NumPy based : Multidimensional Array Filtering Using Apartment

Table of Contents

1. Problem Statement:

- Define a 3D, 3x2x4 numpy array. We defined the indexes of this array as **`floor, apartment and room.`**
- Then, with filtering, we extracted the apartment numbers according to the magnets in the kitchens of the apartments in this building.

2. Initial Part

- We have defined a 3D, 3x2x4 numpy array named array_3D.

In [ ]:

```python
import numpy as np
array_3B = np.arange(0,24).reshape(3,2,4)
print("Our array: ",array_3B)
print("Dimensions of our array: ",array_3B.shape)
print("Our array has a size of ",array_3B.ndim," pieces.")
```

```
Our array:  [[[ 0  1  2  3]
  [ 4  5  6  7]]

 [[ 8  9 10 11]
  [12 13 14 15]]

 [[16 17 18 19]
  [20 21 22 23]]]
Dimensions of our array:  (3, 2, 4)
Our array has a size of  3  pieces.
```

- Let's consider the first axis (dimension) of this array as "floors", the second axis as "apartments", and the third axis as "rooms".

- It has three floors ([0],[1], [2]) and two apartments(left[0], right[1]) on each floor. Each apartment has four rooms (living room[0], bedroom[1], kitchen[2], bathroom[3]).
- There are 3x2x4 = 24 rooms in total, and the values we give are the number of tables in each room.

In [ ]:

```
print("Number of 'floors' on the first axis: ",array_3B.shape[0])
print("Number of 'apartments' on the second axis: ",array_3B.shape[1])
print("Number of 'rooms' on the third axis: ",array_3B.shape[2])
```

```
Number of 'floors' on the first axis:  3
Number of 'apartments' on the second axis:  2
Number of 'rooms' on the third axis:  4
```

Let's get, in one row, the number of tables (or fridge magnet) in the kitchens of each right apartment on each floor.

In [ ]:

```
print("The number of magnets in the kitchen of the right apartments on each floor : ",array_3B[0:3,1,2])
```

```
The number of magnets in the kitchen of the right apartments on each floor :  [ 6 14 22]
```

Let's list the apartments with more than 3 magnets in their kitchens.

In [ ]:

```
array_3B_filter = array_3B[array_3B[:,:,2] > 3]
# Just to get the apartment numbers:
array_3B_apartments = np.delete(array_3B_filter,[0,2,3],1)
print("Apartments with more than 3 magnets in their kitchens", array_3B_apartments.tolist())
```

```
Apartments with more than 3 magnets in their kitchens [[5], [9], [13], [17], [21]]
```

Bonus: Let's define the values of a 3-dimensional array as the sum of the indices of each respective cell. For example: array_3B will be [0,1,2] --> 3.

In [ ]:

```
array_3B = np.empty([3,2,4])
for first_axis in np.arange(array_3B.shape[0]):
    for second_axis in np.arange(array_3B.shape[1]):
        for third_axis in np.arange(array_3B.shape[2]):
            array_3B [first_axis,second_axis,third_axis] = first_axis + second_axis + third_axis
print("Our array :", array_3B)
```

```
Our array : [[[0. 1. 2. 3.]
  [1. 2. 3. 4.]]

 [[1. 2. 3. 4.]
  [2. 3. 4. 5.]]
 [[2. 3. 4. 5.]
  [3. 4. 5. 6.]]]
```

1.Importing Libraries

In [ ]:

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

2.Importing and reading the Dataset

In [ ]:

```
df=pd.read_csv('../Polynomial regression/data/Position_Salaries.csv')
df
```

Out[ ]:

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000
5	Region Manager	6	150000
6	Partner	7	200000
7	Senior Partner	8	300000
8	C-level	9	500000
9	CEO	10	1000000

3.Dividing the dataset into 2 components

Divide dataset into two components that is X and y.X will contain the Column between 1 and 2. y will contain the 2 columns.

In [ ]:

```
X = df.iloc[:, 1:2].values
y = df.iloc[:, 2].values
```

4.Fitting Linear Regression to the dataset

Fitting the linear Regression model On two components.

In [ ]:

```
# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin = LinearRegression()

lin.fit(X, y)
```

Out[ ]:

```
LinearRegression()
```

5.Fitting Polynomial Regression to the dataset

Fitting the Polynomial Regression model on two components X and y.

In [ ]:

```
# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree = 4)
X_poly = poly.fit_transform(X)

poly.fit(X_poly, y)
lin2 = LinearRegression()
lin2.fit(X_poly, y)
```

Out[ ]:

```
LinearRegression()
```

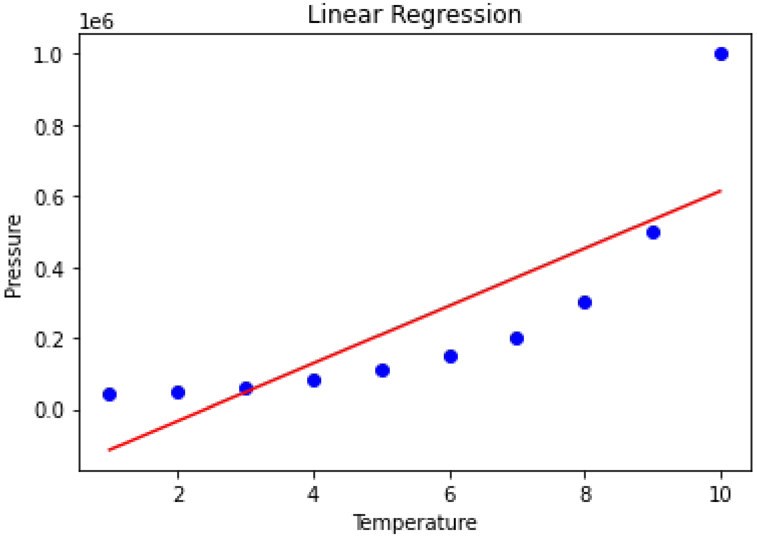
6.In this step, we are Visualising the Linear Regression results using a scatter plot.

In [ ]:

```
# Visualising the Linear Regression results
plt.scatter(X, y, color = 'blue')

plt.plot(X, lin.predict(X), color = 'red')
plt.title('Linear Regression')
plt.xlabel('Temperature')
plt.ylabel('Pressure')

plt.show()
```

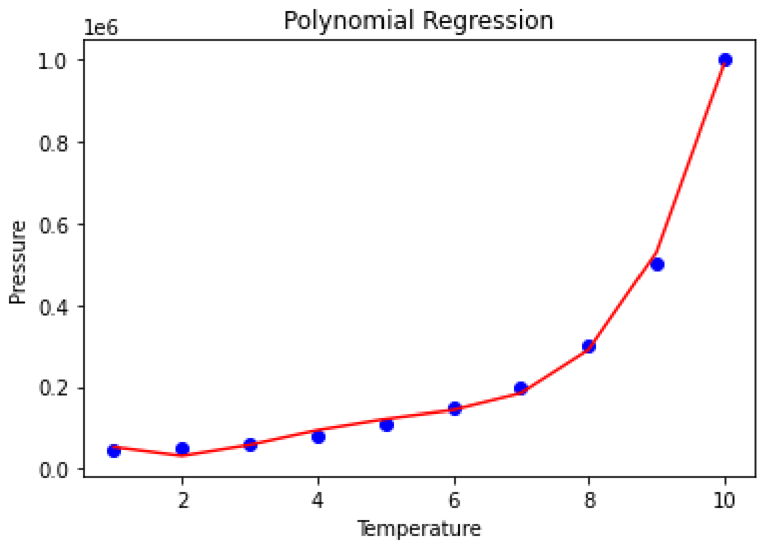


7.Visualising the Polynomial Regression results using a scatter plot.

In [ ]:

```
# Visualising the Polynomial Regression results
plt.scatter(X, y, color = 'blue')
plt.plot(X, lin2.predict(poly.fit_transform(X)), color = 'red')
plt.title('Polynomial Regression')
plt.xlabel('Temperature')
plt.ylabel('Pressure')

plt.show()
```



8.Predicting new results with both Linear and Polynomial Regression. Note that the input variable must be in a numpy 2D array.

In [ ]:

```
# Predicting a new result with Linear Regression after converting predict variable to 2D array
pred = 110.0
predarray = np.array([[pred]])
lin.predict(predarray)
```

Out[ ]:

```
array([8701333.33333333])
```

In [ ]:

```
# Predicting a new result with Polynomial Regression after converting predict variable to 2D array
pred2 = 110.0
```

```
pred2array = np.array([[pred2]])
lin2.predict(poly.fit_transform(pred2array))
```

```
Out[ ]: array([1.10869084e+11])
```