1.Consider the following NumPy array:

arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

What will be the output of:

a) arr[1:, :2]

b) arr[::-1, ::2]

c) arr.reshape(1, -1)

d) np.where(arr > 5)

1. NumPy Array Operations

Given the NumPy array:

```python
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

Click to expand

a) `arr[1:, :2]`

This expression selects all rows starting from index 1 and the first two columns.

- Output:

```
array([[4, 5],
[7, 8]])
```

Click to expand

b) `arr[::-1, ::2]`

This expression reverses the rows and selects every second column.

- Output:

```
array([[7, 9],
[4, 6],
...
```

Click to expand

c) `arr.reshape(1, -1)`

This expression reshapes the array into a single row with as many columns as needed to accommodate all elements.

- Output:

```
array([[1, 2, 3, 4, 5, 6, 7, 8, 9]])
```

Click to expand

d) `np.where(arr > 5)`

This expression returns the indices of the elements that are greater than 5.

- Output:

```
(array([1, 1, 2, 2, 2]), array([2, 1, 0, 1, 2]))
```

Click to expand

This indicates that the elements greater than 5 are located at positions (1, 2), (1, 1), (2, 0), (2, 1), and (2, 2).

2.Given a pandas DataFrame containing student records:

data = {

'Name': ['John', 'Alice', 'Bob'],

'Score1': [85, 92, 78],

'Score2': [90, 88, 85]

}

Write expressions for:

a) Calculate mean score per student

b) Filter students with average score > 85

c) Sort by Score1 in descending order

d) Add a new column for grade based on average score

2. Pandas DataFrame Operations

Given the DataFrame:

python 8 lines Click to expand

```
import pandas as pd
...
```

a) Calculate mean score per student

python 1 lines Click to expand

```
df['Average'] = df[['Score1', 'Score2']].mean(axis=1)
```

b) Filter students with average score > 85

python 1 lines Click to expand

```
filtered_students = df[df['Average'] > 85]
```

c) Sort by Score1 in descending order

python 1 lines Click to expand

```
sorted_df = df.sort_values(by='Score1', ascending=False)
```

d) Add a new column for grade based on average score

python 9 lines Click to close

```
def assign_grade(avg):
if avg >= 90:
...
```

3.Analyze the time complexity and space complexity of:

a) NumPy array broadcasting

b) Pandas DataFrame merging

c) GroupBy operations

d) Sorting in pandas

3. Time Complexity and Space Complexity Analysis

a) NumPy Array Broadcasting

- **Time Complexity**: O(n), where n is the number of elements in the larger array. Broadcasting involves aligning the shapes of the arrays, which can be done in linear time.
- **Space Complexity**: O(1) for the operation itself, but it may require additional space for the resulting array if the shapes are not compatible.

b) Pandas DataFrame Merging

- **Time Complexity**: O(n + m), where n and m are the sizes of the two DataFrames being merged. The merge operation involves scanning both DataFrames.
- **Space Complexity**: O(n + m) for storing the merged DataFrame.

c) GroupBy Operations

- **Time Complexity**: O(n), where n is the number of rows in the DataFrame. Grouping involves iterating through the DataFrame to create groups.
- **Space Complexity**: O(k), where k is the number of unique groups created, as it requires space to store the grouped data.

d) Sorting in Pandas

- **Time Complexity**: O(n log n), where n is the number of elements in the DataFrame. Sorting algorithms like Timsort (used in pandas) have this complexity.
- **Space Complexity**: O(n) for the temporary storage required during the sorting process.

4.Design a data processing pipeline using pandas and NumPy to:

 Load and clean a large dataset

 Handle missing values

 Perform feature engineering

 Create visualizationExplain the rationale behind each step and potential optimization techniques.

4. Data Processing Pipeline Using Pandas and NumPy

Steps in the Pipeline

1. **Load and Clean a Large Dataset**:
   - **Rationale**: Loading data from various sources (CSV, SQL, etc.) and ensuring it is in a usable format.
   - **Optimization**: Use `pd.read_csv()` with parameters like `chunksize` for large datasets to load data in manageable chunks.
2. **Handle Missing Values**:
   - **Rationale**: Missing values can skew analysis and lead to incorrect conclusions.
   - **Techniques**: Use `df.fillna()` for imputation or `df.dropna()` to remove missing values.
   - **Optimization**: Use vectorized operations for efficiency.
3. **Perform Feature Engineering**:
   - **Rationale**: Create new features that can improve model performance (e.g., extracting date features, encoding categorical variables).
   - **Techniques**: Use `pd.get_dummies()` for one-hot encoding and `pd.to_datetime()` for date features.
   - **Optimization**: Use efficient data types (e.g., category for categorical variables) to save memory.
4. **Create Visualization**:
   - **Rationale**: Visualizations help in understanding data distributions and relationships.
   - **Tools**: Use libraries like Matplotlib or Seaborn for plotting.
   - **Optimization**: Pre-compute aggregations to reduce the amount of data being visualized.

5.Three factories A, B, C of an electric bulb manufacturing company produces 45%, 35% and 20% of the total output. Approximately 1.5%, l% and 2% of the bulbs produced by these factories are known to be defective. If a randomly selected bulb manufactured by the company was found to be defective, what is the probability that the bulb was manufactured in factory B?

5. Probability Calculation for Defective Bulbs

Let:

- $P(A) = 0.45$ (Factory A)
- $P(B) = 0.35$ (Factory B)
- $P(C) = 0.20$ (Factory C)
- $P(D|A) = 0.015$ (Defective given A)
- $P(D|B) = 0.01$ (Defective given B)
- $P(D|C) = 0.02$ (Defective given C)

We want to find $ P(B|D) $ using Bayes' theorem:
$ P(B|D) = \frac{P(D|B) \cdot P(B)}{P(D)} $
First, we need to calculate $ P(D) $:
$ P(D) = P(D|A) \cdot P(A) + P(D|B) \cdot P(B) + P(D|C) \cdot P(C) $
Calculating $ P(D) $:
$ P(D) = (0.015 \cdot 0.45) + (0.01 \cdot 0.35) + (0.02 \cdot 0.20) $ $ = 0.00675 + 0.0035 + 0.004 = 0.01425 $
Now, substituting back into Bayes' theorem:
$ P(B|D) = \frac{0.01 \cdot 0.35}{0.01425} $ $ = \frac{0.0035}{0.01425} \approx 0.245 $
Thus, the probability that a randomly selected defective bulb was manufactured in factory B is approximately **24.5%**.