۱.فاز ۱

۱.۱ استخراج عکس

با استفاده از کتابخانه os و یک set() که عکسهای duplicate را از عکسهای دریافت شده حذف می کند.

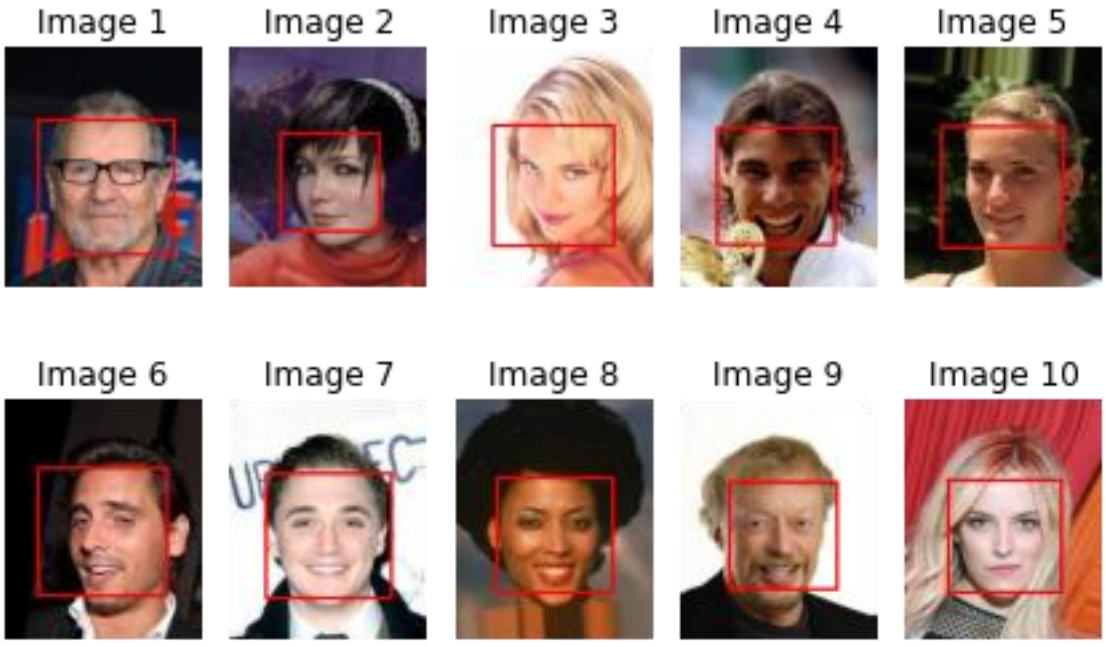
شماره عکس های تکراری : {3727, 7230, 3902, 3903, 3904, 3905, 3906, 3907, 3908, 3909, 3910, 3911, 7231, 7232, 7233, 7234, 7235, 7236, 7237, 7238, 7239, 5232, 5233, 5234, 5235, 5236, 5237, 5238, 5239, 5240}

۱.۲ ابعاد صورت

در این بخش در سه مرحله ابعاد صورت استخراج شده:

الف)'cv2.data.haarcascades + 'haarcascade_frontalface_default.xml :

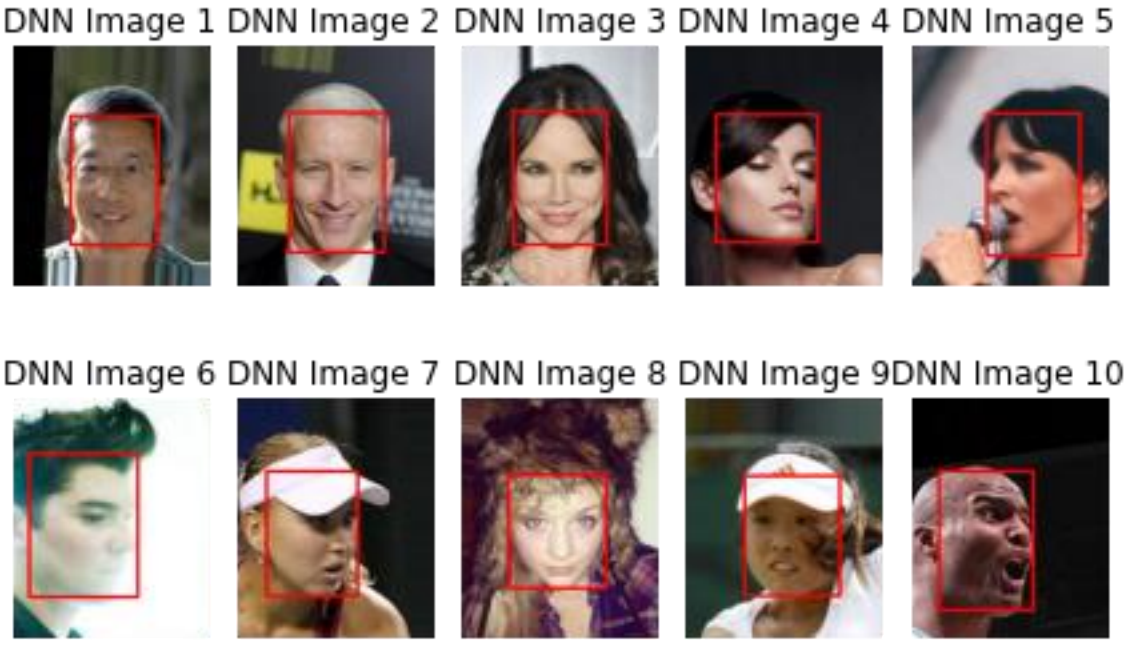
یک روش مؤثر و سریع برای شناسایی چهره‌ها در تصاویر دیجیتال فراهم می‌کند. با استفاده از این کد، برنامه‌نویسان و محققان می‌توانند به راحتی الگوریتم‌های تشخیص چهره را در پروژه‌های خود پیاده‌سازی کنند.



این روش معایبی نیز دارد زیرا برخی تصاویر که ممکن است زاویه سر از حد خاصی نسبت به یک تصویر با زاویه مستقیم از دوربین بیشتر شود نتواند آنرا تشخیص دهد.

ب) deploy.prototxt.txt و res10_300x300_ssd_iter_140000.caffemodel:

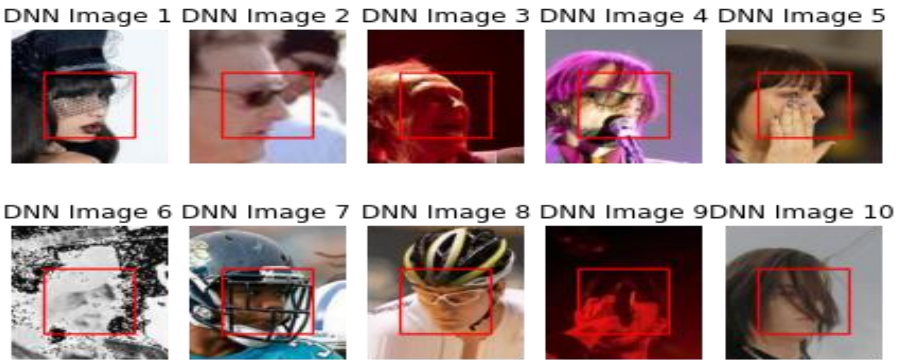
یک مدل DNN که ابزاری مؤثر برای شناسایی چهره‌ها در تصاویر دیجیتال فراهم می‌کند. با استفاده از این کد، توسعه‌دهندگان می‌توانند به راحتی الگوریتم‌های تشخیص چهره را در پروژه‌های خود پیاده‌سازی کنند و از قدرت یادگیری عمیق بهره‌مند شوند.



با استفاده از این مدل تمامی چهره ها بجز ۴۱ تصویر بدست می آید که این تصویر ها ویژگی های مخربی دارند که باعث می شود قابل تشخیص این مدل ها نباشند.

ج) میانگین گیری:

برای پیدا کردن bounding box باقی تصاویر از میانگین گیری bounding box های دیگر استفاده می کنیم.



۳.۱ موقعیت چشم ها

از آنجایی که اکثر تصاویر در موقعیت تقریبا یکسانی قرار گرفته اند با استفاده از ضریب زیر با تقریب بالایی اینکار انجام شده.

```
eyes = []
for i in range(50000):
    eyes.append([faces[i][0] +20,faces[i][1]+30 ,int(faces[i][2]/4) ,int(faces[i][3]/4)])
```

نتایج در مقایسه با opencv:

٤.١ رنگ پوست:

```python
def get_skin_color_rgb(image, bbox):



    image_ycrcb = cv2.cvtColor(image, cv2.COLOR_BGR2YCrCb)


    min_ycrcb = np.array([0, 133, 77], dtype=np.uint8)
    max_ycrcb = np.array([255, 173, 127], dtype=np.uint8)


    skin_mask = cv2.inRange(image_ycrcb, min_ycrcb, max_ycrcb)


    skin_region = cv2.bitwise_and(image, image, mask=skin_mask)


    x, y, w, h = bbox
    roi = skin_region[y:y+h, x:x+w]



    if roi.size == 0:
        return None

    avg_color = cv2.mean(roi, mask=skin_mask[y:y+h, x:x+w])[:3]


    r, g, b = map(int, avg_color[::-1])

    return r, g, b
```

۱ .تبدیل تصویر به فضای رنگYCrCb

ابتدا تصویر ورودی که در فضای رنگ BGR (آبی، سبز، قرمز) قرار دارد، به فضای رنگ YCrCb تبدیل می‌شود. این تبدیل به دلیل قابلیت بالای فضای YCrCb در تفکیک نواحی پوست از سایر نواحی تصویر انجام می‌شود.

۲ .تعریف محدوده رنگ پوست

در ادامه، محدوده‌ای برای رنگ پوست در فضای YCrCb تعیین می‌شود. این محدوده با استفاده از دو آرایه min_ycrcb و max_ycrcb مشخص می‌شود که شامل مقادیر حداقل و حداکثر برای شناسایی نواحی پوست است.

۳ .ایجاد ماسک پوست

با استفاده از تابع cv2.inRange، یک ماسک ایجاد می‌شود که نواحی تصویر که در محدوده مشخص شده قرار دارند را شناسایی می‌کند. نواحی پوست در این ماسک سفید و بقیه نواحی سیاه خواهند بود.

۴ .استخراج ناحیه پوست

سپس با استفاده از ماسک ایجاد شده، تنها نواحی پوست از تصویر اصلی استخراج می‌شوند. این کار با استفاده از تابع cv2.bitwise_and انجام می‌گیرد که بر اساس ماسک عمل می‌کند.

۵ .تعیین ناحیه مورد نظر برای محاسبه رنگ

با استفاده از پارامتر bbox که مختصات و ابعاد یک مستطیل را مشخص می‌کند، ناحیه‌ای از تصویر استخراج شده (ناحیه پوست) انتخاب می‌شود.

۶ .بررسی خالی بودن ناحیه انتخابی

اگر ناحیه انتخابی خالی باشد (به عبارت دیگر، هیچ پیکسل پوستی وجود نداشته باشد)، تابع None بازمی‌گرداند.

۷ .محاسبه میانگین رنگ

در مرحله بعد، میانگین رنگ ناحیه انتخابی محاسبه می‌شود. فقط پیکسل‌های مربوط به پوست در محاسبه میانگین لحاظ می‌شوند.

۸ .تبدیل و بازگشت رنگ‌ها

در نهایت، مقادیر میانگین رنگ به ترتیب RGB (قرمز، سبز، آبی) تبدیل و به صورت یک تاپل بازگردانده می‌شوند.این روش به طور مؤثری رنگ پوست را از یک تصویر استخراج کرده و مقدار میانگین RGB آن را برمی‌گرداند، که برای کاربردهای مختلف مانند پردازش تصویر یا تحلیل داده‌ها مفید است.

استخراج شده تصاویر در دسته های زیر قرار می گیرند.RGBدر انتها با توجه به

```
skin_tones = {

    1: (255, 224, 189), #light_skin
    2: (224, 172, 105), #medium_light_skin
    3: (194, 142, 83), #medium_skin
    4: (143, 85, 54), #medium_dark_skin
    5: (78, 53, 36) #dark_skin
}
```

۵.۱ رنگ چشم

```
def get_eye_color(image, bbox):




    x, y, w, h = bbox

    eye_roi = image[y:y+h, x:x+w]


    eye_roi_rgb = cv2.cvtColor(eye_roi, cv2.COLOR_BGR2RGB)


    average_color = np.mean(eye_roi_rgb, axis=(0, 1))

    return tuple(average_color.astype(int))
```

با استفاده از مختصات استخراج‌شده، ناحیه مربوط به چشم از تصویر اصلی برش داده می‌شود.

سپس با استفاده ازOpenCV ، ناحیه برش داده‌شده به فرمت رنگ RGB تبدیل می‌شود OpenCV .
به طور پیش‌فرض تصاویر را در فرمت BGR ذخیره می‌کند، بنابراین این تبدیل ضروری است.

درمرحله بعد، میانگین رنگ ناحیه چشم محاسبه می‌شود. این کار با استفاده از تابع np.mean انجام می‌شود که میانگین رنگ‌ها را در دو بعد (ارتفاع و عرض) محاسبه می‌کند.

در نهایت، میانگین رنگ به صورت یک تاپل (tuple) بازگردانده می‌شود. برای اطمینان از اینکه مقادیر رنگ صحیح هستند، آنها به نوع عدد صحیح (integer) تبدیل می‌شوند.

در مرحله اخر نیست بر اساس RGB خود در دسته های زیر قرار می گیرند.

```
eye_colors = {
   1: (101, 67, 33), #brown
   2: (0, 0, 255), #blue
   3: (0, 128, 0), #green
   4: (118, 92, 66), #hazel
   5: (128, 128, 128), #gray
   6: (255, 191, 0) #amber
}
```

۲.فاز ۲

ابتدا ضرایب را به روش زیر محاسبه می کنیم:

مراحل محاسبه ضریب همبستگی پیرسون

۱. **محاسبه کوواریانس**:

کوواریانس میزان تغییرات دو متغیر تصادفی را با یکدیگر اندازه‌گیری می‌کند. فرمول کوواریانس بین دو متغیر X و Y به‌صورت زیر است:

$$Cov(X,Y)=1n\sum i=1n(Xi-X^-)(Yi-Y^-)Cov(X,Y)=n1i=1\sum n(Xi-X^-)(Yi-Y^-)$$

که در آن n تعداد نقاط داده، Xi و Yi نقاط داده فردی و X^- و Y^- میانگین‌های X و Y هستند.

۲. **محاسبه انحراف معیار**:

انحراف معیار میزان پراکندگی یک مجموعه داده نسبت به میانگین آن را اندازه‌گیری می‌کند. برای هر متغیر، به‌صورت زیر محاسبه می‌شود:

$$\sigma X=1n\sum i=1n(Xi-X^-)2\sigma X=n1i=1\sum n(Xi-X^-)2$$

و به‌طور مشابه برای Y:

$$\sigma Y=1n\sum i=1n(Yi-Y^{-})2\sigma Y=n1i=1\sum n(Yi-Y^{-})2$$

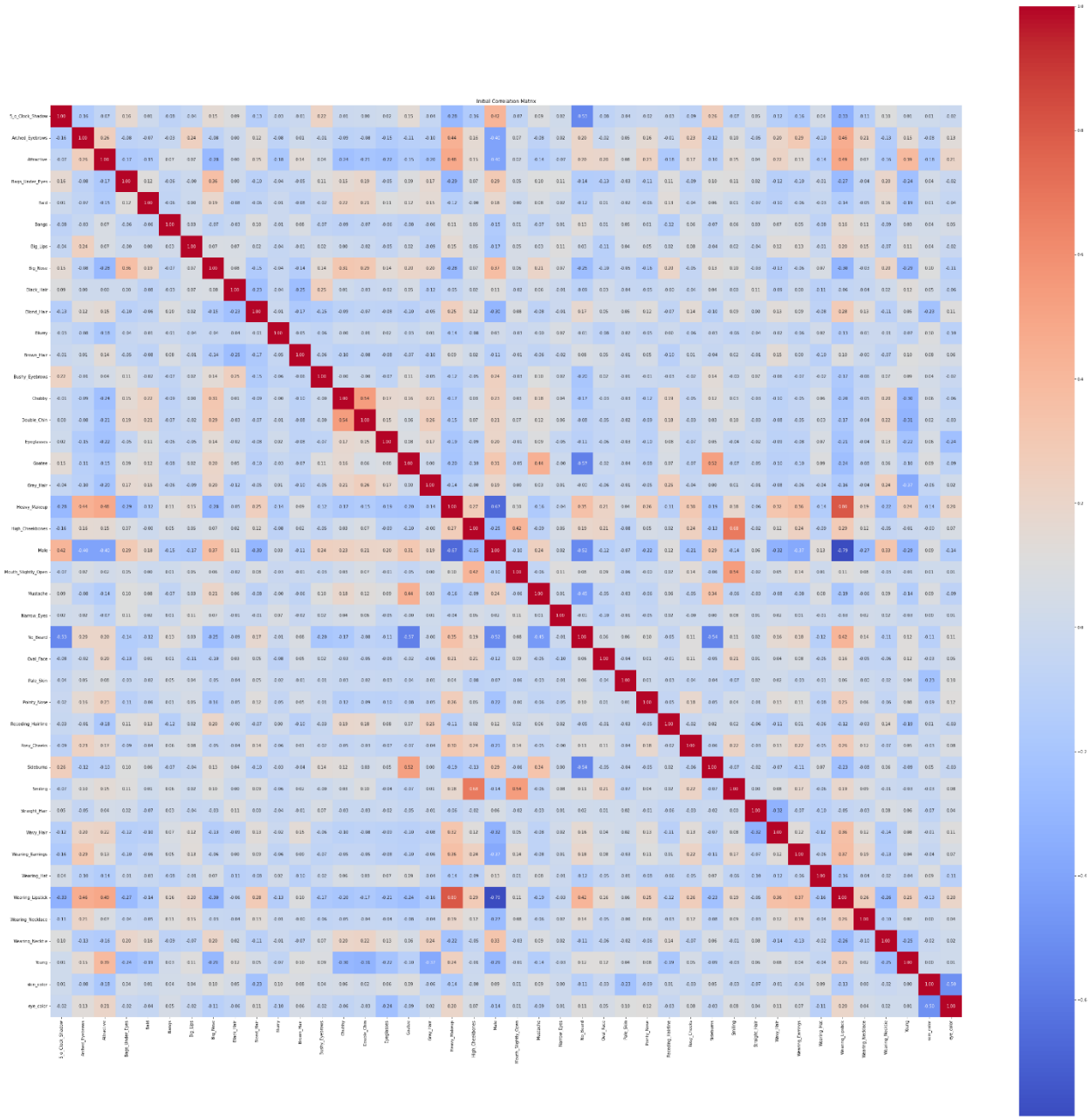
۳. محاسبه ضریب همبستگی پیرسون:

پس از محاسبه کوواریانس و انحراف معیارها، می‌توانید ضریب همبستگی پیرسون را با استفاده از فرمول زیر محاسبه کنید:

$$r=Cov(X,Y)\sigma X\sigma Yr=\sigma X\sigma YCov(X,Y)$$

این فرمول کوواریانس را با حاصل‌ضرب انحراف معیارها نرمالیزه می‌کند و مقداری بین −۱ و ۱ ارائه می‌دهد.

نتایج زیر حاصل می شود:

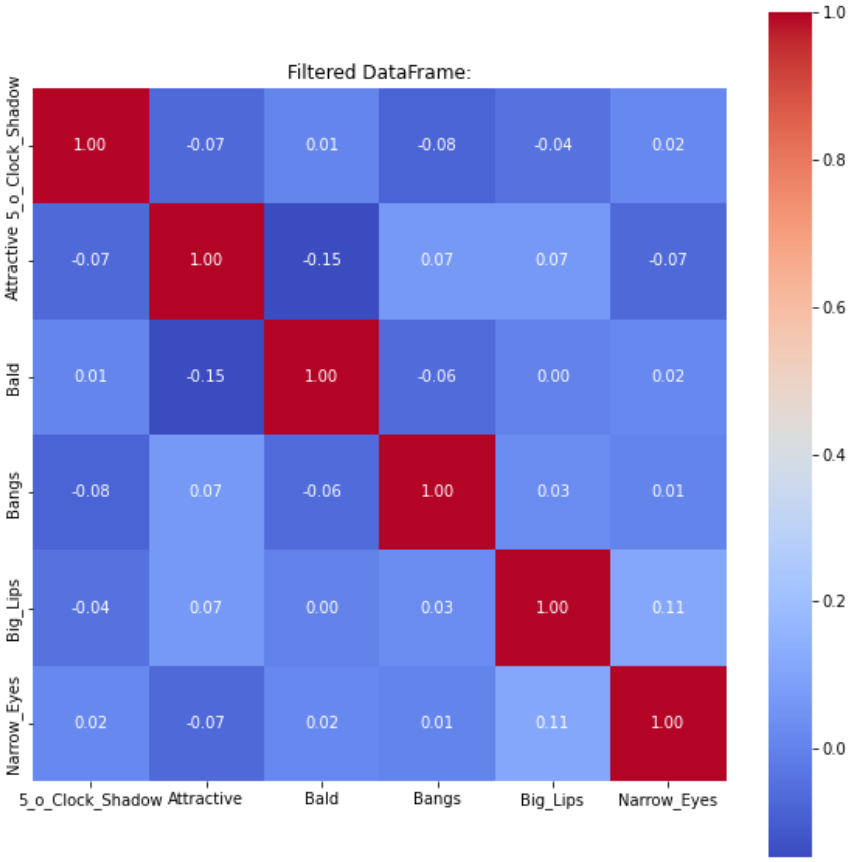


سپس با در نظر گرفتن threshold = 0.15 یک correlation پیدا می کنیم که با تمامی correlation های دیگر زیر threshold باشد و Narrow_Eyes انتخاب می شود.

سپس ۵ فیچر دیگر انتخاب می کنیم که با این فیچر و دیگر فیچرهای انتخاب شده correlation زیر threshold داشته باشند. فیچر های زیر انتخاب می شوند.

{'5_o_Clock_Shadow', 'Bald', 'Attractive', 'Bangs', 'Big_Lips', 'Narrow_Eyes'}

هیت مپ آنها نیز حالت زیر می شود.



در این مرحله فیچر های مناسب انتخاب شده و آنهارا از فایل csv استخراج می کنیم.

۳. فاز ۳

kmeans ۱.۳

با استفاده از elbow method برای این روش k را پیدا می کنیم

روش آرنج(Elbow Method)

روش آرنج یک تکنیک رایج در خوشه‌بندی k-means است که برای تعیین تعداد بهینه خوشه‌ها، که با k نشان داده می‌شود، استفاده می‌شود. در اینجا یک توضیح ساده از چگونگی عملکرد آن آورده شده است:

مرور کلی روش آرنج

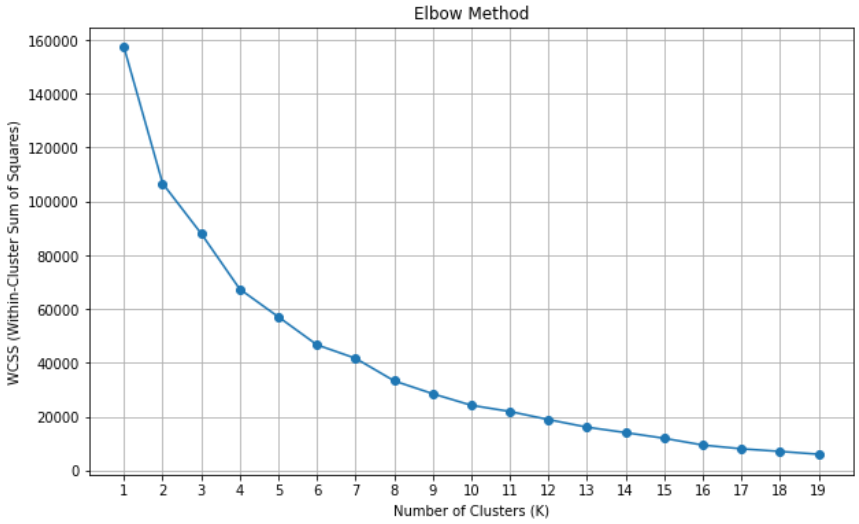
۱. **هدف** :هدف روش آرنج شناسایی نقطه‌ای است که افزودن خوشه‌های بیشتر به‌طور قابل توجهی کیفیت خوشه‌بندی را بهبود نمی‌بخشد. این نقطه به‌صورت "آرنج" در یک نمودار ظاهر می‌شود.

۲. **مجموع مربعات درون خوشه :(WCSS)** این روش به محاسبه WCSS برای مقادیر مختلف kتکیه دارد WCSS .میزان فشردگی خوشه‌ها را اندازه‌گیری می‌کند و به‌عنوان مجموع فاصله‌های مربعی بین هر نقطه داده و مرکز خوشه مربوطه تعریف می‌شود.

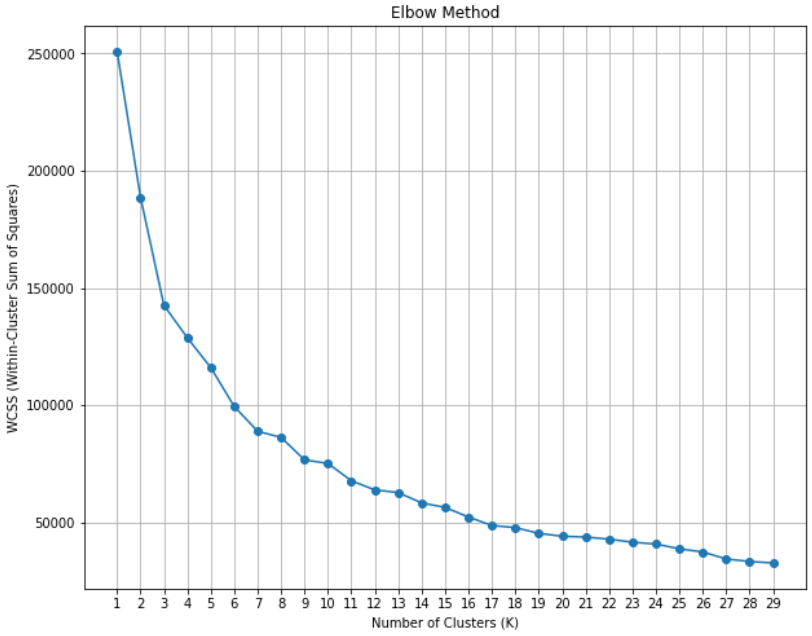
۳. **روش کار**:

- الگوریتم k-means را برای یک دامنه از مقادیر k(به‌عنوان مثال، از ۱ تا ۱۰) اجرا کنید.
- برای هر k، WCSSرا محاسبه کنید.
- نتایج را بر اساس kرسم کنید.

۴. **شناسایی آرنج** :نمودار معمولاً در ابتدا کاهش شدیدی در WCSS نشان می‌دهد که سپس مسطح می‌شود. نقطه‌ای که این تغییر رخ می‌دهد، به‌عنوان kبهینه در نظر گرفته می‌شود. این نشان‌دهنده آن است که افزودن خوشه‌های بیشتر فراتر از این نقطه، بازدهی کمتری در کاهش WCSS دارد.



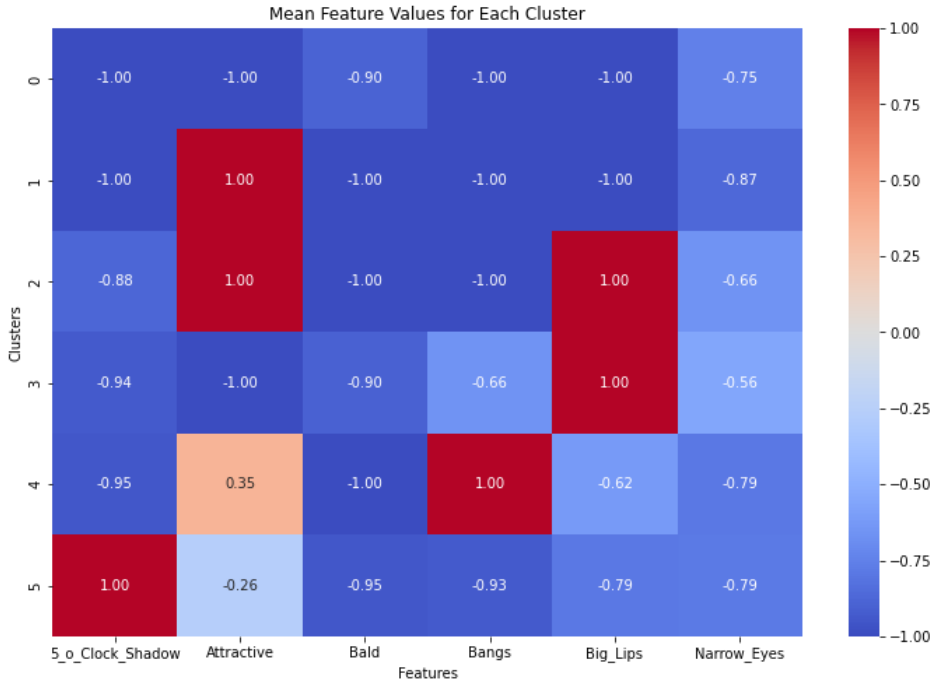
نمودار elbow method برای ۶فیچر اصلی



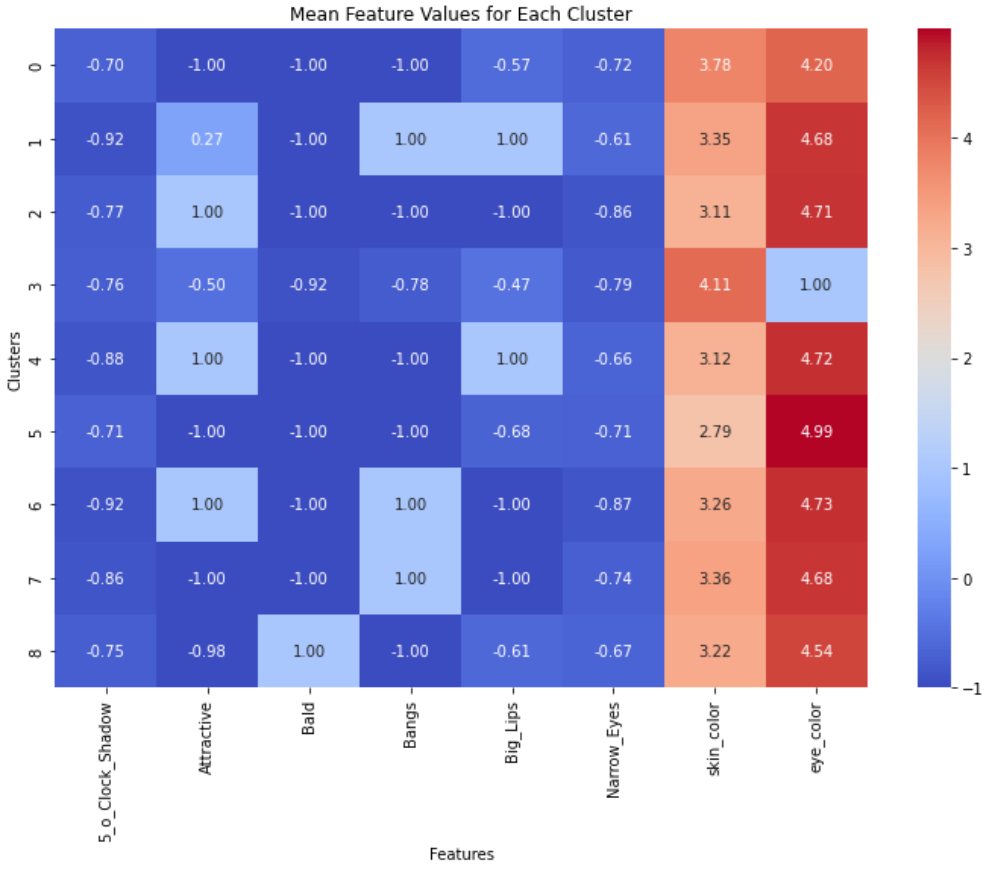
نمودار elbow method برای ۶فیچر اصلی + رنگ چشم و صورت

سپس با توجه به این نمودار ها k را انتخاب می کنیم.(برای ۶ فیچر اصلی cluster ۶ وبرای ۶فیچر+ رنگ چشم و صورت ۹)

بعد از این انتخاب کلاسترینگ انجام شده و هیت مپ کلاستر ها به صورت زیر می شود.



Silhouette Score: 0.7161765349799127



Silhouette Score: 0.4423807707041405

همانطور که از این هیت مپ ها پیداست رنگ چشم و پوست ویژگی خوب و حاکمی برای یک کلاستر نیست این موضوع حتی در امتیاز silhouette کلاستر ها نیز پیداست که وجود این فیچر ها باعث کیفیت پایین کلاستر می شود.

تأثیر ویژگی‌های نامناسب بر خوشه‌بندی

۱. نویز و بی‌ربط بودن:

- ویژگی‌های نامناسب معمولاً نویز یا اطلاعات بی‌ربطی را به مجموعه داده‌ها وارد می‌کنند. این می‌تواند منجر به ایجاد خوشه‌هایی شود که نمایانگر گروه‌بندی‌های معنادار نیستند و در نتیجه امتیاز سیلوئت پایینی را به همراه دارد. امتیاز سیلوئت نزدیک به صفر یا منفی نشان‌دهنده این است که نقاط به‌خوبی به خوشه‌ها اختصاص نیافته‌اند و این نشان می‌دهد که ویژگی‌ها به‌طور مؤثر داده‌ها را به گروه‌های متمایز جدا نمی‌کنند.

۲. افزایش همپوشانی بین خوشه‌ها:

- ویژگی‌هایی که به جداسازی خوشه‌ها کمک نمی‌کنند، می‌توانند باعث همپوشانی قابل توجهی بین آن‌ها شوند. این همپوشانی تشخیص گروه‌های مختلف را برای الگوریتم خوشه‌بندی دشوار می‌کند و منجر به کاهش امتیاز سیلوئت می‌شود. برای داشتن یک امتیاز سیلوئت بالا، نیاز به جداسازی واضح بین خوشه‌ها است که ویژگی‌های نامناسب این جداسازی را تضعیف می‌کنند.

۳. متریک‌های فاصله ضعیف:

- الگوریتم‌های خوشه‌بندی، مانندK-means ، به متریک‌های فاصله (مانند فاصله اقلیدسی) برای تعیین تخصیص خوشه‌ها وابسته هستند. اگر ویژگی‌های نامناسب شامل شوند، می‌توانند این فاصله‌ها را تحریف کنند و باعث شوند که خوشه‌ها بر اساس اطلاعات گمراه‌کننده تشکیل شوند. این تحریف می‌تواند منجر به افزایش میانگین فاصله درون خوشه‌ها و کاهش میانگین فاصله به خوشه‌های همسایه شود و در نتیجه امتیاز سیلوئت ضعیفی ایجاد کند.

۴. مسائل ابعادی:

- فضاهای با ابعاد بالا اغلب از "نفرین ابعاد" رنج می‌برند، جایی که فاصله بین نقاط با افزایش ابعاد کمتر معنی‌دار می‌شود. اگر ویژگی‌های نامناسب به این ابعاد کمک کنند بدون اینکه اطلاعات

مفیدی اضافه کنند، می‌تواند منجر به خوشه‌بندی ناکارآمد و امتیاز سیلوئت پایینی شود زیرا فاصله‌ها بین نقاط درون یک خوشه افزایش می‌یابد.

۵. **مقیاس‌بندی و نرمال‌سازی ویژگی‌ها:**

- ناهماهنگی در مقیاس‌بندی ویژگی‌ها نیز می‌تواند بر نتایج خوشه‌بندی تأثیر بگذارد. اگر برخی از ویژگی‌های نامناسب به دلیل مقیاس خود غالب شوند، می‌توانند الگوریتم خوشه‌بندی را به سمت تشکیل گروه‌بندی‌های نادرست هدایت کنند که منجر به کاهش امتیاز سیلوئت می‌شود.

نتیجه‌گیری

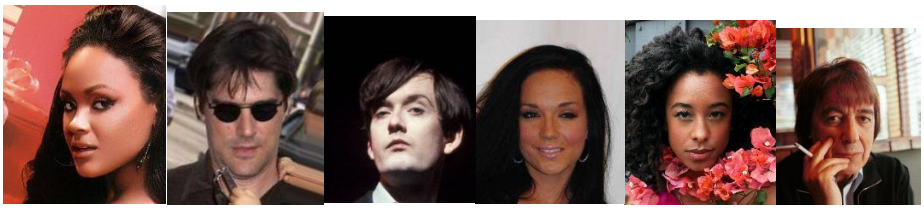
برای دستیابی به خوشه‌بندی مؤثر با امتیاز سیلوئت بالا، انتخاب و پیش‌پردازش دقیق ویژگی‌ها بسیار مهم است. حذف ویژگی‌های نامناسب یا استفاده از تکنیک‌های انتخاب ویژگی می‌تواند کیفیت خوشه‌ها را افزایش دهد و امتیاز سیلوئت را با اطمینان از اینکه فقط ویژگی‌های مرتبط و مفید در فرآیند خوشه‌بندی مشارکت دارند، بهبود بخشد .

برای بهتر مشخص شدن تفاوت کلاسترخها ۵ عکس از هر ۲ کلاستر از در زیر قرار گرفته می شود.

کلاستر ۲ kmean بدون رنگ چشم و پوست:

کلاستر ۵ kmean بدون رنگ چشم و پوست:

کلاستر ٤ kmean با رنگ چشم و پوست:



کلاستر ٦ kmean با رنگ چشم و پوست:

با استفاده از رنگ چشم و پوست تقریبا می توان دید که این معیار تاثیر خود را در کلاسترینگ اعمال کرده ولی از آنجایی که عکسها از کیفیت بالایی برخوردار نبودند ممکن است دچار خطا شویم و از جهتی تمامی فیچر های ما باینری بوده ولی این دو فیچر دارای مقادیر متفاوت هستند.

تأثیر نوع ویژگی‌ها بر خوشه‌بندی

۱. ویژگی‌های باینری:

- **الگوریتم‌های خوشه‌بندی** :الگوریتم‌های خوشه‌بندی سنتی مانند K-Means برای داده‌های باینری مناسب نیستند. دلیل این امر این است که K-Means از فاصله اقلیدسی استفاده می‌کند که در هنگام اعمال به مقادیر باینری مشکل‌ساز می‌شود. این متریک فاصله به‌طور مؤثر اختلافات بین متغیرهای باینری را محاسبه می‌کند که منجر به تخصیص‌های تصادفی خوشه‌ها به دلیل تساوی در محاسبات فاصله می‌شود. این می‌تواند منجر به خوشه‌هایی شود که به‌طور معناداری نمایانگر داده‌های زیرین نیستند.
- **رویکردهای جایگزین** :برای داده‌های باینری، روش‌هایی مانند خوشه‌بندی سلسله‌مراتبی یا K-Modesمناسب‌تر هستند. این روش‌ها از متریک‌های فاصله متفاوتی استفاده می‌کنند که برای داده‌های دسته‌ای مناسب‌تر هستند و بنابراین از مشکلات K-Means جلوگیری می‌کنند.

۲. ویژگی‌های پیوسته:

- ویژگی‌های پیوسته می‌توانند با استفاده از K-Means خوشه‌بندی شوند، اما تأثیر آن‌ها بستگی به مقیاس و توزیع داده‌ها دارد. اگر ویژگی پیوسته دارای دامنه باریکی باشد (مانند ۰ تا ۵)، ممکن است در فرآیند خوشه‌بندی سهم زیادی نداشته باشد اگر ویژگی باینری غالب باشد.
- **مقیاس‌بندی ویژگی‌ها** :هنگام ترکیب ویژگی‌های باینری و پیوسته، مقیاس‌بندی مناسب ویژگی‌های پیوسته بسیار مهم است. در غیر این صورت، الگوریتم خوشه‌بندی ممکن است به‌طور نامتناسبی وزن بیشتری به ویژگی پیوسته بدهد به دلیل دامنه عددی بزرگتر آن نسبت به مقادیر باینری.

۳. نوع ویژگی‌های مختلط:

- هنگام خوشه‌بندی مجموعه داده‌هایی با انواع مختلط ویژگی‌ها، ضروری است که در نظر گرفته شود هر ویژگی چگونه به متریک فاصله استفاده شده در خوشه‌بندی کمک می‌کند. تفاوت قابل توجه در نوع ویژگی‌ها می‌تواند منجر به خوشه‌های گمراه‌کننده شود اگر یک نوع به دلیل دامنه عددی‌اش غالب باشد.
- **کاهش ابعاد** :تکنیک‌هایی مانند تحلیل مولفه اصلی (PCA) یا تحلیل عاملی می‌توانند برای تبدیل داده‌های مختلط به فرمت مناسب‌تر برای خوشه‌بندی مفید باشند.

نتیجه‌گیری

به طور خلاصه، استفاده از ترکیبی از ویژگی‌های باینری و پیوسته در خوشه‌بندی می‌تواند واقعاً منجر به نتایج نامطلوب شود اگر به درستی مدیریت نشود. توصیه می‌شود الگوریتم‌های خوشه‌بندی را انتخاب کنید که به‌طور خاص برای داده‌های باینری طراحی شده‌اند یا داده‌ها را از طریق مقیاس‌بندی یا تکنیک‌های کاهش ابعاد پیش‌پردازش کنید. این اطمینان حاصل می‌کند که همه ویژگی‌ها به‌طور معناداری در فرآیند خوشه‌بندی مشارکت دارند و به جلوگیری از مشکلات ناشی از تفاوت نوع ویژگی‌ها کمک می‌کند.

همچنین دلیل زیر برای کم بودن silhouette کلاسترینگ با رنگ چشم و پوست است:

معیارهای فاصله گمراه‌کننده

انتخاب معیار فاصله می‌تواند تأثیر زیادی بر نتایج خوشه‌بندی داشته باشد. اگر معیار فاصله با ساختار داده‌ها همخوانی نداشته باشد)مثلاً استفاده از فاصله اقلیدسی برای داده‌های غیر اقلیدسی)، ممکن است نحوه

نزدیک یا دور بودن نقاط را به‌طور نادرست نشان دهد و منجر به خوشه‌بندی ضعیف و نمرات سیلوئت پایین شود.

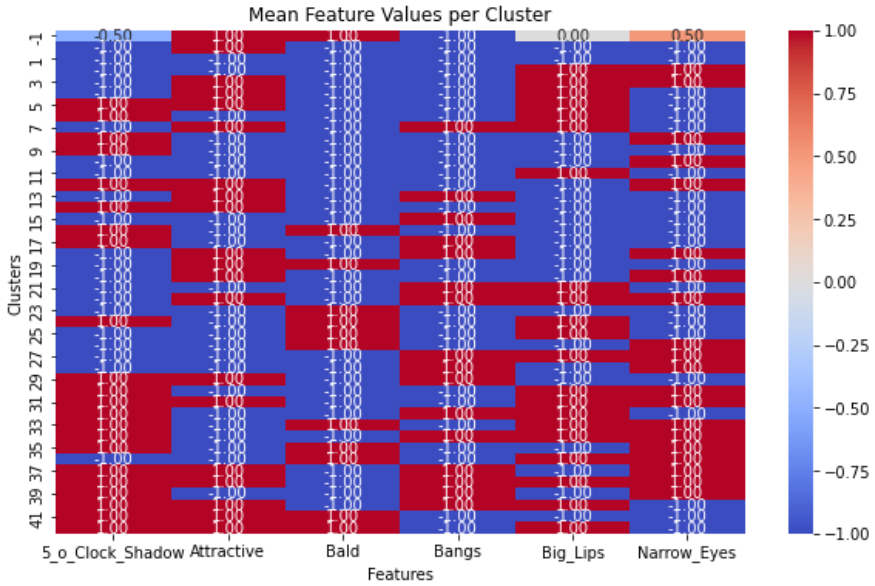
DBscan ۲.۳

```
for eps in np.arange(0.1, 1.0, 0.1):
    for min_samples in range(2, 10):
        dbscan = DBSCAN(eps=eps, min_samples=min_samples)
        labels = dbscan.fit_predict(X_scaled)
        n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
        if n_clusters > 1:
            s_score = silhouette_score(X_scaled, labels)
            if best_score < s_score:
                best_score = s_score
                best_eps = eps
                best_min_samples = min_samples
```

با استفاده از قطعه کد بالا بهترین hyperparameter ها را برای این الگوریتم انتخاب میکنیم به گونه ای که بالاترین silhouette انتخاب شود.

سپس کلاسترینگ را انجام می دهیم.

امتیاز silhouette = ۰.۹۹۹۹۰۴۰۸۸۹۰۰۸۴۵۴

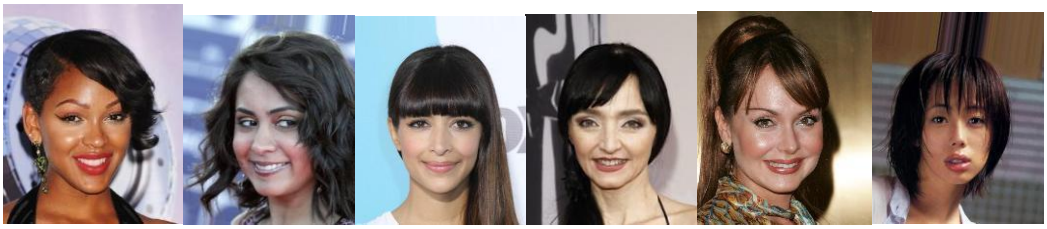


نمودار heatmap

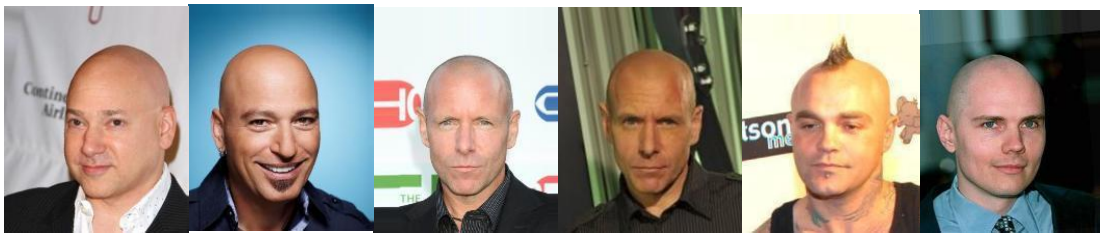
همانطور که مشاهده می شود این الگوریتم کلاستر های قاطعی ساخته است.

نمونه ای از این کلاستر ها:

کلاستر ۸:



کلاستر ۲۰:



meanshift ۳.۳

```
bandwidth = estimate_bandwidth(df, quantile=0.25, n_samples=2500)  # Adjust
quantile as needed
print(bandwidth)
```

از این تابع برای بدست اوردن hyperparameter در این الگوریتم استفاده شده است.

چگونگی پیدا کردن بهترین هایپرپارامتر

۱. **محاسبه فاصله** :تابع فاصله‌های جفتی بین نقاط در مجموعه داده را محاسبه کرده و نزدیک‌ترین همسایه‌ها را بر اساس کوانتیل مشخص شده شناسایی می‌کند. این فرآیند به تعیین چقدر نقاط داده فشرده هستند، کمک می‌کند.

۲. **تخمین پهنای باند** :این فاصله‌ها جمع‌آوری شده و یک مقدار واحد برای پهنای باند محاسبه می‌شود که نشان‌دهنده چگالی نقاط داده است. این مقدار به عنوان یک هایپرپارامتر برای Mean Shift عمل کرده و تأثیرگذار بر نحوه تشکیل خوشه‌هاست.

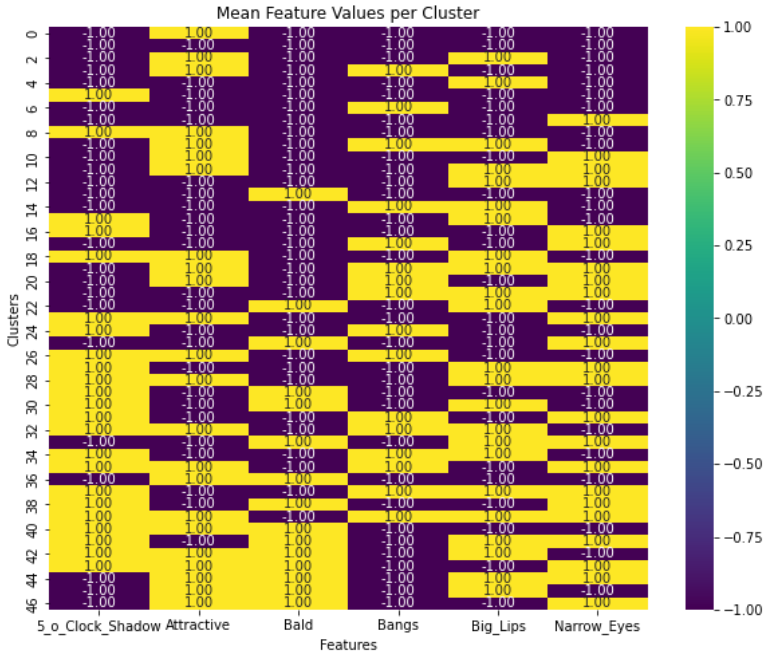
۳. **تأثیر بر خوشه‌بندی** :تخمین زده شده از پهنای باند به طور مستقیم بر نتایج خوشه‌بندی تأثیر می‌گذارد:

- یک **پهنای باند کوچک** ممکن است منجر به بسیاری از خوشه‌های کوچک شود که جزئیات ریز را Captures می‌کند.
- یک **پهنای باند بزرگ** ممکن است خوشه‌های متمایز را ادغام کرده و جزئیات نمایندگی داده‌ها را از دست بدهد.

با اجرای این کد، شما به طور مؤثری انتخاب یک پهنای باند مناسب برای مدل Mean Shift خود را خودکار می‌کنید، به آن اجازه می‌دهید که بهتر با توزیع زیرین داده‌ها سازگار شود و عملکرد خوشه‌بندی را بهبود بخشد.

هایپرپارامتر بدست امده = ۱.۵۹۸۱۶۸۱۱۶۲۸۴۲۹۹۸

سپس کلاسترینگ را انجام میدهیم(Silhouette Score: 0.99992)



هیت مپ featureها

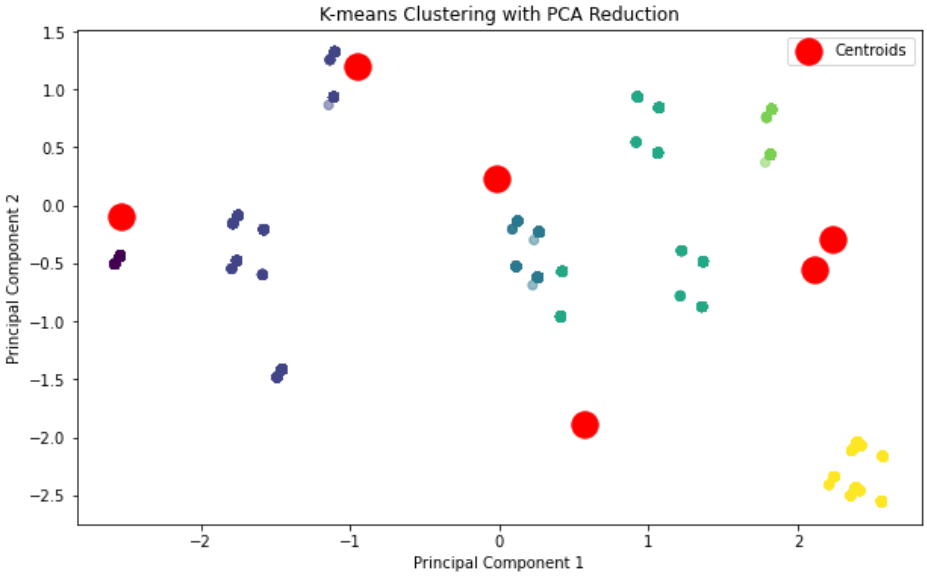
این کلاسترها نیز همانند Dbscan از فیچرهای قاطعی برخوردارند .

نمونه ای از این کلاستر ها:

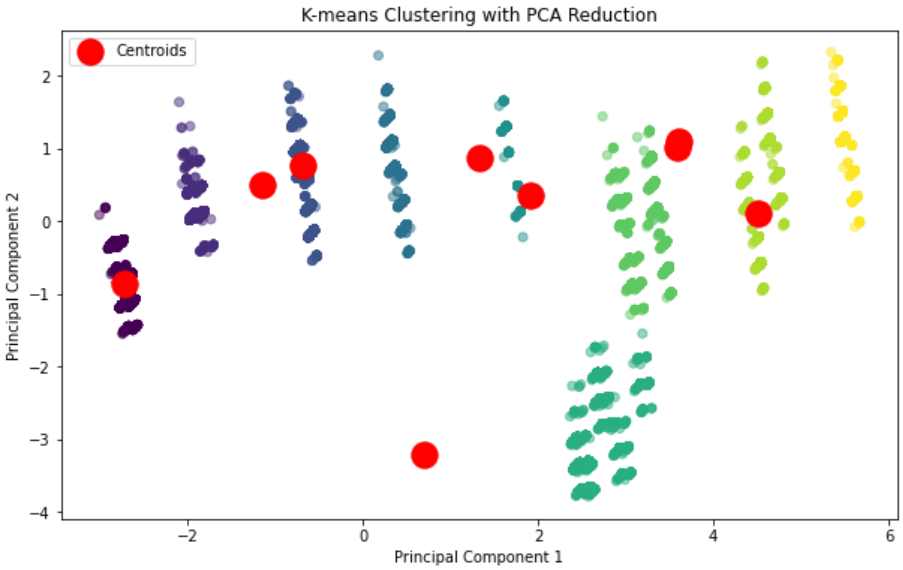
کلاستر ۱۰:

کلاستر۲۵:

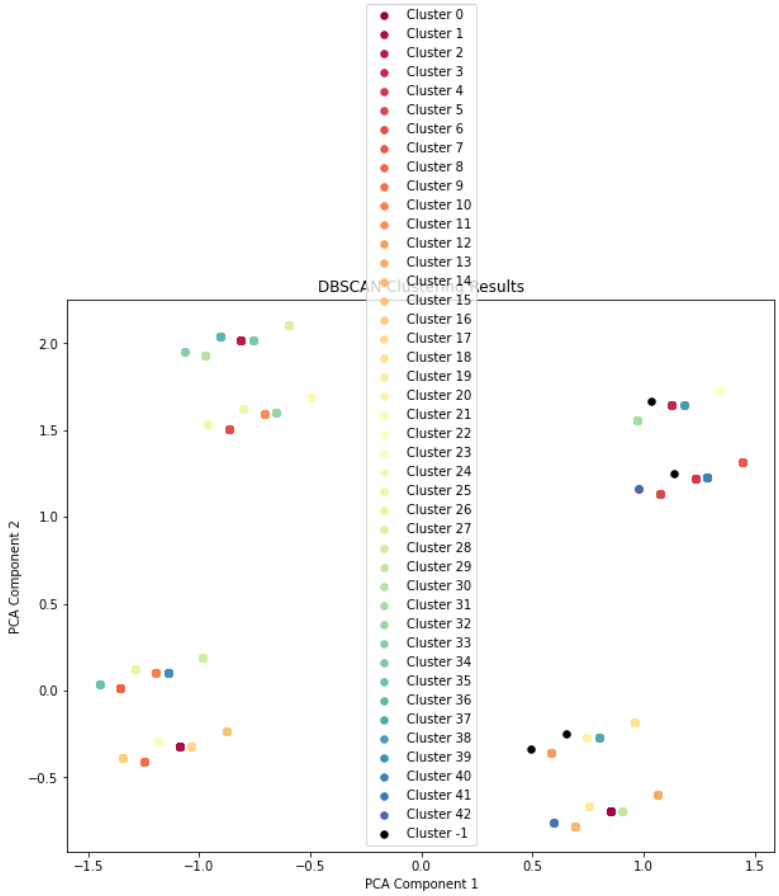
٤. فاز ٤



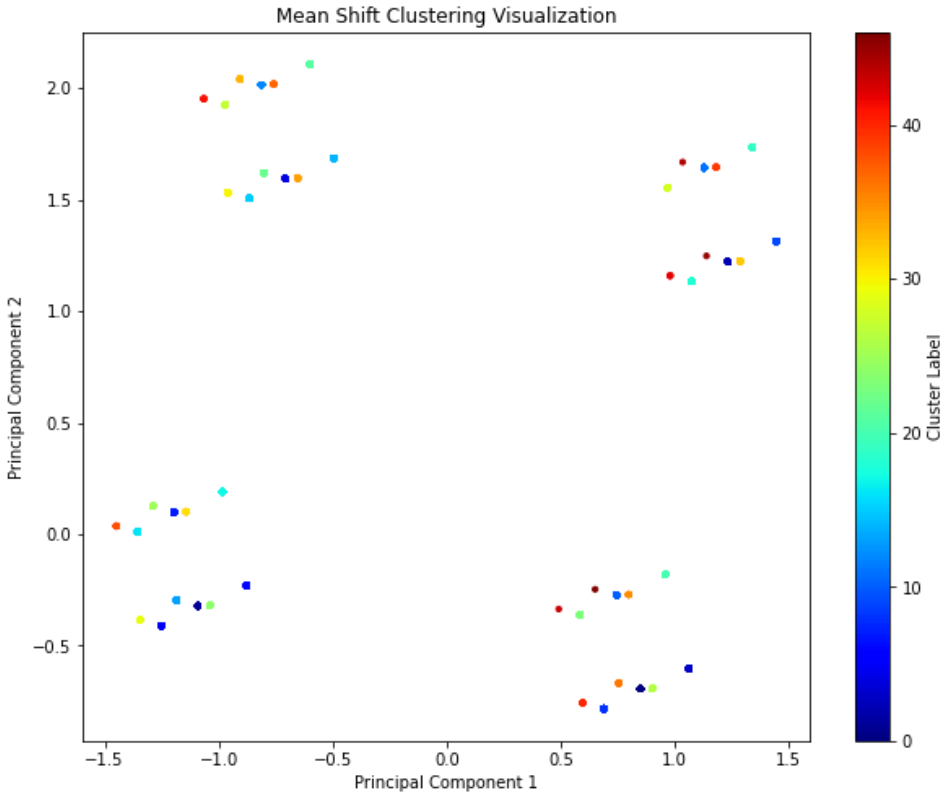
Kmeans 1


K-means Clustering with PCA Reduction
Centroids
Principal Component 2
Principal Component 1
2
1
0
−1
−2
−3
−4
−2
0
2
4
6

Kmeans2


Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9
Cluster 10
Cluster 11
Cluster 12
Cluster 13
Cluster 14
Cluster 15
Cluster 16
Cluster 17
Cluster 18
Cluster 19
Cluster 20
Cluster 21
Cluster 22
Cluster 23
Cluster 24
Cluster 25
Cluster 26
Cluster 27
Cluster 28
Cluster 29
Cluster 30
Cluster 31
Cluster 32
Cluster 33
Cluster 34
Cluster 35
Cluster 36
Cluster 37
Cluster 38
Cluster 39
Cluster 40
Cluster 41
Cluster 42
Cluster -1
PCA Component 2
PCA Component 1
2.0
1.5
1.0
0.5
0.0
−0.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

Dbscan



Meanshift

همانطور که پیداست meanshiftوDbscan دارای خوشه های بیشتری هستند و از امتیاز و مشاهده خوشه هایشان پیداست کیفیت بالاتری نسبت به keamnsدارند ولی kmeans1 نیز دارای خوشه های قابل قبول با کیفیت نسبتا بالاتری از kmeans2 می باشد زیرا فیچرهای بی کیفیت رنگ چشم و صورت در آن تاثیر گذار نیست.

۵.فاز ۵

در بررسی خوشه بندی بدون رنگ پوست و چشم نتایج به صورت زیر است:

```
Cluster 0: Nearest 50 points belong to clusters:
Cluster
0    50
Name: count, dtype: int64

Cluster 1: Nearest 50 points belong to clusters:
Cluster
1    50
```

```
Name: count, dtype: int64

Cluster 2: Nearest 50 points belong to clusters:
Cluster
2    50
Name: count, dtype: int64

Cluster 3: Nearest 50 points belong to clusters:
Cluster
3    50
Name: count, dtype: int64

Cluster 4: Nearest 50 points belong to clusters:
Cluster
4    50
Name: count, dtype: int64

Cluster 5: Nearest 50 points belong to clusters:
Cluster
5    50
Name: count, dtype: int64

Cluster 0: Nearest 3000 points belong to clusters:
Cluster
0    3000
Name: count, dtype: int64

Cluster 1: Nearest 3000 points belong to clusters:
Cluster
1    3000
Name: count, dtype: int64

Cluster 2: Nearest 3000 points belong to clusters:
Cluster
2    3000
Name: count, dtype: int64

Cluster 3: Nearest 3000 points belong to clusters:
Cluster
3    3000
Name: count, dtype: int64

Cluster 4: Nearest 3000 points belong to clusters:
Cluster
4    3000
Name: count, dtype: int64

Cluster 5: Nearest 3000 points belong to clusters:
Cluster
5    3000
Name: count, dtype: int64
```

همانطور که مشاهده می شود تمامی نقاط پیرامون کلاستر خود هستند.

و اما نتایج برای خوشه بندی با رنگ چشم و پوست:

```
Cluster 0: Nearest 50 points belong to clusters:
Cluster
0    50
Name: count, dtype: int64

Cluster 1: Nearest 50 points belong to clusters:
Cluster
1    50
Name: count, dtype: int64

Cluster 2: Nearest 50 points belong to clusters:
Cluster
2    50
Name: count, dtype: int64

Cluster 3: Nearest 50 points belong to clusters:
Cluster
3    50
Name: count, dtype: int64

Cluster 4: Nearest 50 points belong to clusters:
Cluster
4    50
Name: count, dtype: int64

Cluster 5: Nearest 50 points belong to clusters:
Cluster
5    50
Name: count, dtype: int64

Cluster 6: Nearest 50 points belong to clusters:
Cluster
6    50
Name: count, dtype: int64

Cluster 7: Nearest 50 points belong to clusters:
Cluster
7    50
Name: count, dtype: int64

Cluster 8: Nearest 50 points belong to clusters:
Cluster
8    50
Name: count, dtype: int64

Cluster 0: Nearest 3000 points belong to clusters:
Cluster
0    3000
Name: count, dtype: int64
```

```
Cluster 1: Nearest 3000 points belong to clusters:
Cluster
1    1766
6    1234
Name: count, dtype: int64

Cluster 1: Misclassified points belong to clusters:
Cluster
6    1234
Name: count, dtype: int64

Cluster 2: Nearest 3000 points belong to clusters:
Cluster
2    3000
Name: count, dtype: int64

Cluster 3: Nearest 3000 points belong to clusters:
Cluster
3    3000
Name: count, dtype: int64

Cluster 4: Nearest 3000 points belong to clusters:
Cluster
4    3000
Name: count, dtype: int64

Cluster 5: Nearest 3000 points belong to clusters:
Cluster
5    3000
Name: count, dtype: int64

Cluster 6: Nearest 3000 points belong to clusters:
Cluster
6    2999
2       1
Name: count, dtype: int64

Cluster 6: Misclassified points belong to clusters:
Cluster
2    1
Name: count, dtype: int64

Cluster 7: Nearest 3000 points belong to clusters:
Cluster
7    1991
6    1009
Name: count, dtype: int64

Cluster 7: Misclassified points belong to clusters:
Cluster
6    1009
Name: count, dtype: int64
```

```
Cluster 8: Nearest 3000 points belong to clusters:
Cluster
5    2197
8     798
2       5
Name: count, dtype: int64

Cluster 8: Misclassified points belong to clusters:
Cluster
5    2197
2       5
Name: count, dtype: int64
```

این کلاستر ها در اطراف خود نقاطی از کلاستر های دیگر دارند.

این پدیده می تواند به دلایل زیر پدیدار شود:

۱ .ویژگی‌های بی‌ربط

ویژگی‌های بی‌ربط یا نویزی می‌توانند در محاسبات فاصله که KNN به آن‌ها وابسته است، ایجاد سردرگمی کنند. از آنجا که KNN از شباهت ویژگی‌ها برای تعیین نزدیک‌ترین همسایه‌ها استفاده می‌کند، ویژگی‌هایی که اطلاعات معناداری ارائه نمی‌دهند می‌توانند نتایج را تحریف کنند و باعث شوند نقاطی که واقعاً به یک خوشه تعلق ندارند به عنوان همسایه‌های نزدیک شناسایی شوند.

۲. Curse of Dimensionality

با افزایش تعداد ویژگی‌ها، کارایی KNN معمولاً کاهش می‌یابد به دلیل "نفرین ابعاد". در فضاهای با ابعاد بالا، نقاط داده پراکنده می‌شوند و فاصله‌ها بین نقاط تمایل دارند به هم نزدیک شوند که تشخیص مؤثر بین آن‌ها را دشوار می‌کند. این ممکن است منجربه طبقه‌بندی نادرست شود.

۳ .مقیاس‌دهی ویژگی‌ها

اگر ویژگی‌ها در مقیاس‌های مختلفی باشند (مثلاً یک ویژگی از ۰ تا ۱ و دیگری از ۰ تا ۱۰۰۰ متغیر باشد)، KNN به سمت ویژگی‌هایی با مقیاس بزرگ‌تر تمایل خواهد داشت. نرمال‌سازی یا استانداردسازی مناسب ویژگی‌ها برای اطمینان از اینکه همه به‌طور مساوی در محاسبات فاصله مشارکت دارند، حیاتی است.

٤ .همپوشانی خوشه‌ها

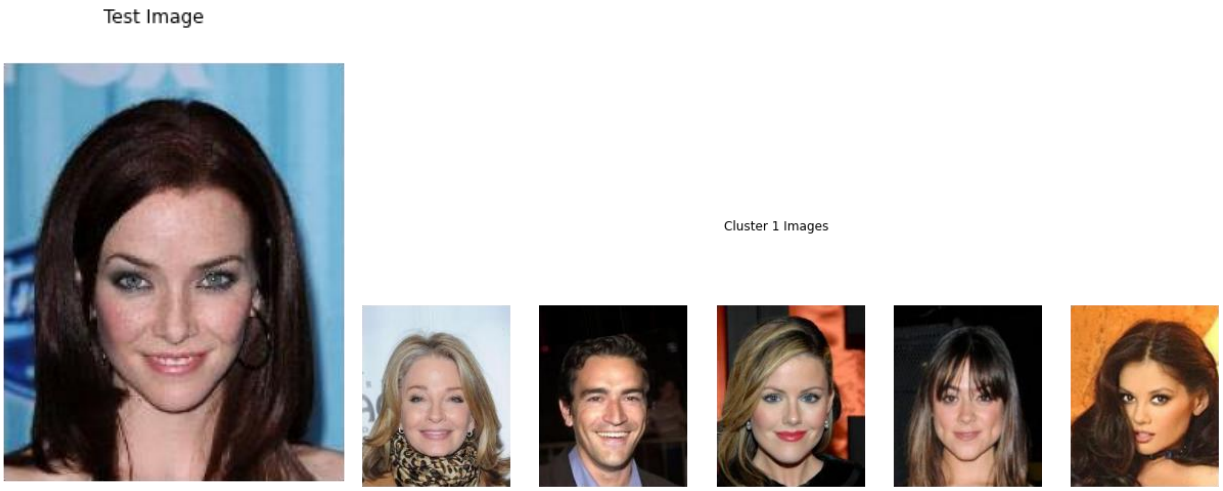
اگر خوشه‌های شما به طور قابل توجهی همپوشانی داشته باشند، KNN ممکن است نقاطی از خوشه‌های همسایه را به عنوان نزدیک‌ترین همسایگان به یک مرکز انتخاب کند. این موضوع به ویژه در مجموعه داده‌هایی با توزیع‌های پیچیده یا زمانی که تعداد خوشه‌ها نسبت به مقدار داده‌ها زیاد باشد، شایع است.

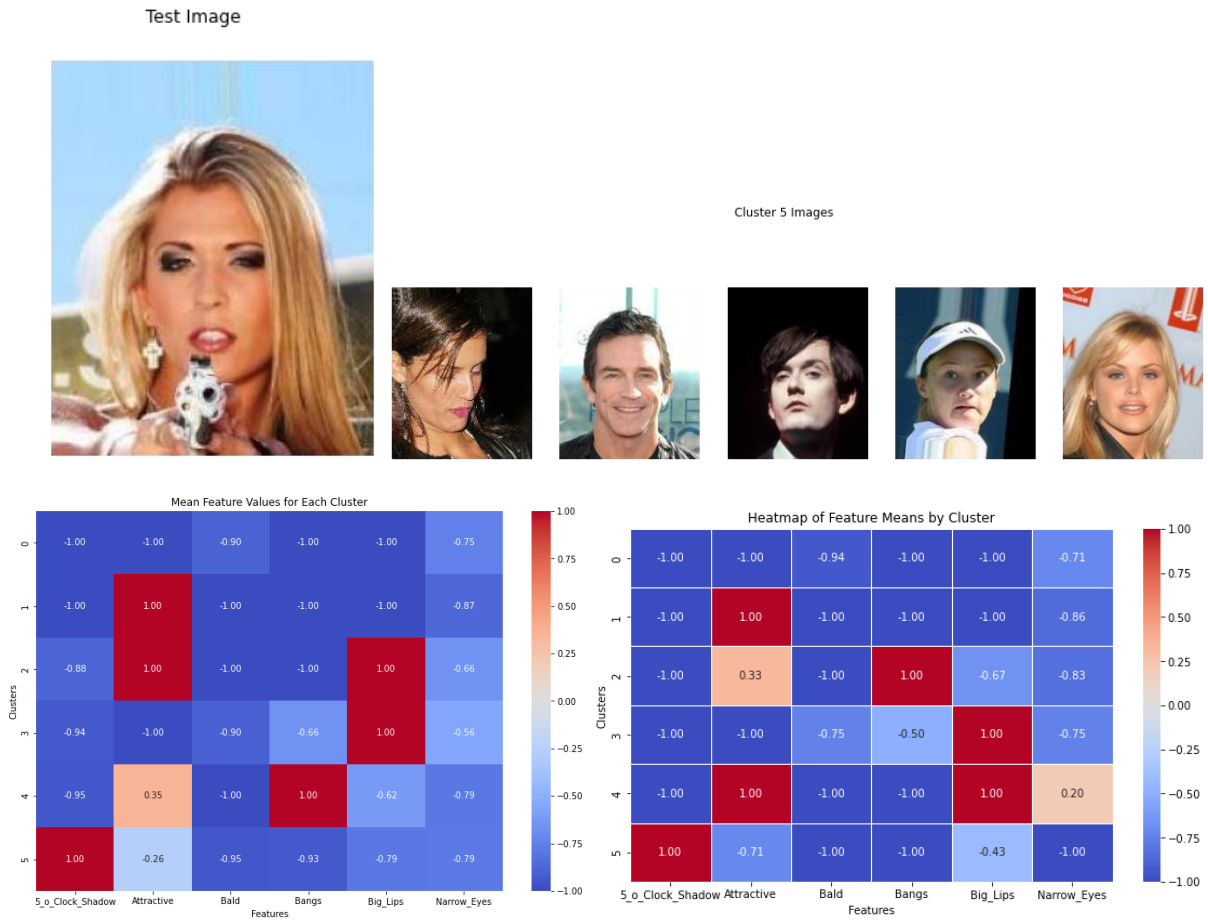
٥ .توزیع داده‌ها

توزیع کلی داده‌های شما نیز می‌تواند بر نتایج تأثیر بگذارد. اگر داده‌های شما به خوبی جدا نشده‌اند یا دارای نقاط پرت هستند، KNNممکن است این نقاط پرت را به عنوان نزدیک‌ترین همسایگان به مراکز طبقه‌بندی کند.

٦.فاز ٦

برخی نتایج خوشه بندی:





Train Test

با مشاهده این heatmap میتوان گفت دسته تست از ویژگی های دسته train با ضریب بالایی پیروی می کند و تفاوت آنها بدلیل کم بودن داده تست است.