NumPy based : Real-World Multidimensional Array Filtering

Table of Contents

1. Problem Statement: Solution for dataset of students scores in different subjects

- We have a dataset of students scores in different subjects and we want to filter the data based on certain criteria.
- We'll assume that we have a dataset of sales transactions with multiple dimensions such as product, region, and time.

Solution for dataset of students scores in different subjects

- imports the NumPy library
- we define a multidimensional array scores using the np.array() function from NumPy. This array represents the scores of different students in various subjects. It is a 2-dimensional array with 5 rows (representing 5 students) and 4 columns (representing 4 subjects).

```
import numpy as np

# Sample multidimensional array representing students' scores
scores = np.array([
    [85, 90, 92, 88],
    [78, 82, 80, 85],
    [92, 88, 95, 90],
    [75, 80, 85, 90],
    [88, 92, 87, 95]
])
```

- We set a threshold value, which represents the minimum score required for a student to be included in the filtered data. In this case, we set it to 90.
- the array scores is filtered based on the condition scores > threshold. This creates a boolean mask where each element in scores is compared to the threshold. Elements that satisfy the condition (scores greater than the threshold) are set to True, and those that do not satisfy the condition are set to False. The resulting boolean mask is then used to index the scores array, which returns a 1-dimensional array of the filtered scores.

```
threshold = 90
filtered_scores = scores[scores > threshold]
print(filtered_scores)
```

```
[92 92 95 92 95]
```

2.Solution for dataset of sales data array

- import the numpy library and create a sample sales data array using np.array().

- Each row in the array represents a sales transaction, with columns for the product, region, month, and sales amount.

```
import numpy as np

# Sample sales data
sales_data = np.array([
    ['Product A', 'Region 1', 'Jan', 100],
    ['Product B', 'Region 2', 'Feb', 200],
    ['Product A', 'Region 2', 'Jan', 150],
    ['Product C', 'Region 1', 'Mar', 300],
    ['Product B', 'Region 1', 'Feb', 250],
    ['Product A', 'Region 1', 'Mar', 120],
])
```

- Next, we define the filter_sales_data function. It takes the data array and optional filter parameters: product, region, and month. Inside the function, we create a variable filtered_data and initialize it with the original data array.

- then check each filter parameter. If a filter parameter is provided (not None), we apply the filter to the filtered_data array using boolean indexing. For example, if product is provided, we filter the data to only include rows where the product in the first column matches the specified product value. We do the same for region and month.

```
# Filter function
def filter_sales_data(data, product=None, region=None, month=None):
    filtered_data = data
    if product:
        filtered_data = filtered_data[filtered_data[:, 0] == product]
    if region:
        filtered_data = filtered_data[filtered_data[:, 1] == region]
    if month:
        filtered_data = filtered_data[filtered_data[:, 2] == month]
    return filtered_data
```

- the filter_sales_data function with the sales_data array and specify the filter values: product='Product A', region='Region 1', and month='Mar'.

- The function filters the sales data based on these parameters and assigns the filtered result to the filtered_data variable. Then, we print the filtered_data array, which contains the rows that match all the specified filters.

```
# Example usage
filtered_data = filter_sales_data(sales_data, product='Product A',
region='Region 1', month='Mar')
print(filtered_data)
```