گزارش:

سوال 1 :

بخش اول:

-در قسمت ابتدایی ما کتابخانه هایی در طی پروژه لازم داریم را import میکنیم

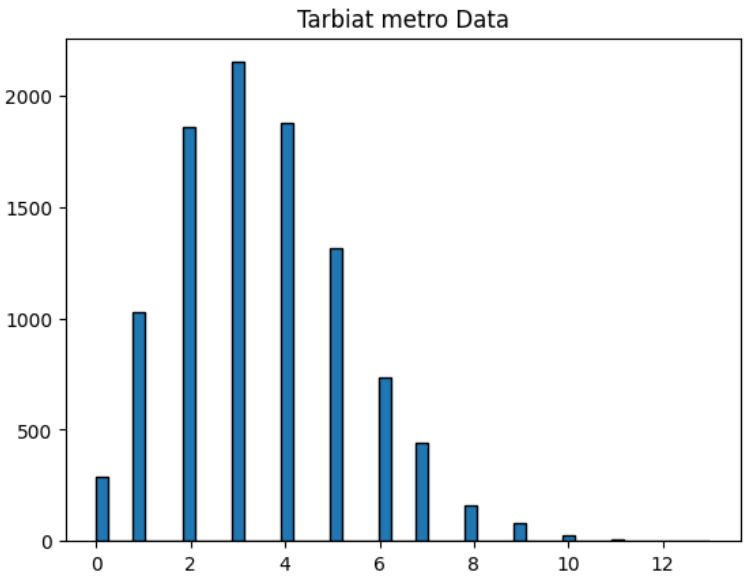
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from scipy.stats import norm
import math
from scipy.stats import poisson
from scipy.stats import binom
```

-در این قسمت در ابتدا باید فایل csv را بخوانیم

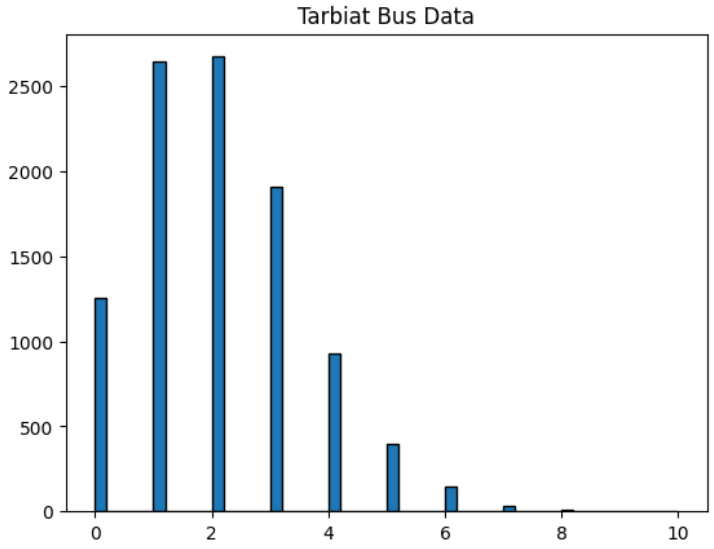
```
df=pd.read_csv('Tarbiat.csv')
```
✓ 0.0s

-حال برای هر کدام از داده ها هیستوگرام رسم میکنیم

```
y = df['metro']
plt.title("Tarbiat metro Data")
plt.hist(y,bins=50,edgecolor="black")
plt.show()
```
✓ 0.2s



```
y = df['BRT']
plt.title("Tarbiat Bus Data")
plt.hist(y,bins=50,edgecolor="black")
plt.show()
```
✓ 0.2s



بخش دوم:

-با توجه به نمودار ها و همچنین طبق مطالب کلاس با توجه به زیاد بودن دیتا میتوان گفت این متغیر ها از توزیع پواسون پیروی میکند که پارامتر لاندا در هر کدام برابر است با:

-همانطور که میدانیم در نوزیع پواسون پارامتر لاندا برابر میانگین و واریانس است که در اینجا نیز این دو تقریبا یکسان شده اند

```
metroaverage=np.average(df['metro'])
print(metroaverage)
metrovar=np.var(df['metro'])
print(metrovar)
```
✓ 0.0s

```
3.5316
3.60320144
```

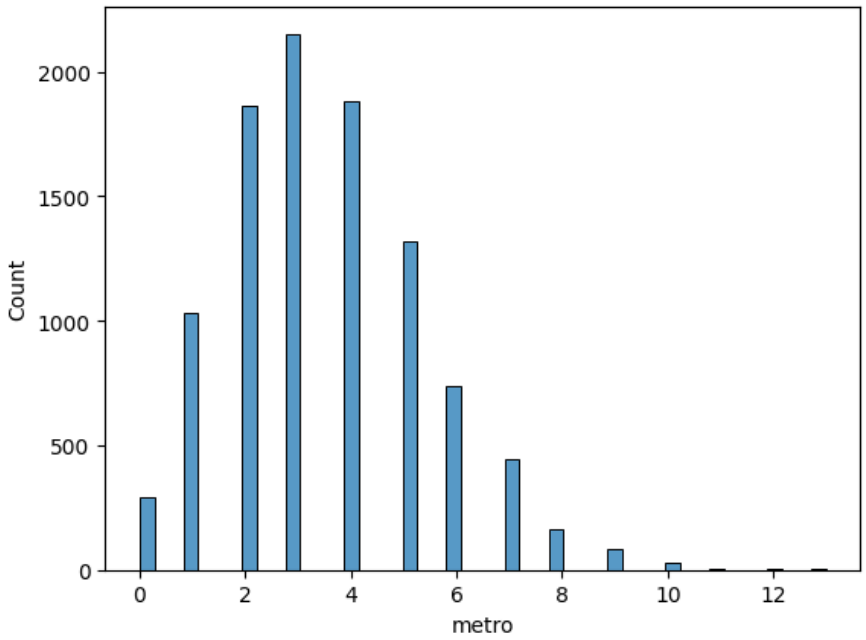
```
metroaverage=np.average(df['BRT'])
print(metroaverage)
metrovar=np.var(df['BRT'])
print(metrovar)
```
✓ 0.0s

```
2.0636
2.0669550400000003
```

بخش سوم:

– برای این کار از seaborn استفاده میکنیم

```
y = df['metro']
sns.histplot(y)
plt.show()
```
✓ 0.3s



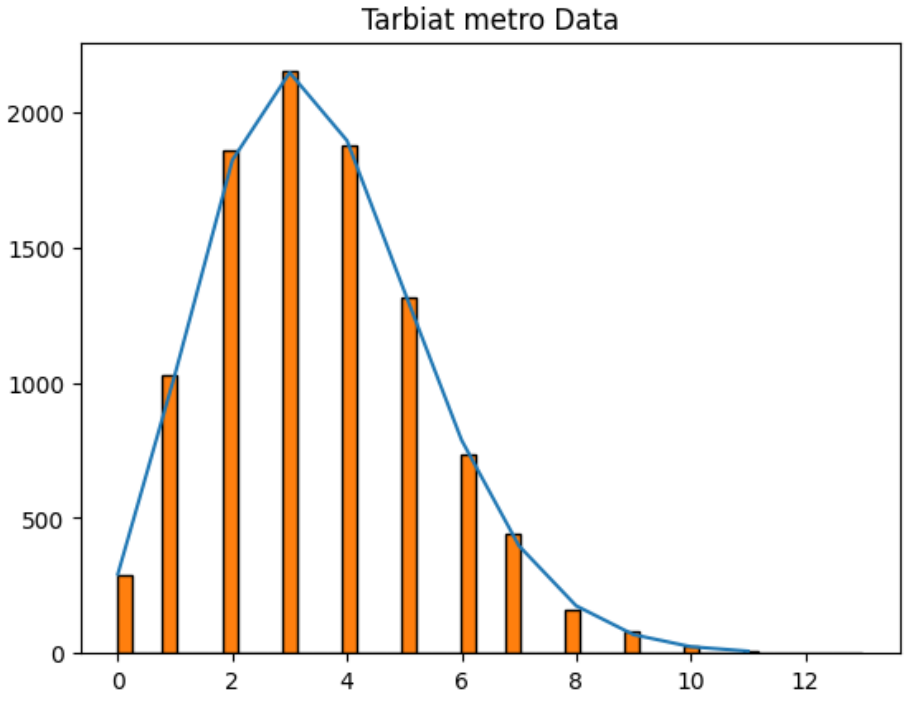
بخش چهارم:

-با توجه به نمودار های زیر میتوان از نرمال بودن توضیع مطمعن شد

```
x = np.arange(12)
y = poisson.pmf(x, mu=3.5316, loc=0)
plt.plot(10000*y)

y2 = df['metro']
plt.hist(y2,bins=50,edgecolor="black")

plt.title("Tarbiat metro Data")
plt.show()
```
✓ 0.3s



بخش پنجم:

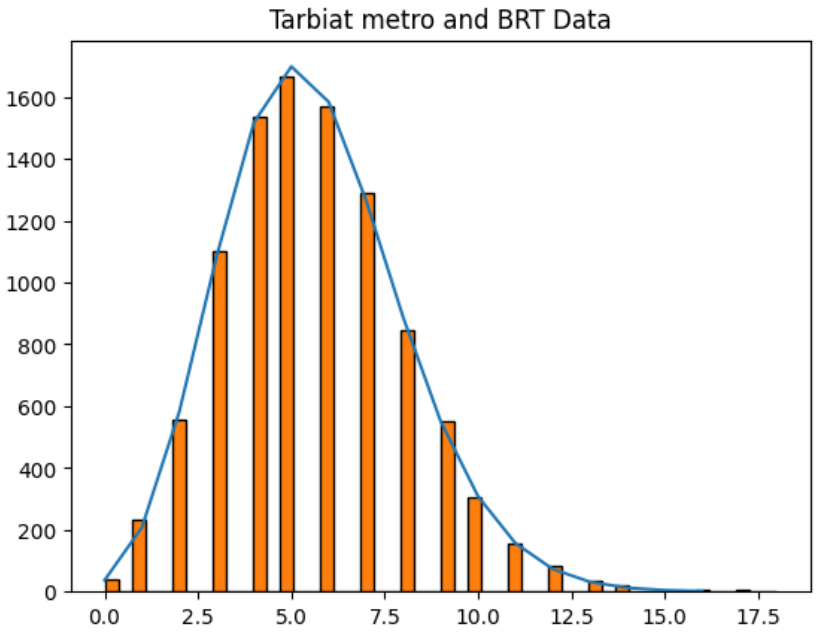
– از انجا که x و y ار هم مستقل هستند بنابر این توزیع z نیز یک توزیع پواسون است که پارامتر لاندا ان برابر مجموع پارامتر های x و y خواهد بود که این را در نمودار پایین میتوان دید

```
x = np.arange(17)
y = poisson.pmf(x, mu=3.5316+2.0636, loc=0)
plt.plot(10000*y)

yx = df['metro']
yy = df['BRT']
plt.hist(yx+yy,bins=50,edgecolor="black")

plt.title("Tarbiat metro and BRT Data")
plt.show()
```

✓ 0.4s



بخش ششم:

-با توجه به مطالب گفته شده در کلاس توزیع w یک توزیع دوجمله ای با پارامتر های n و landa(x)/(landa(x)+landa(y)) بنابر این خواهیم داشت

که پارامتر p در این توزیع دوجمله ای برابر است با:

```
p=3.5316/(3.5316+2.0636)
p
```

✓ 0.0s

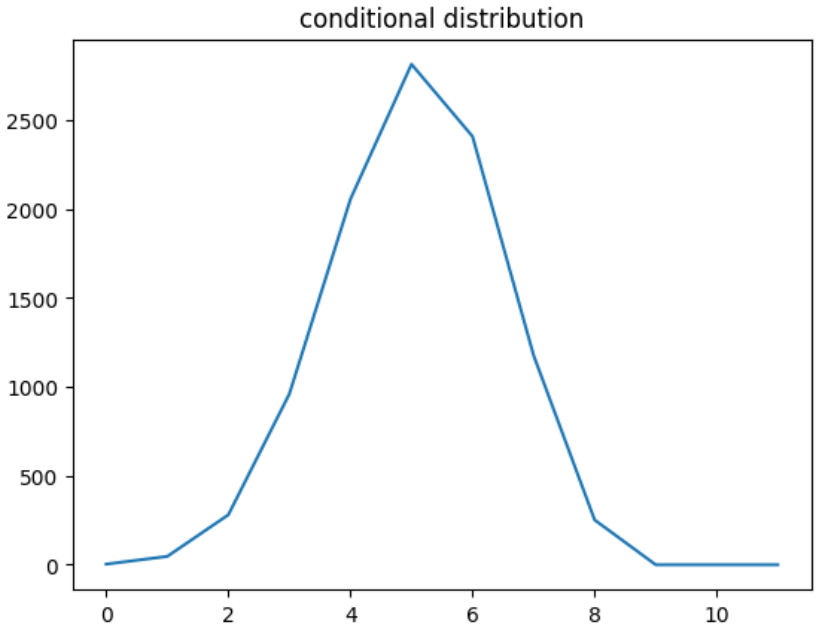
```
0.6311838718901915
```

بخش هفتم:

– خواهیم داشت:

```
x = np.arange(0,12,1)
y = binom.pmf(x,n=8,p=0.6311838718901915)
plt.plot(10000*y)
plt.title("conditional distribution")
plt.show()
```

✓ 0.2s



بخش هشتم:

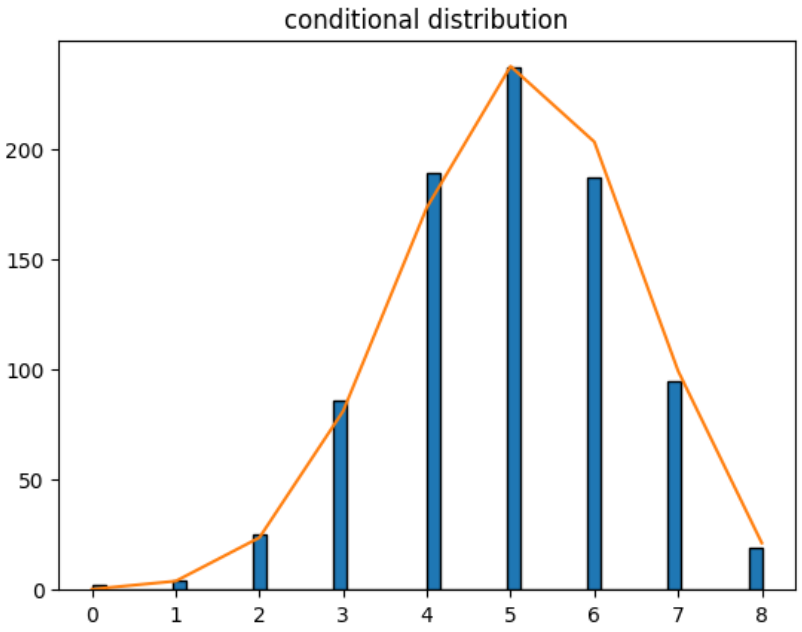
-همانطور که در نمودار پایین مشاهده میکنید توزیع w با همانظور که در سوال قبلس حدس زده بودیم یک توزیع دوجمله ای اس

```
metro=np.array([])
for i in range (0,10000):
    if(df['metro'][i]+df['BRT'][i]==8):
        metro=np.append(metro,df['metro'][i])
plt.hist(metro,bins=50,edgecolor="black")

x = np.arange(9)
y = binom.pmf(x,n=8,p=0.6311838718901915)
plt.plot(len(metro)*y)

plt.title("conditional distribution")
plt.show()
```

✓ 0.3s



سوال 2:

بخش اول:

-در قسمت ابتدایی ما کتابخانه هایی در طی پروژه لازم داریم را import میکنیم

```
import sympy
import numpy as np
import random
```

✓ 0.0s

-در قسمت بعدی تابع را میسازیم

```
def monte(n):
    counter=0
    check=[]
    for i in range(n):
        check=check+[0]
    tedada_type_dide_shode=0
    endcheck=False
    while(not endcheck):
        counter+=1
        x=random.randint(0,n-1)
        if (check[x]==0):
            check[x]=1
            tedada_type_dide_shode += 1
            if(tedada_type_dide_shode==n):
                endcheck=True

    return counter

def monte_avg(n,k):
    sum=0
    for i in range(0,k):
        sum += monte(n)
    return sum/k
```

✓ 0.0s

بخش دوم:

```
print(monte_avg(10,10))
print(monte_avg(10,100))
print(monte_avg(10,1000))
```

✓ 0.0s

```
26.0
27.41
29.312
```

-همانطور که میبینید این اعداد به ۲۹.۳ میل میکند

بخش سوم:

-هر کدام از xi ها دارای توضیع هندسی هستند و از انجا که هر بار پس از مشاهده i کالابرگ مختلف احتمال دیدن کالابرگ جدید برابر (N-i)/N میشود بنابر این برای به دست اوردن تابع مولد هر کدام از xi ها میتوانیم تابع زیر را تعریف کنیم

```
def get_mgf(i,n):
    s = sympy.symbols('s')
    p=(n-(i))/n
    MGF=(p*sympy.exp(s))/(1-(1-p)*sympy.exp(s))
    return MGF
```

✓ 0.0s

بخش چهارم:

-با توجه به اینکه هر کدام از xi ها از هم مستقل هستند پس تابع مولد X برابر با حاصل ضرب تابع مولد هر xi میشود

```
MGFx=1
for i in range(10):
    MGFx*=get_mgf(i,10)
```

✓ 0.0s

بخش پنجم:

-همانطور که مشاهده میکنید در اینجا نیز پاسخ مانند قسمت دو تقریبا برابر با 29.3 شده است

سوال ۳:

بخش ۱:

-در قسمت ابتدایی ما کتابخانه هایی که لازم داریم را import میکنیم

```
import pandas as pd
import numpy as np
```

-حال کافیست اعداد ۲۰۱ام و ۲۰۲ام را در متغیرهایی ذخیره کنید و سپس از دیتافریم حذف کنید.

```
a=df[200:201]
b=df[201:202]
newdf = df[0:200]
```

بخش۲:

-قرار دادن threshold:

```
pixels = newdf.iloc[:, 1:]

def change_value_to_binary(element):
    if element > 128:
        return 1
    else:
        return 0

binarydf = pixels.map(change_value_to_binary)
```

بخش ۳:

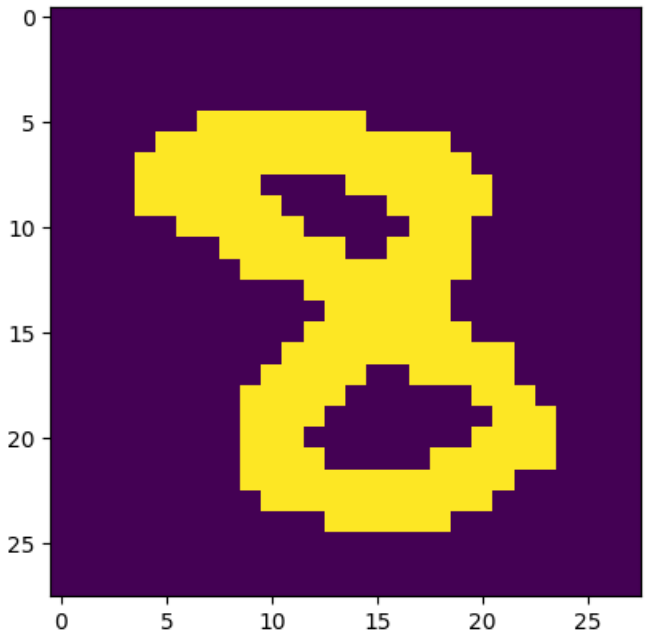
-نمایش یک sample :

```
sample = binarydf.sample(1)
sample_np = sample.to_numpy()
sample_np =sample_np.reshape((28,28))
```

-حال ان را نمایش میدهیم:

```
from matplotlib.pyplot import imshow

imshow(sample_np)
```



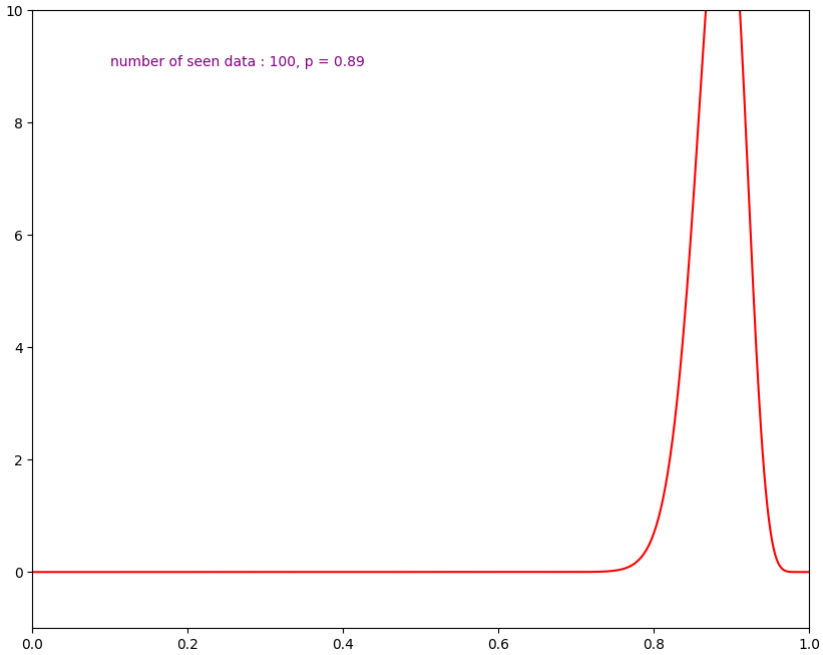
بخش ۴:

-کامل کردن قسمت خالی کد:

```
    pnyfy = []
    for i in range(len(p)):
        y = p[i]
        if n:
            pny = y
        else:
            pny = 1-y
        pnyfy.append(pny * fy[i])

    pnyfy = np.array(pnyfy)
    integral = 0
    for i in range(len(p)):
        integral += (1/t) * pnyfy[i]

    post = pnyfy / integral
    return post
```



با تشکر از توجه شما ☺