به نام خدا

هانیه اسعدی ۹۹۵۲۱۰۵۵

منابع در فایل ipynb. ذکر شده است.

۱- از تابع Where برای حل سوال استفاده می‌کنیم. هر عنصر را با عنصر متناظر مقایسه کرده و با توجه به شرط، مقدار true/false را در آرایه جواب قرار می‌دهیم. برای مثال برای آرایه بزرگتر، در مقایسه عنصر به عنصر، اگر عنصر در آرایه 1 از عنصر متناظر در آرایه 2 بزرگتر باشد، مقدار true در آرایه جواب قرار می‌دهیم.

2- براساس متد گرفته شده در ورودی تابع، اگر متد element-wise باشد، از تابع multiply که برای ضرب عنصر به عنصر است استفاده می‌کنیم. در غیر این صورت، از تابع matmul که برای ضرب ماتریسی دو ماتریس است استفاده می‌کنیم.

3- با توجه به متد ورودی، ماتریس را به ماتریس ستونی یا سطری تبدیل می‌کنیم. سپس چک می‌کنیم که آیا تعداد ستون‌ها/سطرهای ماتریس دوم با ماتریس اول همخوانی دارد یا خیر. اگر نداشته باشد، نمیتوان آن را اضافه کرد و مقدار invalid در خروجی چاپ می‌شود. اگر خطایی نداشته باشیم ماتریس دوم را با استفاده از عملگر + به ماتریس اول اضافه می‌کنیم.

4- با استفاده از random.randint یک ماتریس 4 در 4 با عناصر بین 1 تا 10 تولید می‌کنیم. برای نرمال کردن مقادیر، ابتدا مقادیر ماکسیمم و مینیمم ماتریس را

پیدا می‌کنیم. برای نرمال کردن هر عنصر، اختلاف عنصر و مقدار مینیمم را بر اختلاف مقدار ماکسیمم و مینیمم تقسیم می‌کنیم.

5- 1) با استفاده از تابع diff مقدار ستون closing price هر سطر را از سطر قبلی کم کرده و دخیره می‌کنیم. برای بدست آوردن مقدار بازده روزانه: ابتدا مقداری که قبلا محاسبه کرده بودیم را یکی شیفت می‌دهیم(چون اختلاف روز اول و قبلش NaN است)، و ردیف آخر را حساب نمی‌کنیم. این ستون را بر مقدار ستون closing price تقسیم میکنیم.(ردیف آخر closing price را هم در نظر نمی‌گیریم چون روز آخر برای حساب بازدهی لازم نیست)

2) با تابع mean، میانگین داده هایی که در قسمت قبل محاسبه شد را بدست می‌آوریم.

3) با تابع std واریانس داده های قسمت اول را بدست می‌آوریم.

4) با استفاده از تابع plot، نمودار closing price را بر حسب date رسم می‌کنیم.

5) مقدار بازده روزانه از قسمت اول را بر حسب date(بجز روز اول) رسم می‌کنیم.

6) برای پیدا کردن ایندکس بیشترین و کمترین مقدار بازده روزانه، از تابع idxmax و idxmin استفاده می‌کنیم. پس از پیدا کردن ایندکس ها، مقدار ستون date ردیف بعدی همان ایندکس ها را نمایش می‌دهیم.

7) اطلاعات ردیفی که مقدار ستون Closing price آن برابر با کوچکترین/بیشترین مقدار closing price باشد، نمایش می‌دهیم.

6- برای پیاده سازی با حلقه، ابتدا یک ماتریس ستونی تمام صفر با تعداد ردیف ماتریس دیتا می‌سازیم. سپس با استفاده از حلقه، هر ردیف ماتریس دیتا را در ماتریس وزن ضرب و حاصل جمع آنها را در ردیف مربوطه قرار می‌دهیم.

در روش vectorization، از تابع matmul برای بدست آوردن پاسخ استفاده میکنیم.

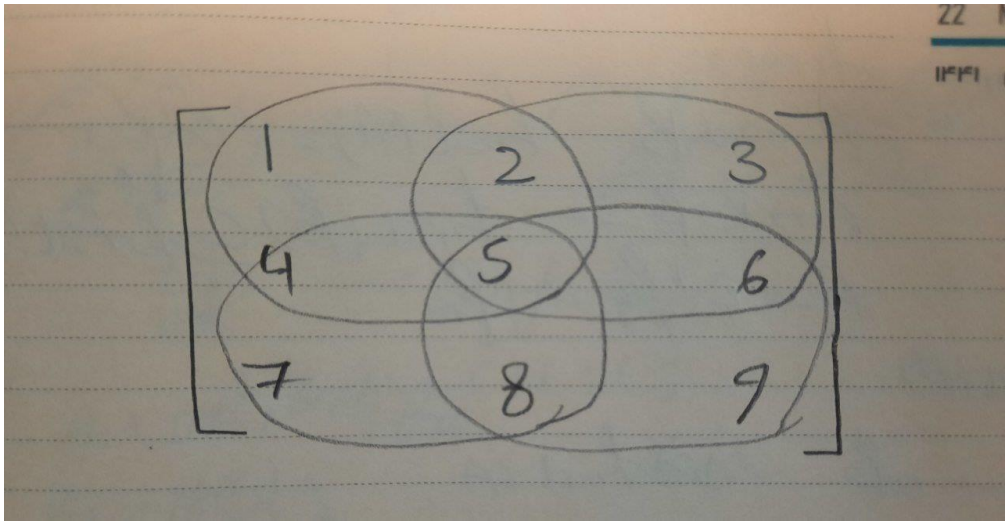
تفاوت این دو در این است که تابع آماده ضرب ماتریسی matmul از کتابخانه numpy بهینه تر است و سرعت اجرای بیشتری هم دارد. Numpy بهینه سازی های مختلفی دارد که به اجرای سریعتر و عملکرد بهتر کمک میکند.

7- مثل سوال اول، هر عنصر را با threshold داده شده مقایسه کرده، اگر مقدار کوچکتر یا مساوی ترشلد باشد، مقدار صفر و در غیر این صورت، 1 جایگزین میشود.

8- 1) با عملگر == مساوی بودن دو ماتریس را بررسی می‌کنیم.

2) ابتدا یک ماتریس دو بعدی با همان ابعاد ماتریس ورودی ایجاد می‌کنیم و همه اعضای آن را به صورت دیفالت false قرار می‌دهیم. با دو حلقه For تو در تو، هر element را با element متناظر مقایسه می‌کنیم. اگر عضو در ماتریس اول، از ایندکس متناظر در ماتریس دوم بزرگتر مساوی باشد، همان ایندکس را در ماتریس خروجی true می‌کنیم. در غیر این صورت همان false می‌ماند.

3) ابعاد ماتریس کوچکتر (که باید زیر مجموعه بودن آن بررسی شود) را به دست می‌آوریم. تعداد جایشگت های متمایزی که ماتریس میتواند در ماتریس بزرگتر وجود داشته باشد را حساب میکنیم. با دو حلقه for تو در تو، هر زیرماتریس را مانند شکل زیر بدست آورده و با ماتریس کوچکتر مقایسه می‌کنیم.



اگر یکی از زیرماتریس ها با ماتریس اصلی برابر بود مقدار flag را true می‌کنیم. در غیر این صورت مقدار پرچم false باقی می‌ماند که یعنی اولین ماتریس ورودی زیر مجموعه ماتریس دوم نیست.

4) ابتدا یک ماتریس تمام صفر با تعداد سطرهای ماتریس اول و ستون های ماتریس دوم ایجاد میکنیم. با دو حلقه فور، مقدار sum of products هر ستون و سطر را بدست می‌آوریم.