به نام خدا

گزارش تمرین اول درس هوش محاسباتی

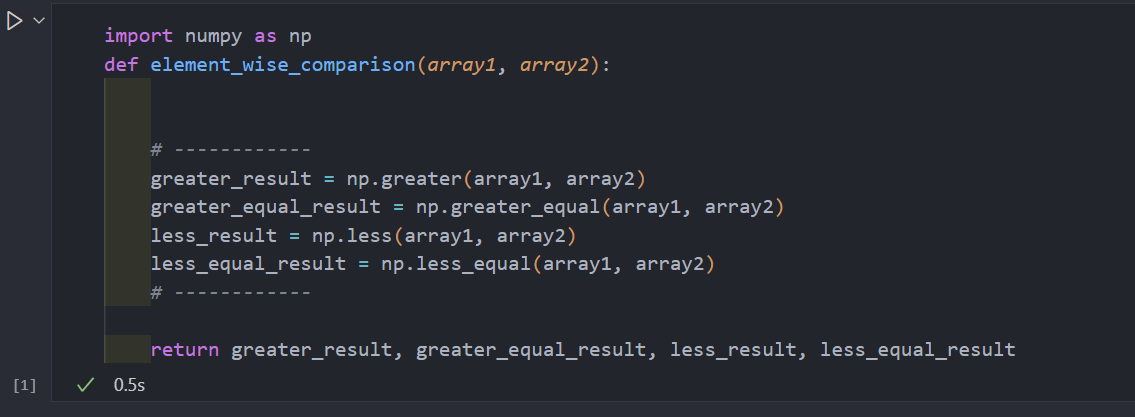
علیرضا اسلامی خواه

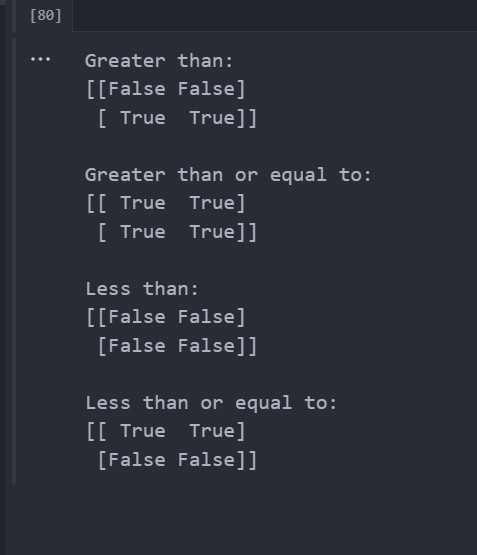
99521064

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.

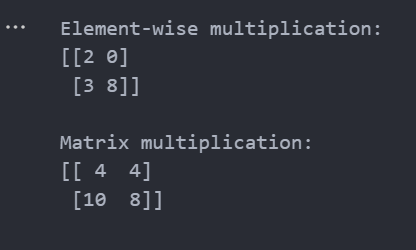




در اینجا بولین های نشان داده شده بیانگر درستی یا غلط بودن عبارت متد مورد نظر میباشند.

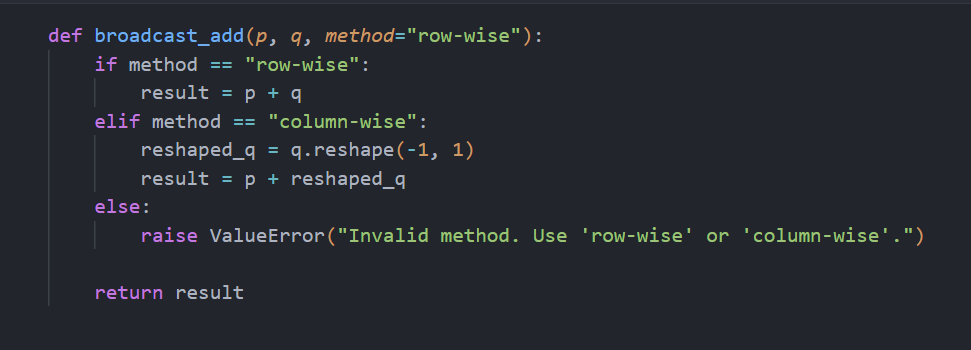
سوال دوم :

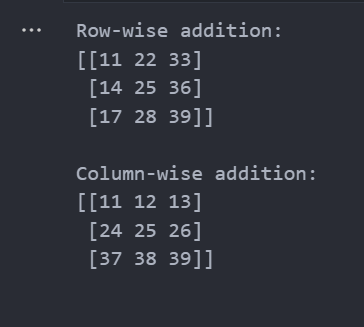
در این سوال مقصود ضرب عنصر به عنصر و ماتریسی بود که با استفاده از توابع multiply و dot انجام دادیم و همچنین جواب در فایل Jupyter قابل مشاهده است.



سوال سوم :

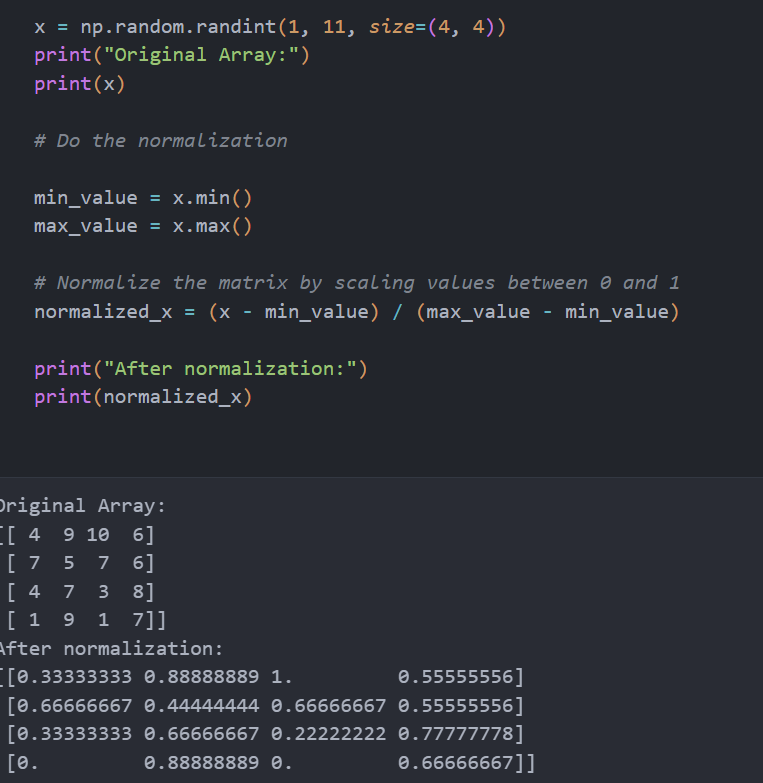
در این سوال ماتریس دوم را باید یکبار به صورت افقی یا همان سطری و یکبار به صورت عمودی یا همان ستونی در می اوردیم (که برای اینکار از reshape کمک گرفتیم) و سپس عملیات اضافه کردن را انجام میدادیم. خروجی این سوال در کد قابل مشاهده است.





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.



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')

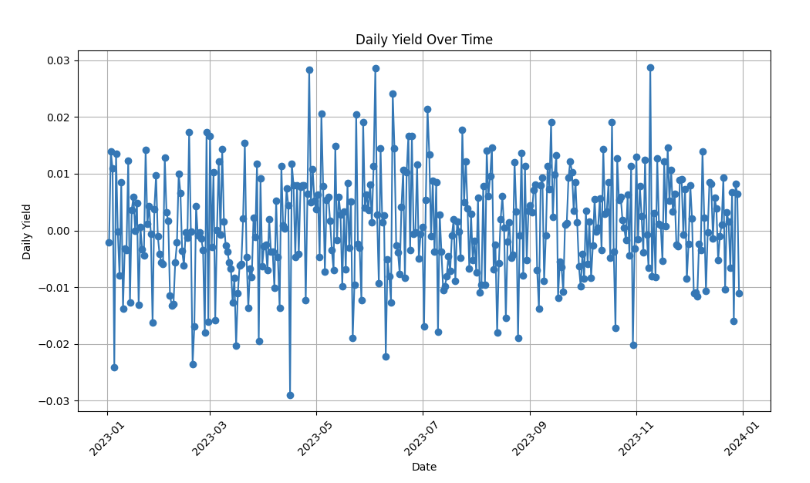
plt.grid(True)

plt.xticks(*rotation*=45)

plt.tight\_layout()

plt.show()

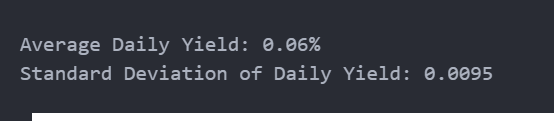
و خروجی دلخواه را میگیریم :



بخش دوم و سوم :

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

خروجی :



بخش چهارم :

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

*#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')

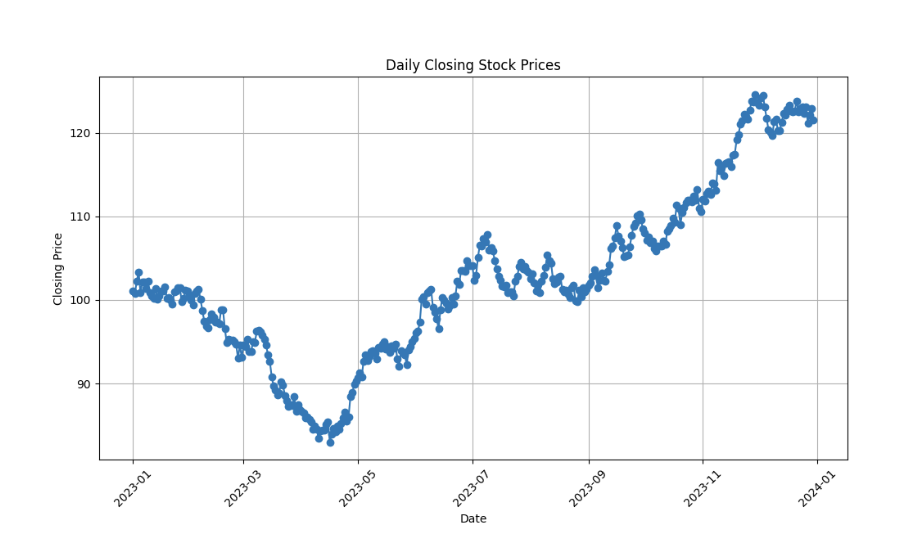
plt.ylabel('Closing Price')

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')

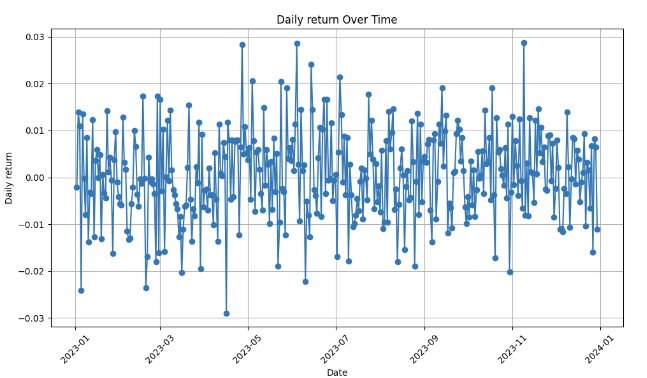
plt.grid(True)

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()]

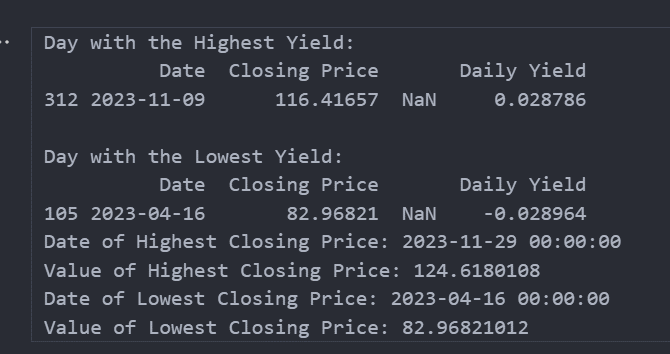
lowest\_closing\_price\_value = data['Closing Price'].min()

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}')



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

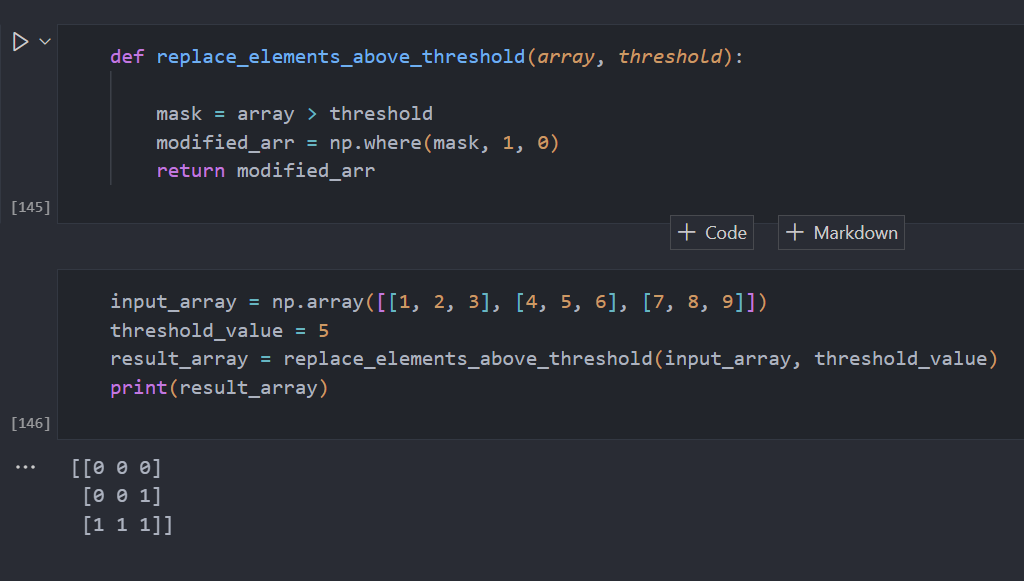
def vectorized\_feed\_forward(*X*, *w*):

    outputs = np.dot(X , w)

    return outputs

سوال هفتم :

مانند تصویر و متودهای ذکر شده به خروجی دلخواه رسیدیم.



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:

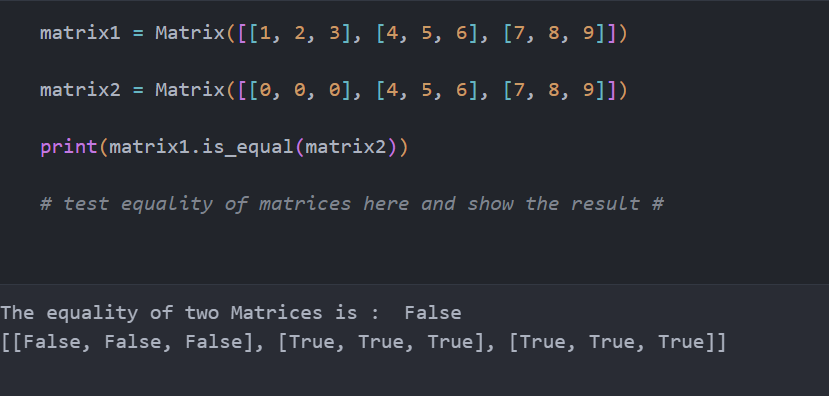
                    output[i][j] = False

                    flag = False

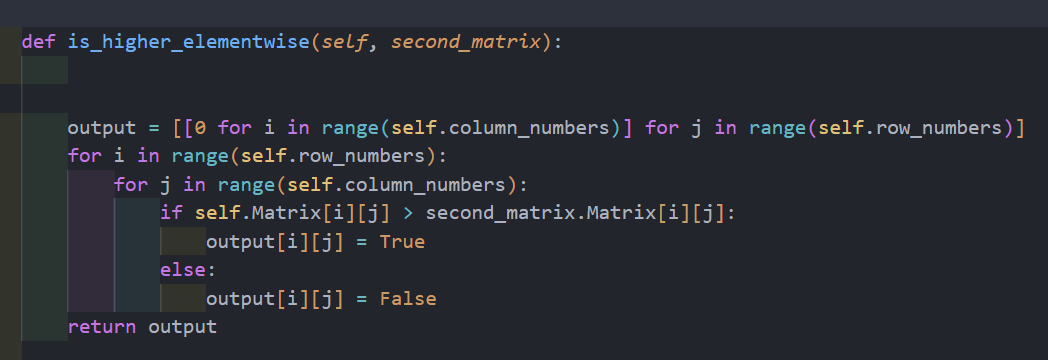
        print("The equality of two Matrices is : ", flag)

        return output

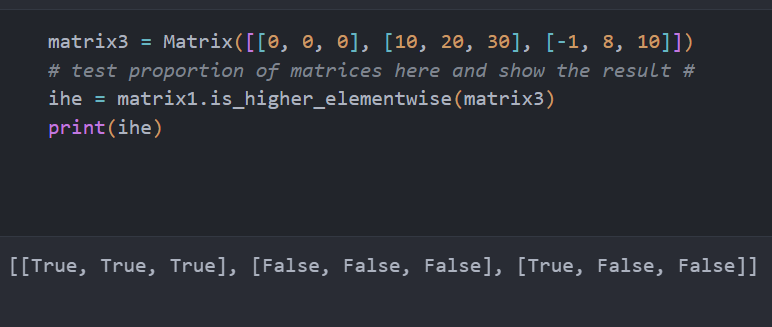
و به خروجی دلخواه رسیدیم :



برای تست بزرگتر یا کوچکتر بودن :



خروجی :



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):

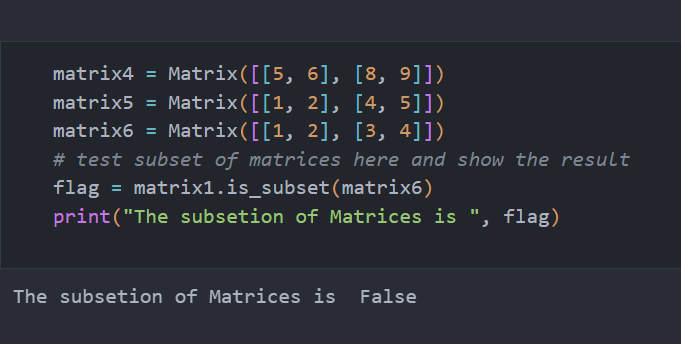
            for j in range(cols1 - cols2 + 1):

                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 :



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):

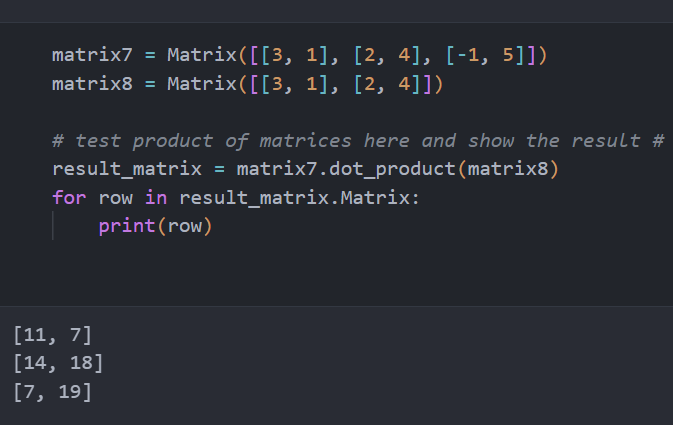
                    dot\_product += self.Matrix[i][k] \* second\_matrix.Matrix[k][j]

                row.append(dot\_product)

            result.append(row)

        return Matrix(result)

خروجی :



منابع مورد استفاده در تمرین : chat.openai