CA#2

رضا چهرقانی ۸۱۰۱۰۱۴۰۱

---

1. به وسیلهٔ تابع pd.read_csv فایل را می‌خوانیم. ستون metro را در متغیر metro و ستون BRT را در متغیر brt قرار می‌دهیم. چونکه در این سوال در کل باید چهار شکل بکشیم، چهار subplot به صورت دو در دو درست می کنیم.

1.1. به وسیلهٔ تابع plt.hist هیستوگرام metro و BRT را یکجا رسم می کنیم. همچنین به آنها label نیز می‌دهیم. نام این نمودار را part 1 می گذاریم. مقدار bins را از کمترین مقدار تا بیشترین مقدار به صورت یک واحد یک واحد قرار می‌دهیم. (برای زیباتر شدن هیستوگرام بازه‌های binها را به صورت x-0.5 و x+0.5 قرار می‌دهیم که x مقداری صحیح است.)

1.2. به دلیل آنکه X و Y متغیر تصادفی گسسته می‌باشند و طبق شکل آن‌ها، می‌توان حدس زد که دارای توزیع پواسون هستند. پارامتر توزیع پواسون (λ) برابر میانگین آن می‌باشد. پس به وسیلهٔ تابع np.mean میانگین و در نتیجه پارامتر X و Y را بدست می‌آوریم.

1.3. برای رسم توزیع مقادیر metro همانند بخش 1.1 عمل می کنیم با این تفاوت که density را برابر True قرار می‌دهیم. زیرا با این کار ارتفاع هر ستون را به تعداد کل مقادیر تقسیم می کند و در نتیجه مساحت زیر هیستوگرام برابر یک می‌شود.

1.4. برای رسم نمودار توزیع X، ابتدا بازه‌ای را که می‌خواهیم نمایش دهیم مشخص می کنیم. سپس به کمک تابع scipy.stats.poisson.pmf که همان تابع جرمی احتمال توزیع پواسون هست، نقاط مقادیر بازه را بدست می‌آوریم و نمایش می‌دهیم.

1.5. می‌دانیم مجموع دو متغیر پواسون، متغیری پواسون است که پارامتر آن برابر مجموع پارامتر توزیع‌ها می‌باشد. پس Z نیز متغیری تصادفی با توزیع پواسون می‌باشد. همانند بخش 1.4 نمودار توزیع Z را رسم می کنیم. همانند بخش 1.3 هیستوگرام مجموع دو ستون مترو و BRT را رسم می کنیم.

۱.6. تابع جرمی احتمال متغیر تصادفی W را حساب می‌کنیم $(X + Y = Z)$:

$$P_{X|Z}(x|n) = \frac{P_{XZ}(x,n)}{P_Z(n)} = \frac{P_{XY}(x,n-x)}{P_Z(n)} = \frac{P_X(x)P_Y(n-x)}{P_Z(n)} = \frac{\frac{\lambda_X^{\ x}}{x!}e^{-\lambda_X}\times\frac{\lambda_Y^{\ n-x}}{(n-x)!}e^{-\lambda_Y}}{\frac{(\lambda_X+\lambda_Y)^n}{n!}e^{-(\lambda_X+\lambda_Y)}} = \binom{n}{x}\left(\frac{\lambda_X}{\lambda_X+\lambda_Y}\right)^x\left(\frac{\lambda_Y}{\lambda_X+\lambda_Y}\right)^{n-x}$$

$$\Rightarrow W\sim Bin(n,\ \frac{\lambda_X}{\lambda_X+\lambda_Y}) \Rightarrow W\sim Bin(n,\ \frac{\lambda_X}{\lambda_Z})$$

۱.7. همانند قبل اما این بار به کمک تابع scipy.stats.binom.pmf ، تابع جرمی احتمال W را رسم می‌کنیم.

۱.8. مقادیر ستون metro ردیف‌هایی که مجموع مترو و BRT آنها برابر ۸ می‌شود را در متغیر conditional_metro قرار می‌دهیم. همانند قبل بازه‌ی این مقادیر را بدست می‌آوریم و هیستوگرام را رسم می‌کنیم. همانطور که می‌شود مشاهده کرد، این دو نمودار بر روی هم افتاده‌اند. پس محاسبات تئوری با آزمایش عمل نتیجهٔ یکسانی داشته‌اند.

2. مسئلهٔ جمع کنندهٔ کالابرگ

۲.1. در ابتدا تابع coupon_collector_average را می‌نویسیم که مسئله را به کمک تابع monte_carlo به تعداد k بار اجرا می‌کند و میانگین مقادیر بدست آمده را برمی‌گرداند. در تابع monte_carlo یک set تعریف می‌کنیم تا بفهمیم که آیا تمام کالابرگ‌ها را دیده‌ایم یا نه. سپس حلقهٔ while را تا وقتی که همهٔ کالابرگ‌ها را ندیده‌ایم اجرا می‌کنیم. هر سری به کمک تابع np.random.randint عدد صحیحی در بازهٔ ۰ تا n می‌گیریم. (۰ بسته و n باز) و همچنین یکی به تعداد کالابرگ‌هایمان اضافه می‌کنیم. در نهایت نیز تعداد کالابرگ‌هایی را که دیده‌ایم باز می‌گردانیم.

۲.2. همانطور که از جواب‌ها پیداست، مقادیر به عدد 29 همگرا می‌شوند.

متغیر تصادفی X_i برابر تعداد کالابرگ‌هایی است که باید مشاهده کنیم تا تعداد کالابرگ‌های یکتایمان برابر i شود پس از اینکه تعداد کالابرگ‌های یکتایمان برابر i-1 شده بود. پس X_i دارای توزیع هندسی با احتمال $\frac{n-(i-1)}{n}$ است، زیرا از بین n نوع کالابرگ، n-(i-1) نوع آن برای ما مطلوب است.
شرایط تمام کالابرگ‌ها یکسان است و در هر مرحله با احتمال یکسانی ممکن است هر کدام از کالابرگ‌های را ببینیم. پس احتمال مشاهدهٔ کالابرگ نوع i برای اولین بار برای همهٔ iهای ۱ تا n یکسان است.

۲.3. دو نماد s و i را تعریف می‌کنیم. براساس صحبت قبل مقدار p_{X_i} را تعیین می‌کنیم. سپس تابع مولد گشتاور X_i $\emptyset_{X_i}(s)$ را بدست می‌آوریم. در ابتدا آمدم و از روابط زیر استفاده کردم:

$$P_{X_i}(x) = p_{X_i}(1 - p_{X_i})^{x-1} \Rightarrow \emptyset_{X_i}(s) = E[e^{sX}] = \sum_{x=1}^{\infty} e^{sx}P_{X_i}(x)$$

اما جواب غلط در می‌آمد. زیرا مقدار $\emptyset_{X_1}(s)$ صفر محاسبه می‌شد. در صورتی که برابر e^s باید می‌شد. پس از تست کردن مقادیر مختلف فهمیدم که کتابخانهٔ sympy باگ دارد. زیرا خروجی کد زیر را صفر می‌دهد:

```
from sympy import *
x = symbols('x')
print(Sum(0**(x-1), (x, 1, oo)).doit())
```

در صورتی که به ازای $x = 1$ مقدار برابر 0^0 می‌شود که برابر 1 است. پس خروجی باید یک بشود نه صفر. پس آمدم و از روش دیگری استفاده کردم. یادم آمد که استاد سر کلاس رابطهٔ تابع مولد گشتاور توزیع هندسی را بدست آورده‌بودند که به صورت زیر است:

$$X_i \sim Geo(p_i) \Rightarrow \emptyset_{X_i}(s) = \frac{p_i e^s}{1-(1-p_i)e^s}$$

پس از این رابطه برای محاسبهٔ تابع مولد گشتاور استفاده کردم.

2.4. می‌دانیم که متغیرهای تصادفی X_i مستقل هستند. همچنین می‌دانیم که تابع مولد گشتاور مجموع متغیرهای مستقل برابر ضرب تابع مولد گشتاور تک‌تک متغیرهای می‌باشد: $\emptyset_X(s) = \emptyset_{X_1}(s) \times \emptyset_{X_2}(s) \times ... \times \emptyset_{X_n}(s)$

2.5. می‌دانیم میانگین متغیر تصادفی از روی تابع مولد گشتاور به صورت روبه‌رو محاسبه می‌شود: $E[X] = \frac{d\emptyset_X(0)}{ds}$
پس به کمک تابع sympy.diff مشتق $\emptyset_X(s)$ را حساب می‌کنیم و سپس s را برابر صفر قرار می‌دهیم. در نهایت مقدار میانگین X را پرینت می‌کنیم.

3. ابتدا فایل csv را می‌خوانیم.

3.1. اعداد ۲۰۱ام و ۲۰۲ام را در متغیرهای digit_201 و digit_202 ذخیره می‌کنیم و سپس به کمک تابع pandas.DataFrame.drop آن دو ردیف را حذف می‌کنیم.

3.2. تابعی به نام set_threshold می‌نویسیم تا براساس THRESHOLD = 128 مقادیر پیکسل‌ها را باینری کند. سپس این تابع را به کمک pandas.DataFrame.map بر روی مقادیر دیتافریم به جز ستون label یعنی ستون‌های pixel اعمال می‌کنیم.

3.3. توسط تابع np.random.randint یک عدد رندوم از میان شماره‌ی ردیف‌های دیتافریم بدست می‌آوریم و ردیف متناظر را در متغیر random_row قرار می‌دهیم. باید حواسمان باشد که اولین خانه‌ی این ردیف مقدار label است، پس بقیهٔ خانه‌ها را به numpy.ndarray تبدیل کرده و در نهایت آنها را نیز توسط تابع

به آرایه دو بعدی ۲۸×۲۸ تغییر شکل می‌دهیم. حال با استفاده از تابع numpy.reshape آن را نمایش می‌دهیم. matplotlib.pyplot.imshow

3.4. طبق توضیحات صورت سوال می‌توان مقدار متغیر pny را به صورت زیر بدست آورد:

$$N|Y \sim Ber(Y) \Rightarrow P_{N|Y}(n|p) = p^{n}(1-p)^{1-n}$$

حال به محاسبهٔ مخرج کسر می‌پردازیم. براساس تخمینی که در صورت سوال گفته شده است داریم:

$$\int_0^1 f(x)dx \approx \frac{1-0}{N}\sum_{i=0}^{N} f(x_i);\ x_i = 0 + i\frac{1-0}{N} = \frac{1}{N}\sum_{i=0}^{N} f(\frac{i}{N})$$

پس مخرج کسر بدین صورت می‌شود:

$$\int_0^1 P_{N|Y}(n|p)f_Y(p) = \frac{1}{N}\sum_{i=0}^{N} P_{N|Y}(n\,|\frac{i}{N})f_Y(\frac{i}{N})$$

در این مسئله پیاده‌سازی توابع احتمالاتی به صورت آرایه‌ای از مقادیری است که جواب تابع در تمام نقاط بازهٔ مورد نظر می‌باشند. این مقادیر را به صورت np.linspace تعریف می‌کنیم که تعداد نقاط آن برابر t است. پس برای محاسبهٔ سیگمای بالا فقط کافیست مجموع مقادیر آرایهٔ $P_{N|Y} \times f_Y$ را به کمک تابع np.sum محاسبه کنیم. و سپس جواب را بر t تقسیم می‌کنیم. در نهایت نیز طبق فرمول صورت سوال مقدار post را محاسبه و برمی‌گردانیم. همان طور که از شکل پیداست مد توزیع بتا برابر با p=0.67 می‌شود؛ پس احتمال روشن بودن پیکسل ۴۰۴ام برابر ۰.۶۷ می‌باشد.

3.5. در ابتدا $P(X|label)$ را محاسبه می‌کنیم. به این صورت که دیتا فریم را بر اساس label گروه‌بندی می‌کنیم و سپس میانگین ستون‌های گروه‌ها را بدست می‌آوریم و چونکه در صورت پروژه گفته شده به صورت آرایه باید باشند، آن را به آرایهٔ دو بعدی تبدیل می‌کنیم به گونه‌ای که بعد اول ایندکس labelها و بعد دوم احتمال روشن بودن پیکسل iام یعنی همان x_i باشد.

3.6. به محاسبهٔ $P(label)$ می‌پردازیم. تعداد کل اعداد را ذخیره می‌کنیم و با تابع collections.Counter تعداد تکرار هر label را بدست می‌آوریم و با تقسیم بر تعداد کل اعداد، احتمال را محاسبه می‌کنیم. همچنین برای راحتی کار با آرایه‌ها، به هر label همان ایندکسی را که در دیکشنری دارند نسبت می‌دهیم.

برای اعمال توابع احتمالاتی بر روی اعداد ۲۰۱ام و ۲۰۲ام، تابع threshold را بر روی آن‌ها صدا می‌کنیم.

سه تابع calc_pX_condLabel، calc_pX و calc_pLabel_condX را برای محاسبهٔ $P(X|label)$، $P(X)$ و $P(label|X)$ در قانون بیز دقیقا به همان شکلی که در صورت پروژه گفته شده است، پیاده‌سازی می‌کنیم.

حال که توابع احتمالاتی مورد نیاز را پیاده‌سازی کردیم؛ به راحتی مقادیر خواسته شده را بدست می‌آوریم.

مقادیر خروجی به خوبی دلالت بر درستی و دقت حدس ما از label اعداد ۲۰۱ام و ۲۰۲ام دارد.