به نام خدا

تمرین هوش محاسباتی اول

استاد : دکتر مزینی

دستیاران آموزشی :سینا اسکندری محمد میرزایی

سارا سادات یونسی ۹۸۵۳۳۰۵۳

فهرست

سوال ۱ ..... صفحه ۳

سوال ۲ ..... صفحه۴

سوال ۳ ..... صفحه ۶

سوال ۴ ..... صفحه۷

سوال ۵ ..... صفحه ۸

سوال ۶ ..... صفحه۱۴

سوال ۷ ..... صفحه ۱۶

سوال ۸ ..... صفحه۱۷

منابع صفحه۲۳

سوال ۱

تابعی بنویسید که یک مقایسه عنصر به عنصر)بزرگتر، بزرگتر مساوی، کوچکتر و کوچکتر مساوی(از دو آرایه داده شده)هر دو آرایه با هر اندازه ی مساوی(انجام دهد.)از کتابخانه ی NumPy برای حل این مسئله استفاده شود.

پاسخ سوال ۱)

Q1

```python
def element_wise_comparison(array1, array2):
    """
    Perform element-wise comparisons between two NumPy arrays.

    Parameters:
    - array1 (numpy.ndarray): First input NumPy array.
    - array2 (numpy.ndarray): Second input NumPy array.

    Returns:
    - tuple: A tuple of NumPy arrays containing the following element-wise comparison results:
        - greater_result (numpy.ndarray): Element-wise greater than comparison.
        - greater_equal_result (numpy.ndarray): Element-wise greater than or equal to comparison.
        - less_result (numpy.ndarray): Element-wise less than comparison.
        - less_equal_result (numpy.ndarray): Element-wise less than or equal to comparison.
    """

    # they return boolean true or false depend on condition    less_result = np.less(array1, array2)
    less_equal_result = np.less_equal(array1, array2)
    greater_result = np.greater(array1, array2)
    greater_equal_result = np.greater_equal(array1, array2)
    less_result = np.less(array1, array2)


    return greater_result, greater_equal_result, less_result, less_equal_result
```

✓ 0.0s

با استفاده از توابع numpy که در کتابخانه ان وجود دارد پیاده سازی را انجام دادم و اگر آن ایندکس شرط های مورد نظر را داشت true و در غیر این صورت false بر میگرداند.

نتیجه :

برای مثال برای مورد اول دو عدد ردیف ما یکسان اند و بزرگتر بودن فالس می شود ولی شرط دوم بزرگتر مساوی بودن را نقض نمیکند. همچنین کوچکتر از هم نیستند ولی کوچکتر مساوی بودن را نقض نمی کنند و درست می باشند.

```
array1 = np.array([[1, 2], [3, 4]])
array2 = np.array([[1, 2], [2, 3]])

greater, greater_equal, less, less_equal = element_wise_comparisons(array1, array2)

print("Greater than:")
print(greater)
print("\nGreater than or equal to:")
print(greater_equal)
print("\nLess than:")
print(less)
print("\nLess than or equal to:")
print(less_equal)
```

✓ 0.0s

```
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]]
```

سوال ۲

تابعی طراحی کنید که دو آرایه و یک پارامتر method مشخص شده را دریافت کرده و بر اساس method مشخص شده، عملیات ضرب عنصر به عنصر یا ضرب ماتریسی را انجام دهد. (از کتابخانهی NumPy برای حل این مسئله استفاده شود)

پاسخ سوال ۲)

Q2

```
def array_multiply(array1, array2, method="element-wise"):
    """
    Perform multiplication between two NumPy arrays using the specified method.

    Parameters:
    - array1 (numpy.ndarray): First input NumPy array.
    - array2 (numpy.ndarray): Second input NumPy array.
    - method (str, optional): The multiplication method to use. Defaults to "element-wise".

    Returns:
    - numpy.ndarray: The result of the multiplication operation based on the chosen method.
    """
    result = np.matmul(array1, array2) if method == 'matrix-multiply' else np.multiply(array1, array2)

    return result
```

در این سوال اگر متد ضرب ماتریسی انتخاب شود به صورت ماتریسی ضورب مشود عدد اول ردیف ماتریس اول در عدد ستون اول ماتریس دوم ضرب+ عدد دوم ردیف ماتریس اول در عدد ستون دوم ماتریس دوم ضرب و اولین درایه ی ماتریس نهایی را می دهد و همین کار را ادامه می دهیم تا تمام ماتریس ساخته شود اما در ضرب نظیر به نظیر هر درایه در درایه نظیر خود ضرب می شود. و در نهایت در result نتیجه برگردانده می شود.

یک پارامتر به نام method که مشخص میکند که عملیات ماتریسی چه نوعی باشد. اگر method برابر با "matrix-multiply" باشد، حاصلضرب ماتریسی دو آرایه انجام میشود. اگر method برابر با هر چیز دیگری باشد، ضرب عنصری دو آرایه انجام میشود.

- تابع np.matmul که حاصلضرب ماتریسی دو آرایه را محاسبه میکند. این تابع از کتابخانه numpy استفاده میکند که یک کتابخانه محبوب و قدرتمند برای کار با آرایهها و محاسبات علمی در پایتون است .این تابع برای آرایه های دو بعدی معادل حاصلضرب ماتریسی معمولی است، اما برای آرایههای با بعد بالاتر، آنها را به عنوان پشته ای از ماتریس ها در نظر میگیرد و با قوانین خاصی آنها را ضرب میکند
- تابع np.multiply که ضرب عنصری دو آرایه را محاسبه میکند. این تابع نیز از کتابخانه numpy استفاده میکند و هر عنصر از یک آرایه را در عنصر متناظر از آرایه دیگر ضرب میکنداین تابع برای آرایه های با ابعاد مختلف نیز کار میکند، اما باید قوانین broadcasting را رعایت کنند.
- یک عبارت شرطی if-else که بر اساس مقدار method عملیات مورد نظر را انتخاب میکند. اگر method برابر با "matrix-multiply" باشد، تابع np.matmul فراخوانی میشود. در غیر این صورت، تابع np.multiply فراخوانی میشود.
- یک دستور return که آرایه result را بر میگرداند.

نتیجه :

در المنت وایز ۱*۲=۲و ۲*۰=۰ و ۳ *۱=۳ و ۴*۲ =۸ و ضرب ماتریسی که طبق الگو ضرب جمع شده.

```
    array1 = np.array([[1, 2], [3, 4]])
    array2 = np.array([[2, 0], [1, 2]])

    # Perform element-wise multiplication
    element_wise_result = array_multiply(array1, array2, method="element-wise")
    print("Element-wise multiplication:")
    print(element_wise_result)

    # Perform matrix multiplication
    matrix_multiply_result = array_multip Loading... , array2, method="matrix-multiply")
    print("\nMatrix multiplication:")
    print(matrix_multiply_result)
[60] ✓ 0.0s

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

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

سوال ۳

تابعی بنویسید که دو آرایه ورودی با ابعاد n×n و یک بردار با ابعاد ۱ ×n و همچنین یک ورودی method بگیرد و بر اساس methodمشخ ص شده بردار دوم را به صورت افقی یا عمودی به بردار اول اضافه کند.

پاسخ سوال ۳)

Q3

```python
def broadcast_add(p, q, method="row-wise"):
    """
    Perform addition between two NumPy arrays using broadcasting and the specified method.

    Parameters:
    - p (numpy.ndarray): First input NumPy array.
    - q (numpy.ndarray): Second input NumPy array.
    - method (str, optional): The addition method to use. Defaults to "row-wise".
        - "row-wise": Perform row-wise addition, broadcasting q to match the number of rows in p.
        - "column-wise": Perform column-wise addition, adding q to each column of p.

    Returns:
    - numpy.ndarray: The result of the addition operation based on the chosen method.

    Raises:
    - ValueError: If an invalid method is provided or if the shapes are incompatible for the chosen method.
    """
    if method == 'row-wise':
            q = q.reshape(1, p.shape[1])
            result = p+q
    if method == 'column-wise':
            q = q.reshape(p.shape[0], 1)
            result = p+q

    return result
```

0.0s

طبق این تابع شرطی یا باید درایه ها کسری جمع شوند یا به صورت ستونی جمع شوند اگر سطری باشد باید ماتریس به حالت سطری دراوریم تا با هر ردیف قابل جمع شدن باشد و اگر ستونی هم باشد این مراحل تکرار می شود.

• یک پارامتر به نام method که مشخص میکند که p و q باید به صورت سطری یا ستونی جمع شوند. اگر method برابر با "row-wise" باشد، p و q به صورت سطری جمع میشوند. اگر method برابر با "column-wise" باشد، p و q به صورت ستونی جمع میشوند.

• دو شرط if که بر اساس مقدار method عملیات مورد نظر را انجام میدهند. در این شرطها، از تابع reshape کتابخانه numpy استفاده میشود که شکل آرایه q را تغییر میدهد. این کار باعث میشود که p و q قابل جمع شدن باشند. برای مثال، اگر p یک آرایه با شکل (۳،۴) و q یک آرایه با شکل (،۴) باشند، q.reshape(1, p.shape[1]) یک آرایه با شکل (۱،۴) و q.reshape(p.shape[0], 1) یک آرایه با شکل (۳،۱) خواهند بود

• تابع جمع که دو آرایه را به صورت عنصری جمع میکند. این تابع از قاعده broadcasting استفاده میکند که بیان میکند که چگونه دو آرایه با ابعاد مختلف را میتوان با هم جمع برای مثال، اگر p یک آرایه با شکل (۳،۴) و q یک آرایه با شکل (۱،۴) باشند، p+q یک آرایه با شکل (۳،۴) خواهد بود که هر سطر p با q جمع شده است. اگر q یک آرایه با شکل (۳،۱) باشد، p+q یک آرایه با شکل (۳،۴) خواهد بود که هر ستون p با q جمع شده است.

- یک دستور return که آرایه result را بر میگرداند.

۱+۱۰و۱۰+۲ و ۱۰+۳ = ۱۱و۱۲ و۱۳ و به همین ترتیب ادامه دارد.

```python
# Example usage with different-shaped arrays
p = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
q = np.array([10, 20, 30])

# Add q row-wise to p
row_wise_result = broadcast_add(p, q, method="row-wise")
print("Row-wise addition:")
print(row_wise_result)

# Add q column-wise to p
column_wise_result = broadcast_add(p, q, method="column-wise")
print("\nColumn-wise addition:")
print(column_wise_result)
```

✓ 0.0s

```
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 ۴

یک ماتریس ۴×۴ به شکل رندوم با مقادیر بین ۱ تا ۱۰ تشکیل دهید ، سپس مقادیر آن را نرمال سازی کنید (. پس از نرمال سازی، مقادیر حداقل ۰ و حداکثر ۱ میشوند)

پاسخ سوال ۴)

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

فرمول بالا یک فرمول نرمالسازی می باشد که ما هم پس از ساخت یک ماتریس رندوم 4*4 که اعداد ان دربازه ی ۱ تا ۱۱ قرار گرفته اند فرمول را پیاده سازی می کنیم و همانگونه که در نتیجه می بینیم همه۸ه ی اعداد ماتریس نهایی بین ۰ و ۱ قرار می گیرند.مقادیر ماکس و مین را هم با استفاده از توابع اماده پیدا می کنیم.

Q4

```
# Initialize the random matrix
# x =
x=np.random.randint(1,11,(4,4))

print("Original Array:")
print(x)

# Do the normalization
s=(x-np.min(x))
v=(np.max(x)-np.min(x))
x=s/v


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

✓ 0.0s

```
Original Array:
[[ 4  6  2 10]
 [ 1  9  1  4]
 [ 7 10  6  2]
 [ 3  8  1  7]]
After normalization:
[[0.33333333 0.55555556 0.11111111 1.        ]
 [0.         0.88888889 0.         0.33333333]
 [0.66666667 1.         0.55555556 0.11111111]
 [0.22222222 0.77777778 0.         0.66666667]]
```

سوال ۵

ابتدا داده های موجود در فایل csv.data را بخوانید و سپس- :

میزان بازده روزانه را محاسبه کنید: قیمت پایانی روز قبلی از قیمت پایانی روز جاری کم کرده و بر قیمت پایانی روز قبلی تقسیم کنید

-میانگین بازده روزانه را محاسبه کنید.

- انحراف معیار بازده روزانه را محاسبه کنید.

- قیمت های پایانی روزانه سهام را به مرور زمان نمایش دهید.

- میزان بازده روزانه را به مرور زمان نمایش دهید.

- روزهای با بیشترین و کمترین بازده را شناسایی کنید.

- تاریخ و مقدار بیشترین و کمترین قیمت های تاریخی سهام را پیدا کنید(.از دو کتابخانه ی NumPy و matplotlib استفاده کنید)

پاسخ سوال ۵)

ابتدا دیتاست خود را با دستور رید می خوانیم و ان را در دیتا ست ذخیر می کنیم.سپس با شیفت دادن یک سطر از دیتاست خود به دیتای روز قبل دست پیدا می کنیم .

و طبق فرمول گفته شده پیاده سازی را انجام می دهیم و با استفاده از توابع اماده mean و std میانگین و انحراف معیار را بدست می آوریم.

- ابتدا کتابخانههای matplotlib و pandas را با نامهای مستعار plt و pd وارد میکند.
- سپس دادهست را از فایل csv با استفاده از تابع pd.read_csv خوانده و در یک شیء DataFrame به نام data_set ذخیره میکند.
- بعد از آن، با استفاده از تابع shift از کتابخانه pandas، قیمت پایانی روز قبل را در یک ستون جدید به نام last_day_return محاسبه میکند. این تابع مقادیر یک ستون را به تعداد دلخواه به جلو یا عقب منتقل میکند. در اینجا، مقدار ۱ به عنوان پارامتر periods به تابع داده شده است، که به این معنی است که مقادیر یک خانه به جلو منتقل شوند. بنابراین، قیمت پایانی روز قبل در همان خانهای که قیمت پایانی روز جاری است، قرار میگیرد.
- سپس قیمت پایانی روز جاری را در یک ستون جدید به نام today_return ذخیره میکند. این ستون همان ستون Closing Price از دادهست اصلی است.
- سپس با استفاده از فرمول (last_day_return - today_return) / last_day_return، بازده روزانه را در یک ستون جدید به نام total_return محاسبه میکند. این فرمول نشان میدهد که قیمت پایانی چقدر درصد تغییر کرده است نسبت به روز قبل.
- سپس با استفاده از تابع print، پاسخ سوال اول را چاپ میکند. این پاسخ شامل مقادیر بازده روزانه برای هر روز از دادهست است.
- سپس با استفاده از تابع mean از کتابخانه pandas، میانگین بازده روزانه را در یک متغیر به نام mean_total_returns ذخیره میکند. این تابع میانگین مقادیر یک ستون را محاسبه میکند. در اینجا، محور ۰ به عنوان پارامتر axis به تابع داده شده است، که به این معنی است که محاسبه بر روی ستونها انجام شود.
- سپس با استفاده از تابع print و f-string، پاسخ سوال دوم را چاپ میکند. این پاسخ شامل مقدار میانگین بازده روزانه است.
- سپس با استفاده از تابع std از کتابخانه pandas، انحراف معیار بازده روزانه را در یک متغیر به نام std_total_returns ذخیره میکند. این تابع انحراف معیار مقادیر یک ستون را محاسبه میکند. انحراف معیار یک معیار از پراکندگی یا تغییرپذیری

دادهها است. در اینجا، مقدار ۱ به عنوان پارامتر ddof به تابع داده شده است، که به این معنی است که محاسبه با تقسیم بر N-1 انجام شود، که N تعداد مقادیر است. این روش استاندارد است و به نرمالسازی با درجه آزادی N-1 معروف است.

- سپس با استفاده از تابع print و f-string، پاسخ سوال سوم را چاپ میکند. این پاسخ شامل مقدار انحراف معیار بازده روزانه است.

Q5

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

data_set = pd.read_csv('data.csv')
last_day_return=data_set ['Closing Price'].shift(1)
today_return=data_set['Closing Price']
total_return = (last_day_return - today_return) / last_day_return
print("Answer 1 : ")
print(total_return)
print("_____")


mean_total_returns = total_return.mean()
print(f'Answer 2 ={mean_returns} ')
print("_____")



std_total_returns=total_return.std()
print(f'Answer 3 ={std_total_returns} ')
print("_____")
```

✓ 0.0s

```
Answer 1 :
0           NaN
1      0.002145
2     -0.013911
3     -0.010884
4      0.024109
         ...
359   -0.006720
360    0.015886
361   -0.008170
362   -0.006454
363    0.011010
Name: Closing Price, Length: 364, dtype: float64
_____
Answer 2 =-0.0005548260008486606
_____
Answer 3 =0.009455978850317192
_____
```

```
print("_____")


mean_total_returns = total_return.mean()
print(f'Answer 2 ={mean_returns} ')
print("_____")



std_total_returns=total_return.std()
print(f'Answer 3 ={std_total_returns} ')
print("_____")
```

[110] ✓ 0.0s

```
_____
Answer 2 =-0.0005548260008486606
_____
Answer 3 =0.009455978850317192
_____
```

با کمک کتابخانه متپلات به رسم نمودار ها می پردازیم و با توجه به نمودار ها ان را رسم می کنیم .

• سپس با استفاده از تابع plot از کتابخانه matplotlib، نمودار خطی قیمت پایانی را رسم میکند. این تابع مقادیر ستون Closing Price از دادهست را به عنوان y میگیرد و مقادیر پیشفرض ۰, ۱, ۲, ... به عنوان x میگیرد. نمودار یک خط را از نقطه به نقطه میکشد.

• سپس با استفاده از تابع xlabel و ylabel از کتابخانه matplotlib، برچسبهای محور x و y را به ترتیب "TIME" و "PRICE CLOSING" میگذارد. این توابع عنوان محورها را تعیین میکنند.

• سپس با استفاده از تابع show از کتابخانه matplotlib، نمودار را نمایش میدهد. این تابع نمودار را در یک پنجره جدید باز میکند.

• سپس با استفاده از تابع plot از کتابخانه matplotlib، نمودار خطی بازده روزانه را رسم میکند. این تابع مقادیر ستون total_return را به عنوان y میگیرد و مقادیر پیشفرض ۰, ۱, ۲, ... به عنوان x میگیرد. نمودار یک خط را از نقطه به نقطه میکشد.

• سپس با استفاده از تابع xlabel و ylabel از کتابخانه matplotlib، برچسبهای محور x و y را به ترتیب "TIME" و "RETURN DALY" میگذارد. این توابع عنوان محورها را تعیین میکنند.

• سپس با استفاده از تابع show از کتابخانه matplotlib، نمودار را نمایش میدهد. این تابع نمودار را در یک پنجره جدید باز میکند.

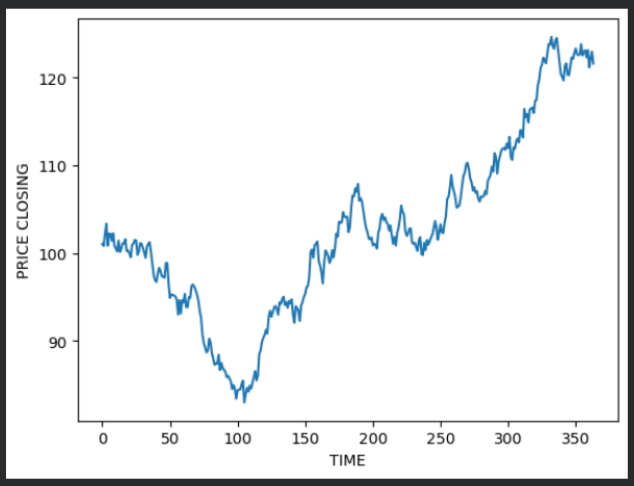
```
print ("Answer 4 :")

plt.plot(data_set['Closing Price'])
plt.xlabel('TIME')
plt.ylabel('PRICE CLOSING')
plt.show()


print("_____")
print ("Answer 5 :")
plt.plot(total_return)
plt.xlabel('TIME')
plt.ylabel('RETURN DALY')
plt.show()
```

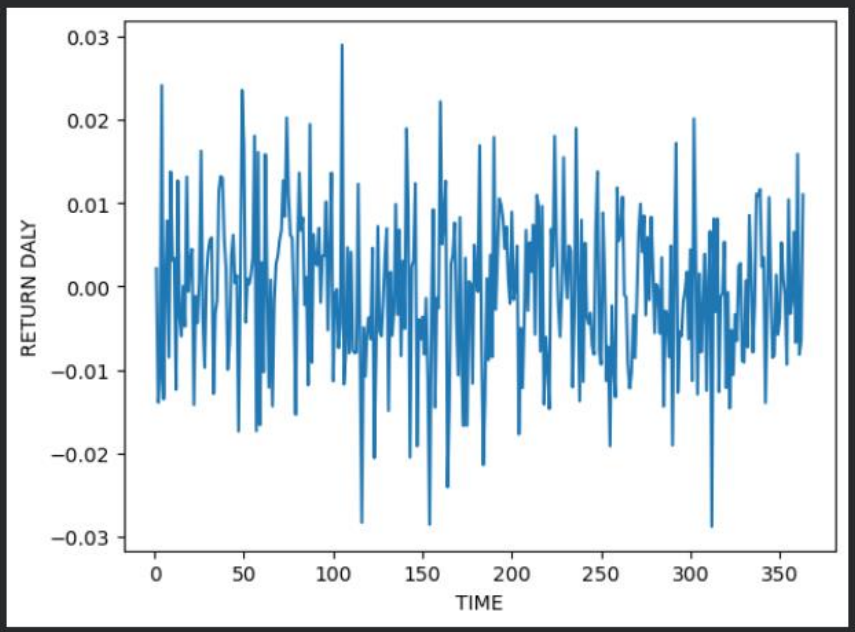
[79] ✓ 0.4s

```
Answer 4 :
```




```
_____
Answer 5 :
```

Answer 5 :



این کد چند سوال را دربارهی بازده و قیمت پایانی یک دادهست از قیمت سهام در یک بازار مالی پاسخ میدهد. دادهست شامل تاریخ، قیمت پایانی، قیمت بازگشایی، حجم معاملات و تغییرات درصدی است. کد به این صورت عمل میکند:

• سپس با استفاده از تابع print، عبارت "Answer 6 :" را چاپ میکند. این عبارت نشان میدهد که این بخش پاسخ سوال ششم است.

• سپس با استفاده از تابع loc از کتابخانه pandas، بازده روزانه را در یک ستون جدید به نام returns محاسبه میکند. این تابع به ما اجازه میدهد که یک گروه از سطرها و ستونها را با برچسب یا یک آرایه بولی دسترسی پیدا کنیم. در اینجا، ما بازده روزانه را با فرمول (today_return - last_day_return) / last_day_return محاسبه میکنیم. این فرمول نشان میدهد که قیمت پایانی چقدر درصد تغییر کرده است نسبت به روز قبل. ما همچنین مقادیر NaN را با تابع fillna از کتابخانه pandas با صفر جایگزین میکنیم. این تابع مقادیر NaN را با یک مقدار دلخواه پر میکند.

• سپس با استفاده از تابع idxmax و idxmin از کتابخانه pandas، شاخص روزی که بازده بیشترین و کمترین را داشته است را در دو متغیر به نام max_return_day و min_return_day ذخیره میکنیم. این توابع شاخص اولین بروزرسانی بیشترین و کمترین را در یک محور درخواستی برمیگردانند. در اینجا، ما محور ۰ را به عنوان پارامتر axis به توابع میدهیم، که به این معنی است که محاسبه بر روی سطرها انجام شود.

• سپس با استفاده از تابع print و f-string، تاریخ و مقدار بازده بیشترین و کمترین را چاپ میکنیم. ما با استفاده از تابع loc از کتابخانه pandas، مقدار بازده مربوط به شاخص max_return_day و min_return_day را از ستون returns استخراج میکنیم. ما همچنین با استفاده از تابع loc از کتابخانه pandas، مقدار تاریخ مربوط به شاخص max_return_day و min_return_day را از ستون Date استخراج میکنیم. ما این مقادیر را در قالب f-string قرار میدهیم و با تابع print چاپ میکنیم..

• سپس با استفاده از تابع print، عبارت "Answer 7 :" را چاپ میکنیم. این عبارت نشان میدهد که این بخش پاسخ سوال هفتم است.

• سپس با استفاده از تابع idxmax و idxmin از کتابخانه pandas، شاخص روزی که قیمت پایانی بیشترین و کمترین را داشته است را در دو متغیر به نام Pr_Max و Pr_Min ذخیره میکنیم. این توابع شاخص اولین بروزرسانی بیشترین و کمترین را در یک محور درخواستی برمیگردانند. در اینجا، ما محور ۰ را به عنوان پارامتر axis به توابع میدهیم، که به این معنی است که محاسبه بر روی سطرها انجام شود.

• سپس با استفاده از تابع loc از کتابخانه pandas، مقدار قیمت پایانی مربوط به شاخص Pr_Max و Pr_Min را در دو متغیر به نام Val_Max و Val_Min ذخیره میکنیم. این تابع به ما اجازه میدهد که یک گروه از سطرها و ستونها را با برچسب یا یک آرایه بولی دسترسی پیدا کنیم. در اینجا، ما مقدار قیمت پایانی را از ستون Closing Price استخراج میکنیم.

• سپس با استفاده از تابع print و f-string، تاریخ و مقدار قیمت پایانی بیشترین و کمترین را چاپ میکنیم. ما با استفاده از تابع loc از کتابخانه pandas، مقدار تاریخ مربوط به شاخص Pr_Max و Pr_Min را از ستون Date استخراج میکنیم. ما این مقادیر را در قالب f-string قرار میدهیم و با تابع print چاپ میکنیم.

به این ترتیب، کد ما پاسخهای مورد نظر را بر اساس دادهست ارائه میدهد.

```
print("_____________________________________________________________________________________________")
print ("Answer 6 :")

print(f"Day Max Return: {data_set['Date'][max_return_day]}")
print(f"Max Return Val: {returns.loc[returns.idxmax()]}")
print(f"Day Min Return : {data_set['Date'][min_return_day]}")
print(f"Min Return Va:{returns.loc[returns.idxmin()]}")



print("_____________________________________________________________________________________________")
print ("Answer 7 :")


Pr_Max = data.loc[data_set['Closing Price'].idxmax(), 'Date']
Val_Max = data.loc[data_set['Closing Price'].idxmax(), 'Closing Price']

Pr_Min = data.loc[data_set['Closing Price'].idxmin(), 'Date']
Val_Min = data.loc[data_set['Closing Price'].idxmin(), 'Closing Price']

print(f"Day Max Pr:{Pr_Max}")
print(f"Max Pr Val: {Val_Max}")
print(f"Day Min Pr: {Pr_Min}")
print(f"Min Pr Val: {Val_Min}")
```

[108] ✓ 0.0s

```
_____________________________________________________________________________________________
Answer 6 :
Day Max Return: 4/16/2023
Max Return Val: 0.028963574613605738
Day Min Return : 11/9/2023
Min Return Va:-0.02878633838810639
_____________________________________________________________________________________________
Answer 7 :
Day Max Pr:11/29/2023
Max Pr Val: 124.6180108
Day Min Pr: 4/16/2023
Min Pr Val: 82.96821012
```

سوال ۶

در این سوال با مرحله forward feed در شبکه های عصبی آشنا می شوید . در این مرحله نمونه بردار های ورودی Xکه n ویژگی دارد در بردار وزن W به صورت زیر ضرب می شوند

$f = \sum xiwi\ n\ i=1$.گر هر ویژگی در بردار ورودی و نمایان که وزن متناظر هر ویژگی است .به طور مثال یک ورودی [۳ ,۲ ,۱] که بردار وزن [۱ -۱, ,۱] دارد، خروجی forward feed برابر است با:

$$(1 \times 1) + (-1 \times 2) + (1 \times 3)$$

فرض کنید در یک شبکه عصبی ۱۰۰۰ نمونه ورودی داریم و به ازای هر نمونه یکبار باید این عملیات محاسبه شود و ۱۰۰۰ خروجی بدست بیاید .

برای پیاده سازی این محاسبات دو روش وجود دارد. در روش اول با استفاده از حلقه for برای هر نمونه این مقادیر ضرب و جمع می شوند . روش دوم که به روش vectorization معروف است؛ در این روش تمامی نمونه ها در کنار هم در یک ماتریس بزرگتر

قرار می گیرند و این ماتریس بزرگتر در بردار وزن ضرب داخلی می شود. به طور مثال برای ۳ نمونه A و B و C بردار وزن W به صورت زیر می باشد

$$[\ a1\ b1\ c1\ a2\ b2\ c2\ a3\ b3\ c3\ ].[\ w1\ w2\ w3\ ] = [\ f1\ f2\ f3\ ]$$

برای این سوال، در نظر داریم که برای ۱۰۰۰ نمونه که هر نمونه ۵۰۰ ویژگی دارد عملیات forward feed را انجام دهیم. ابتدا یک ماتریس رندوم با ابعاد ۵۰۰×۱۰۰۰ و یک بردار وزن با ابعاد ۱× ۵۰۰ تولید کنید . با استفاده از حلقه های for و سپس با استفاده از روش vectorization محاسبات را انجام دهید. در آخر این دو روش را از نظر سرعت و مدت زمان اجرا مقایسه کنید

پاسخ سوال ۶)

در قسمت حلقه همانگونه که سوال گفته با حلقه زدن روی نمونه ها و ضرب و جمع کردن ماتریس وزن و فیچر جواب را بدست می اوریم و در قسمت وکتور هم با پیاده سازی تابع آماده dot این کار را انجام میدهیم.

دو تابع برای انجام یک عمل feed-forward در یک شبکه عصبی ساده با یک لایه وزنی را تعریف میکند. عمل feed-forward به این معنی است که دادههای ورودی را با وزنهای شبکه ضرب کرده و خروجی را محاسبه کنیم. این کد دو روش مختلف برای انجام این عمل را نشان میدهد: یکی با استفاده از حلقه for و دیگری با استفاده از بردارسازی.

- تابع for_loop_feed_forward با استفاده از دو حلقه for بر روی سطرها و ستونهای ماتریس ورودی X حرکت میکند و هر سطر را با ماتریس وزن w ضرب میکند و نتیجه را در یک ماتریس خروجی outputs ذخیره میکند. این تابع دو پارامتر میگیرد: X که ماتریس دادههای ورودی با شکل (num_samples, num_features) است و w که ماتریس وزن با شکل (num_features, 1) است. این تابع یک مقدار برمیگرداند: outputs که ماتریس خروجی با شکل (num_samples, 1) است.

- تابع vectorized_feed_forward با استفاده از تابع dot از کتابخانه numpy، ماتریس ورودی X را با ماتریس وزن w ضرب میکند و نتیجه را در یک ماتریس خروجی outputs ذخیره میکند. این تابع دو پارامتر میگیرد: X که ماتریس دادههای ورودی با شکل (num_samples, num_features) است و w که ماتریس وزن با شکل (num_features, 1) است. این تابع یک مقدار برمیگرداند: outputs که ماتریس خروجی با شکل (num_samples, 1) است.

روش بردارسازی سریعتر و بهینهتر از روش حلقه for است، زیرا از عملیات ماتریسی بهره میبرد و نیازی به تکرار بر روی هر عنصر ندارد. و محاسبات را به صورت موازی انجام می دهد و در وقت صرفه جویی می شود.

روش بردارسازی سریعتر از روش for است، زیرا از عملیات موازی بر روی چندین عنصر به جای اجرای تکتک عناصر استفاده میکند. این روش باعث میشود که پردازنده بتواند بهرهوری و سرعت خود را افزایش دهد. برای مثال، اگر بخواهیم یک بردار را در یک عدد ضرب کنیم، روش بردارسازی میتواند چندین عنصر از بردار را همزمان در عدد ضرب کند، در حالی که روش for باید برای هر عنصر یک عمل ضرب انجام دهد. بنابراین، روش بردارسازی نیاز به کمترین دستورالعمل دارد و زمان اجرای کمتری نیاز دارد.

Q6

```python
def for_loop_feed_forward(X, w):
    """
    Perform a feed-forward operation using a for loop.

    Parameters:
    - X (numpy.ndarray): Input data matrix of shape (num_samples, num_features).
    - w (numpy.ndarray): Weight matrix of shape (num_features, 1).

    Returns:
    - numpy.ndarray: Output matrix of shape (num_samples, 1).
    """

    outputs = np.zeros((X.shape[0], 1))
    for i in range(0, X.shape[0]):
        s = 0
        for j in range(0, X.shape[1]):
            outputs[i] += X[i][j] * w[j]

    return outputs

def vectorized_feed_forward(X, w):
    """
    Perform a feed-forward operation using vectorization.

    Parameters:
    - X (numpy.ndarray): Input data matrix of shape (num_samples, num_features).
    - w (numpy.ndarray): Weight matrix of shape (num_features, 1).

    Returns:
    - numpy.ndarray: Output matrix of shape (num_samples, 1).
    """
    outputs = np.dot(X, w)

    return outputs
```

2] ✓ 0.0s

```python
import time

# generate random samples

X = np.random.rand(1000, 500)
w = np.random.rand(500, 1)

start_time = time.time()
outputs = for_loop_feed_forward(X, w)

print("Time spent on calculating the outputs using for loops: ")
print(time.time() - start_time)

start_time = time.time()
outputs = vectorized_feed_forward(X, w)

print("Time spent on calculating the outputs using vectorization: ")
print(time.time() - start_time)
```

13] ✓ 1.2s

```
Time spent on calculating the outputs using for loops:
1.2737257480621338
Time spent on calculating the outputs using vectorization:
0.009953975677490234
```

سوال ۷

تابعی بنویسید که ی ک آرای ه داده شده و یک مقدار مشخص با نام threshold را دریافت کند. این تابع باید تمام عناصر آرایه را بررس ی کرده و عناصری که بیشتر از مقدار threshold هستند را به ی ک مقدار دلخواه)مثالً ۱(و عناصری که کمتر یا مساو ی threshold هستند را به ی ک مقدار دلخواه د یگر)مثالً ۰(تغییر دهد. سپس آرای ه جدید را با تغیی رات اعمال شده برگرداند.)سوال با استفاده از کتابخانه NumPy حل شود و استفاده از حلقه مجاز نیست.( ) ۱۰ امتیاز)

پاسخ سوال ۷)

Q7

```python
def replace_elements_above_threshold(array, threshold):
    """
    Replace elements in a NumPy array that are higher than the given threshold with a specified value.

    Parameters:
    - array (numpy.ndarray): Input NumPy array.
    - threshold (float): Threshold value to compare elements with.

    Returns:
    - numpy.ndarray: NumPy array with elements replaced above the threshold.
    """

    UP = 0
    HiGH = 1

    dics ={ "UP":0,"HIGH":1}
    modified_arr=np.where(array>threshold,HiGH,UP)
    return modified_arr
```

[5] ✓ 0.0s

```python
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)
```

[6] ✓ 0.0s

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

بخش های مختلف کد به شرح زیر است:

- def replace_elements_above_threshold(array, threshold): این خط تعریف تابع است که دو پارامتر ورودی می گیرد: array که یک آرایه نامپای است و threshold که یک عدد اعشاری است.

- UP = 0 و HIGH = 1 این دو خط دو متغیر را با مقادیر صفر و یک مقداردهی می کنند. این مقادیر برای جایگزین کردن عناصر آرایه استفاده می شوند.

- modified_arr=np.where(array>threshold,HIGH,UP) این خط یک تابع نامپای به نام np.where را فراخوانی می کند که سه آرگومان می گیرد: یک شرط، یک مقدار برای عناصری که شرط را برآورده می کنند و یک مقدار برای عناصری که شرط را برآورده نمی کنند. این تابع یک آرایه جدید با همان شکل ورودی ایجاد می کند که عناصر آن بر اساس شرط و مقادیر مشخص شده تغییر می کنند. در این مورد، شرط array>threshold است که بررسی می کند که کدام عناصر آرایه ورودی بیشتر از آستانه هستند. مقدار HIGH برای عناصری که شرط را برآورده می کنند و مقدار UP برای عناصری که شرط را برآورده نمی کنند انتخاب می شود. نتیجه این تابع در متغیر modified_arr ذخیره می شود.

- return modified_arr این خط مقدار متغیر modified_arr را به عنوان خروجی تابع برمی گرداند.

همانگونه که میبینیم در نهایت اعداد ۱ ا ۵ که زیر استانه بودند تبدیل به ۰ شدند و بقیه ی اعداد تبدیل به یک شده اند.

سوال ۸

بدون استفاده از کتابخانهnumpy ، کالسی برای داده س اختار ماتریس، تعریف کنید. این کالس باید شامل متد های زیر باشد : متدی برای بررسی کردن مساوی بودن ماتریس با ماتریس دیگر. در صورتی که ماتریس ها برابر بودند، خروجی True و در غیر این صورت False باشد.

متدی برای بررسی بزرگ تر یا کوچک تر بودن المان های ماتریس. خروجی باید ماتریسی با همان ابعاد ماتریسهای ورودی باشد و هر المان ماتریس بیان گر مقایسه بین المانهای دو ماتریس باشد. به عنوان مثال اگر در ماتریس اول، در اندیس [۲،۱] مقدار ۵ داشته باشد و در ماتریس دوم مقدار این اندیس برابر 3باشد، در ماتریس خروجی باید در این اندیس True قرار بگیرد.

-متدی برای بررسی زیر مجموعه بودن دو ماتریس.

- متدی برای محاسبه ضرب ۲ ماتریس)شبیه به عملکرد تابعdot.numpy

پاسخ سوال ۸)

Q8

```python
class Matrix:
    def __init__(self, matrix):

        self.values = matrix

    def is_equal(self, second_matrix):

        if not (len(self.values) == len(second_matrix.values) and len(self.values[0]) == len(second_matrix.values[0])):
            return False
        res = True
        num1 = len(self.values)
        num2 = len(self.values[0])
        for i in range(num1):
            for j in range(num2):
                if self.values[i][j] != second_matrix.values[i][j]:
                    res = False
                    break
            if not res:

                break
        return res

    def is_higher_elementwise(self, second_matrix):
        num1 = (len(self.values[0]))
        num2 = (len(self.values))

        res = [[True if self.values[i][j] > second_matrix.values[i][j] else False \
                        for j in range (num1)]
                            for i in range (num2)]
        return res
```

در قسمت اول برابر بودن ماتریس ها

و در قسمت دوم اگر ماتریسی درایه ی بزرگتر داشت در ماتریس نهایی true برگردانده شده است.

def __init__(self, matrix): این متد یک تابع سازنده است که یک پارامتر ورودی به نام matrix می گیرد که یک لیست از لیست های عددی است. این متد مقدار matrix را در یک متغیر نمونه به نام self.values ذخیره می کند که نشان دهنده مقادیر ماتریس است.

- def is_equal(self, second_matrix): این متد یک تابع عضو است که یک پارامتر ورودی به نام second_matrix می گیرد که یک شیء از کلاس Matrix است. این متد بررسی می کند که آیا ماتریس فعلی (self) با ماتریس داده شده (second_matrix) برابر است یا خیر. برای این کار، ابتدا تعداد سطرها و ستون های هر دو ماتریس را با هم مقایسه می کند و اگر برابر نبودند، مقدار False را برمی گرداند. سپس، یک متغیر بولی به نام res را با مقدار True مقداردهی می کند و با استفاده از دو حلقه for تمام عناصر هر دو ماتریس را با هم مقایسه می کند. اگر هر عنصری از ماتریس فعلی با عنصر متناظر از ماتریس داده شده مخالف بود، مقدار res را به False تغییر می دهد و از حلقه ها خارج می شود. در نهایت، مقدار res را به عنوان خروجی متد برمی گرداند.

• def is_higher_elementwise(self, second_matrix): این متد یک تابع عضو است که یک پارامتر ورودی به نام second_matrix می گیرد که یک شیء از کلاس Matrix است. این متد یک ماتریس بولی جدید ایجاد می کند که نشان می دهد که کدام عناصر ماتریس فعلی (self) بیشتر از عناصر ماتریس داده شده (second_matrix) هستند. برای این کار، ابتدا تعداد ستون ها و سطرهای ماتریس فعلی را در دو متغیر به نام num1 و num2 ذخیره می کند. سپس، با استفاده از یک فهم لیست، یک لیست از لیست های بولی ایجاد می کند که هر عنصر آن بر اساس شرط self.values[i][j] > second_matrix.values[i][j] مقدار True یا False می گیرد. این لیست را در یک متغیر به نام res ذخیره می کند و آن را به عنوان خروجی متد برمی گرداند.

نتیجه بخش ۱ و ۲:

در بخش اول همانگونه که میبینیم دو ماتریس که برابر نبودند فالس و یک ماتریس که با خود برابر است درست بازگردانده می شود.

در بخش دو هم سه درایه ی اول ماتریس اول که بزرگتر است درست و بقیه ی درایه ها کوچک تر ایت فالس برگردانده شده.

```
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))
print (matrix1.is_equal(matrix1))



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

[450] ✓ 0.0s

```
False
True
```

```
matrix3 = Matrix([[0, 0, 0], [10, 20, 30], [-1, 8, 10]])

# test proportion of matrices here and show the result #


print(matrix1.is_higher_elementwise(matrix2))
print(matrix1.is_higher_elementwise(matrix3))
```

[451] ✓ 0.0s

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

در این بخش زیر مجموعه بودن یا نبودن ماتریس یعنی ماتریس کوچکتر و یا دومی بخشی از ماتریس بزرگتر می باشد یا نه و ضرب ماتریس ها را پیاده سازی کردیم.

این بخش کد دو متد دیگر از کلاس Matrix را شامل می شود: is_subset و dot_product. توضیحات کد به شرح زیر است:

- def is_subset(self, second_matrix): این متد یک تابع عضو است که یک پارامتر ورودی به نام second_matrix می گیرد که یک شیء از کلاس Matrix است. این متد بررسی می کند که آیا ماتریس فعلی (self) یک زیرمجموعه از ماتریس داده شده (second_matrix) است یا خیر. به عبارت دیگر، آیا ماتریس فعلی در یکی از بخش های ماتریس داده شده قرار می گیرد یا خیر. برای این کار، ابتدا تعداد سطرها و ستون های هر دو ماتریس را در چهار متغیر به نام num1، num2، num3 و num4 ذخیره می کند. سپس، شکل ماتریس فعلی را در یک متغیر به نام cutShape و شکل ماتریس داده شده را در یک متغیر به نام restrict ذخیره می کند. اگر تعداد سطرها یا ستون های ماتریس فعلی بیشتر از ماتریس داده شده باشد، مقدار False را برمی گرداند. در غیر این صورت، با استفاده از دو حلقه for، تمام بخش های ممکن از ماتریس داده شده را که همان اندازه ماتریس فعلی هستند را بررسی می کند. برای این کار، از تابع Matrix برای ایجاد یک شیء ماتریس از بخش مورد نظر و از متد is_equal برای مقایسه آن با ماتریس فعلی استفاده می کند. اگر هر بخشی با ماتریس فعلی برابر بود، مقدار True را برمی گرداند. در نهایت، اگر هیچ بخشی با ماتریس فعلی برابر نبود، مقدار False را برمی گرداند.

- def dot_product(self, second_matrix): این متد یک تابع عضو است که یک پارامتر ورودی به نام second_matrix می گیرد که یک شیء از کلاس Matrix است. این متد حاصلضرب نقطه ای دو ماتریس را محاسبه می کند. برای این کار، ابتدا تعداد ستون های ماتریس فعلی را در یک متغیر به نام num1، تعداد سطرهای ماتریس داده شده را در یک متغیر به نام num2، تعداد ستون های ماتریس داده شده را در یک متغیر به نام num3 و تعداد سطرهای ماتریس فعلی را در یک متغیر به نام num4 ذخیره می کند. سپس، یک ماتریس صفر با شکل (num4, num3) را در یک متغیر به نام res ایجاد می کند. اگر تعداد ستون های ماتریس فعلی با تعداد سطرهای ماتریس داده شده برابر نباشد، یک خطای مقدار (ValueError) با پیام Cant Dot Together را پرتاب می کند. در غیر این صورت، با استفاده از سه حلقه for، تمام عناصر ماتریس خروجی را محاسبه می کند. برای این کار، هر سطر از ماتریس فعلی را در یک متغیر به نام row و هر ستون از ماتریس داده شده را در یک متغیر به نام column ذخیره می کند. سپس، یک متغیر به نام part را با مقدار صفر مقداردهی می کند و با استفاده از یک حلقه for، حاصل جمع ضرب عناصر متناظر از row و column را در آن ذخیره می کند. این مقدار را در عنصر متناظر ماتریس res قرار می دهد. در نهایت، مقدار متغیر res را به عنوان خروجی متد برمی گرداند.

```
    def is_subset(self, second_matrix):
        num1 = len(self.values)
        num2 = len(self.values[0])
        num3 = len(second_matrix.values)
        num4 = len(second_matrix.values[0])
        cutShape = (num1, num2)
        restrict = (num3, num4)

        part1 = cutShape[0]
        part2 = restrict[0]

        if part1 > part2 :
            return False

        if cutShape[1] > restrict[1] :
            return False


        for i in range(part2-part1 + 1):

            for j in range(restrict[1] - cutShape[1] + 1):

                if Matrix([row[j:j+cutShape[1]] for row in second_matrix.values[i:i+part1]]).is_equal(self):
                    return True
        return False

    def dot_product(self, second_matrix):

        num1 = len(self.values[0])
        num2 = len(second_matrix.values)
        num3 = (len(second_matrix.values[0]))
        num4 = (len(self.values))

        res = [[0 for j in range(num3)] for i in range (num4)]

        if num1 != num2:
            raise ValueError('Cant Dot Together')

        for i in range(num4):
            for j in range(num1):
                row = self.values[i]
                column = [row[j] for row in second_matrix.values]
                part =0
                for k in range(len(row)):
                    part += row[k] * column[k]
                res[i][j] = part
        return res
```

```
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 #
print(matrix4.is_subset(matrix1))
print(matrix5.is_subset(matrix1))
print(matrix6.is_subset(matrix1))
print(matrix5.is_subset(matrix2))
print(matrix6.is_subset(matrix2))
print(matrix5.is_subset(matrix2))
```

456] ✓ 0.0s

```
True
True
False
False
False
False
```

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

# test product of matrices here and show the result #
print(matrix7.dot_product(matrix8))
```

455] ✓ 0.0s

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

منابع :

https://numpy.org/

chatGPT