به نام خدا

# تمرین سری اول درس مبانی هوش محاسباتی دکتر ناصر مزینی

# فرزان رحمانی 99521271

## سوال اول

کد پایتون این سوال یک تابع element\_wise\_comparison را تعریف می‌کند که مقایسه‌های عنصری بین دو آرایه NumPy را انجام می‌دهد. دو آرایه NumPy را به عنوان ورودی می گیرد و چندین آرایه NumPy را برمی گرداند که شامل نتایج چهار مقایسه مختلف از نظر عنصر است:

greater\_result: از نظر عنصر بیشتر از مقایسه.

greater\_equal\_result: از نظر عنصر بزرگتر یا مساوی مقایسه است.

less\_result: از نظر عنصر کمتر از مقایسه.

less\_equal\_result: از نظر عنصر کمتر یا مساوی با مقایسه.

در اینجا توضیح مفصلی در مورد هر قسمت از کد ارائه شده است:

greater\_result = np.greater(array1, array2)

این خط از تابع NumPy np.greater برای اجرای یک عنصر بزرگتر از مقایسه بین عناصر array1 و array2 استفاده می کند.

greater\_result یک آرایه NumPy خواهد بود که در آن هر عنصر True است اگر عنصر مربوطه در array1 بزرگتر از عنصر مربوطه در array2 باشد و در غیر این صورت False.

نتیجه در متغیر greater\_result ذخیره می شود.

greater\_equal\_result = np.greater\_equal(array1, array2)

این خط از تابع NumPy np.greater\_equal برای اجرای یک عنصر بزرگتر یا مساوی با مقایسه بین عناصر آرایه1 و آرایه2 استفاده می کند.

greater\_equal\_result یک آرایه NumPy خواهد بود که در آن هر عنصر True است اگر عنصر مربوطه در array1 بزرگتر یا مساوی عنصر مربوطه در array2 باشد و در غیر این صورت False.

نتیجه در متغیر greater\_equal\_result ذخیره می شود.

less\_result = np.less(array1, array2)

این خط از تابع NumPy np.less برای مقایسه عنصری بین عناصر array1 و array2 استفاده می کند.

less\_result یک آرایه NumPy خواهد بود که در آن هر عنصر True است اگر عنصر مربوطه در array1 کمتر از عنصر مربوطه در array2 باشد و در غیر این صورت False.

نتیجه در متغیر less\_result ذخیره می شود.

less\_equal\_result = np.less\_equal(array1, array2)

این خط از تابع NumPy np.less\_equal برای اجرای یک عنصر کمتر یا مساوی با مقایسه بین عناصر array1 و array2 استفاده می کند.

less\_equal\_result یک آرایه NumPy خواهد بود که در آن هر عنصر True است اگر عنصر مربوطه در array1 کمتر یا برابر با عنصر مربوطه در array2 باشد و در غیر این صورت False.

نتیجه در متغیر less\_equal\_result ذخیره می شود.

در نهایت، تابع یک تاپل حاوی چهار نتیجه مقایسه را برمی‌گرداند:

greater\_result: مقایسه عنصر به عنصر بیشتر.

greater\_equal\_result: مقایسه عنصر به عنصر بیشتر یا مساوی.

less\_result: مقایسه عنصر به عنصر کمتر.

less\_equal\_result: مقایسه عنصر به عنصر کمتر یا مساوی.

مراجع:

<https://numpy.org/doc/stable/reference/generated/numpy.greater.html>   
<https://numpy.org/doc/stable/reference/generated/numpy.greater.html>   
<https://numpy.org/doc/stable/reference/generated/numpy.less.html>  
<https://numpy.org/doc/stable/reference/generated/numpy.less_equal.html>

## سوال دوم

این تابع بر اساس روش مشخص شده ضرب را روی دو آرایه NumPy انجام می دهد.

از دو آرایه NumPy (آرایه 1 و آرایه 2) و یک پارامتر رشته اختیاری (روش) استفاده می‌کند که به‌طور پیش‌فرض روی " element-wise" تنظیم می‌شود.

اگر روش " element-wise" باشد، با استفاده از تابع numpy.multiply، یک ضرب عنصر را انجام می دهد. این کار هر عنصر در آرایه 1 را با عنصر مربوطه در آرایه 2 ضرب می کند.

اگر روش "matrix-multiply" باشد، با استفاده از تابع numpy.matmul ضرب ماتریسی را انجام می دهد. این آرایه ها را به عنوان ماتریس در نظر می گیرد و آنها را ضرب می کند.

هر مقدار دیگری از متد یک ValueError را با پیامی مبنی بر نامعتبر بودن متد ایجاد می کند.

تابع آرایه نتیجه را از ضرب برمی گرداند.

کامنت ها نشان می دهد که چگونه می توان از عملگرهای \* و @ به ترتیب برای ضرب عنصر به عنصر و ماتریس به عنوان مخفف استفاده کرد.  
بنابراین به طور خلاصه، این کار ضرب آرایه را بر روی آرایه‌های NumPy بر اساس اینکه آیا می‌خواهیم از نظر عنصر یا ضرب ماتریسی انجام دهیم، آرایه نتیجه را برمی‌گرداند.

## سوال سوم

این تابع جمع بین دو آرایه p و q را با استفاده از پخش(broadcast) بر اساس روش مشخص شده انجام می دهد.

این تابع دو آرایه Numpy p و q و یک پارامتر متد را می گیرد که به طور پیش فرض روی "row-wise" قرار می گیرد.

اگر متد "سطری" باشد، جمع سطر را با پخش q انجام می دهد تا با استفاده از q[:, np.newaxis] با تعداد ردیف های p مطابقت داشته باشد. این q را به هر ردیف p اضافه می کند.

اگر روش "ستون محور" باشد، با اضافه کردن q به هر ستون p با استفاده از پخش، جمع ستونی را انجام می دهد. این q را به هر ستون p اضافه می کند.

ابتدا بررسی می کند که آیا اشکال با روش انتخابی سازگار هستند یا خیر.

برای "ردیفی"، تعداد ستون های p باید با تعداد اعضای q یکی باشد.

برای "ستونی"، تعداد سطر های p باید با تعداد اعضای q یکی باشد.

اگر اشکال ناسازگار باشند ValueError داده می شود.

هر روش نامعتبر دیگری یک ValueError ایجاد می کند.

در نهایت نتیجه اضافه کردن برگردانده می شود.

بنابراین به طور خلاصه، بر روی دو آرایه NumPy بر اساس پارامتر متد، پخش(broadcast) را بر اساس ردیف یا ستون انجام می دهد. اشکال سازگار مورد نیاز است.  
منابع:

<https://numpy.org/doc/stable/user/basics.broadcasting.html>

## سوال چهارم

این کد با استفاده از np.random.randint یک ماتریس تصادفی 4\*4 تولید می کند، آن را چاپ می کند، سپس قبل از چاپ مجدد، مقادیر بین 0 و 1 را عادی سازی می کند.

در اینجا یک توضیح مفصل است:

x = np.random.randint(1, 11, (4, 4))

یک آرایه 4×4 با ورودی های تصادفی بین 1 تا 10 ایجاد می کند

print(x)

آرایه تصادفی اصلی را چاپ می کند

max = np.max(x)

حداکثر مقدار را در آرایه پیدا میکند.

min = np.min(x)

مقدار حداقل را در آرایه پیدا میکند.

x = (x - min) / (max - min)

آرایه را با کم کردن min و تقسیم بر max-min عادی سازی میکند به محدوده 0 تا 1 میبرد.

print(x)

آرایه نرمال شده را چاپ میکند.  
بنابراین به طور خلاصه، یک ماتریس تصادفی تولید می کند، مقادیر را در محدوده 0-1 نرمال می کند و قبل و بعد چاپ می کند.

منابع:

<https://numpy.org/doc/stable/reference/random/generated/numpy.random.randint.html>  
<https://www.statology.org/numpy-normalize-between-0-and-1/>

## سوال پنجم

این کد از کتابخانه pandas برای دستکاری داده ها(manipulation) و matplotlib برای تجسم داده ها(visualization) استفاده می کند. این یک فایل CSV حاوی داده‌های سهام بر اساس تاریخ را می‌خواند، معیارهای مختلف را محاسبه می‌کند و نمودارها را تولید می‌کند. در اینجا خلاصه ای از آنچه هر قسمت از کد انجام می دهد آورده شده است:

وارد کردن کتابخانه های لازم:

matplotlib.pyplot برای ایجاد نمودار.

pandas برای دستکاری داده ها

فایل CSV را در DataFrame پاندا میخوانیم:

کد یک فایل CSV به نام "data.csv" را می خواند و داده ها را در یک DataFrame پاندا به نام dataframe ذخیره می کند.

محاسبه بازده روزانه:

این بازده روزانه را با گرفتن درصد تغییر ستون "قیمت بسته" محاسبه می کند و این اطلاعات را به عنوان یک ستون جدید به نام "بازده روزانه" در DataFrame اضافه می کند.

میانگین بازده روزانه را محاسبه و چاپ میکنیم:

میانگین (میانگین) ستون "بازده روزانه" را محاسبه می کند و نتیجه را چاپ می کنیم.

محاسبه و چاپ انحراف استاندارد بازده روزانه:

انحراف استاندارد ستون "بازگشت روزانه" را محاسبه می کند و نتیجه را چاپ می کند.

قیمت های بسته شدن روزانه را در طول زمان ترسیم میکنیم:

با استفاده از ستون‌های «تاریخ» و «قیمت بسته» از DataFrame، یک نمودار خطی از قیمت‌های بسته شدن روزانه در طول زمان ایجاد می‌کند. طرح شامل عنوان، برچسب ها و گزینه های قالب بندی است.

بازده روزانه را در طول زمان ترسیم میکنیم:

مشابه طرح قبلی، با استفاده از ستون‌های «تاریخ» و «بازگشت روزانه» از DataFrame، یک نمودار خطی از بازده روزانه در طول زمان ایجاد می‌کند.

روزهایی را با بیشترین و کمترین بازده روزانه پیدا میکنیم:

ایندکس سطر با بالاترین و کمترین مقدار را در ستون "بازگشت روزانه" شناسایی می کند و تاریخ و مقدار مربوطه را بازیابی می کند.

تاریخ و ارزش بالاترین و کمترین قیمت سهام تاریخی را می یابیم:

مشابه بازده‌های روزانه، شاخص ردیفی را با بالاترین و کمترین مقدار در ستون «قیمت بسته» شناسایی می‌کند و تاریخ و مقدار مربوطه را بازیابی می‌کند.

چاپ نتایج:

این کد بالاترین و کمترین مقادیر بازده روزانه و تاریخ مربوط به آنها و همچنین بالاترین و کمترین ارزش بسته شدن و تاریخ مربوطه را چاپ می کند.  
چند مرحله کلیدی:

* pct\_change() برای محاسبه درصد تغییر
* mean() و std() برای محاسبه آمار
* matplotlib برای تجسم داده ها
* idxmax() و idxmin() برای یافتن شاخص max/min
* loc برای فهرست بندی در DataFrame
* رشته های f برای چاپ خروجی فرمت شده

این کد تجزیه و تحلیل داده های تاریخی سهام، از جمله تجسم قیمت های بسته شدن و بازده روزانه، و همچنین شناسایی بهترین و بدترین روزهای عملکرد از نظر بازده روزانه و قیمت های بسته شدن را ارائه می دهد.

منابع:

<https://matplotlib.org/3.2.2/api/_as_gen/matplotlib.axis.XAxis.set_major_locator.html>  
<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.pct_change.html>  
<https://pandas.pydata.org/docs/>

## سوال ششم

کد پایتون ارائه شده تفاوت بین اجرای یک عملیات فید فوروارد با استفاده از حلقه for و استفاده از vectorization را نشان می دهد. این عملیات اغلب در محاسبات یادگیری ماشین و شبکه های عصبی، جایی که داده ها و وزن های ورودی دارید و می خواهید مقادیر خروجی را محاسبه کنید، مشاهده می شود.

در اینجا توضیحی گام به گام درباره کاری که کد انجام می دهد آورده شده است:

1. عملکرد `for\_loop\_feed\_forward`:

- این تابع با استفاده از یک حلقه for یک عملیات فید فوروارد را انجام می دهد.

- دو پارامتر را به عنوان ورودی می گیرد: «X» که یک ماتریس داده ورودی شکل «(num\_samples، num\_features)» است و «w» که یک ماتریس وزنی از شکل «(num\_features, 1)» است.

- یک آرایه خالی «خروجی» را برای ذخیره نتایج مقداردهی می کند. «خروجی‌ها» شکل «(تعداد\_نمونه‌ها، 1)» برای ذخیره خروجی هر نمونه خواهند داشت.

- روی هر نمونه در ماتریس داده ورودی 'X' تکرار می شود و خروجی آن نمونه را با استفاده از یک حلقه تو در تو محاسبه می کند:

- برای هر نمونه، «y\_i» به 0 مقداردهی اولیه می شود.

- سپس روی هر ویژگی برای آن نمونه تکرار می‌شود و با ضرب مقدار ویژگی در وزن مربوطه و جمع‌آوری نتیجه، «y\_i» را محاسبه می‌کند.

- در نهایت، "y\_i" را به موقعیت مربوطه در آرایه "خروجی ها" اختصاص می دهد.

- تابع آرایه "خروجی" را که حاوی نتایج است برمی گرداند.

2. عملکرد `'feed\_forward vectorized:

- این تابع با استفاده از بردارسازی یک عملیات feed-froward را انجام می دهد.

- پارامترهای ورودی مشابه «for\_loop\_feed\_forward» را می گیرد.

- خروجی را با استفاده از ضرب ماتریس محاسبه می کند. کد «np.dot(X, w)» حاصل ضرب نقطه‌ای ماتریس داده ورودی «X» و ماتریس وزن «w» را محاسبه می‌کند و در نتیجه ماتریس خروجی «خروجی‌ها» ایجاد می‌شود.  
3. تولید داده های نمونه تصادفی:

- قبل از مقایسه دو روش، کد داده های نمونه تصادفی تولید می کند.

- «X»، یک ماتریس داده تصادفی با ردیف‌های «تعداد\_نمونه‌ها» و ستون‌های «تعداد\_ویژگی‌ها» و «w»، یک ماتریس وزن تصادفی با همان تعداد ویژگی «X» ایجاد می‌کند.

- تعداد نمونه ها (`num\_samples`) و ویژگی ها (`num\_features`) در این مثال به ترتیب 1000 و 500 تنظیم شده است.

4. مقایسه عملکرد:

- کد زمان صرف شده برای محاسبه خروجی ها را با استفاده از روش حلقه for و روش برداری اندازه گیری می کند.

- از تابع "time.time()" برای ثبت زمان شروع، انجام محاسبات و سپس محاسبه زمان سپری شده استفاده می کند.

- زمان صرف شده برای هر روش را چاپ می کند و به شما امکان می دهد کارایی دو روش را مقایسه کنید.

به طور خلاصه، کد تفاوت عملکرد بین یک پیاده سازی مبتنی بر حلقه for و یک پیاده سازی برداری شده برای یک عملیات feedforward را نشان می دهد. بردارسازی به طور کلی برای مجموعه داده های بزرگ کارآمدتر است و یک تکنیک رایج است که برای سرعت بخشیدن به محاسبات عددی در پایتون، به ویژه هنگام کار با کتابخانه هایی مانند NumPy استفاده می شود.

بردارسازی در بسیاری از موارد به چند دلیل سریعتر از استفاده از حلقه ها است:

1. کد کامپایل شده: بسیاری از کتابخانه ها و عملیات در زبان های برنامه نویسی مانند NumPy، که اغلب برای عملیات برداری استفاده می شود، در زبان های سطح پایین تر مانند C یا Fortran پیاده سازی می شوند. این کتابخانه‌ها بسیار بهینه‌سازی شده‌اند و می‌توانند از ویژگی‌های خاص سخت‌افزار بهره ببرند و به طور قابل‌توجهی سریع‌تر از کد پایتون تفسیر شده باشند.

2. موازی سازی : عملیات بردار را می توان به طور خودکار توسط کتابخانه های زیرین موازی کرد. CPU های مدرن دارای چندین هسته هستند و کد برداری می تواند به طور موثر کار را در بین این هسته ها توزیع کند و از پردازنده های چند هسته ای کامل استفاده کند.

3. سربار مفسر پایتون کاهش یافته: پایتون یک زبان تفسیر شده است و اجرای کد پایتون شامل سربار مفسر است. در یک حلقه for، ممکن است تعداد زیادی فراخوانی تابع پایتون برای هر تکرار داشته باشید که می تواند نسبتا کند باشد. در عملیات برداری، کتابخانه زیربنایی بسیاری از عملیات را در یک فراخوانی تابع انجام می دهد و این سربار را کاهش می دهد.

4. الگوریتم های بهینه شده: کتابخانه هایی مانند NumPy برای استفاده از الگوریتم های بسیار بهینه شده برای عملیات رایج طراحی شده اند. این الگوریتم‌ها اغلب کارآمدتر از چیزی هستند که ممکن است در یک حلقه for پیاده‌سازی کنید.

5. کارایی حافظه: عملیات بردار می تواند از نظر حافظه کارآمدتر باشد زیرا می تواند عملیات را در محل انجام دهد یا حافظه را هوشمندتر تخصیص دهد. در مقابل، برای حلقه‌ها ممکن است نیاز به ایجاد آرایه‌ها یا اشیاء موقت داشته باشند، که می‌تواند کند و حافظه فشرده باشد.

6. سادگی و خوانایی: کد برداری اغلب مختصرتر است و خواندن و نگهداری آن آسان تر است. این می تواند منجر به اشکالات کمتری شود و درک هدف کد را آسان تر کند.

7. سربار قفل مفسر جهانی (Global Interpreter Lock): در مورد پایتون، قفل مترجم جهانی (GIL) می تواند موازی بودن کدهای چند رشته ای پایتون را محدود کند. عملیات برداری اغلب می تواند GIL را دور بزند و امکان استفاده بهتر از چند هسته را فراهم کند.

توجه به این نکته مهم است که همه عملیات را نمی توان به طور موثر بردار کرد. برای مثال، کارهایی که شامل شاخه‌بندی پیچیده، تخصیص حافظه پویا، یا الگوهای دسترسی نامنظم به داده‌ها می‌شوند، ممکن است از بردارسازی سود نبرند. با این حال، برای کارهایی که شامل محاسبات عددی روی آرایه‌ها یا ماتریس‌های بزرگ است، بردارسازی اغلب رویکرد ترجیحی برای دستیابی به عملکرد و سادگی کد است.

منابع:

<https://numpy.org/doc/stable/reference/generated/numpy.matmul.html>  
<https://numpy.org/doc/stable/reference/generated/numpy.dot.html>   
<https://chat.openai.com/>

## سوال هفتم

این تابع یک آرایه NumPy و یک مقدار آستانه را به عنوان ورودی می گیرد. یک آرایه NumPy جدید برمی گرداند که در آن مقادیر بالای آستانه با 1 و مقادیر زیر آستانه با 0 جایگزین می شوند.

در اینجا کاری است که گام به گام انجام می دهد:

1. تابع replace\_elements\_above\_threshold را با دو پارامتر - آرایه و آستانه تعریف میکنیم.

2. آرایه را به آرایه بولی تبدیل میکنیم که نشان می دهد کدام عناصر بالاتر از آستانه هستند:

آرایه > آستانه

این منجر به یک آرایه بولی از مقادیر True/False می شود.

3. آرایه بولی را با استفاده از .astype(int) به اعداد صحیح 0/1 تبدیل می کنیم.

True می شود 1، False می شود 0.

4. همچنین می توانیم از np.greater برای مقایسه آرایه با آستانه استفاده کنیم.

5. گزینه بهتر استفاده از np.where برای جایگزینی عناصر در یک مرحله است:

- جایی که آرایه > آستانه، با 1 جایگزین کنید.

- در غیر این صورت، با 0 جایگزین کنید.

6. آرایه اصلاح شده را با 0 و 1 بر اساس آستانه بر میگردانیم.

بنابراین به طور خلاصه، یک آرایه NumPy می گیرد، آن را با یک آستانه مقایسه می کند و یک آرایه جدید را برمی گرداند که در آن مقادیر بالای آستانه با 1 و پایین تر با 0 جایگزین می شوند.

مراحل کلیدی استفاده از مقایسه بولی، astype، و np.where برای آستانه گذاری برداری است.  
منابع:

<https://numpy.org/doc/stable/reference/generated/numpy.greater.html>  
<https://numpy.org/doc/stable/reference/generated/numpy.where.html>

## سوال هشتم

کد پایتون ارائه شده یک کلاس 'Matrix' را تعریف می کند که نشان دهنده یک ماتریس دو بعدی است و شامل چندین method برای انجام عملیات روی ماتریس ها است. بیایید عملکرد این کلاس و متدهای آن را بررسی کنیم:

1. Constructor «\_\_init\_\_»:

- سازنده یک شی «ماتریس» را با یک لیست معین از لیست‌هایی که ماتریس را نشان می‌دهند مقداردهی اولیه می‌کند. ماتریس را به عنوان یک ویژگی(attribute) ذخیره می کند و تعداد سطرها و ستون های ماتریس را محاسبه می کند. در واقع، متد «\_\_init\_\_()» سازنده کلاس است. این یک پارامتر "ماتریس" را می گیرد که لیستی از لیست هایی است که عناصر ماتریس را نشان می دهد. در داخل سازنده، این ماتریس را به عنوان یک ویژگی به نام "self.matrix" ذخیره می کند. همچنین تعداد "ردیف" و "ستون" را در ماتریس محاسبه و ذخیره می کند.

2. Method `\_\_str\_\_`:

- این روش برای ایجاد یک نمایش رشته ای قابل خواندن توسط انسان از ماتریس استفاده می شود. ماتریس را در یک رشته با قالب بندی زیبا، با سطرها و ستون هایی که با فاصله و سطرهایی با کاراکترهای خط جدید از هم جدا شده اند، برمی گرداند.در واقع، متد `\_\_str\_\_()` برای ایجاد نمایش رشته ای از ماتریس هنگام چاپ استفاده می شود. از میان ردیف ها حلقه می زند، هر عنصر را به یک رشته تبدیل می کند و آنها را با فاصله به هم می پیوندد. سپس رشته های ردیف را با کاراکترهای خط جدید به هم می پیوندد.

3. Method " is\_equal":

- این روش بررسی می کند که آیا ماتریس فعلی برابر با ماتریس دیگری است (به عنوان یک شی «ماتریس» ارائه می شود). ابتدا ابعاد (ردیف‌ها و ستون‌ها) را با هم مقایسه می‌کند و سپس از نظر عنصر بررسی می‌کند که آیا ماتریس‌ها برابر هستند. اگر مساوی باشند «درست» و در غیر این صورت «نادرست» را برمی‌گرداند. در واقع، متد «is\_equal()» بررسی می‌کند که آیا نمونه ماتریس فعلی برابر است یا خیر. ابتدا بررسی می کند که آیا ابعاد (ردیف ها و ستون ها) برابر هستند یا خیر. اگر چنین است، سپس از طریق هر عنصر تکرار می شود و بررسی می کند که آیا آنها برابر هستند یا خیر. اگر همه عناصر برابر باشند True و در غیر این صورت False را برمی‌گرداند.

4. Method "is\_higher\_elementwise":

- این روش عناصر ماتریس جاری را با عناصر ماتریس دیگری از نظر عنصر(element-wise) مقایسه می کند. یک ماتریس جدید (به عنوان یک شی «ماتریس») با همان شکل ماتریس های ورودی برمی گرداند. اگر عنصر مربوطه در ماتریس فعلی بالاتر از عنصر در ماتریس دیگر باشد، هر عنصر ماتریس جدید "درست" است. در غیر این صورت، "نادرست" است. در واقع، متد «is\_higher\_elementwise()» نمونه ماتریس فعلی را از نظر عنصر با نمونه ماتریس دیگری مقایسه می‌کند. ابتدا مطابقت ابعاد را بررسی می کند. سپس یک ماتریس جدید به نام `higher\_matrix` برای ذخیره نتایج ایجاد می کند. از طریق هر عنصر تکرار می شود و دو ماتریس را با هم مقایسه می کند. اگر عنصر ماتریس فعلی بالاتر باشد، آن موقعیت را در «ماتریس\_بالا» روی True، else False تنظیم می‌کند. ماتریس "ماتریس\_بالا" را برمی گرداند.

5. Method "is\_subset":

- این روش بررسی می کند که آیا ماتریس فعلی زیرمجموعه ای از ماتریس دیگری است یا خیر. ابتدا ابعاد را مقایسه می کند و سپس در ماتریس دیگر تکرار می کند تا زیرماتریسی را پیدا کند که با ماتریس فعلی مطابقت داشته باشد. اگر مطابقتی پیدا شود، «درست» را برمی‌گرداند. در غیر این صورت، "نادرست" را برمی گرداند. در واقع، متد «is\_subset()» بررسی می‌کند که آیا ماتریس فعلی زیرمجموعه‌ای از ماتریس دیگری است یا خیر. ابتدا ابعاد را با هم مقایسه می کند. سپس از طریق برش‌هایی از ماتریس دوم به اندازه ماتریس فعلی تکرار می‌شود و برابری را با استفاده از «is\_equal()» بررسی می‌کند. اگر هر برشی برابر باشد، True را برمی‌گرداند، در غیر این صورت پس از بررسی همه برش‌ها، False را برمی‌گرداند.

6. Method "dot\_product":

- متد dot\_product در کلاس Matrix برای محاسبه حاصل ضرب نقطه ای بین دو شی ماتریس طراحی شده است. حاصل ضرب نقطه‌ای که به‌عنوان حاصلضرب اسکالر یا ضرب داخلی نیز شناخته می‌شود، یک عملیات ریاضی است که دو ماتریس (یا بردار) را می‌گیرد و بسته به ابعاد ماتریس‌های ورودی، یک مقدار اسکالر یا ماتریس دیگری را در نتیجه تولید می‌کند. در اینجا توضیح مفصلی از آنچه این روش انجام می دهد آورده شده است:

این متد یک آرگومان به نام second\_matrix می گیرد که یک شی ماتریس دیگر است.

این روش ابتدا ابعاد دو ماتریس را بررسی می کند تا مشخص کند کدام یک از موارد زیر اعمال می شود:

آ. حالت ماتریس "2-D در 2-D" (ضرب ماتریسی): این زمانی اعمال می شود که هر دو ماتریس بیش از یک ردیف و بیش از یک ستون داشته باشند. این روش بررسی می‌کند که آیا تعداد ستون‌های ماتریس اول (خود) با تعداد ردیف‌های ماتریس دوم (second\_matrix) مطابقت دارد یا خیر. اگر ابعاد با هم سازگار نباشند، یک ValueError ایجاد می کند. اگر آنها سازگار باشند، با استفاده از حلقه‌های تودرتو به محاسبه محصول نقطه‌ای می‌پردازد.

ب. حالت ماتریس "1-D در 1-D" (ضرب داخلی بردارها): این حالت زمانی اعمال می شود که هر دو ماتریس بردار ستون یا بردار ردیف باشند. این روش بررسی می‌کند که آیا تعداد ردیف‌های هر دو ماتریس مطابقت دارند یا خیر. اگر ابعاد سازگار نداشته باشند، یک ValueError ایجاد می کند. اگر آنها سازگار باشند، محصول نقطه ای را با استفاده از یک حلقه محاسبه می کند و نتیجه را به عنوان یک ماتریس 1\*1 برمی گرداند.

ج. حالت ماتریس "0-D در 0-D" (ضرب دو اسکالر): این حالت زمانی اعمال می شود که هر دو ماتریس ماتریس 1\*1 (مقادیر اسکالر) باشند. به سادگی دو مقدار اسکالر را با هم ضرب می کند و نتیجه را به صورت یک ماتریس 1\*1 برمی گرداند.

د. حالت ماتریس "0-D در 2-D" یا "0-D در 1-D" (ضرب اسکالر-بردار یا اسکالر-ماتریس): در این حالت، یکی از ماتریس ها یک مقدار اسکالر (ماتریس 1\*1) است. و دیگری ماتریسی با بیش از یک سطر و ستون است. این روش مقدار اسکالر را با هر عنصر ماتریس دوم ضرب می کند و نتیجه را به عنوان یک ماتریس جدید برمی گرداند.

ه. حالت ماتریس "1-D در 2-D" (sum product over the last axis): این حالت زمانی اعمال می شود که یک ماتریس یک بردار 1-D (بردار ردیف یا ستون) باشد و ماتریس دیگر یک ماتریس دو بعدی باشد. این روش بررسی می‌کند که آیا تعداد سطر و ستون‌های بردار 1-D با تعداد سطر و ستون‌های ماتریس دو بعدی مطابقت دارد یا خیر. اگر ابعاد سازگار نداشته باشند، یک ValueError ایجاد می کند. اگر آنها سازگار باشند، محصول نقطه‌ای را با استفاده از حلقه‌های تودرتو محاسبه می‌کند.

این method ماتریس نتیجه را بر اساس مقادیر محاسبه شده می سازد و ماتریس نتیجه به عنوان یک شی ماتریس جدید برگردانده می شود.

به طور خلاصه، روش dot\_product موارد مختلفی از محاسبات نقطه ای بین ماتریس ها یا بردارها را مدیریت می کند و نتیجه را به عنوان یک شی ماتریس برمی گرداند. ابعاد را برای اطمینان از سازگاری بررسی می کند و عملیات ضرب یا محصول داخلی مناسب را بر اساس شکل ماتریس های ورودی انجام می دهد.

هر یک از این روش‌ها(methods) برای کار بر روی نمونه‌هایی(instances) از کلاس «Matrix» و انجام عملیات‌های مختلف مرتبط با ماتریس‌ها، مانند بررسی‌های برابری، مقایسه‌های عنصری، بررسی‌های زیرمجموعه و محصولات نقطه‌ای طراحی شده‌اند. از این روش ها می توان برای دستکاری و تجزیه و تحلیل ماتریس ها به روشی ساختاریافته با استفاده از کلاس «ماتریس» استفاده کرد.  
  
  
منابع:

<https://numpy.org/doc/stable/reference/generated/numpy.empty.html>  
<https://stackoverflow.com/questions/45737880/how-to-iterate-over-this-n-dimensional-dataset>  
<https://chat.openai.com/>

پایان