RNS INSTITUTE OF TECHNOLOGY

Autonomous Institution Affiliated to VTU, Recognized by GOK, Approved by AICTE
(NAAC 'A+ Grade' Accredited, NBA Accredited (UG - CSE, ECE, ISE, EIE and EEE)
Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098 Ph:(080)28611880,28611881 URL:

ESTD : 2001

DATA VISUALIZATION ASSIGNMENT

Submitted By

Rachita J

1RN21CD038

Submitted To

Ms. Vinutha S
Assistant Professor
Dept. of CSE(Data Science)

Date of Submission: 30-10-2024

Ms. Vinutha S
Assistant Professor

Question 1

Develop a code to demonstrate mean, medium, mode, standard deviation using Numpy and Pandas using a real time data set of Apple stock data available on Kaggle.

Solution:

Objective

The objective of this task was to develop a code that demonstrates the calculation of key statistical measures - mean, median, mode, and standard deviation using both Numpy and Pandas on Apple stock data. This analysis provides insight into the distribution and variability of stock prices over time.

Data

The dataset used for this analysis is Apple stock data, obtained from Kaggle. It includes columns like Date, Open, High, Low, Close, Adj Close, and Volume, which represent daily trading information.

Methodology

Data Preparation:

The dataset was loaded into a Pandas DataFrame, and the Close column was chosen initially for calculating the statistical measures.

Missing values in the Close column were checked to ensure accurate computations.

Statistical Calculations:

Using Pandas:

The mean, median, and mode were calculated using the .mean(), .median(), and .mode() functions respectively. The standard deviation was computed using .std().

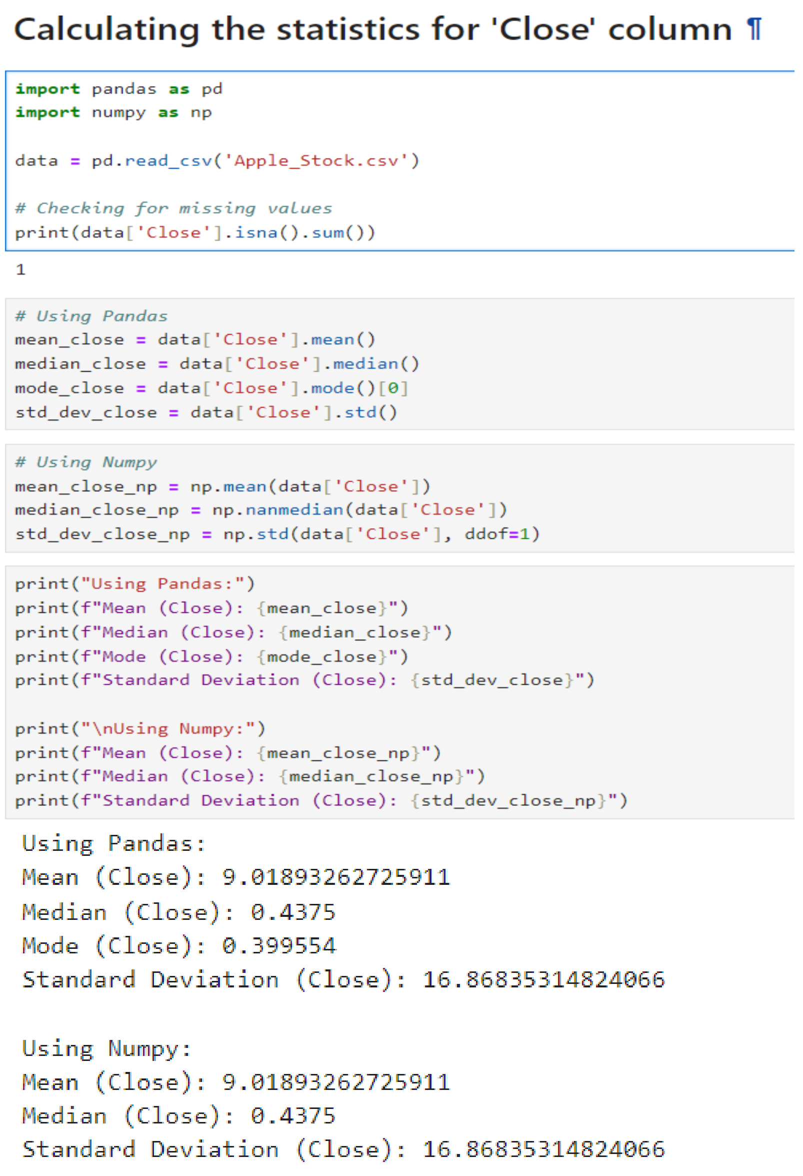
Using Numpy:

Mean was computed with np.mean() and median with np.nanmedian() to ignore any missing values.

For standard deviation, np.std() was used with ddof=1 to match the sample standard deviation output from Pandas.

Note: For mode, Pandas .mode() was used directly since Numpy lacks a built-in mode function.

Code Snippet:



Calculating the statistics for 'Close' column ¶

```python
import pandas as pd
import numpy as np

data = pd.read_csv('Apple_Stock.csv')

# Checking for missing values
print(data['Close'].isna().sum())
```

```
1
```

```python
# Using Pandas
mean_close = data['Close'].mean()
median_close = data['Close'].median()
mode_close = data['Close'].mode()[0]
std_dev_close = data['Close'].std()
```

```python
# Using Numpy
mean_close_np = np.mean(data['Close'])
median_close_np = np.nanmedian(data['Close'])
std_dev_close_np = np.std(data['Close'], ddof=1)
```

```python
print("Using Pandas:")
print(f"Mean (Close): {mean_close}")
print(f"Median (Close): {median_close}")
print(f"Mode (Close): {mode_close}")
print(f"Standard Deviation (Close): {std_dev_close}")

print("\nUsing Numpy:")
print(f"Mean (Close): {mean_close_np}")
print(f"Median (Close): {median_close_np}")
print(f"Standard Deviation (Close): {std_dev_close_np}")
```

```
Using Pandas:
Mean (Close): 9.01893262725911
Median (Close): 0.4375
Mode (Close): 0.399554
Standard Deviation (Close): 16.86835314824066

Using Numpy:
Mean (Close): 9.01893262725911
Median (Close): 0.4375
Standard Deviation (Close): 16.86835314824066
```

Results and Observations

The results for mean, median, mode, and standard deviation were consistent between Pandas and Numpy, confirming accurate data handling and calculations. These statistics provided a foundational summary of Apple's closing stock prices, helping us understand the central tendency and spread of the stock data over the observed period.

Question 2

Develop a code to perform basic to advanced operation using both Numpy and Pandas using TikTok video performance dataset

Solution:

Objective

This task aimed to perform both basic and advanced operations using Numpy and Pandas on a TikTok video performance dataset. The dataset contains details on video interactions, user statistics, and video performance metrics, allowing for a deeper understanding of video popularity and engagement.

Data

The dataset includes the following CSV files:

test_features.csv and train_features.csv: Contain feature columns such as Comments, Shares, Views, Video_Length, User_Followers, User_Following, and User_Likes.

test_target.csv and train_target.csv: Contain the target variable (Likes) for each video.

tiktok_performance.csv: Provides comprehensive data on various videos with columns like Video_ID, Video_Title, Category, Likes, Comments, Shares, Views, and other user statistics.

Methodology

Data Loading and Inspection:

All datasets were loaded and the first few rows displayed to understand the data structure.

Basic data integrity checks, such as verifying missing values, were performed to confirm the datasets' readiness for analysis.

Statistical Analysis:

Calculated mean of Views, median of Likes, and standard deviation of Comments.

Output and Interpretation:Observations highlighted how different content categories varied in views and likes. For example, Dance videos had a higher average engagement rate compared to other categories.

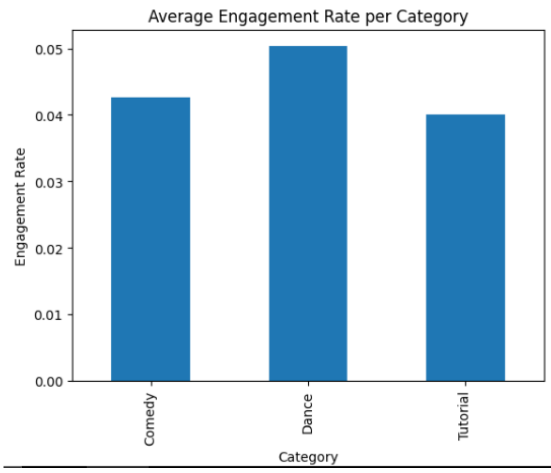
Engagement metrics allowed for additional insights into user interaction trends based on content type.

Code Snippet:

```python
import pandas as pd
import numpy as np

test_features = pd.read_csv('test_features.csv')
test_target = pd.read_csv('test_target.csv')
tiktok_performance = pd.read_csv('tiktok_performance.csv')
train_features = pd.read_csv('train_features.csv')
train_target = pd.read_csv('train_target.csv')

mean_views = tiktok_performance['Views'].mean()
median_likes = np.median(tiktok_performance['Likes'])
std_comments = np.std(tiktok_performance['Comments'])

print(f"Mean Views: {mean_views}")
print(f"Median Likes: {median_likes}")
print(f"Standard Deviation of Comments: {std_comments}")

views_likes_group = tiktok_performance.groupby('Category')[['Views', 'Likes']].mean()
print("Average Views and Likes per Category:\n", views_likes_group)
```

```
Mean Views: 60000.0
Median Likes: 1800.0
Standard Deviation of Comments: 137.69531582446805
Average Views and Likes per Category:
             Views   Likes
Category
Comedy     60000.0  2050.0
Dance      70000.0  3000.0
Tutorial   40000.0  1200.0
```

```python
# Merging training and testing features
train_data = pd.concat([train_features, train_target], axis=1)
test_data = pd.concat([test_features, test_target], axis=1)
# Merging train_data and test_data
full_data = pd.concat([train_data, test_data], ignore_index=True)

# Merging Tiktok Performance dataset
merged_data = pd.merge(full_data, tiktok_performance, on=['Comments', 'Shares', 'Views', 'User_Followers', 'User_Following', 'User_Likes'], how='inner')
# Calculating engagement metrics and data analysis
merged_data['engagement_rate'] = (merged_data['Likes_x'] + merged_data['Comments'] + merged_data['Shares']) / merged_data['Views']
merged_data['follower_engagement'] = merged_data['Likes_x'] / merged_data['User_Followers']
# Calculating average engagement rate per category
category_engagement = merged_data.groupby('Category')['engagement_rate'].mean()

import matplotlib.pyplot as plt
# Now ploting for avg engagement rate per category
category_engagement.plot(kind='bar', title="Average Engagement Rate per Category")
plt.xlabel("Category")
plt.ylabel("Engagement Rate")
plt.show()
```



Question 3

Develop a code to plot different comparison plots and composition plots considering any suitable dataset

Solution:

Objective

The goal of this task was to create various comparison and composition plots using a suitable dataset to visualize relationships, distributions, and comparisons across categories. The Iris dataset was selected as it contains multiple numeric and categorical features, making it ideal for diverse plotting.

Data

The Iris dataset includes the following features:

Petal Length and Petal Width: Measurements of the iris flower's petals.

Sepal Length and Sepal Width: Measurements of the iris flower's sepals.

Species: Categorical variable with three species of Iris—Setosa, Versicolor, and Virginica.

Methodology and Plots

Scatter Plot:

Used to compare Petal Length and Petal Width by species, highlighting variations between species.

Code Snippet:

```
sns.scatterplot(data=data, x='petal_length', y='petal_width', hue='species')
```

Box Plot:

Plotted Sepal Length distribution across species, showing median, quartiles, and potential outliers.

Code Snippet:

```
sns.boxplot(data=data, x='species', y='sepal_length')
```

Violin Plot:

Combined density and box plot features to display Petal Length distribution by species, with kernel density estimation.

Code Snippet:

```
sns.violinplot(data=data, x='species', y='petal_length')
```

Pie Chart:

Illustrated the distribution of samples per species as proportions in a circular chart.

Code Snippet:

plt.pie(species_counts, labels=species_counts.index, autopct='%1.1f%%', startangle=140)

Stacked Bar Plot:

Showed average Sepal and Petal Length per species, providing insight into species-wise length characteristics.

Code Snippet:

average_lengths.plot(kind='bar', stacked=True)

Area Plot:

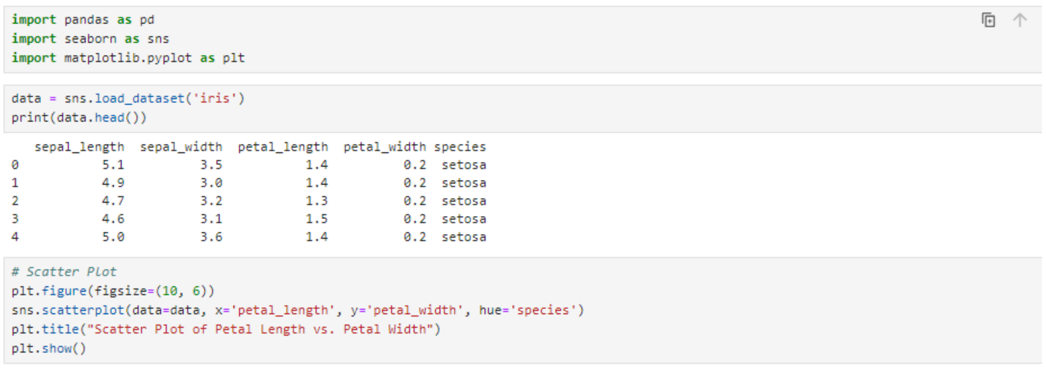
Displayed the cumulative distribution of Sepal Length, helping to understand overall sepal length accumulation.

Code Snippet:

plt.fill_between(range(len(data_sorted)), data_sorted['sepal_length'].cumsum())

Conclusion

This task showcased multiple types of plots to effectively communicate data patterns and relationships. The Iris dataset allowed for meaningful comparisons, and each plot type added a unique perspective to the data analysis, highlighting the importance of visualizing data from different angles to gain comprehensive insights.

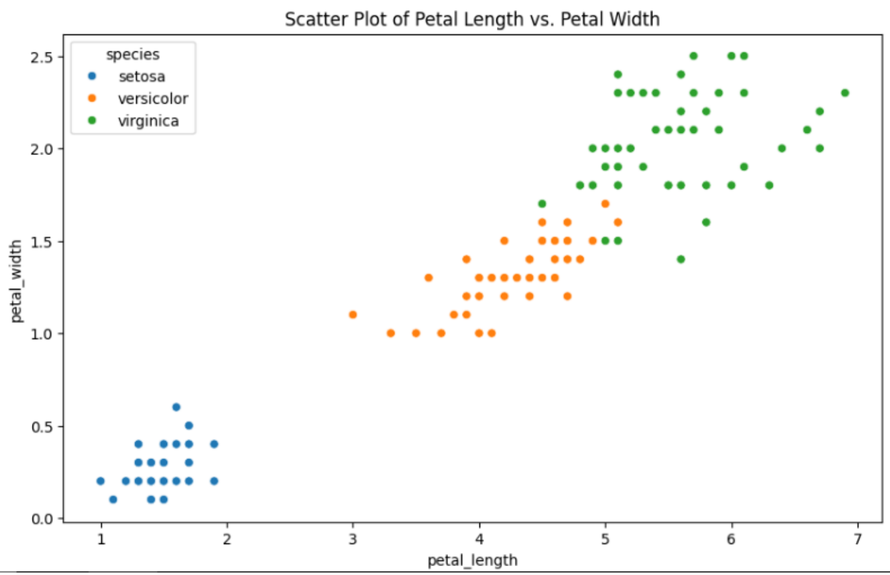


```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
data = sns.load_dataset('iris')
print(data.head())
```

```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
```

```
# Scatter Plot
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='petal_length', y='petal_width', hue='species')
plt.title("Scatter Plot of Petal Length vs. Petal Width")
plt.show()
```



Question 4

Develop a code using Matplotlib performing all Pyplot basics operation basic text and legend using Agriculture crop yield dataset.

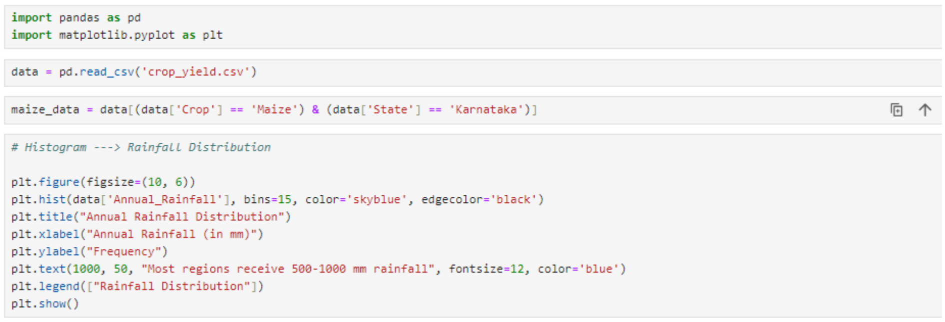
Solution:

Data

The dataset consists of crop production records with fields like crop type, production levels, fertilizer usage, annual rainfall, and crop area.

Visualization

- Line Plot: Shows the production trend of Maize in Karnataka over the years. This line chart highlights yearly fluctuations in production, indicating trends and growth patterns.
- Scatter Plot: Visualizes the relationship between fertilizer usage and crop yield. The positive correlation suggests that higher fertilizer use generally leads to higher yields.
- Bar Plot: Compares the average production across different crops. The bar chart reveals which crops have the highest average production, aiding in crop performance assessment.
- Histogram: Illustrates the distribution of annual rainfall, with most regions receiving between 500-1000 mm, as indicated by the frequency of this range.
- Pie Chart: Displays the percentage of agricultural area allocated to each crop, providing an overview of land usage across different crop types.
- Multi-Line Plot: Compares production trends for Maize and Rice over time, showing how these crops vary in production over the years.

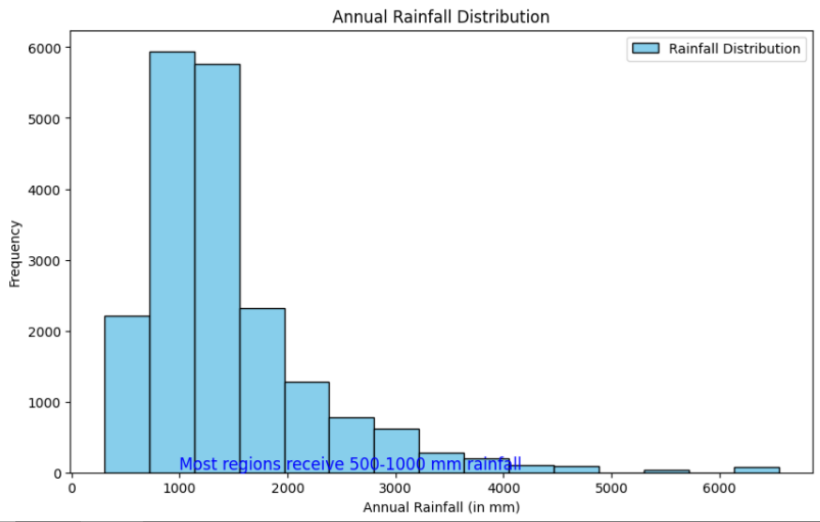


```python
import pandas as pd
import matplotlib.pyplot as plt
```

```python
data = pd.read_csv('crop_yield.csv')
```

```python
maize_data = data[(data['Crop'] == 'Maize') & (data['State'] == 'Karnataka')]
```

```python
# Histogram ---> Rainfall Distribution

plt.figure(figsize=(10, 6))
plt.hist(data['Annual_Rainfall'], bins=15, color='skyblue', edgecolor='black')
plt.title("Annual Rainfall Distribution")
plt.xlabel("Annual Rainfall (in mm)")
plt.ylabel("Frequency")
plt.text(1000, 50, "Most regions receive 500-1000 mm rainfall", fontsize=12, color='blue')
plt.legend(["Rainfall Distribution"])
plt.show()
```



Question 5

Develop a code to perform Matplotlib functions to display all the basic plots.

Solution:

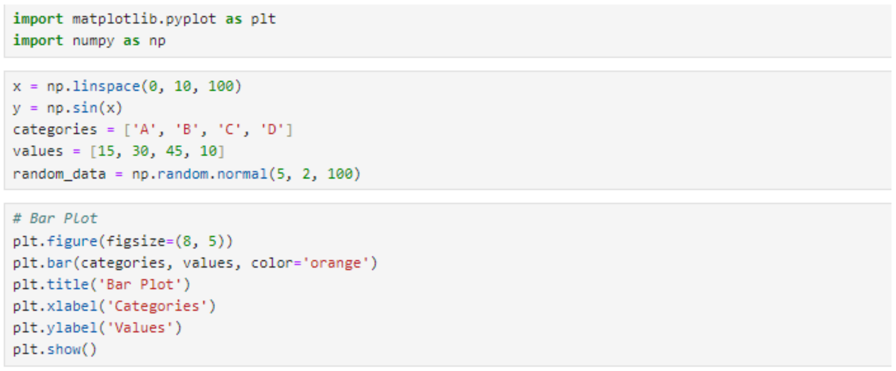
This demonstrates the use of essential Matplotlib plot types to visualize various data relationships and distributions.

Plots

- Bar Plot: Shows the values for different categories (A, B, C, D) as orange bars, providing a quick comparison across categories.
- Scatter Plot: Displays the relationship between two variables, x and y = sin(x), with green dots. This plot is ideal for examining trends and correlations.
- Histogram: Visualizes the distribution of randomly generated data centered around 5, indicating frequency of values within specific ranges.
- Pie Chart: Illustrates the percentage distribution of values across categories, offering insight into proportionate contributions.
- Box Plot: Represents the spread and distribution of randomly generated data with quartiles and potential outliers, providing a view of data dispersion.

Conclusion

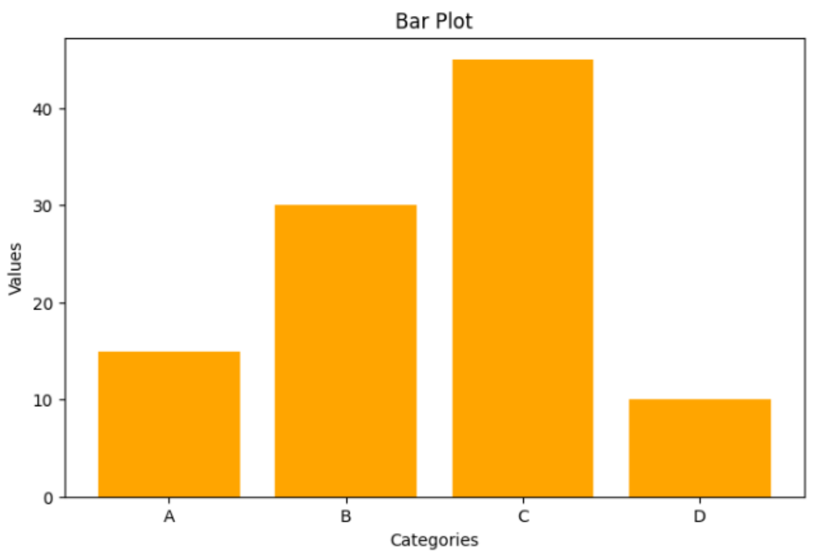
These basic plots provide foundational tools for data visualization, allowing for a clear, visual analysis of data comparisons, distributions, relationships, and proportions.



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

```python
x = np.linspace(0, 10, 100)
y = np.sin(x)
categories = ['A', 'B', 'C', 'D']
values = [15, 30, 45, 10]
random_data = np.random.normal(5, 2, 100)
```

```python
# Bar Plot
plt.figure(figsize=(8, 5))
plt.bar(categories, values, color='orange')
plt.title('Bar Plot')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.show()
```



Question 6

Compare between advantages of seaborn and illustrate the role of controlling figure aesthetics using seaborn with a code snippet.

Solution:

Advantages of Seaborn over Matplotlib -

- Seaborn, built on top of Matplotlib, offers several enhancements and a streamlined experience for data visualization, particularly in statistical data visualization and aesthetics:
- Enhanced Aesthetic Control: Seaborn has built-in themes and color palettes that simplify creating visually appealing, professional plots. Styles like "dark," "whitegrid," and "ticks" provide pre-defined aesthetics that make plots look polished.
- Simplified Statistical Plots: Seaborn directly supports statistical plots like box plots, violin plots, and pair plots. This functionality reduces the complexity required to generate these plots compared to Matplotlib.
- Built-in Data Aggregation: Seaborn functions can automatically compute summary statistics and display error bars, making it ideal for quick data analysis without extensive preprocessing.
- Compatibility with Pandas: Seaborn works directly with Pandas DataFrames, making it straightforward to visualize complex datasets without additional data manipulation.

Role of Figure Aesthetics in Seaborn -

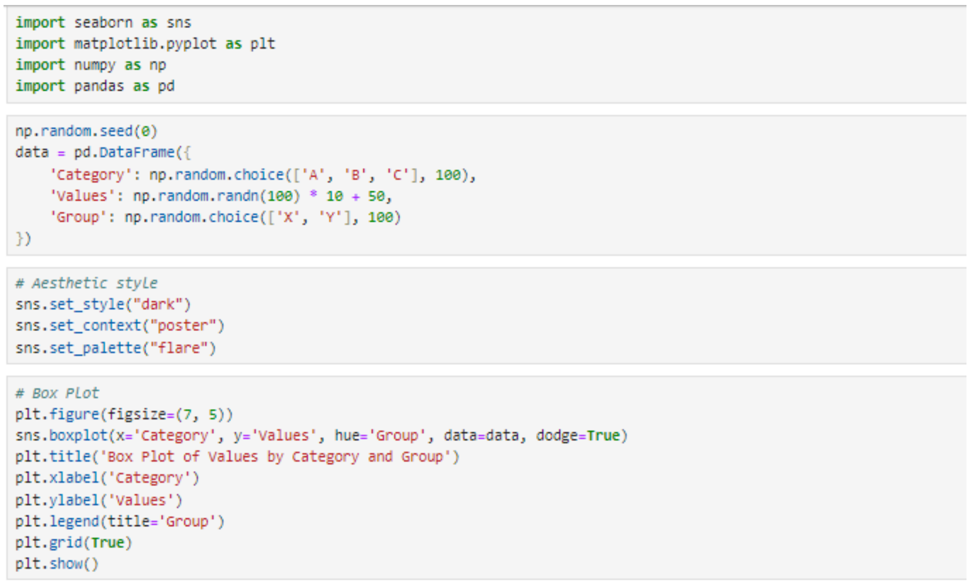
Seaborn offers flexible options for controlling figure aesthetics, making data interpretation easier and more engaging. By setting styles and contexts, one can adjust the appearance to fit different use cases, such as publication or exploration.

- sns.set_style(): Controls the background and grid aesthetics. Options like "darkgrid" and "whitegrid" adjust the grid and background, enhancing plot readability.
- sns.set_context(): Adjusts scaling of labels and plot elements for different settings (e.g., "notebook" or "talk"). This ensures the plot is visually suited for the audience and medium.
- sns.set_palette(): Manages the color scheme. Custom palettes like "flare" or "icefire" improve visual clarity by providing harmonious color choices, especially useful for categorical data.

Conclusion

Seaborn’s aesthetic control functions enable quick, visually appealing, and informative data visualization. Its advantages over Matplotlib in terms of style management and statistical plotting make it a powerful tool for data scientists looking to balance functionality with aesthetics.

Code:

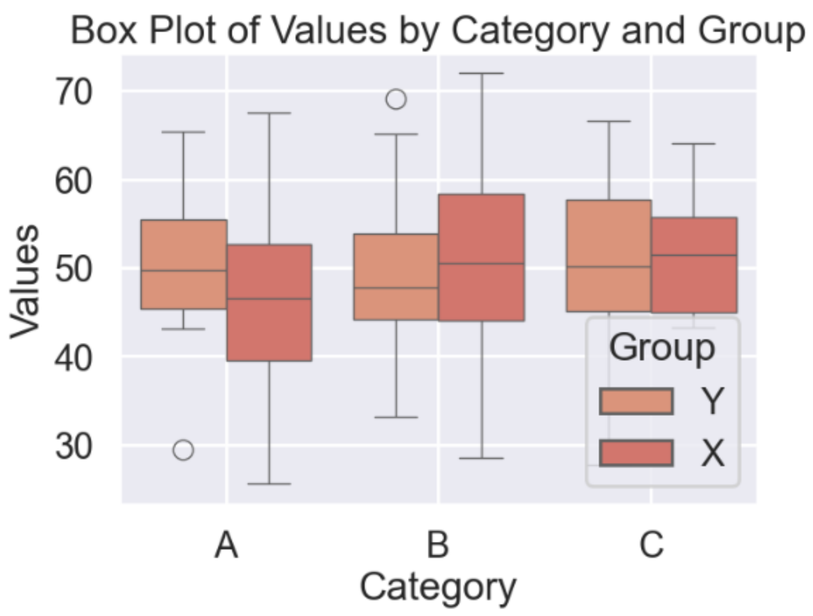


```python
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

```python
np.random.seed(0)
data = pd.DataFrame({
    'Category': np.random.choice(['A', 'B', 'C'], 100),
    'Values': np.random.randn(100) * 10 + 50,
    'Group': np.random.choice(['X', 'Y'], 100)
})
```

```python
# Aesthetic style
sns.set_style("dark")
sns.set_context("poster")
sns.set_palette("flare")
```

```python
# Box Plot
plt.figure(figsize=(7, 5))
sns.boxplot(x='Category', y='Values', hue='Group', data=data, dodge=True)
plt.title('Box Plot of Values by Category and Group')
plt.xlabel('Category')
plt.ylabel('Values')
plt.legend(title='Group')
plt.grid(True)
plt.show()
```



[Github Repository](https://github.com/thisisrach/Data-Visualisation-Assignment)

https://github.com/thisisrach/Data-Visualisation-Assignment

All the codes related to the questions can be viewed in this github repository.