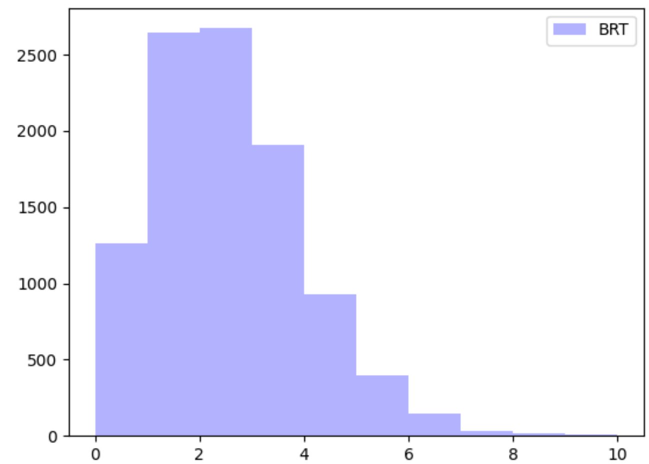
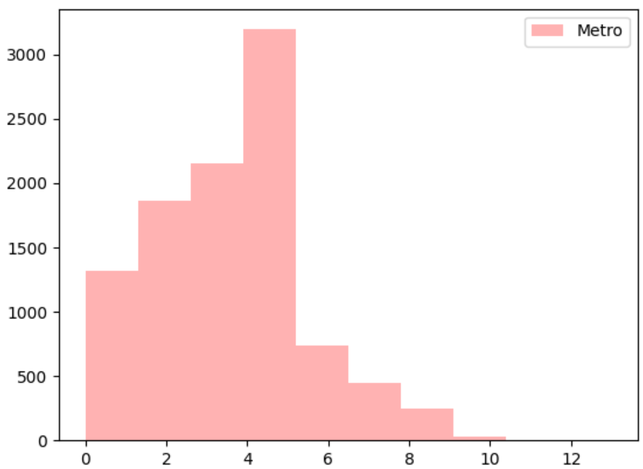
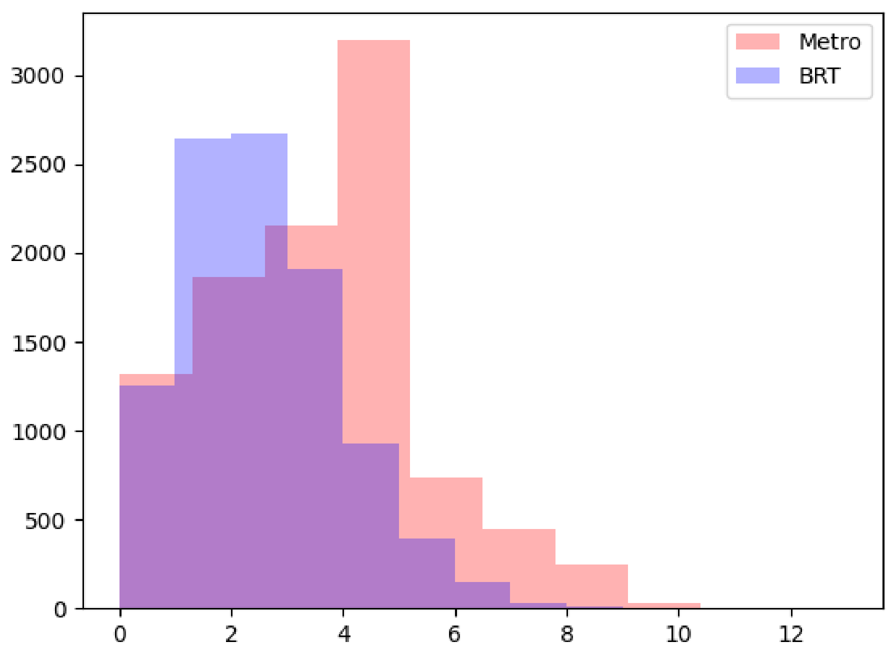
سوال ۱:

بخش ۱: داده ها با استفاده از کتابخانه pandas از فایل خوانده می شوند و سپس از ستون را که مربوط به مترو و BRT هستند را جهت محاسبات آینده جداسازی کرده و در دو لیست جدا ذخیره می کنیم.

با استفاده از کتابخانه matplotlib بخش hist نمودار هیستوگرام داده های مربوط به مترو و BRT را رسم می کنیم که نمودار به شکل زیر است.







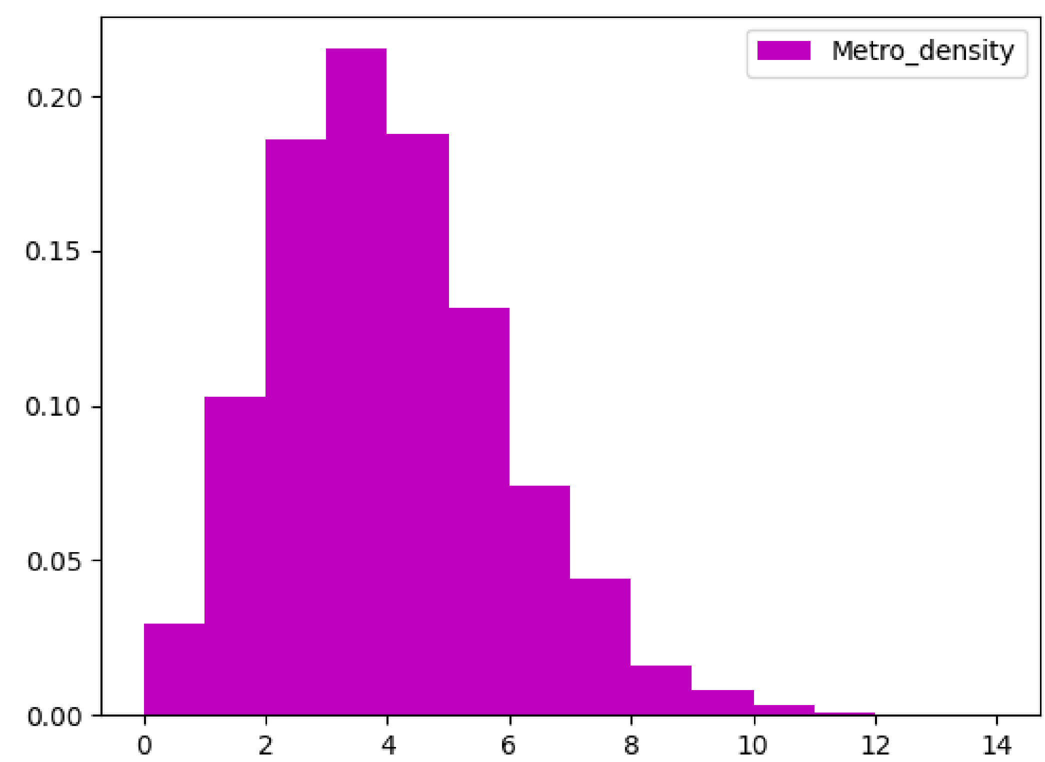
نمودار قرمز مربوط به داده های مترو و نمودار آبی مربوط به داده های BRT می باشد.

بخش ۲: با توجه به شکل نمودار های بالا، می توان در نتیجه گرفت که توزیع مربوط به گذر های مترو و BRT توزیع پواسون هستند. برای محاسبه λ هر توزیع پواسون، کافی است واریانس یا میانگین داده ها را محاسبه کنیم. چون همان طور که می دانیم در توزیع پواسون میانگین و واریانس باهم برابر هستند و مقدار آنها برابر پارامتر توزیع پواسون یعنی λ است.

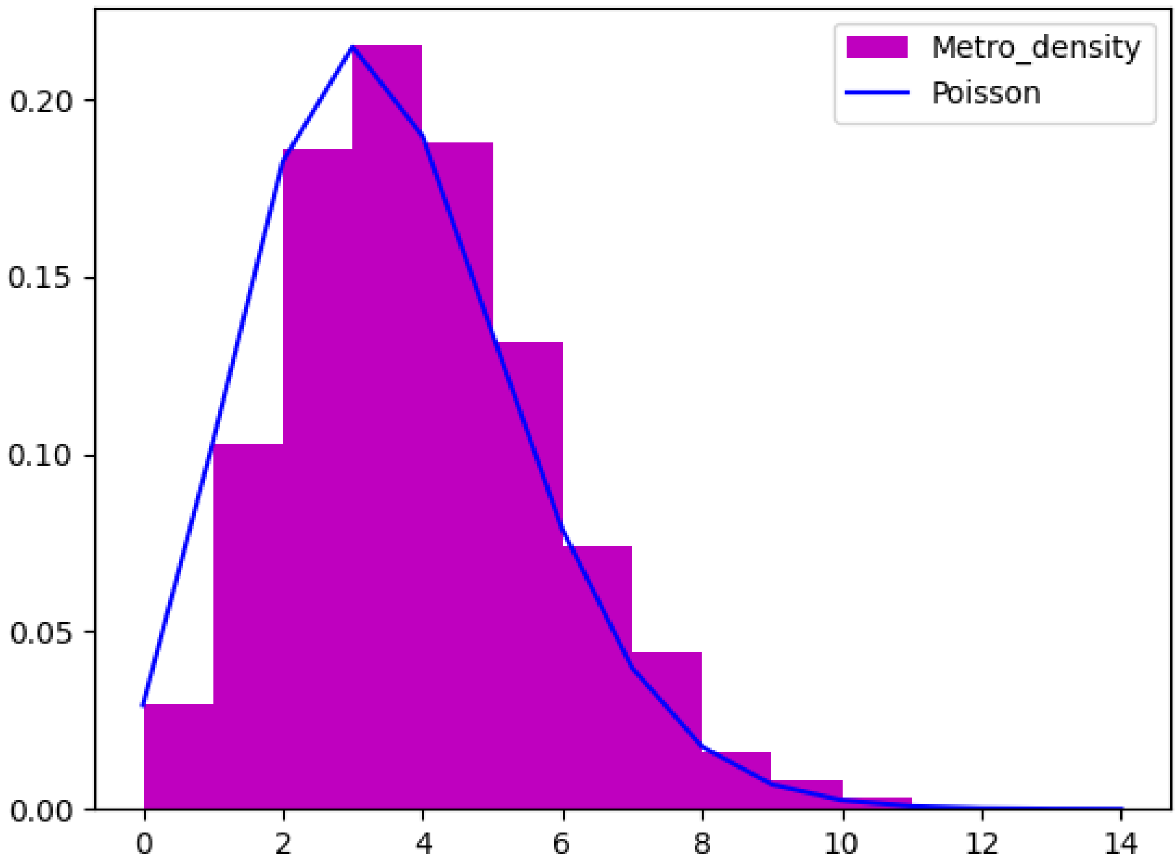
در اینجا به کمک dataframe پایتون و تابع mean میانگین گذر های مترو و BRT محاسبه شده است که نتایج به صورت زیر است:

```
paramters of poisson distribution:
 metro:  3.5316  BRT:  2.0636
```

بخش ۳: برای رسم density histogram کافی است پارامتر density در hist را برابر مقدار true قرار دهیم و نمودار را رسم کنیم. تنها تفاوت آن با هیستوگرام قبلی، مقدار محور عمودی است. نمودار به شکل زیر خواهد بود.



بخش ۴: با توجه به محاسبات انجام شده به این نتیجه رسیدیم که توزیع مربوط به X پواسون با پارامتر تقریبا ۳.۵ است. با استفاده از کتابخانه scipy و بخش poisson ابتدا بازه داده ها را مشخص می کنیم. این بازه با داده های dataframe و ستون مترو قابل محاسبه است. این بازه از مینیمم داده ستون مترو تا ماکزیمم داده ستون مترو خواهد بود. این بازه را به تابع poisson.pmf می‌دهیم تا بتوان نمودار آن را رسم کرد. همچنین λ را برابر مقدار محاسبه شده قرار می دهیم. جهت بررسی راحت تر توزیع پواسون را به شکل plot رسم می کنیم. نمودار به شکل زیر است.



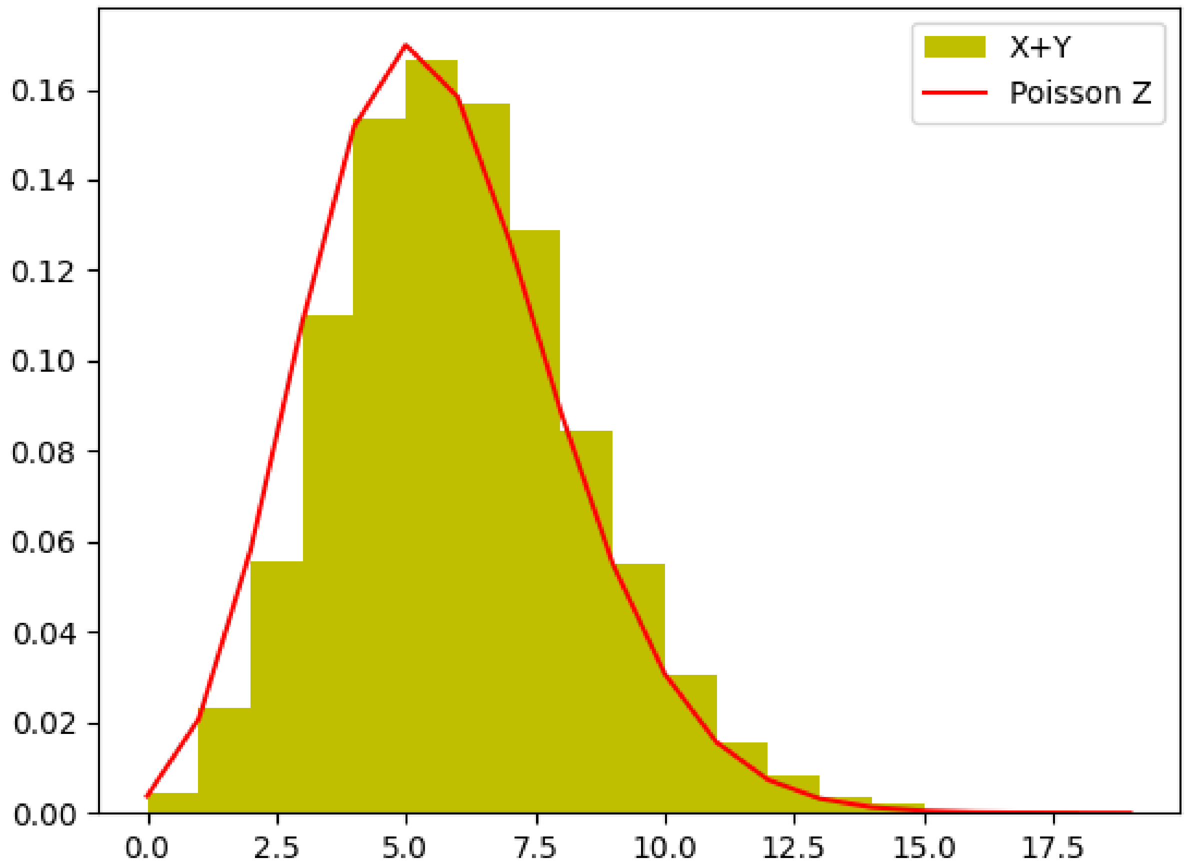
همان طور که واضح است نمودار توزیع پواسون با λ = ۳.۵ تا حد خوبی بر هیستوگرام داده های متغیر تصادفی X منطبق است. بنابراین داده ها به درستی با توزیع پواسون مدل شده اند.

بخش ۵: با توجه به اینکه X و Y از هم مستقل هستند و هر دو دارای توزیع پواسون هستند، می دانیم متغیر تصادفی Z که حاصل جمع آنهاست هم دارای توزیع پواسون خواهد بود که پارامتر آن برابر حاصل جمع پارامتر های X و Y خواهد بود. $\lambda_Z = \lambda_X + \lambda_Y$

که برابر ۵.۵۹ است.

برای راحت تر شدن محاسبات و جلوگیری از استفاده از حلقه های for داده های مربوط به مترو و BRT را که در لیست پایتون ذخیره کردیم به آرایه numpy تبدیل می کنیم. چون امکان جمع کردن آرایه های numpy مانند ماتریس وجود دارد. بنابراین تنها

با جمع کردن n1 + n2 به داده های Z می رسیم. به کمک این داده ها هیستوگرام مربوط به متغیر تصادفی Z را می کشیم. با توجه به اینکه به این نتیجه رسیدیم که توزیع این متغیر تصادفی پواسون است مانند بخش های قبلی ابتدا بازه نتایج را که برابر مینیمم و ماکزیمم داده های Z است مشخص کرده و به λ محاسبه شده به تابع poisson.pmf می دهیم و در نهایت به شکل plot رسم می کنیم. نتیجه به شکل زیر است:



همان طور که از نمودار مشخص است به درستی توزیع پواسون برای حاصل جمع انتخاب شده است و تا حد خوبی پواسون با λ محاسبه شده با هیستوگرام داده های مربوط به متغیر تصادفی Z منطبق است.

بخش ۶:

توزیع X و X+Y هردو پواسون هستند با پارامتر های متفاوت. همان طور که می دانیم فرمول محاسبه احتمال یا pmf به شکل زیر است:

$$P_x(k) = \frac{e^{-\lambda}\lambda^k}{k!}$$

بنابراین در فرمول احتمال شرطی، برای هر توزیع از فرمول بالا استفاده می کنیم و محاسبات و ساده سازی ها را انجام می دهیم که به صورت زیر است:

$$w \sim p(X \mid X+Y=n) = \frac{P(X=x,\ Y=n-x)}{P(X+Y=n)} \xrightarrow{\text{استقلال}} = \frac{P(X=x)\,P(Y=n-x)}{P(X+Y=n)}$$

$$= \frac{\dfrac{e^{-\lambda_x}\lambda_1^{x}}{x!} \times \dfrac{e^{-\lambda_y}\lambda_y^{(n-x)}}{(n-x)!}}{\dfrac{e^{-(\lambda_x+\lambda_y)}(\lambda_x+\lambda_y)^{(n)}}{n!}} = \frac{n!}{x!(n-x)!} \times \frac{\lambda_x^{x}\lambda_y^{n-x}}{(\lambda_x+\lambda_y)^{n}} = \binom{n}{x}\left(\frac{\lambda_x}{\lambda_x+\lambda_y}\right)^{x}\left(\frac{\lambda_y}{\lambda_x+\lambda_y}\right)^{n-x}$$

$$\Rightarrow (X \mid X+Y=n) \sim Bin\left(n, \frac{\lambda_x}{\lambda_x+\lambda_y}\right)$$

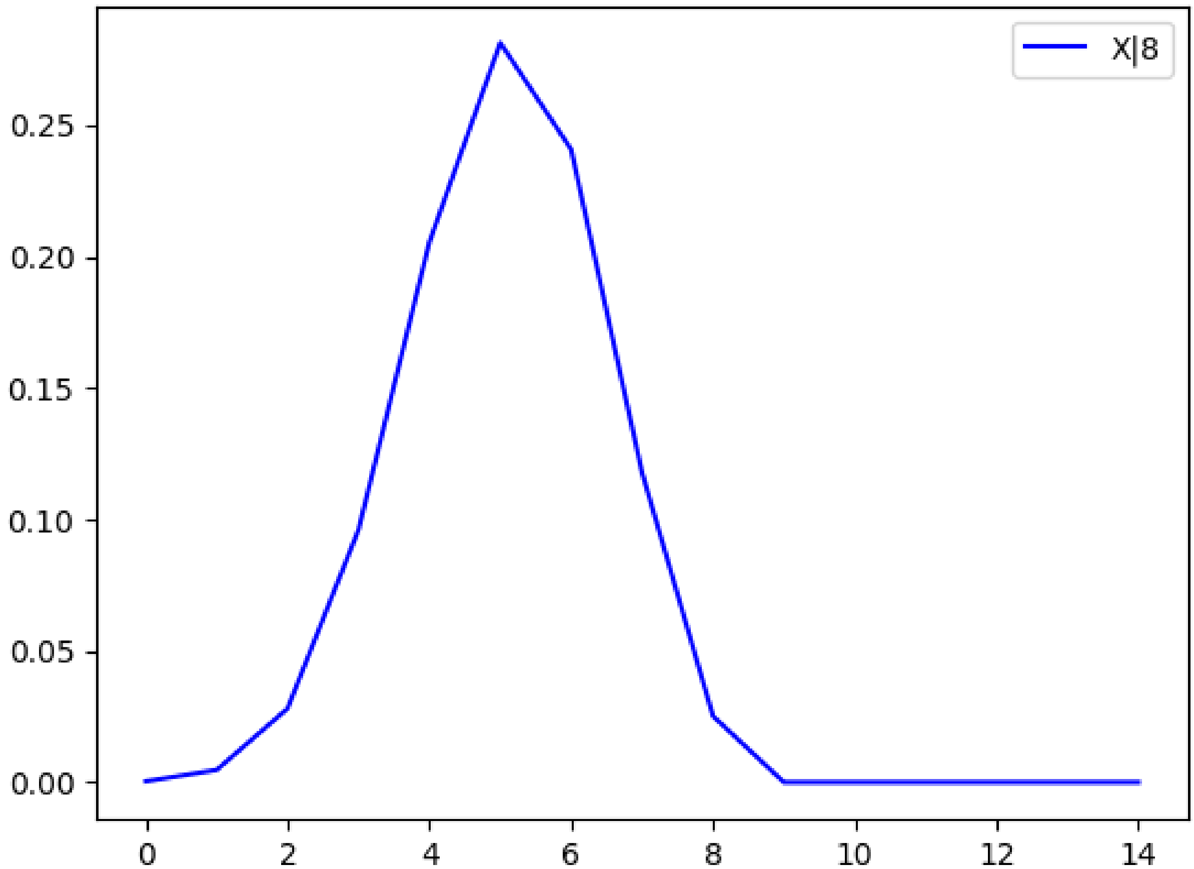
بنابراین در نهایت پس از ساده سازی به توزیع دوجمله ای می رسیم چراکه فرمول pmf توزیع دو جمله ای به فرم زیر است:

$$P_x(k) = \binom{n}{k} p^k q^{n-k}$$

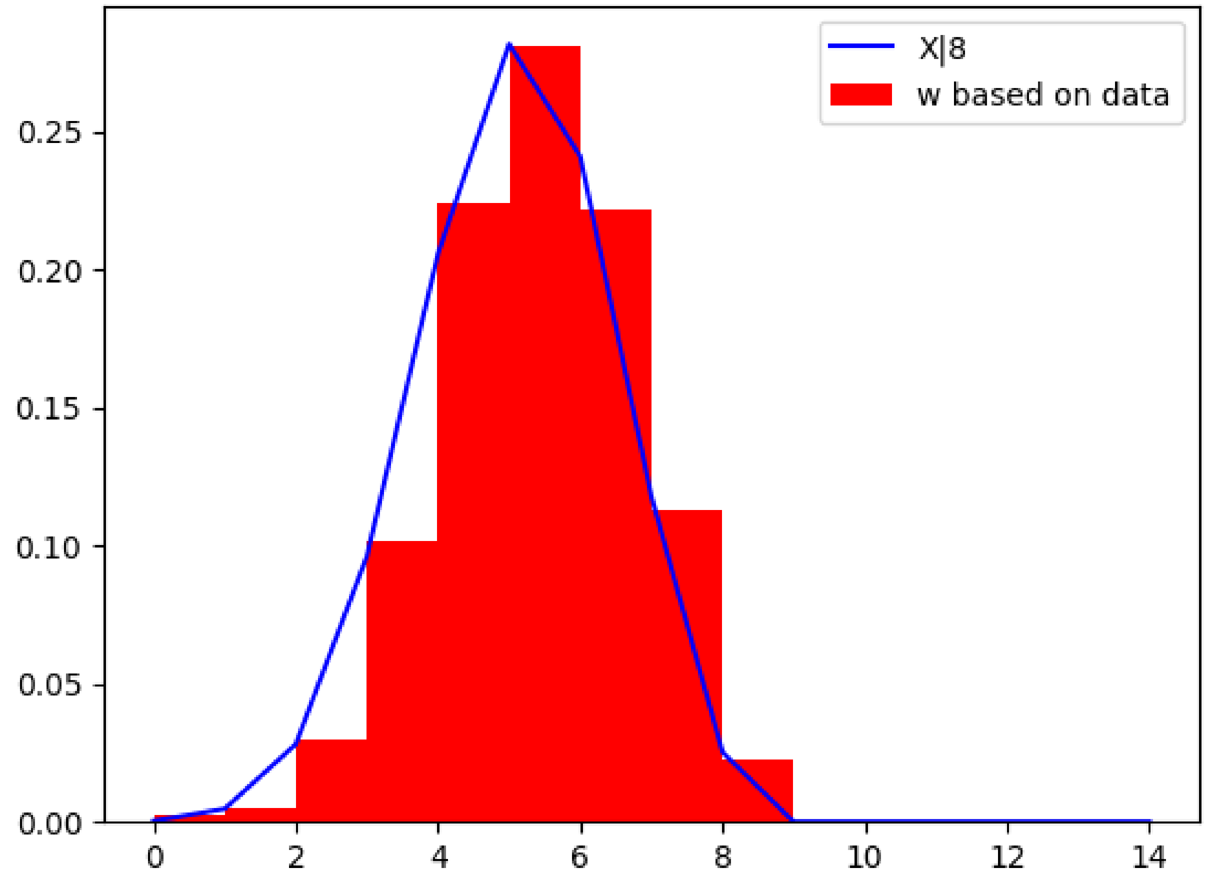
پس از انجام محاسبات پارامتر های توزیع W به شکل زیر خواهند بود:

$$w \sim Bin(n, 0.63)$$

بخش ۷: همان طور که گفته شد توزیع مربوطه، توزیع دو جمله ای است و پارامتر های آن را در بخش قبلی محاسبه کردیم. حال در اینجا مقدار n ثابت است و باید به ازای تمام مقدار x نمودار را رسم کنیم. بنابراین تمام مقادیر x را به binom.pmf می دهیم و نمودار را به صورت plot رسم می کنیم. نتیجه به شکل زیر خواهد بود.



بخش ۸: در اینجا به کمک dataframe به طور عملی محاسبه می کنیم که در کجاها مجموع مترو و BRT برابر ۸ شده و مقدار مربوط به مترو را ذخیره می کنیم. سپس این داده ها را به فرم هیستوگرام رسم می کنیم و نتیجه به شکل زیر است. همان طور که در نمودار مشخص است، توزیعی که برای این متغیر تصادفی در نظر گرفته بودیم با نمودار هیستوگرام ناشی از محاسبه خواسته سوال به صورت عملی منطبق است.



سوال۲:

بخش۱: دراین بخش قصد داریم مسئله را به روش عملی حل کنیم. با توابع *random.redit* اعداد رندوم در بازه ۱ تا *n* دریافت می کنیم و تا زمانی که همه اعداد را ندیده ایم به این برداشتن تصادفی ادامه میدهیم. در نهایت تعداد دفعاتی که عدد تصادفی برداشته ایم را در لیستی ذخیره کرده و در نهایت از آن میانگین می گیریم.

بخش ۲: به ازای *k*های مختلف این کار را انجام می دهیم. با توجه به اینکه در هر دفعه در حال برداشتن رندوم هستیم با هر بار اجرا کردن برنامه خروجی های متفاوتی خواهیم داشت. اما به طور کلی میانگین دفعاتی که باید انجام دهیم تا همه کوپن ها را ببینیم به عدد ۲۹ میل می کند. برای *k* ها به ترتیب داریم:

```
28.2
30.15
29.349
```

بخش ۳: تابع مولد گشتاور اینگونه است:

$$\phi_X(s) = \mathrm{E}\left(e^{sX}\right) = \sum_i e^{sx_i} P_X(x_i)$$

برای هر متغیر تصادفی *xi* توزیع آن هندسی است بنابراین به جای *p*از توابع مربوط به توزیع هندسی استفاده می کنیم. احتمال دیده شدن کارت جدید هربار متفاوت خواهد بود چون کارت را بر میگردانیم و ممکن است به ازای برداشتن یک کارت تعداد کارت های متمایز دیده شده عوض شود. بنابراین هر مرحله احتمال موفقیت برابر رابطه زیر است:

$$p_i = \frac{n-(i-1)}{n}$$

برای جلوگیری از توابع طولانی از فرمول مستقیم استفاده میکنیم:

$$\phi_X(s) = \sum_{k=1}^{\infty} (1-p)^{k-1} p e^{ks} = pe^s \sum_{m=0}^{\infty} \left((1-p)e^s\right)^m = \frac{pe^s}{1-(1-p)e^s}$$

بنابراین تابع مولد گشتاور برای *xi* برابر زیر است:

```python
#3
def moment_generating_Xi(i):
    p = (10-i+1)/10
    return (p*sympy.exp(s))/(1-(1-p)*sympy.exp(s))
```

با توجه به اینکه *s* را یک نماد تعریف کرده ایم خروجی این تابع یک *expression* خواهد بود.

بخش ۴: همان طور که در مسئله گفته شد مشکل را به چندین بخش کوچکتر و مستقل تبدیل کردیم که هر کدام توزیع هندسی داشتند. حال برای محاسبه تابع مولد گشتاور *x* که حاصل جمع *xi* هاست باید از رابطه زیر استفاده کنیم:

- اگر دو متغیر تصادفی X و Y مستقل باشند و $Z = X + Y$ باشد، داریم:

$$\phi_Z(s) = \phi_X(s)\phi_Y(s)$$

بنابراین توابع مولد گشتاور *xi* ها را باید در هم ضرب کنیم تا به تابع مولد نهایی برسیم بنابراین تابع مولد گشتاور *x* به شکل زیر است:

```python
#4
def moment_generating_X(n):
    fi_X = moment_generating_Xi(1)
    for i in range(2,n+1):
        fi_X *= moment_generating_Xi(i)
    return fi_X
```

بخش ۵: میدانیم اگر بخواهیم از روی تابع مولد گشتاور به میانگین یا *E[X]* برسیم داریم:

1) قضیه گشتاور:

$$m_n = E(X^n) = \Phi^{(n)}(0)$$

یعنی مشتق nام تابع مولد در نقطه صفر، گشتاور nام را می‌دهد.

بنابراین برای به دست آوردن میانگین، باید از رابطه ای که در بخش قبل به دست اوردیم مشتق بگیریم و متغیر *s* را برابر با صفر قرار دهیم.

```
#5
moment_generating_function_X = moment_generating_X(10)
expectation_function = sympy.diff(moment_generating_function_X, s)
print("expectation = ", expectation_function.subs({s : 0}))
```

در صورت اجرای این کد به ازای *n = 10* خروجی به شکل زیر خواهد بود:

```
28.098
expectation =  29.2896825396826
```

بنابراین طبق محاسبات میانگین برابر ۲۹ است که به بخش ۲ که به صورت عملی حساب کردیم برابر است.

سوال ۳:

بخش ۱: ابتدا فایل *csv* را میخوانیم. با توجه به اینکه ستون اول مربوط به *label* است باید به ایندکس هایی که میخواهیم یکی اضافه کنیم. بنابراین داده هایی که در ستون های ۲۰۲ و ۲۰۳ قرار دارند را در متغیر ریخته و سپس *drop* می کنیم. پارامتر *inplace* را به این علت درست گذاشتیم که پس از حذف از دیتافریم کپی نگیرد بلکه همان دیتافریم قبلی را تغییر دهد.

```
#1
data = pd.read_csv("digits.csv")
index_201 = data.iloc[:,202]
index_202 = data.iloc[:,203]
data.drop(data.columns[[202,203]], axis=1, inplace=True)
```

با این کد ابتدا داده های این دو ستون را ذخیره کرده و سپس حذف می کنیم.

بخش ۲: برای تبدیل داده ها به باینری، تمام داده هایی را که کمتر مساوی ۱۲۸ باشد را صفر و بزرگتر ها را یک می کنیم. برای این کار روش های مختلفی وجود دارد که یکی از آنها لامبدا است که با بررسی شرط روی ستون ها داده های آن را عوض می کند. از انجایی که ستون *label* بدون تغییر باید بماند، ایندکس ستون ها را از ۱ شروع کرده ایم تا داده های این ستون تغییری نکند.

```
threshold = 128
for i in range(1,len(data.axes[1])):
    data.iloc[:, i] = data.iloc[:, i].apply(lambda u : 0 if u <= threshold else 1)
```

بخش ۴: می دانیم فرمول احتمال توزیع یکنواخت اینگونه است:

اگر در توزیع بتا $a = b = 1$ به توزیع یکنواخت می رسیم:

$$f_X(x) = \begin{cases} \dfrac{1}{\beta(a,b)} x^{a-1}(1-x)^{b-1} & 0 < x < 1 \\ 0 & \text{otherwise} \end{cases}$$

چراکه تمام ترم ها یک می شوند و $\beta(a,b)$ برابر $b - a$ می شود پس به توزیع یکنواخت رسیدیم. چون n برنولی است و y یکنواخت بنابراین ترکیب شرطی هم متغیر پیوسته هم گسسته دارد :