzing Correlations Between Variables**

Understanding the relationships between variables is crucial in data analysis. **Correlation** is a statistical measure that expresses the extent to which two variables are linearly related. The correlation coefficient ranges from -1 (perfect negative correlation) to +1 (perfect positive correlation), with 0 meaning no correlation.

- **Pearson Correlation:** Measures the linear relationship between two continuous variables.
- **Spearman Rank Correlation:** Used for ordinal data or when the relationship between variables is not linear.
- **Example: Correlation Analysis**

```
import pandas as pd

# Sample dataset
data = pd.DataFrame({
    'Age': [23, 45, 22, 34, 40],
    'Salary': [45000, 54000, 47000, 58000, 60000],
    'Experience': [1, 10, 2, 8, 12]})
# Calculate correlation matrix
corr_matrix = data.corr()
print(corr_matrix)
```

Output:

Markdown

```
    Age   Salary  Experience

Age        1.000000  0.967858    0.822845
Salary     0.967858  1.000000    0.970010
Experience  0.822845  0.970010    1.000000
```

From the output, we can see:

- The **Salary** and **Experience** variables are highly positively correlated with each other (0.97).
- There is a strong positive correlation between **Age** and **Salary** (0.97), indicating that older individuals in this sample tend to have higher salaries.

3. Inferential Analysis

Inferential analysis involves making predictions or inferences about a population based on a sample. This typically involves hypothesis testing, regression analysis, and confidence intervals. Key techniques include:

- **Hypothesis Testing:**
  - **Null Hypothesis (H0):** A statement of no effect or no difference.
  - **Alternative Hypothesis (H1):** The statement that there is an effect or difference.

    - **P-value:** Used to assess the strength of the evidence against the null hypothesis (usually, $p < 0.05$ is considered statistically significant).
    - **t-tests / ANOVA:** Used to compare means between groups.
- **Regression Analysis:**
    - **Linear Regression:** Used to predict the value of a dependent variable based on one or more independent variables.
    - **Logistic Regression:** Used when the dependent variable is categorical (e.g., binary classification).

03-04-2025 # Training Day – 41

Topic: Creating a Dashboard

- Integrated multiple Matplotlib visualizations into one figure.
- Example: Combined a line chart, bar chart, and pie chart in subplots.

Matplotlib allows you to combine multiple visualizations (such as line charts, bar charts, and pie charts) into a single figure using **subplots**. This is useful when you want to display different types of visualizations side-by-side for comparative purposes or for a more comprehensive view of the data.

1. Using Subplots in Matplotlib

Subplots allow you to arrange multiple plots in a grid layout. You can specify the number of rows and columns in the grid, and then plot different visualizations in each grid cell.

2. Example: Combining Line Chart, Bar Chart, and Pie Chart in Subplots

In this example, we'll create a figure that contains three different plots:

A **line chart** showing a trend over time.

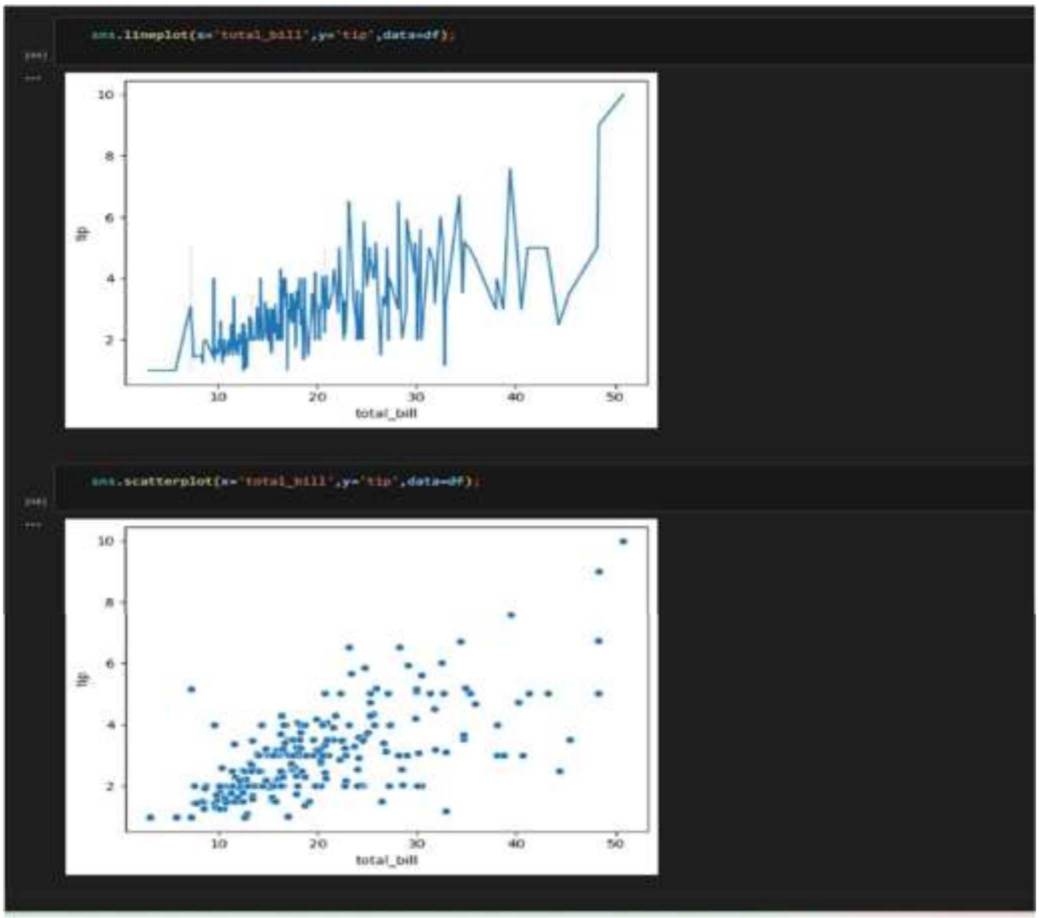
A **bar chart** representing categorical data.

A **pie chart** showing the proportions of categories.

Code Example:

```
import matplotlib.pyplot as plt
import numpy as np

# Sample data
x = np.arange(1, 6)
y1 = [2, 4, 6, 8, 10]  # Line chart data
y2 = [5, 3, 6, 2, 7]   # Bar chart data
labels = ['A', 'B', 'C', 'D', 'E']  # Pie chart categories
sizes = [15, 30, 45, 10, 20]  # Pie chart data

# Create a figure with 1 row and 3 columns
fig, axs = plt.subplots(1, 3, figsize=(15, 5))

# Line chart in the first subplot
axs[0].plot(x, y1, marker='o', color='b', label='Trend')
axs[0].set_title('Line Chart')
axs[0].set_xlabel('X Axis')
axs[0].set_ylabel('Y Axis')
axs[0].legend()

# Bar chart in the second subplot
axs[1].bar(x, y2, color='g', label='Values')
axs[1].set_title('Bar Chart')
axs[1].set_xlabel('Categories')
axs[1].set_ylabel('Values')
axs[1].set_xticks(x)
axs[1].set_xticklabels(labels)
```

```
axs[1].legend()

# Pie chart in the third subplot
axs[2].pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
axs[2].set_title('Pie Chart')

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()
```



04-04-2025 **Training Day – 42**

Topic: Summary of Key Learnings

- Documented techniques learned over the past weeks.
- Example: Listed best practices for data cleaning and visualization.

1. ***Data Cleaning Techniques***

Handling Missing Data:

Imputation: Filling missing values using mean, median, or mode (for numerical data) or the most frequent value (for categorical data).

Removal: Dropping rows or columns with too many missing values.

Interpolation: For time series or sequential data, missing values can be interpolated based on surrounding data points.

Example:

```
df.fillna(df.mean(), inplace=True)    # Impute missing values with column mean
```

Data Transformation:

Normalization/Standardization: Scaling numeric data to a standard range, often required for machine learning models.

Log Transformation: Used to deal with skewed distributions by applying a logarithmic scale.

Categorical Encoding: Converting categorical variables into numeric formats using one-hot encoding or label encoding.

Example:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df['scaled_column'] = scaler.fit_transform(df[['column']])
```

Outlier Detection and Removal:

Z-Score Method: Identifying and removing data points that deviate significantly from the mean (e.g., z-scores greater than 3).

IQR Method: Removing data points outside the interquartile range (Q1 - 1.5 * IQR, Q3 + 1.5 * IQR).

Example:

```
from scipy import stats
df = df[(np.abs(stats.zscore(df['column'])) < 3)]    # Remove outliers based on Z-score
```

2. Combining Multiple Datasets

Concatenation: Combining datasets vertically (stacking rows) or horizontally (adding columns) using `concat()`.

Merging: Joining datasets based on common columns or indices using `merge()` (similar to SQL joins).

1st Insights: How many Flights with respect to their Stopages ?

```
df['Total_Stops'].value_counts()
```

[77]

```
Total_Stops
1    5626
0    3491
2    1520
3      45
4       1
Name: count, dtype: int64
```

```
# From This Histogram  we can see that no.of flights and their Stopages
# In this Data maximum flights have 1 stopages
# And there are few flights which have 3rd and 4th stopages

plt.title('No. of fligths Stopage')
plt.hist(df['Total_Stops'],color='purple')
plt.xlabel('Stopages')
plt.ylabel('No.of Flights')
plt.xticks(df['Total_Stops'].unique())
plt.show()
```

[78]