Name : Ravi kumar yadav
E-mail : kumaryadavravi016@gmail.com
Assignment name : Data structure

Drive : drive

Github : Github

1 Assignment Data Toolkit

1.1 Thoery questions

1.1.1 1)What is NumPy, and why is it widely used in Python ?

NumPy (Numerical Python) is a powerful library used for numerical computing in Python. It provides support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy is widely used because it is highly optimized for performance, especially in numerical computations, and provides efficient handling of arrays with vectorized operations.

1.1.2 2) How does broadcasting work in NumPy ?

Broadcasting in NumPy refers to the ability to perform element-wise operations on arrays of different shapes. NumPy automatically adjusts the shape of one or both arrays in a way that allows operations to be performed without explicit looping. This helps with memory efficiency and makes the code concise.

1.1.3 3) What is a Pandas DataFrame?

A Pandas DataFrame is a two-dimensional, size-mutable, and potentially heterogeneous tabular data structure. It is similar to a table in a relational database or a data frame in R. It allows for the manipulation and analysis of data in rows and columns, where each column can have different data types.

1.1.4 4) Explain the use of the groupby() method in Pandas.

The groupby() method in Pandas is used to split data into groups based on some criteria (like column values), apply a function to each group, and then combine the results. It's a powerful tool for aggregation and transformation of data.

1.1.5 5) Why is Seaborn preferred for statistical visualizations ?

Seaborn is built on top of Matplotlib and provides a higher-level interface for creating attractive and informative statistical graphics. It comes with many built-in themes, color palettes, and functions to make complex statistical visualizations easier to produce.

1.1.6 6) What are the differences between NumPy arrays and Python lists?

(i) Performance: NumPy arrays are more memory-efficient and faster for numerical operations than Python lists.

(ii)Homogeneity: NumPy arrays store elements of the same data type, while Python lists can contain elements of different types.

(iii) Functionality: NumPy provides a wide range of vectorized operations and mathematical functions, whereas Python lists do not have these features.

1.1.7 7) What is a heatmap, and when should it be used?

A heatmap is a graphical representation of data where individual values are represented by colors. It is often used to visualize correlation matrices, patterns, or values across two-dimensional data, such as in machine learning, statistics, or data analysis.

1.1.8 8) What does the term "vectorized operation" mean in NumPy?

A vectorized operation refers to the ability to apply operations on entire arrays or large datasets without the need for explicit loops. This is a key feature in NumPy, which allows it to perform operations more efficiently and in a more readable way.

1.1.9 9) How does Matplotlib differ from Plotly?

(i) Matplotlib: A static plotting library used for creating a wide range of 2D plots and visualizations. It is highly customizable but lacks interactivity.

(ii) Plotly: A library for creating interactive visualizations with rich user interfaces, including zoom, hover, and dynamic updates. Plotly is particularly suited for web applications.

1.1.10 10) What is the significance of hierarchical indexing in Pandas?

Hierarchical indexing in Pandas allows for multiple levels of indexing on a DataFrame or Series, enabling the storage and manipulation of multi-dimensional data in a two-dimensional table. It is useful for working with complex datasets, such as time-series data with multiple categories.

1.1.11 11) What is the role of Seaborn's pairplot() function?

The pairplot() function in Seaborn is used to visualize pairwise relationships in a dataset. It creates a matrix of scatterplots (for numerical features) and histograms (for distributions) to explore the relationships between different variables in a dataset.

1.1.12 12) What is the purpose of the describe() function in Pandas?

The describe() function in Pandas provides summary statistics (mean, standard deviation, min, max, etc.) for numerical columns in a DataFrame or Series. It is used for quick exploration and understanding of the distribution of the data.

1.1.13 13) Why is handling missing data important in Pandas ?

Handling missing data is important to ensure the integrity and accuracy of data analysis. Missing values can introduce biases, lead to incorrect results, and distort statistical computations. Pandas provides methods like isna(), fillna(), and dropna() for managing missing data.

1.1.14 14) What are the benefits of using Plotly for data visualization?

(i) Interactivity: Plotly allows users to zoom, pan, and hover over visualizations to reveal data points.

(ii) Customization: Plotly provides extensive options for customizing graphs and charts.

(iii) Web-Ready: Plotly graphs are easy to integrate into web applications.

1.1.15 15) How does NumPy handle multidimensional arrays?

NumPy supports multidimensional arrays (e.g., matrices, tensors) through its ndarray object. You can perform element-wise operations, slicing, and reshaping on these arrays. NumPy provides functions to handle arrays with any number of dimensions efficiently.

1.1.16 16) What is the role of Bokeh in data visualization?

Bokeh is a powerful interactive visualization library that can produce high-quality, browser-based visualizations. It is designed to create interactive plots for web applications and supports both small and large-scale visualizations.

1.1.17 17) Explain the difference between apply() and map() in Pandas.

(i) apply(): Applies a function along the axis of a DataFrame or to each element of a Series.

(ii) map(): Applies a function element-wise to a Series. It is more limited compared to apply() since it is only for Series and can be used for transforming data.

1.1.18 18) What are some advanced features of NumPy?

(i) Linear algebra operations: Matrix multiplication, eigenvalues, and solving linear systems.

(ii) Random sampling: Generating random numbers and distributions.

(iii) Fourier transforms: Fast Fourier Transform (FFT) operations.

(iv) Advanced indexing: Boolean, fancy, and slice indexing.

1.1.19 19) How does Pandas simplify time series analysis?

Pandas provides a rich set of tools for working with time series data, including:

(i) Datetime indexing: Efficient handling of time-based data.

(ii) Resampling: Changing the frequency of time series data.

(iii) Rolling windows: Calculating rolling statistics.

1.1.20 20) What is the role of a pivot table in Pandas?

A pivot table in Pandas allows for the summarization and transformation of data. It can be used to aggregate and group data, typically to compute summary statistics like sums, means, or counts across multiple dimensions.

1.1.21 21) Why is NumPy's array slicing faster than Python's list slicing?

NumPy arrays are stored in contiguous blocks of memory, allowing for faster access and manipulation. In contrast, Python lists are more flexible and are stored as objects with overhead, making them slower for numerical operations and slicing.

1.1.22 22) What are some common use cases for Seaborn?

Some common use cases for Seaborn include:

(i) Correlation heatmaps: To visualize relationships between numerical variables.

(ii) Categorical plots: Boxplots, violin plots, and bar plots for categorical data.

(iii) Statistical distributions: Visualizing the distribution of numerical variables.

1.2 Practical questions

1.2.1 How do you create a 2D NumPy array and calculate the sum of each row?

[60]:
```python
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
row_sums = arr.sum(axis=1)
print("Sum of each row:", row_sums)
```

```
Sum of each row: [ 6 15 24]
```

1.2.2 Write a Pandas script to find the mean of a specific column in a DataFrame.

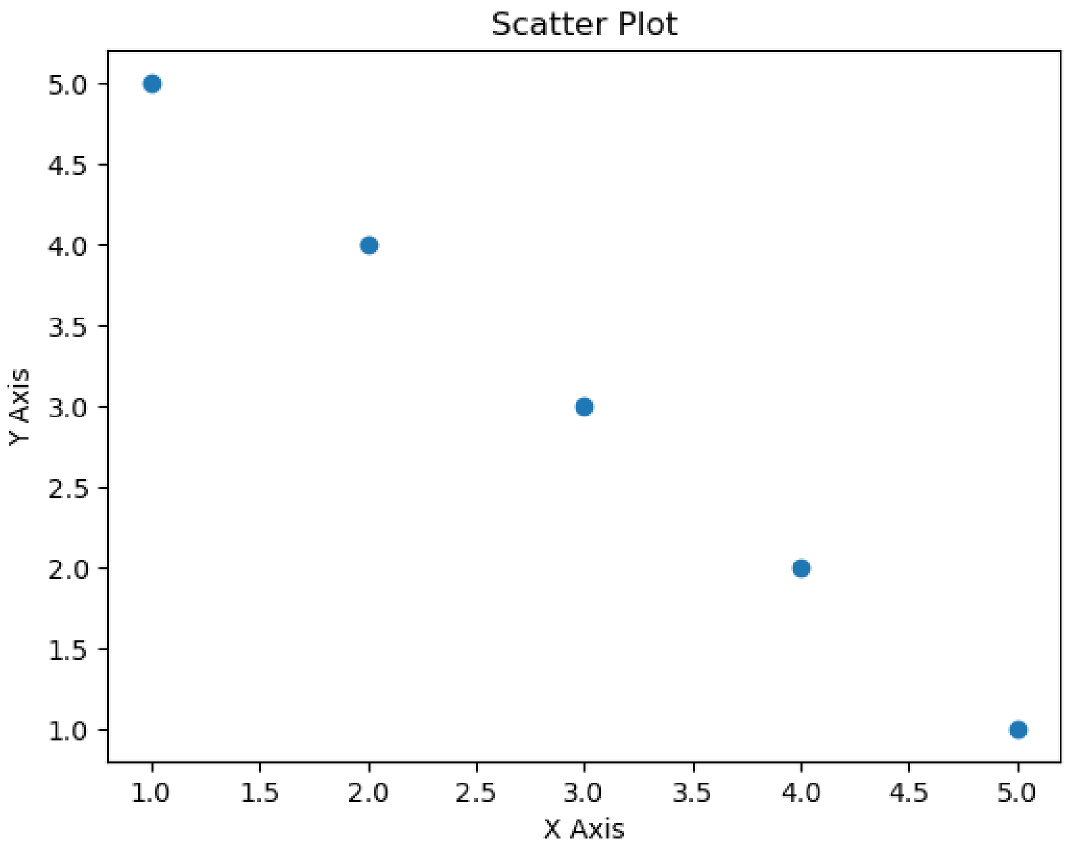
[62]:
```python
import pandas as pd
data = {'A': [1, 2, 3, 4, 5], 'B': [10, 20, 30, 40, 50]}
df = pd.DataFrame(data)
mean_a = df['A'].mean()
```

```
print("Mean of column A:", mean_a)
```

```
Mean of column A: 3.0
```

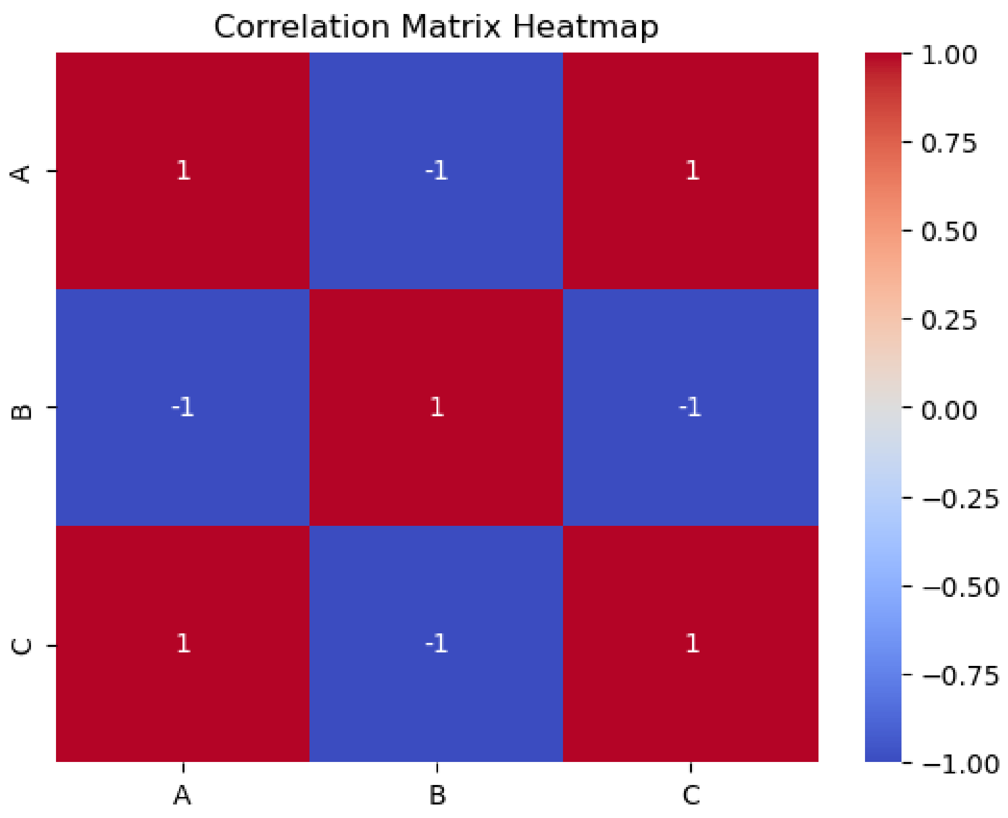
1.2.3 Create a scatter plot using Matplotlib.

[64]:
```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [5, 4, 3, 2, 1]
plt.scatter(x, y)
plt.title("Scatter Plot")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.show()
```



1.2.4 How do you calculate the correlation matrix using Seaborn and visualize it with a heatmap.

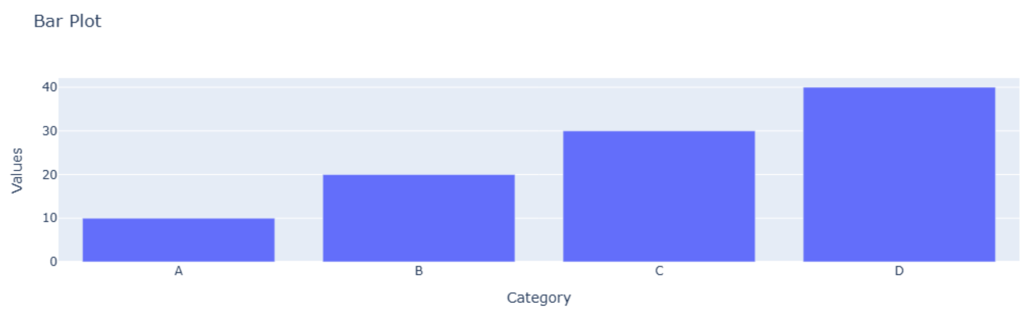
[66]:
```
import seaborn as sns
import pandas as pd
df = pd.DataFrame({
    'A': [1, 2, 3, 4, 5],
    'B': [5, 4, 3, 2, 1],
    'C': [2, 3, 4, 5, 6]
})
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title("Correlation Matrix Heatmap")
plt.show()
```



1.2.5 Generate a bar plot using Plotly.

[68]:
```
import plotly.express as px
data = {'Category': ['A', 'B', 'C', 'D'], 'Values': [10, 20, 30, 40]}
df = pd.DataFrame(data)
fig = px.bar(df, x='Category', y='Values', title="Bar Plot")
```

```
fig.show()
```



1.2.6 Create a DataFrame and add a new column based on an existing column.

[70]:
```
import pandas as pd
df = pd.DataFrame({
    'A': [1, 2, 3, 4, 5]
})
df['B'] = df['A'] * 2
print(df)
```

```
   A   B
0  1   2
1  2   4
2  3   6
3  4   8
4  5  10
```

1.2.7 Write a program to perform element-wise multiplication of two NumPy arrays.

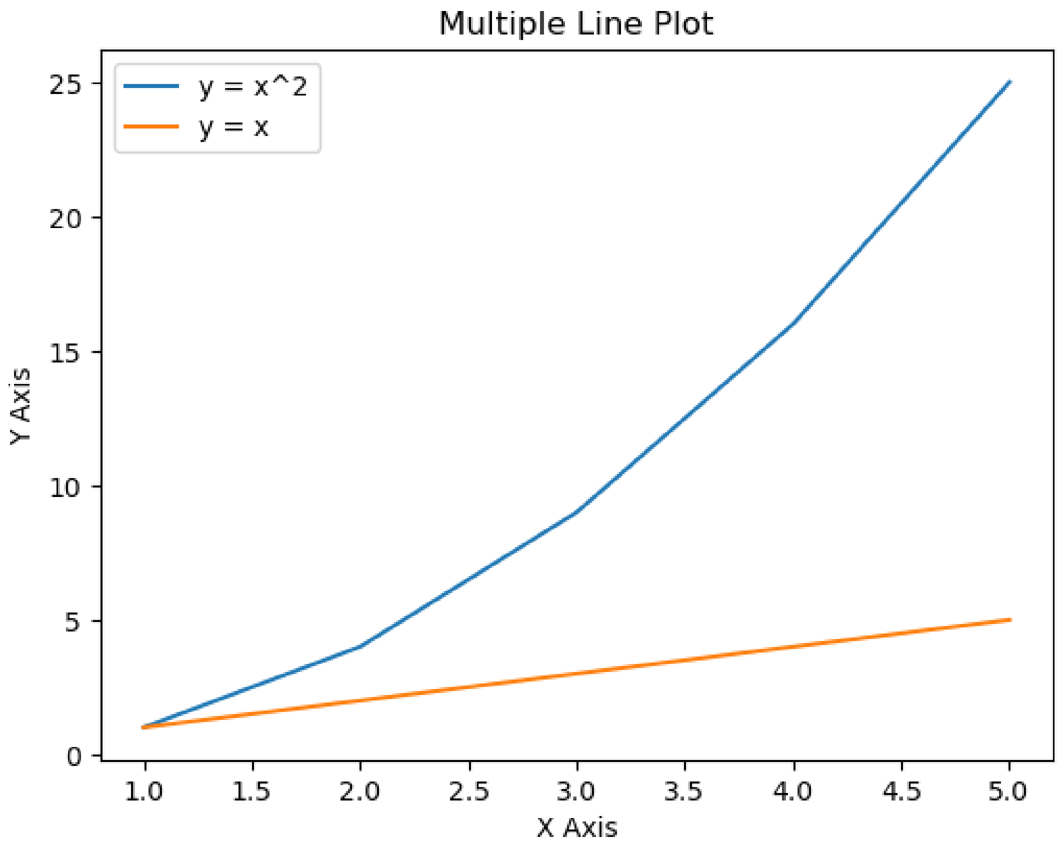
[72]:
```
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
result = arr1 * arr2
print("Element-wise multiplication:", result)
```

```
Element-wise multiplication: [ 4 10 18]
```

1.2.8 Create a line plot with multiple lines using Matplotlib.

[74]:
```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y1 = [1, 4, 9, 16, 25]
y2 = [1, 2, 3, 4, 5]
plt.plot(x, y1, label='y = x^2')
```

```python
plt.plot(x, y2, label='y = x')
plt.title("Multiple Line Plot")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.legend()
plt.show()
```



1.2.9 Generate a Pandas DataFrame and filter rows where a column value is greater than a threshold.

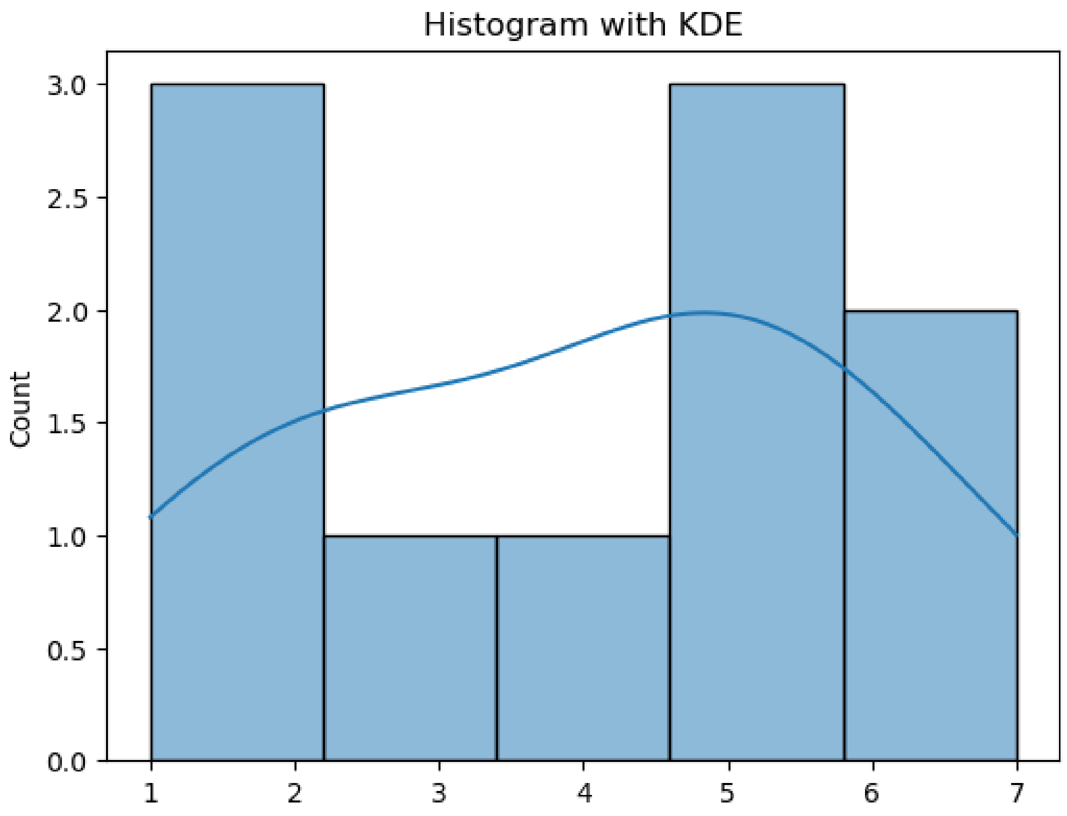
[76]:
```python
import pandas as pd
df = pd.DataFrame({
    'A': [1, 2, 3, 4, 5],
    'B': [10, 20, 30, 40, 50]
})
filtered_df = df[df['A'] > 2]
print(filtered_df)
```

```
   A   B
2  3  30
```

```
3  4  40
4  5  50
```

1.2.10 Create a histogram using Seaborn to visualize a distribution.

[78]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
data = [1, 2, 2, 3, 4, 5, 5, 5, 6, 7]
sns.histplot(data, kde=True)
plt.title("Histogram with KDE")
plt.show()
```



1.2.11 Perform matrix multiplication using NumPy.

[80]:
```python
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
result = np.dot(A, B)
print("Matrix multiplication result:\n", result)
```

```
Matrix multiplication result:
 [[19 22]
 [43 50]]
```

1.2.12 Use Pandas to load a CSV file and display its first 5 rows.

[82]:
```
import pandas as pd
x=pd.read_csv('StudentsPerformance.csv')
x.head()
```

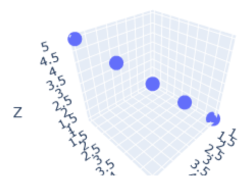
[82]:
```
   gender race/ethnicity parental level of education         lunch  \
0  female        group B           bachelor's degree      standard
1  female        group C                some college      standard
2  female        group B             master's degree      standard
3    male        group A          associate's degree  free/reduced
4    male        group C                some college      standard

  test preparation course  math score  reading score  writing score
0                    none          72             72             74
1               completed          69             90             88
2                    none          90             95             93
3                    none          47             57             44
4                    none          76             78             75
```

1.2.13 Create a 3D scatter plot using Plotly.

[85]:
```
data = {'X': [1, 2, 3, 4, 5], 'Y': [5, 4, 3, 2, 1], 'Z': [1, 2, 3, 4, 5]}
df = pd.DataFrame(data)
fig = px.scatter_3d(df, x='X', y='Y', z='Z', title="3D Scatter Plot")
fig.show()
```

3D Scatter Plot



[ ]:
```
```