به نام خدا

تمرین

کامپیوتری دوم

درس: آمار و احتمال مهندسی

مجید صادقی نژاد
۸۱۰۱۰۱۴۵۹

- توضیحات

کد اصلی پروژه google colab نوشته شد و آدرس آن در زیر آمده است پیشنهاد می شود برای تجربه بهتر کد در همین محیط اجرا شود هر چند که فایل ها .ipynb , .py که ذخیره شده از همین پروژه در colab هستند هم در ضمیمه فایل های تحویل شده آمده اند **نکته قابل توجه است که در صورت اجرا کد در google colab در ابتدا باید تمامی فایل csv تمرین را در آن محیط بارگذاری کنید ...**

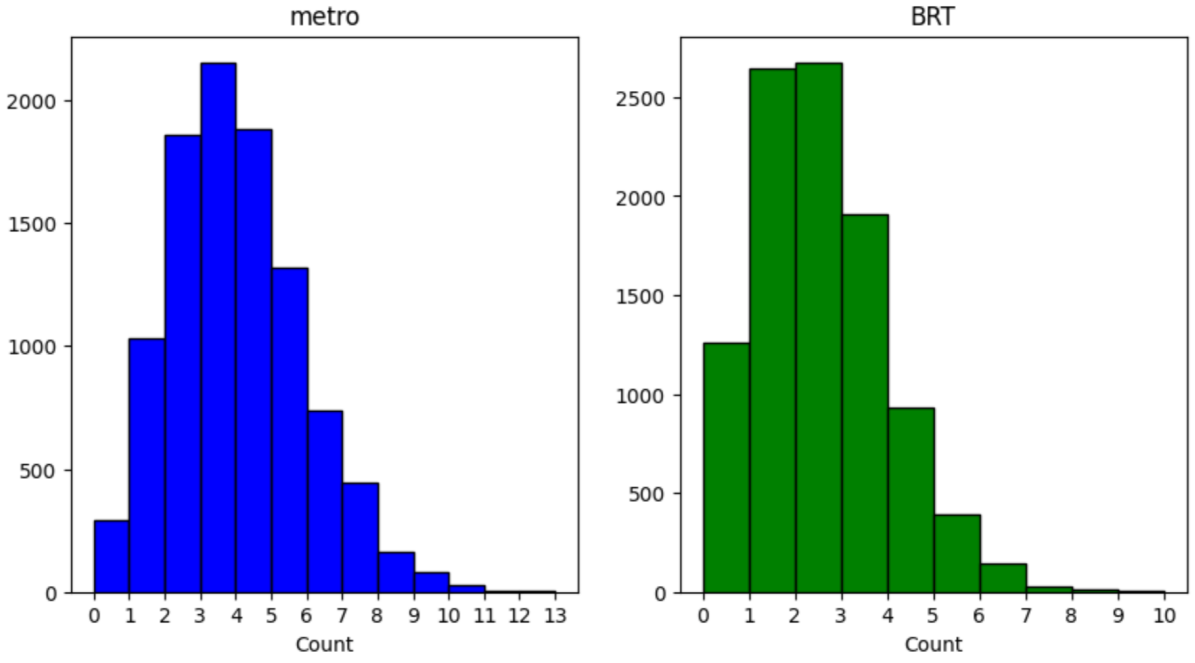
لینک پروژه: https://colab.research.google.com/drive/۱۷RJMfenb۸Ftl_RaBD۳myt-۹Cl۴vcRKZe?usp=sharing

این تمرین با استفاده از ۱ روز Grace تحویل داده شد.

- توزیع شرطی:

- سوال اول:

با استفاده از کتابخانه ی pandas فایل های csv را خوانده و سپس با استفاده از کتابخانی matplotlib هیستوگرام دو ستون مترو و بی آر تی را رسم می کنیم:



که کد آن به شکل زیر است :

```python
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('Tarbiat.csv')

metro = df['metro']
BRT = df['BRT']

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.hist(metro, bins=metro.max()-metro.min(), color='blue', edgecolor='black')
plt.title('metro')
plt.xlabel('Count')
plt.xticks(range(int(metro.min()), int(metro.max()) + 1))

plt.subplot(1, 2, 2)
plt.hist(BRT, bins=BRT.max()-BRT.min(), color='green', edgecolor='black')
plt.title('BRT')
plt.xlabel('Count')
plt.xticks(range(int(BRT.min()), int(BRT.max()) + 1))

plt.show()
```

- **سوال دوم:**

با توجه به این که متغیر های زمانی ما به فرمت تعداد رخداد در یک بازه ی زمانی خاص و همچنین گسسته هستند پس از توزیع پواسون پیروی می کنند و از آنجا که پارامتر پواسون برابر با میانگین داده ها می باشد: **پارامتر پواسون مترو = ۳.۵۳ پارامتر پواسون بی آر تی : ۲.۰۶**

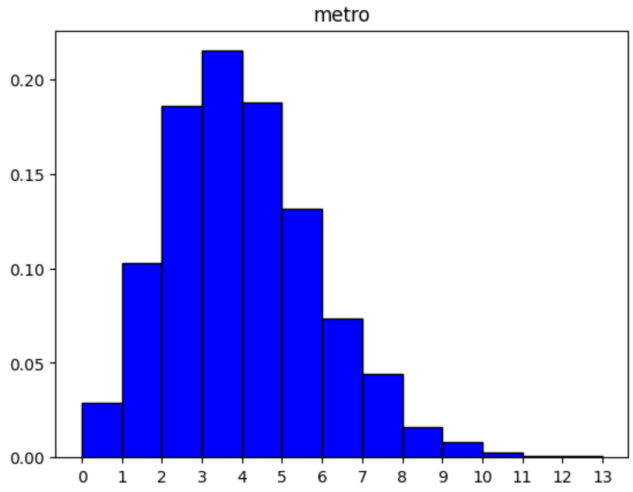
```python
import numpy as np

metroAverage = np.mean(metro)
BRT_Average = np.mean(BRT)

print(metroAverage)
print(BRT_Average)
```

```
3.5316
2.0636
```

- **سوال سوم:**

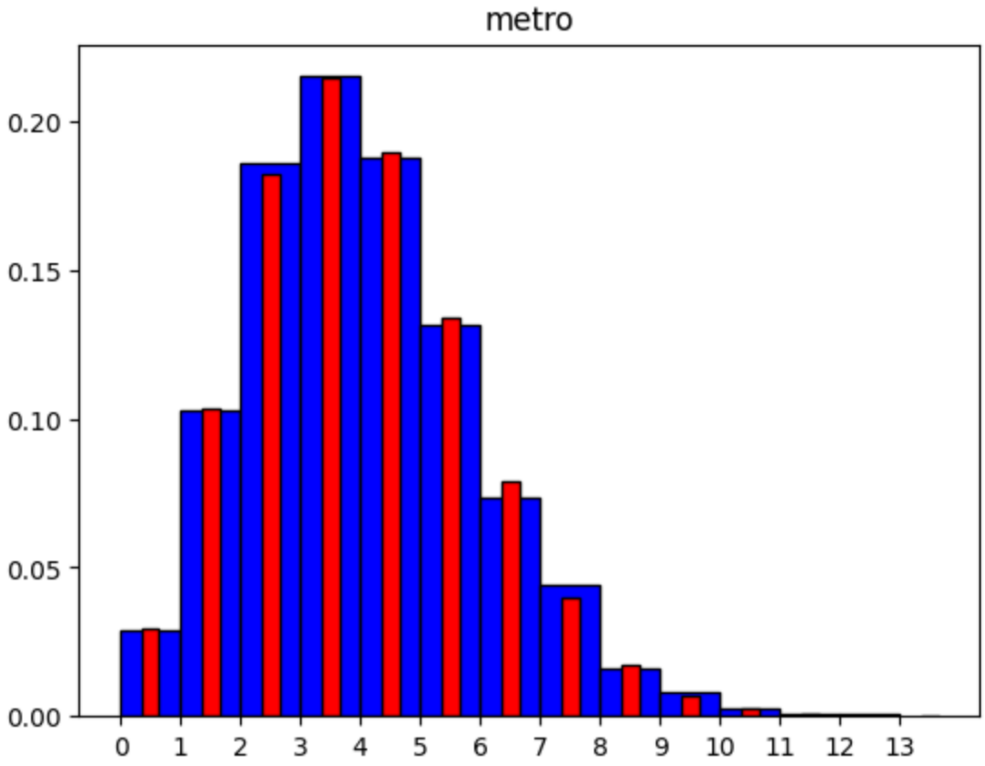
نمودار چگالی رسم شده برای ستون مترو و کد آن به شکل زیر است:

```python
plt.hist(metro, bins=metro.max()-metro.min(), color='blue', edgecolor='black' , density=True)
plt.xticks(range(int(metro.min()), int(metro.max()) + 1))

plt.title('metro')

plt.show()
```



- **سوال چهارم:**

با استفاده از کتابخانه ی scipy نمودار توزیع پواسون (قرمز) با پارامتر میانگین که در قسمت قبل محاسبه شد را بر روی هیستوگرام چگالی داده های مترو (آبی) رسم می کنیم مشاهده می شود بر هم منطبق :

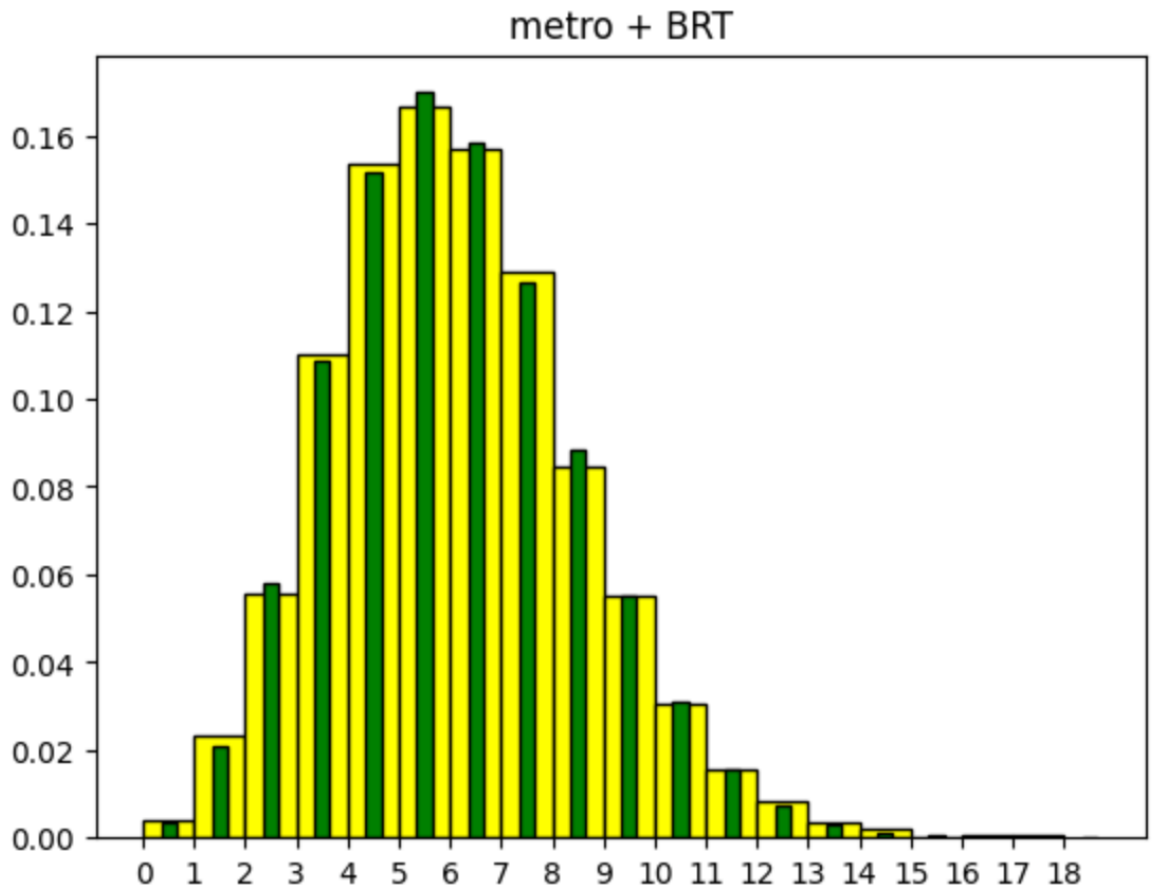


کد این قسمت :

```
from scipy.stats import poisson

x = np.arange(int(metro.min()), int(metro.max()) + 1)

plt.hist(metro, bins=metro.max()-metro.min(), color='blue', edgecolor='black' , density=True)
plt.bar(x+0.5, poisson.pmf(x, metroAverage), color='red', edgecolor='black', width= 0.3)
plt.xticks(x)

plt.title('metro')

plt.show()
```

- **سوال پنجم:**

در درس دیدیم مجموع دو توزیع پواسون توزیعی پواسون دارد و پارامتر آن برابر با مجموع دو توزیع پواسون جمع شده است بنابراین همانند بخش قبل نمودار توزیع پواسون با پارامتر جمع پارامتر های توزیع مترو و بی آرتی (سبز رنگ) و همچنین هیستوگرام جمع داده های مترو و بی آرتی (زرد رنگ) را نمایش می دهیم مشاهده می شود که منطبق هستند:



کد این قسمت نیز به شکل زیر است :

```
x = np.arange(int((metro+BRT).min()), int((metro+BRT).max()) + 1)

plt.hist(metro+BRT, bins=(metro+BRT).max()-(metro+BRT).min(), color='yellow', edgecolor='black' , density=True)
plt.bar(x+0.5, poisson.pmf(x, metroAverage + BRT_Average), color='green', edgecolor='black', width= 0.3)
plt.xticks(x)

plt.title('metro + BRT')

plt.show()
```

- **سوال ششم:**

$$P(X|X+Y=n) = \frac{P(X, X+Y=n)}{P(X+Y=n)} = \frac{P(X+Y=n|X=k)\,P_X(k)}{P(X+Y=n)}$$

$$= \frac{P(Y=n-k)\,P_X(k)}{P_Z(n)} = \frac{P_Y(n-k)\,P_X(k)}{P_Z(n)}$$

$$\Rightarrow P(X|X+Y=n) = \frac{e^{-\lambda_Y} \times \frac{\lambda_Y^{n-k}}{(n-k)!} \times e^{-\lambda_X} \times \frac{\lambda_X^{k}}{k!}}{e^{-(\lambda_Y+\lambda_X)} \times \frac{(\lambda_Y+\lambda_X)^n}{n!}}$$

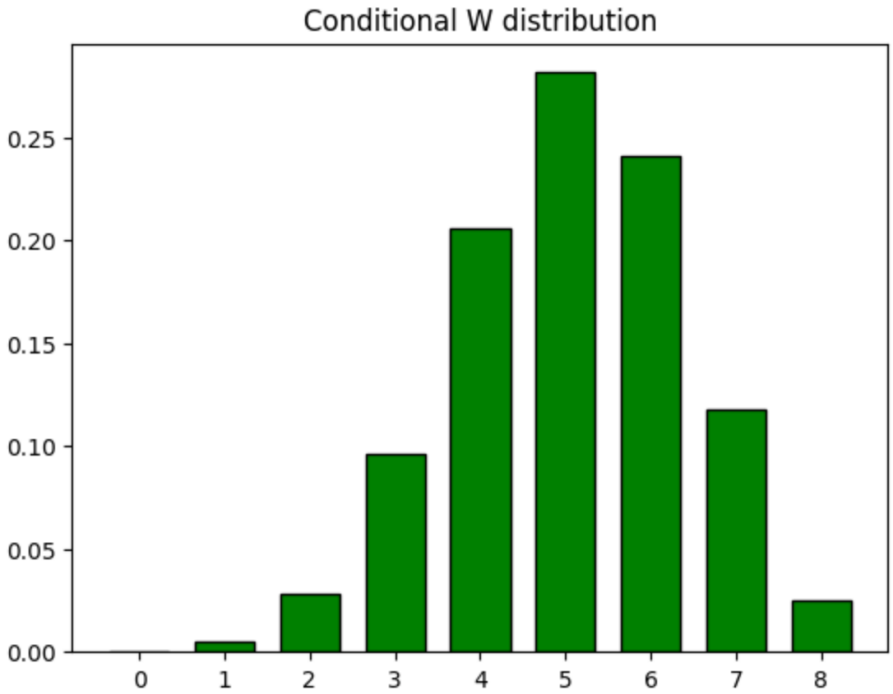
$$= \frac{\lambda_Y^{n-k}}{(n-k)!} \times \frac{\lambda_X^{k}}{k!} \times \frac{n!}{(\lambda_Y+\lambda_X)^n} = \frac{n!}{k!\,(n-k)!} \times \frac{\lambda_Y^{n-k} \times \lambda_X^{k}}{(\lambda_Y+\lambda_X)^n} = \frac{\binom{n}{k}\lambda_Y^{n-k} \times \lambda_X^{k}}{\sum_{i=0}^{i=n}\binom{n}{i}\lambda_X^{i}(\lambda_Y)^{n-i}}$$

$$\Rightarrow P_W(w|n) = \frac{\binom{n}{w}\lambda_Y^{n-w}\lambda_X^{w}}{\sum_{i=0}^{i=n}\binom{n}{i}\lambda_X^{i}(\lambda_Y)^{n-i}} \qquad \Rightarrow W \sim newDistrb(n)$$

همانطور که در محاسبات بالا مشاهده می شود توزیع w یک توزیع جدید می شود که به پارامتر n بستگی دارد.

- **سوال هفتم:**

تصویر تابع جرمی احتمال توزیعی که در سوال قبل به دست آماده به شکل زیر است:

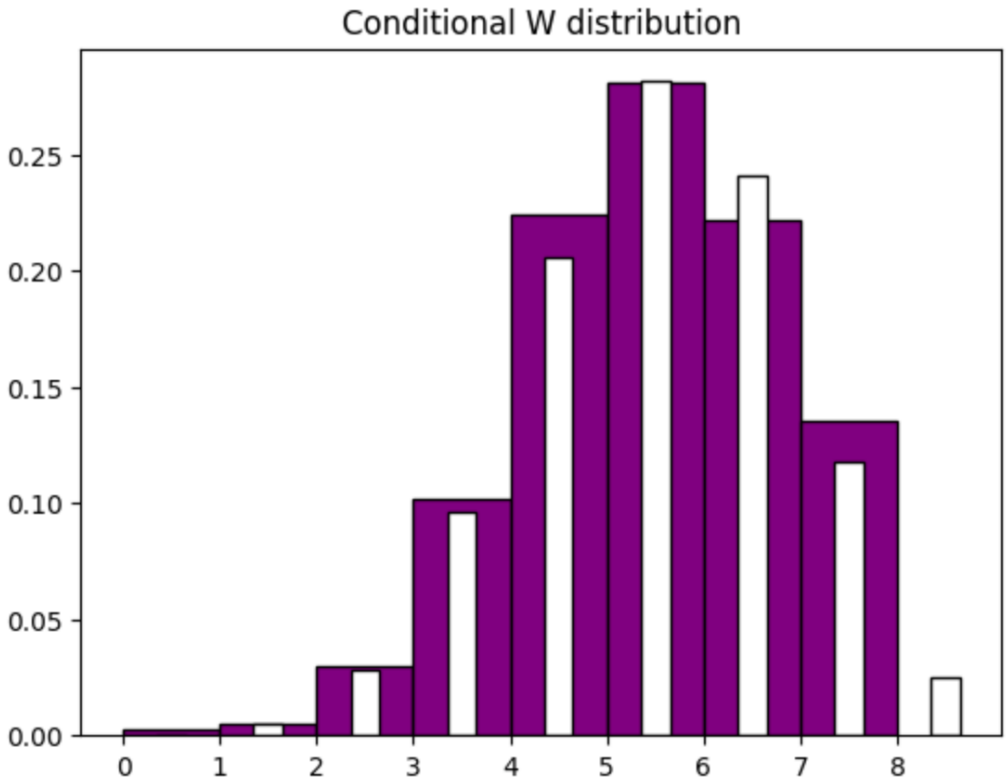


همچنین کدی که برای محاسبه مقادیر این تابع بر اساس محاسبات سوال قبل نوشته شده است به شکل زیر است:

```
from scipy.special import comb

n = 8
x = np.linspace(0,n,n+1)

def newDistrbFunc(n,k):
  return (comb(n, k) * (metroAverage ** k) * (BRT_Average ** (n-k)))

def wDisturb(w,n):
  base = 0
  for i in range(0,n+1,1):
    base += newDistrbFunc(n,i)
  return (newDistrbFunc(n,w)/base)


plt.bar(x, wDisturb(x, n), color='green', edgecolor='black', width= 0.7)
plt.xticks(x)
plt.title('Conditional W distribution')

plt.show()
```

- **سوال هشتم:**

حال با رسم هیستوگرام از داده هایی که با توجه به شرط مسئله استخراج شده (نمودار بنفش) و مقایسه آن با تابع توزیع جرمی احتمال به دست آمده در سوال قبل (نمودار سفید) به تطابق نسبی این دو شکل پی می بریم:



همچنین کد این بخش بدین شکل است:

```
from scipy.special import comb

n = 8
x = np.linspace(0,n,n+1)

def newDistrbFunc(n,k):
  return (comb(n, k) * (metroAverage ** k) * (BRT_Average ** (n-k)))

def wDisturb(w,n):
  base = 0
  for i in range(0,n+1,1):
    base += newDistrbFunc(n,i)
  return (newDistrbFunc(n,w)/base)


condition_result = df['metro'] + df['BRT'] == n

result_list = df.loc[condition_result, 'metro']

plt.hist(result_list, bins=(result_list).max()-(result_list).min(), color='purple', edgecolor='black' , density=True)
plt.bar(x+0.5, wDisturb(x, n), color='white', edgecolor='black', width= 0.3)
plt.xticks(x)
plt.title('Conditional W distribution')

plt.show()
```

- تابع مولد گشتاور:

سوال اول و دوم :

تابع خواسته شده در صورت سوال بدین نحو پیاده شده است که با استفاده از یک تولید کننده عدد رندوم که نماد کوپن ها هستند عدد تولید کرده و در داخل مجموعه قرار می دهیم از آن جا که مجموعه عضو تکراری نمی پذیرد تا هنگامی که تعداد اعضای مجموعه به تعداد نوع کوپن نرسیده این عمل را انجام می دهیم و تعداد انجام این عمل را می شماریم و در نهایت با یک حلقه با ازای k بار اینکار را انجام داده و در هر با تعداد انجام عمل را ذخیره کرده و در نهایت از آن میانگین میگیرم و مقدار میانگین را باز می گردانیم در زیر حاصل تابع به ازای n=۱۰ و k=۱۰, k=۱۰۰, k= ۱۰۰۰ آمده است که همانطور که مشاهده می شود تعداد مورد نظر مسئله به سمت **عدد ۲۹** همگرا می شود :

```python
import random

def montoCarlo(n,k) :
  answers = list()
  for i in range(0,k,1):
    setOfCoupons = set() ;
    times = 0 ;

    while (len(setOfCoupons) != n):
      res = random.randint(1, n)
      setOfCoupons.add(res)
      times += 1
    answers.append(times)
  return np.mean(answers)


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

```
30.8
28.93
29.505
```

- **سوال سوم، چهارم، پنجم:**

احتمال مشاهده کالابرگ ها یکسان نیست و احتمال مشاهده ی کالابرگ نوع i برای اولین بار از این رابطه به دست می آید :

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

زیر پیش از کالابرگ نوع i ، i-۱ نوع کالابرگ مشاهده کرده ایم و در کل n کالابرگ داریم پس احتمال مشاهده کالابرگ نوع i بدین شکل است. و توزیع X_i ها یک توزیع هندسی است.

از آن جا که هر X_i دارای توزیع هندسی است پس تابع تابع جرمی احتمال آن و در نتیجه تابع مولد گشتاور آن به صورت زیر محاسبه می شود :

$$P_X(x) = (1-P)^{x-1} \times P \quad \Rightarrow \quad P = \frac{n-(i-1)}{n}$$

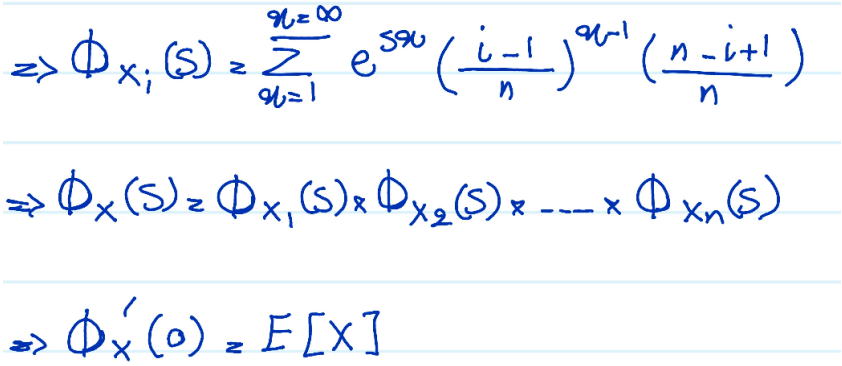
$$\phi_{X_i}(s) = \sum_{1}^{\infty} e^{sx} P_{X_i}(x) = \sum_{0}^{\infty} e^{sx} (1-P)^{x-1} \times P$$

$$= \sum_{x=1}^{x=\infty} e^{sx} \left(1 - \frac{n-i+1}{n}\right)^{x-1} \times \left(\frac{n-i+1}{n}\right)$$

$$= \sum_{x=1}^{x=\infty} e^{sx} \left(\frac{n-n+i-1}{n}\right)^{x-1} \left(\frac{n-i+1}{n}\right)$$

$$= \sum_{x=1}^{x=\infty} e^{sx} \left(\frac{i-1}{n}\right)^{x-1} \left(\frac{n-i+1}{n}\right)$$

و از آنجا که X جمع X_i هاست تابع مولد گشتاور آن به شکل زیر به دست می آید :



بنابراین با استفاده از کتابخانه ی sympy و تعریف دو متغییر s , x سیگمای لازم را تعریف کرده (توجه شود که در سیگما عبارت مثبت بی نهایت وجود داشت اما از آن جا که این مقدار غیرقابل دسترس است مشاهده می شود با عدد بزرگی همچون دویست جواب نهایی با دقت ۴ رقم اعشار به دست آماده و اعداد بزرگ تر از آن برای سقف سیگما صرفا برای افزایش دقت بسیار جزئی است و لزومی ندارد) و سپس با استفاده از یک حلقه مقادیر مختلف تابع مولد گشتاور به ازای i های متفاوت را در هم ضرب کرده سپس از عبارت نهایی بر حسب s مشتق می گیریم و s را برابر صفر قرار می دهیم حاصل برابر میانگین متغیر تصادفی X است. در نهایت عدد میانگین برابر با : **۲۹.۲۸۹۶**

```
from sympy import symbols, summation , diff
from math import exp

x = symbols('x')
s = symbols('s')
e = exp(1)

torque = 1
n=10

for i in range(1,n+1,1):
  expression = (e**(s*x)) * (((i-1)/n)**(x-1)) * ((n-i+1)/n)
  torque *= summation(expression, (x, 1, 200))


res = diff(torque, s).subs({s:0})
print(res)
```

```
29.2896823779169
```

که با عدد به دست آماده در بخش دوم سوال که برابر با ۲۹ بود تطابق دارد .

- تخمین بیزی و استنباط بیزی

- سوال اول و دوم:

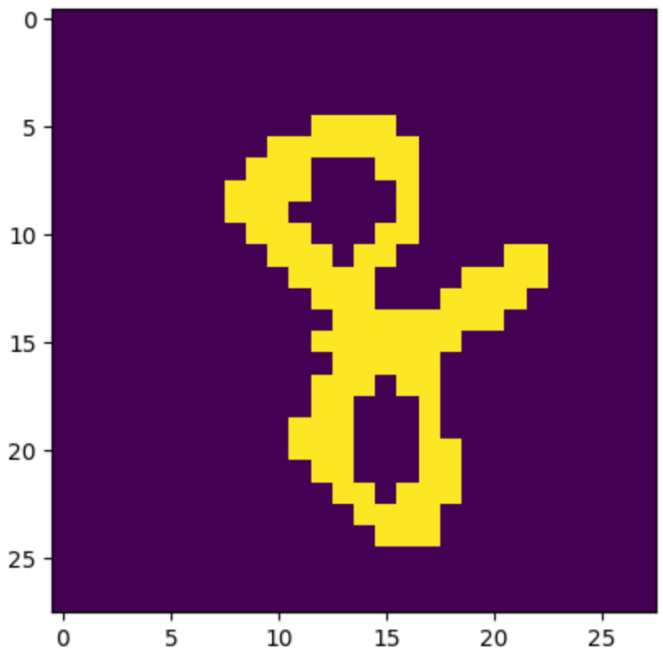
ابتدا اعداد بزرگتر مساوی ۱۲۸ را تبدیل به یک و اعداد کوچکتر از آن را تبدیل به صفر می کنیم به غیر از ستون اول دیتا فریم سپس دو سطر آخر را از دیتا فریم حذف کرده و در دو متغیر جداگانه ذخیره می کنیم

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('digits.csv')

def process_row(row):
    row[1:] = (row[1:] >= 128).astype(int)
    return row

df = df.apply(process_row, axis=1)

row201 = df.loc[200].copy()
row202 = df.loc[201].copy()
df = df.drop([200, 201])
```

- **سوال سوم:**

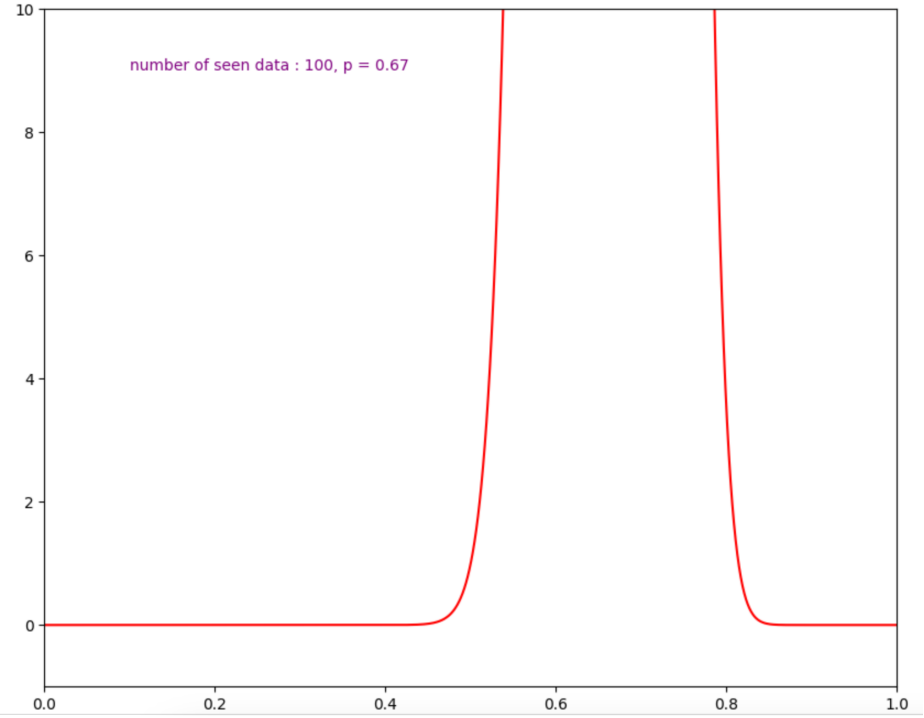
با استفاده از تابع matplotlib.pyplot.imshow سطر ۶۵ ام دیتا فریم را رسم می کنیم شکل حاصل به صورت زیر است :

```
plt.imshow(np.reshape(df.loc[65].values[1:], (28, 28)))
plt.show()
```



- **سوال چهارم:**

در نهایت با تکمیل کد داده شده که در آن با یک شرط ساده متغیر برنولی و با استفاده از کتابخانه ی scipy انتگرال مورد نیاز را پیاده سازی کردیم نتیجه ی تغییر توزیع بتا و نزدیک شدن آن به حالت نهایی را مشاهده می کنیم تصویر نهایی توزیع بتا در زیر آمده است همچنین طبق تصویر مد این توزیع عدد **۰.۶۷** است :



همچنین کد تکمیل شده به صورت زیر است :

```
from scipy import integrate, stats
from time import sleep
from IPython import display

t = 1000
p = np.linspace(0,1,t)
fy = stats.beta.pdf(p, a=1, b=1)

def update(fy: np.array, n:bool) -> np.array:
    p = np.linspace(0,1,t)
    pny = 1 - p
    if n:
      pny = p

    integrand = lambda x: stats.beta.pdf(x, a=1, b=1) * (1 - x)
    integral, error = integrate.quad(integrand, 0, 1)


    post = (pny * fy)/(integral)
    return post

plt.figure(figsize=(10,8))
for i in range(100):
    n =  df[df['label'] == 8].iloc[i, df.columns.get_loc('pixel404')]
    fy = update(fy, n)

    plt.plot(p, fy, 'r', label='1')
    plt.ylim(-1, 10)
    plt.xlim(0, 1)
    plt.text(0.1,9,f'number of seen data : {i + 1}, p = {fy.argmax() / t :.2f}', color='purple')
    display.clear_output(wait=True)
    display.display(plt.gcf())
    plt.clf()
    sleep(0.05)
```

- سوال پنجم و ششم:

در ابتدا از تمامی ستون های دیتا فریم به غیر از ستون اول که لیبل است میانگین میگیریم یک بار به ازای لیبل های ۸ و یک بار به ازای لیبل های ۹ که احتمال X به شرط لیبل است P(X | lable) حال طبق فرمول های داده شده در صورت سوال تابع محاسبه P(lable | X) پیاده سازی می کنیم نتایج خواسته شده در صورت سوال به شرح زیر است که با توجه به اینکه سطر ۲۰۱ از لیبل ۸ است و سطر ۲۰۲ از لیبل ۹ است نتایج کاملا صحیحی است :

```
P(lable=8, X=201)= 0.9997406457898158
P(lable=8, X=202)= 5.9151166121701555e-19
P(lable=9, X=201)= 0.00025935421018423725
P(lable=9, X=202)= 1.0
```

همچنین کد این سوال به صورت زیر است:

```python
PXlabel8 = df[df['label'] == 8].iloc[:, 1:].mean(axis=0)
PXlabel9 = df[df['label'] == 9].iloc[:, 1:].mean(axis=0)

def pXlable(X, choosedLabel):
  ans = 1
  for i in range(0,784,1):
    ans *= (choosedLabel.iloc[i] ** X.iloc[i+1] ) * ((1 - choosedLabel.iloc[i]) ** (1 - X.iloc[i+1]))
  return ans

def pLableX(choosedLable, X):
  return (pXlable(X, choosedLable)) / (pXlable(X,PXlabel8) + pXlable(X,PXlabel9))

print(f'P(lable=8, X=201)= {pLableX(PXlabel8, row201)}')
print(f'P(lable=8, X=202)= {pLableX(PXlabel8, row202)}')
print(f'P(lable=9, X=201)= {pLableX(PXlabel9, row201)}')
print(f'P(lable=9, X=202)= {pLableX(PXlabel9, row202)}')
```