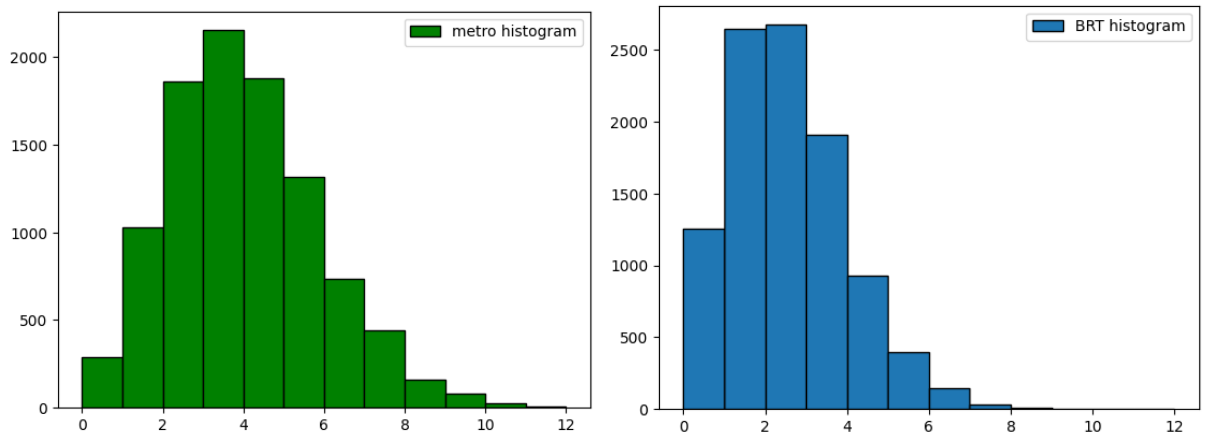
سوال 1:

بخش 1:

ابتدا با استفاده از کتابخانه پانداس دیتا ها رو خوندم سپس با استفاده از matplotlib هیستوگرام های مربوطه را به راحتی نمایش دادم. در نمودار ها همانطور که دیده می شود برای BRT نقطه اوج نمودار در 2 اتفاق می افتد و در مترو این نقطه در 3 است.



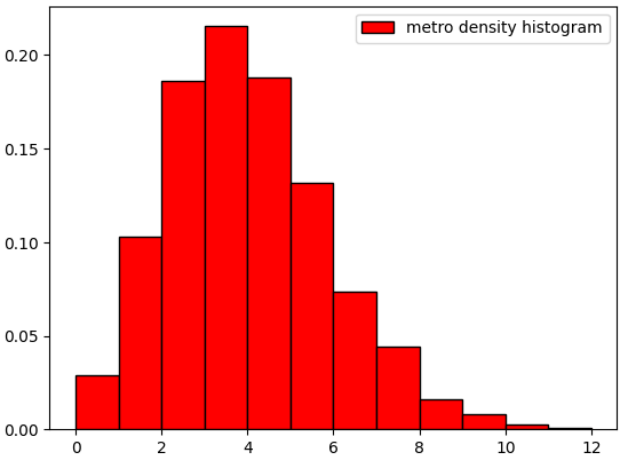

بخش 2:

هر دو تابع از توزیع پواسن پیروی می کنند که پارامتر توزیع پواسن برابر میانگین متغیر تصادفی است. در این بخش میانگین را حساب کردیم که می توانید طبق خروجی پارامتر های آن ها را ببینید

```
for BRT the poisson parameter is 2.0636
for metro the poisson parameter is 3.531
```

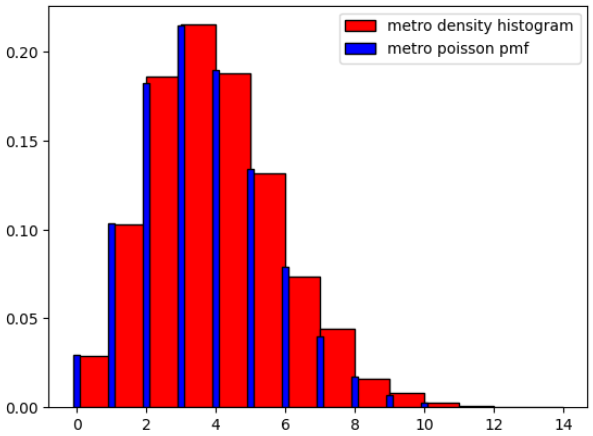
بخش 3:

برای اینکه بتوانیم `Density` را نشان دهیم کافی است فقط متغیر آن را در هیستوگرام مربوطه برابر true بگذاریم که در واقع تراکم را نمایش میدهد.

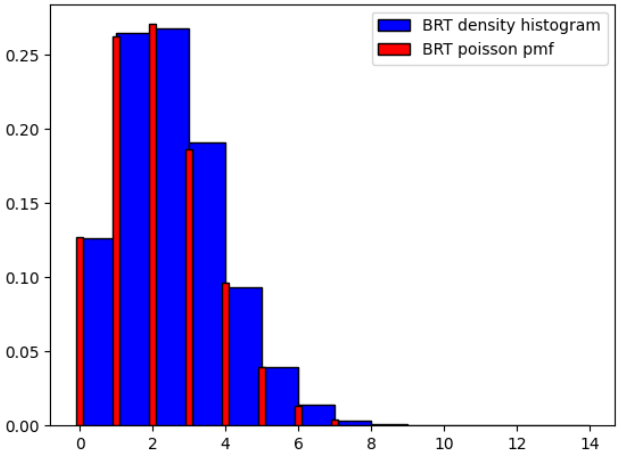


همانطور که در این نمودار انتظار می رفت چون تراکم را نشان می دهد عملا احتمال را در هر نقطه نشان می دهد و با هیستوگرام اولیه مترو تناقضی ندارد.

بخش 4: با استفاده از پواسن و توابع آن در scipy نمودار pmf توزیع پواسن برای مترو را کشیدم (با استفاده از پارامتر آن که در بالا در آوردم) و روی نمودار تراکمی که در بخش قبل قرار دادیم همانطور که دیده می شود بسیار خوب این توزیع را می توان با پواسن مدل کرد.




از طرفی برای این که اطمینان حاصل کنم که پواسن توزیع BRT را نیز به درستی مدل می کند برای این متغیر تصادفی هم این نمودار را رسم کردم و به همین نتیجه رسیدم.

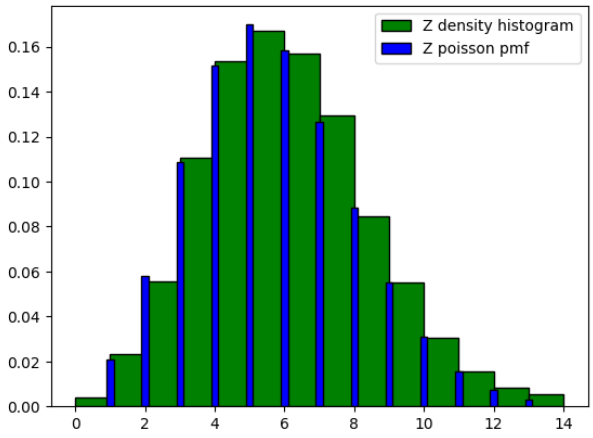



بخش 5:

همان طور که در درس خواندیم مجموع دو متغیر تصادفی مستقل که از توزیع پواسن پیروی می کنند نیز از توزیع پواسن با پارامتر مجموع میانگین آن ها پیروی می کند در این بخش درستی این ادعا را با استفاده از نمودار نشان می دهیم.

(در اینجا برای اینکه تمام مقادیر Z را داشته باشم بازه تست رو از صفر تا 12 به صفر تا 15 تبدیل کردم)

نمودار در صفحه بعد نمایش داده شده.



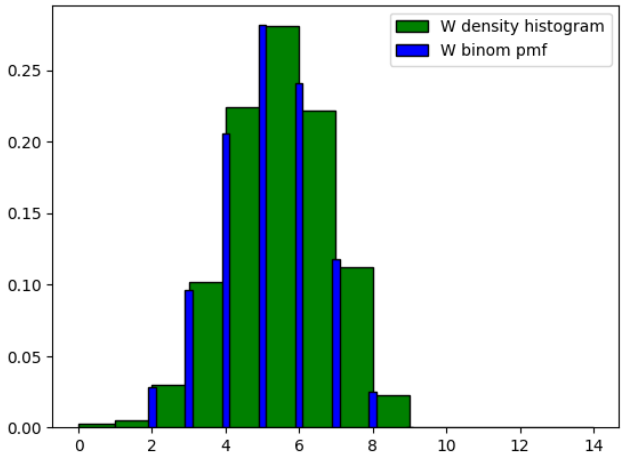

طبق نمودار به دست آمده همانطور که مشاهده می شود به طرز خوبی از توزیع پواسن با پارامتر مجموع میانگین ها پیروی می کند.

بخش 6:

این بخش در واقع از ما توزیع X به شرط Z را می خواهد که طبق چیزی که در درس خواندیم توزیع شرطی دو توزیع پواسن از توزیع دو جمله ای با پارامترn برابر مقدار فیکس Z که در سوال گفته 8 در نظر بگیریم و pبرابر تقسیم پارامتر ها به دست می آید.(در اینجا با تقسیم پارامتر(میانگین) X به پارامتر(میانگین Z) به دست می آید)

بخش7 و 8:

این دو بخش را با هم نوشتم. با انتخاب توزیع دو جمله ای به عنوان توزیع W (نمودار میله ای آبی) یک توزیعی از این متغیر تصادفی به دست آوردم و آن را روی نمودار تراکم(چگالی) این متغیر انداختم و همانطور که انتظار می رفت این دو نمودار به طور خوبی با هم تطابق دارند. (برای به دست آوردن مقادیر W ، مقادیر ایکسی را برداشتم که مجموع X , Y در آن ردیف برابر 8 باشند.)




(سوال 2 صفحه بعد)

سوال 2:

بخش 1:

برای این بخش یک تابع به شکل زیر تعریف کردم که یک مجموع تلاش ها داریم که به صورت آرایه ای تعریف شده از طرفی به ازای k بار آزمایش را انجام می دهیم به طوری که یک آرایه n تایی در نظر گرفتیم به طوری که هر اندیس آن نشانگر یک نوع کوپن است که ابتدا صفر بار دیده شده سپس تا وقتی که همه این کوپن ها را ببینیم رندوم یکی برمیداریم و تلاش ها را یک بار اضافه می کنیم. در انتها میانگین تعداد تلاش ها را برمیگردانیم که برابر مجموع تلاش هایی است که در آرایه ذخیره شده تقسیم بر k باری که آزمایش را تکرار کردیم.

```
def monte_carlo(n, k):
    total_tries = []
    for _ in range(k):
        tries = 0
        coupons = [0] * n
        while not all(c != 0 for c in coupons):
            tries += 1
            coupon = np.random.randint(0,n)
            coupons[coupon] += 1
        total_tries.append(tries)
    return sum(total_tries) / k
```

بخش 2:

به ازای داده های گفته شده چندین بار تابع را صدا کردیم که نتایج آن به صورت زیر بود.

با توجه به این خروجی ها جواب به سمت 29 میل میکند(با کمی ضریب خطا به نظر اینجوری می آید)

(به ازای k های بسیار بزرگ مثلا 100000 به سمت 29.3 میل می کند)

for n = 10 and k = 10 it returns 30.2

for n = 10 and k = 100 it returns 29.59

for n = 10 and k = 1000 it returns 28.879

سوال بین بخش دو و سه:

با توجه به توضیح در مورد Xi ها، این متغیر های تصادفی از توزیع هندسی پیروی می کنند. و پارامتر هر کدام از این متغیر ها اینگونه تعریف میشود که کوپن iم را برای اولین بار در یک انتخاب ببینیم در حالی که کوپن های قبلی را دیده باشیم که احتمال آن برابر p = (n – i + 1) / nاست. از طرفی برای اینکه در بار x ام این کوپن را ببینیم برابر

p(1-p)^(x-1) است. در بخش بعدی با استفاده از این رابطه که اسم آن را P(x) میگذاریم، تابع مولد گشتاور را محاسبه میکنیم.

بخش 3 ، 4، 5:

برای محاسبه تابع مولد گشتاور کافی است P(x) را به ازای هر x ممکن( 1 تا بی نهایت) در e^sx ضرب کنیم و سپس این ها را باهم جمع کنیم که سیگما مورد نظر برابر معادله زیر می شود که این در واقع تابع مولد گشتاور است.

(p * e^s) / (1 – (1 – p) * e^s)

که این رابطه چون p به i وابسته است، به i وابستگی دارد.

از آنجایی که می دانیم تابع مولد گشتاور یک متغیر که از جمع متغیر های دیگر به دست می آید برابر ضرب توابع مولد گشتاور این متغیر هاست، پس کافی است با توجه به ساده سازی هایی که سوال انجام داده برای یافتن تابع مولد گشتاور این سوال، برای i به ازای تمام مقادیرش (1 تا n)تابع مولد گشتاور را حساب کنیم و در هم ضرب کنیم. سپس با مشتق گرفتن از آن نسبت به s و جاگذاری s = 0 می توانیم امید ریاضی را حساب کنیم

حال با استفاده از کتاب خانه گفته شده معادله بالا را می نویسیم و محاسبات را روی آن انجام می دهیم.

```
i = sp.symbols('i', integer=True)
s = sp.symbols('s', real=True)
p = (n - i + 1) / n
mgf_Xi = (p * sp.exp(s)) / (1 - (1 - p) * sp.exp(s))
mgf_X = 1
for k in range(1, 11):
    mgf_X *= mgf_Xi.subs({i : k})
print(f"expectation is {sp.diff(mgf_X, s).subs({s:0})}")
```

در اینجا ابتدا دو متغیر i و s رو تعریف کردم سپس p را طبق فرمول بالا گه قبلا توضیح دادم به دست آوردم و با آن تابع مولد گشتاور برای هر Xi ساختم و در تابع مولد گشتاور اصلی که در ابتدا یک گذاشته بودم ضرب کردم. سپس با مشتق گرفتن از تابع گشتاور نهایی و قرار دادن s=0 ، جواب 7381/252 شد که تقریبا برابر 29.289 است که بسیار به جواب بخش دو نزدیک است همان طور که انتظار می رفت.

سوال 3:

بخش 1:

در این جا ابتدا فایل را با استفاده از کتابخانه پانداس خواندیم بعد آن را تبدیل به یک دیتا فریم کردیم و سپس خط های 201 و 202 طبق صورت سوال را ذخیره کردیم(چون ایندکس از صفر شروع می شد در واقع ایندکس های 200 و 201 منظور سوال بود) سپس با استفاده از تابع drop این دو سطر را از دیتافریم حذف کردیم.

```
df = pd.read_csv("digits.csv")
df = pd.DataFrame(df)
row_201 = df.iloc[200]
row_202 = df.iloc[201]
df.drop(200,axis=0, inplace=True)
df.drop(201,axis=0, inplace=True)
```

بخش 2:

در این سه خط ابتدا یک دیتافریم باینری تعریف کردم و دیتافریم اصلی رو در آن ذخیره کردم سپس خانه هایی که در ردیتا فریم اصلی (به غیر از لیبل) بیشتر یا مساوی 128 هستند را یک و در غیر این صورت برابر صفر قرار دادم.

```
binary_data = df.copy()
binary_data[df.iloc[:, df.columns != 'label'] >= 128] = 1
binary_data[df.iloc[:, df.columns != 'label'] < 128] = 0
```

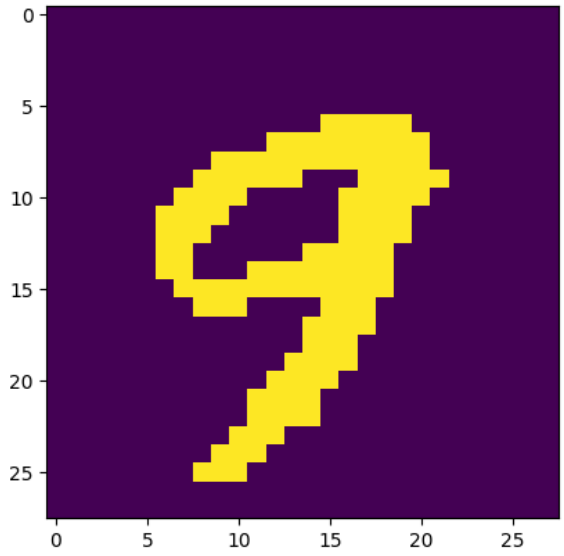
بخش 3:

ابتدا یک عدد رندوم بین صفر تا 199 که برابر شماره سطر منهای یک است انتخاب می کنیم. سپس اعداد آن سطر را به وسیله آرایه نامپای در یک آرایه یک بعدی ذخیره می کنیم و به وسیله reshape این آرایه را به یک آرایه دو بعدی 28 در 28 تبدیل می کنیم.در آخر هم با استفاده از plt.show آن را نمایش می دهیم. نتیجه کار هم بعد کد آورده شده.

```
random_number = np.random.randint(0, 200)
array1 = np.array(binary_data.iloc[random_number, 1:])
array2 = np.reshape(array1, (28,28))
print(random_number)
plt.imshow(array2)
```

دقت داشته باشید که چون مقدار هر سری رندوم انتخاب می شود خروجی شما با خروجی من ممکن است متفاوت باشد ولی به ازای عدد 169 خروجی به شکل زیر است.

random_number = 169




بخش 4:

در این بخش فقط کافی است مقادیر را در تابع آپدیت حساب کنیم طبق روابطی که به ما داده شده این تابع به این صورت در می آید.

```
def update(fy: np.array, n:bool) -> np.array:
    p = np.linspace(0,1,t)
    # calculate P(N = n| Y = p) which is a bernouli distribution
    # calculate integral(0 -> 1) fy * pny
    pny = stats.bernoulli.pmf(n, p)
```

```
    integral = np.sum(fy * pny) / t
    post = fy * pny / integral
    return post
```

که ابتدا احتمال pny را به وسیله تابع pmf آن به دست آوردیم سپس با استفاده از رابطه داده شده برای انتگرال ، حاصل انتگرال مخرج را به دست آوردیم سپس با جا گذاری این مقادیر در رابطه داده شده به جواب نهایی رسیدیم.

پس از دیدن تمام صد نمونه دیتا احتمال ما به 0.67 می رسد که در واقع این همان میانگین یا مد توزیع بتا است.