به نام خدا

گزارش تمرین اول درس هوش محاسباتی

علیرضا اسلامی خواه

99521064

پاییز 1402

سوال اول :

در اینجا خواسته شده بود که مقایسه عنصر به عنصری انجام دهیم. با توجه به متد های آماده کتابخانه numpy به جواب مورد نظر رسیدیم.

```python
import numpy as np
def element_wise_comparison(array1, array2):


    # ------------
    greater_result = np.greater(array1, array2)
    greater_equal_result = np.greater_equal(array1, array2)
    less_result = np.less(array1, array2)
    less_equal_result = np.less_equal(array1, array2)
    # ------------

    return greater_result, greater_equal_result, less_result, less_equal_result
```

[1] ✓ 0.5s

```
[80]
...    Greater than:
       [[False False]
        [ True  True]]

       Greater than or equal to:
       [[ True  True]
        [ True  True]]

       Less than:
       [[False False]
        [False False]]

       Less than or equal to:
       [[ True  True]
        [False False]]
```

در اینجا بولین های نشان داده شده بیانگر درستی یا غلط بودن عبارت متد مورد نظر میباشند.

سوال دوم :

در این سوال مقصود ضرب عنصر به عنصر و ماتریسی بود که با استفاده از توابع multiply و dot انجام دادیم و همچنین جواب در فایل Jupyter قابل مشاهده است.

```
Element-wise multiplication:
[[2 0]
 [3 8]]

Matrix multiplication:
[[ 4  4]
 [10  8]]
```

سوال سوم :

در این سوال ماتریس دوم را باید یکبار به صورت افقی یا همان سطری و یکبار به صورت عمودی یا همان ستونی در می اوردیم (که برای اینکار از reshape کمک گرفتیم) و سپس عملیات اضافه کردن را انجام میدادیم. خروجی این سوال در کد قابل مشاهده است.

```python
def broadcast_add(p, q, method="row-wise"):
    if method == "row-wise":
        result = p + q
    elif method == "column-wise":
        reshaped_q = q.reshape(-1, 1)
        result = p + reshaped_q
    else:
        raise ValueError("Invalid method. Use 'row-wise' or 'column-wise'.")

    return result
```

```
Row-wise addition:
[[11 22 33]
 [14 25 36]
 [17 28 39]]

Column-wise addition:
[[11 12 13]
 [24 25 26]
 [37 38 39]]
```

سوال چهارم :

در این سوال به جهت نرمال سازی ابتدا کوچکترین و سپس بزرگترین را انتخاب کردیم و با توجه به فرمولی که نوشتیم به جواب مربوطه رسیدیم.

```python
x = np.random.randint(1, 11, size=(4, 4))
print("Original Array:")
print(x)

# Do the normalization

min_value = x.min()
max_value = x.max()

# Normalize the matrix by scaling values between 0 and 1
normalized_x = (x - min_value) / (max_value - min_value)

print("After normalization:")
print(normalized_x)
```

```
Original Array:
[[ 4  9 10  6]
 [ 7  5  7  6]
 [ 4  7  3  8]
 [ 1  9  1  7]]
After normalization:
[[0.33333333 0.88888889 1.         0.55555556]
 [0.66666667 0.44444444 0.66666667 0.55555556]
 [0.33333333 0.66666667 0.22222222 0.77777778]
 [0.         0.88888889 0.         0.66666667]]
```

سوال پنجم :

بخش اول :

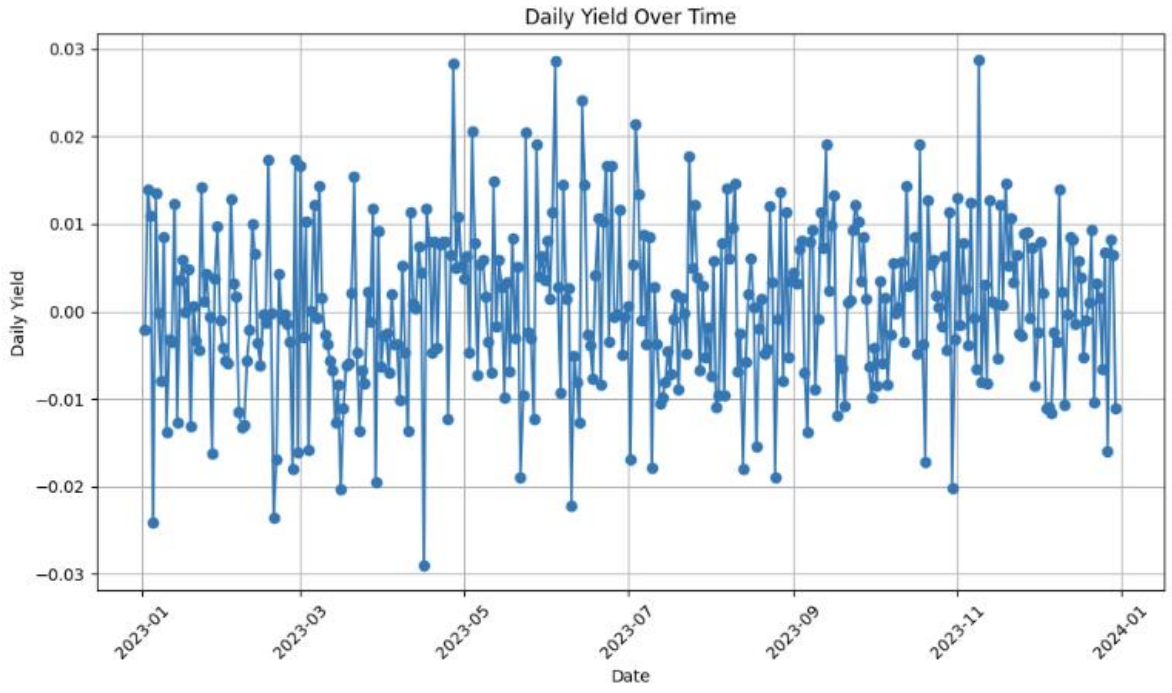
پس از خواندن فایل csv و تبدیل تاریخ به تاریخ واقعی یک ستون دیگر به متغیر ایجاد شده به نام Daily Yield اضافه میکنیم و مقادیر آن را طبق فرمول ذکر شده در صورت سوال قرار میدهیم.

```
data['Daily Yield'] = (data['Closing Price'] - data['Closing Price'].shift(1)) /
data['Closing Price'].shift(1)
```

سپس مقادیر را نشان میدهیم.

```
plt.figure(figsize=(10, 6))
plt.plot(data['Date'], data['Daily Yield'], marker='o', linestyle='-')
plt.title('Daily Yield Over Time')
plt.xlabel('Date')
plt.ylabel('Daily Yield')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

و خروجی دلخواه را میگیریم :



بخش دوم و سوم :

میانگین را با تابع mean و واریانس را با std میگیریم.

خروجی :

```
Average Daily Yield: 0.06%
Standard Deviation of Daily Yield: 0.0095
```

بخش چهارم :

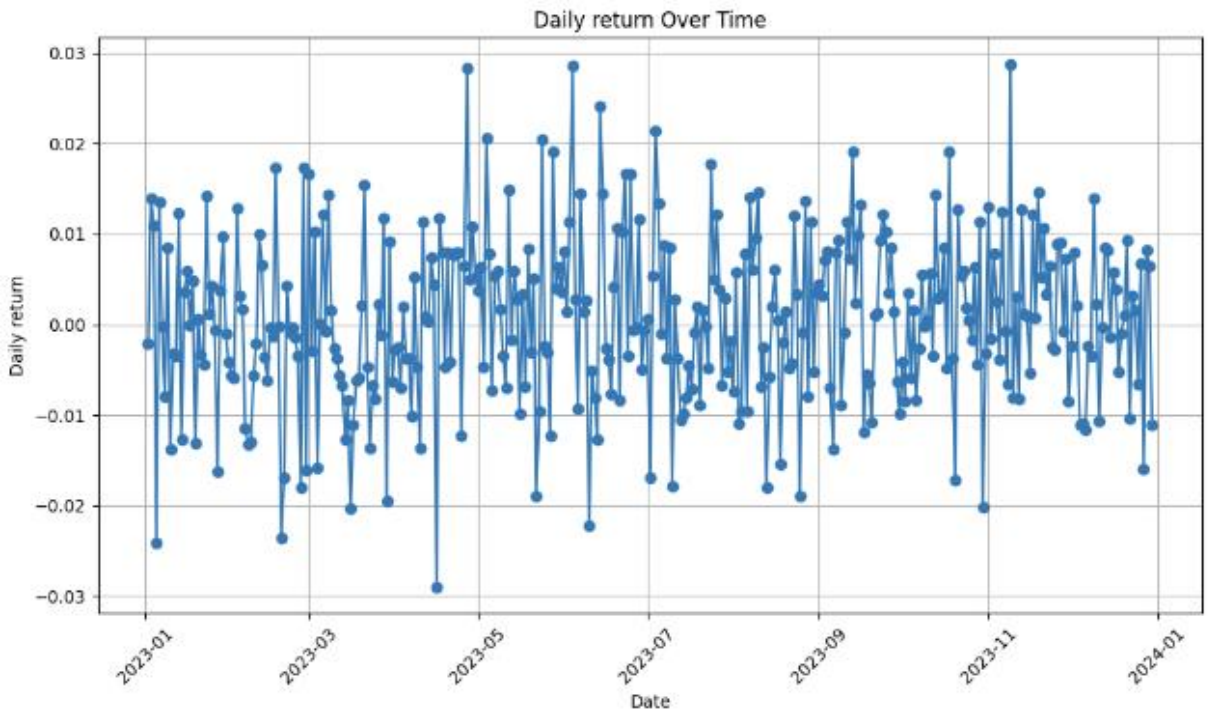
برای نشان دادن قیمت پایانی به مرور زمان احتیاج به مرتب سازی زمانی داشتیم.

```python
#Q4
data = data.sort_values(by='Date')
plt.figure(figsize=(10, 6))
plt.plot(data['Date'], data['Closing Price'], marker='o', linestyle='-')
plt.title('Daily Closing Stock Prices')
plt.xlabel('Date')
plt.ylabel('Closing Price')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

بخش پنجم :

```
plt.figure(figsize=(10, 6))
plt.plot(data['Date'], data['Daily Yield'], marker='o', linestyle='-')
plt.title('Daily return Over Time')
plt.xlabel('Date')
plt.ylabel('Daily return')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

خروجی :



بخش ششم و هفتم:

```
highest_yield_day = data[data['Daily Yield'] == data['Daily Yield'].max()]
lowest_yield_day = data[data['Daily Yield'] == data['Daily Yield'].min()]
print(f'Day with the Highest Yield:')
print(highest_yield_day)
print(f'\nDay with the Lowest Yield:')
print(lowest_yield_day)
```

```
highest_closing_price_date = data['Date'][data['Closing Price'].idxmax()]
highest_closing_price_value = data['Closing Price'].max()

lowest_closing_price_date = data['Date'][data['Closing Price'].idxmin()]
lowest_closing_price_value = data['Closing Price'].min()
print(f'Date of Highest Closing Price: {highest_closing_price_date}')
print(f'Value of Highest Closing Price: {highest_closing_price_value}')
print(f'Date of Lowest Closing Price: {lowest_closing_price_date}')
print(f'Value of Lowest Closing Price: {lowest_closing_price_value}')
```

```
Day with the Highest Yield:
          Date  Closing Price       Daily Yield
312 2023-11-09      116.41657  NaN      0.028786

Day with the Lowest Yield:
          Date  Closing Price       Daily Yield
105 2023-04-16       82.96821  NaN     -0.028964
Date of Highest Closing Price: 2023-11-29 00:00:00
Value of Highest Closing Price: 124.6180108
Date of Lowest Closing Price: 2023-04-16 00:00:00
Value of Lowest Closing Price: 82.96821012
```

سوال ششم :

برای این سوال عملیات feed forward را به سادگی و مانند صورت سوال انجام دادیم.

```
def for_loop_feed_forward(X, w):
    outputs = np.zeros((1000, 1))
    for i in range(1000):
        for k in range(500):
            outputs[i, 0] += X[i, k] * w[k, 0]
    return outputs
def vectorized_feed_forward(X, w):
    outputs = np.dot(X , w)
    return outputs
```

سوال هفتم :

مانند تصویر و متودهای ذکر شده به خروجی دلخواه رسیدیم.

```
[145]
def replace_elements_above_threshold(array, threshold):

    mask = array > threshold
    modified_arr = np.where(mask, 1, 0)
    return modified_arr
```

+ Code + Markdown

```
[146]
input_array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
threshold_value = 5
result_array = replace_elements_above_threshold(input_array, threshold_value)
print(result_array)
```

```
[[0 0 0]
 [0 0 1]
 [1 1 1]]
```

سوال هشتم :

در این تابع برای مساوی بودن دو ماتریس اینگونه کار کردیم :

```
def is_equal(self, second_matrix):

        flag = True
        # output = np.zeros((self.row_numbers, self.column_numbers))
        output = [[0 for i in range(self.column_numbers)] for j in
range(self.row_numbers)]
        for i in range(self.row_numbers):
            for j in range(self.column_numbers):
                if self.Matrix[i][j] == second_matrix.Matrix[i][j]:
                    output[i][j] = True
                else:
                    output[i][j] = False
                    flag = False
        print("The equality of two Matrices is : ", flag)
        return output
```

و به خروجی دلخواه رسیدیم :

```
matrix1 = Matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

matrix2 = Matrix([[0, 0, 0], [4, 5, 6], [7, 8, 9]])

print(matrix1.is_equal(matrix2))

# test equality of matrices here and show the result #
```

```
The equality of two Matrices is :  False
[[False, False, False], [True, True, True], [True, True, True]]
```

برای تست بزرگتر یا کوچکتر بودن :

```
def is_higher_elementwise(self, second_matrix):


    output = [[0 for i in range(self.column_numbers)] for j in range(self.row_numbers)]
    for i in range(self.row_numbers):
        for j in range(self.column_numbers):
            if self.Matrix[i][j] > second_matrix.Matrix[i][j]:
                output[i][j] = True
            else:
                output[i][j] = False
    return output
```

خروجی :

```
    matrix3 = Matrix([[0, 0, 0], [10, 20, 30], [-1, 8, 10]])
    # test proportion of matrices here and show the result #
    ihe = matrix1.is_higher_elementwise(matrix3)
    print(ihe)
```

```
[[True, True, True], [False, False, False], [True, False, False]]
```

برای تست کردن زیرمجموعگی :

```
def is_subset(self, second_matrix):
        rows2, cols2 = len(second_matrix.Matrix), len(second_matrix.Matrix[0])
        rows1 = self.row_numbers
        cols1 = self.column_numbers
        if rows2 > rows1 or cols2 > cols1:
            return False
        for i in range(rows1 - rows2 + 1):
            for j in range(cols1 - cols2 + 1):
                if all(self.Matrix[i + x][j + y] == second_matrix.Matrix[x][y]
for x in range(rows2) for y in range(cols2)):
                    return True
                return False
```

خروجی :

```
matrix4 = Matrix([[5, 6], [8, 9]])
matrix5 = Matrix([[1, 2], [4, 5]])
matrix6 = Matrix([[1, 2], [3, 4]])
# test subset of matrices here and show the result
flag = matrix1.is_subset(matrix6)
print("The subsetion of Matrices is ", flag)
```

```
The subsetion of Matrices is  False
```

برای شبیه سازی dot :

```
def dot_product(self, second_matrix):
        result = []
        for i in range(self.row_numbers):
            row = []
            for j in range(second_matrix.column_numbers):
                dot_product = 0
                for k in range(self.column_numbers):
                    dot_product += self.Matrix[i][k] * second_matrix.Matrix[k][j]
                row.append(dot_product)
            result.append(row)
```

```
        return Matrix(result)
```

خروجی :

```
matrix7 = Matrix([[3, 1], [2, 4], [-1, 5]])
matrix8 = Matrix([[3, 1], [2, 4]])

# test product of matrices here and show the result #
result_matrix = matrix7.dot_product(matrix8)
for row in result_matrix.Matrix:
    print(row)
```

```
[11, 7]
[14, 18]
[7, 19]
```

منابع مورد استفاده در تمرین : chat.openai