# به نام خدا

# پرو‌ژه‌ی سوم داده‌کاوی

## خوشه‌بندی



نام و نام خانوادگی دانشجو : امیررضا صدیقین

## شماره دانشجویی دانشجو:953611133050

# بخش اول : پیش پردازش

در این بخش داده‌ها را پیش‌پردازش می‌شوند.

کد مربوط به این بخش در فایل preprocess.py است که شامل توابع زیر است:

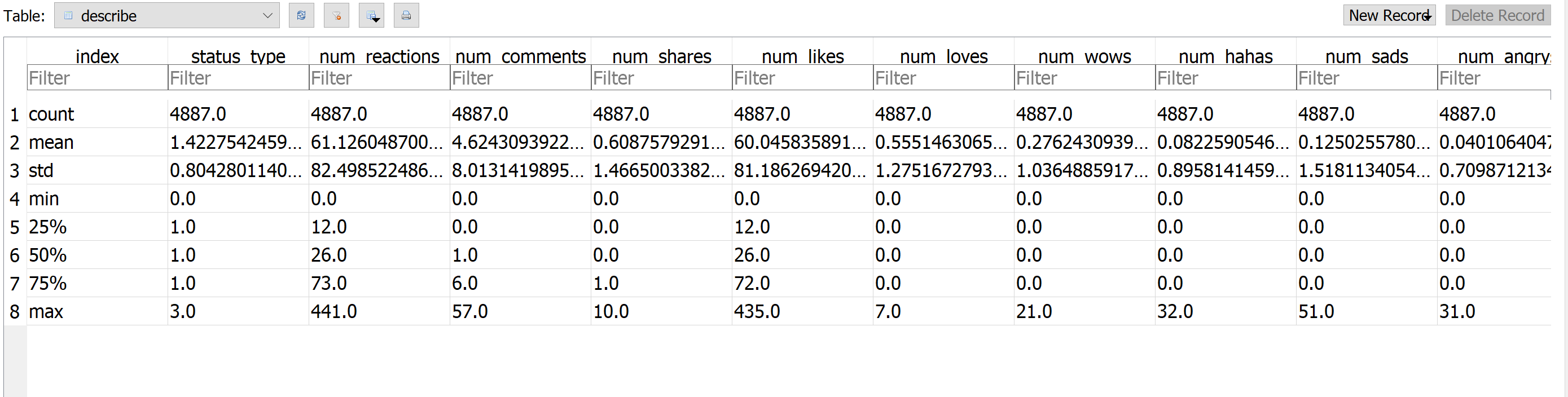
* Preprocessing که عملیات‌های اصلی پیش پردازش ر انجام می‌دهد.
* Drop\_numerical\_outliers : که با استفاده از قانون IQR به شرطی که iqr صفر نباشد داده‌های پرت را حذف می‌کند.
* Label\_encode : که داده‌هایی که نامینال هستند را به عددی تبدیل می‌کند.
* Missing\_data : داده‌های missing را حذف می‌کند.

مراحل کار به شرح زیر هستند:

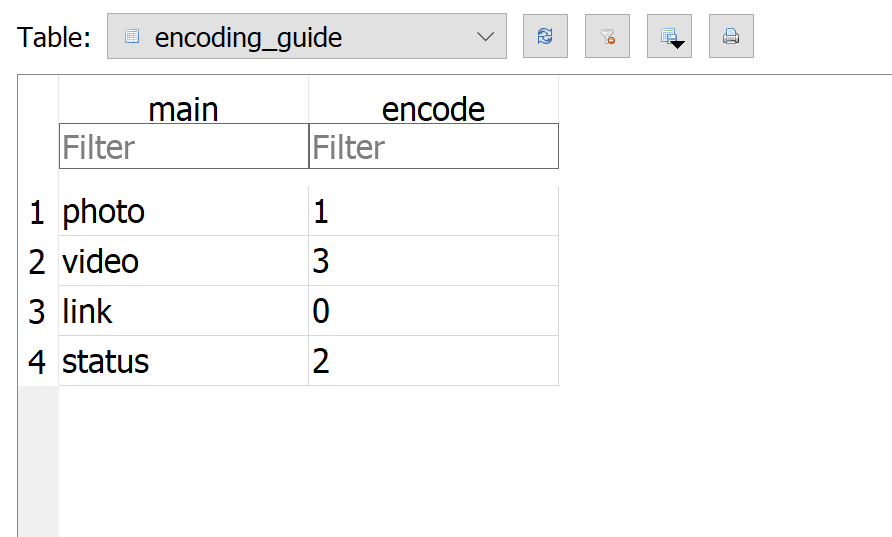
* Missing\_data ها را حذف می‌کنیم.
* ستون‌های status\_id و status\_published به دلیل نداشتن اطلاعات مفید برای خوشه‌بندی حذف می‌کنیم.
* ستون‌های columns1 تا 4 هم به دلیل عدم وجود دیتا حذف می‌کنیم.
* داده‌های پرت را حذف می‌کنیم.
* داده‌های نامینال به نامریک تبدیل می‌کنیم.
* داده‌ها را از لحاظ اسکیلی یکسان ‌می‌کنیم تا تاثیر یکسانی تقریبا داشته‌باشند .

خروجی‌ها :

* Outs/dtypes.txt نوع هر ستون را مشخص میکند.
* دیتابیس information.sqlite : جدول‌های این دیتابیس به صورت زیر است:
  + Describe : توصیفی از ستون‌ها مثل تعداد، میانگین،چارک ها می‌دهد.



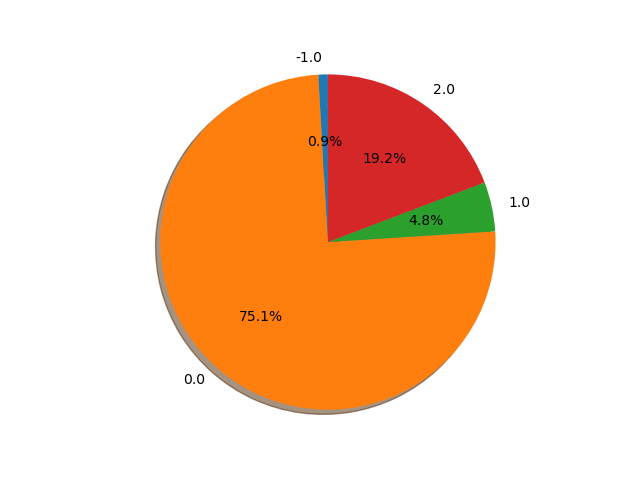
* + Before\_process : داده‌ها را قبل از process نشان می‌دهد.
  + After\_clear : داده‌ها بعد از حذف داده‌های پرت و missing\_value ها
  + Missing\_information : اطلاعات و توصیفی از داده‌های پرت
  + Information: داده‌های اصلی که در آینده از آن‌ها استفاده می‌شود و به عبارتی داده‌های پیش‌پردازش شده است.
  + Outliers : داده‌های پرت
  + Encoding\_guide : راهنمایی برای encoding



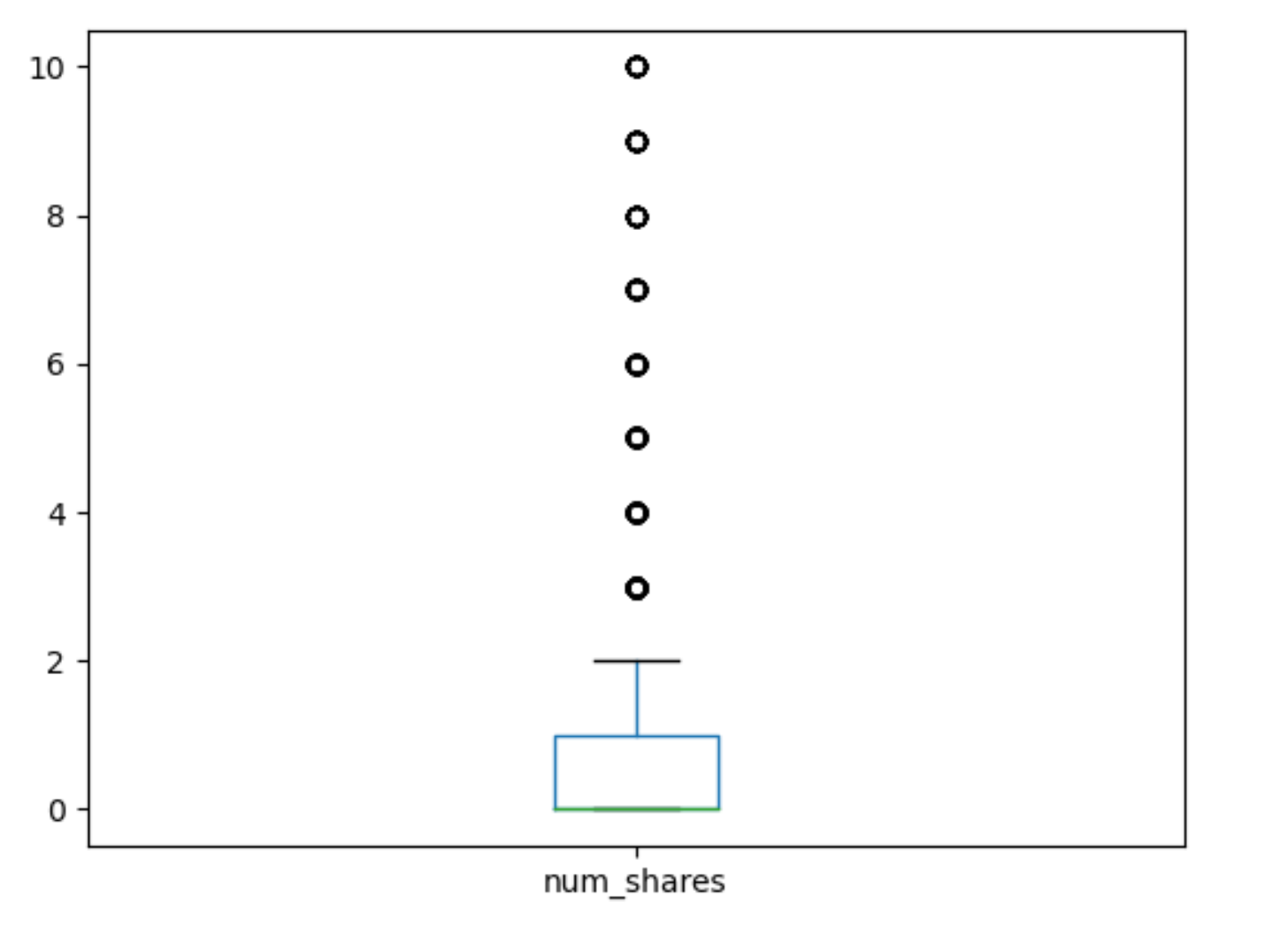
# بخش دوم : درک داده‌ها و بررسی و نمودار‌های داده‌ها:

کد این بخش در فایل distribution.py است که شامل توابع زیر است:

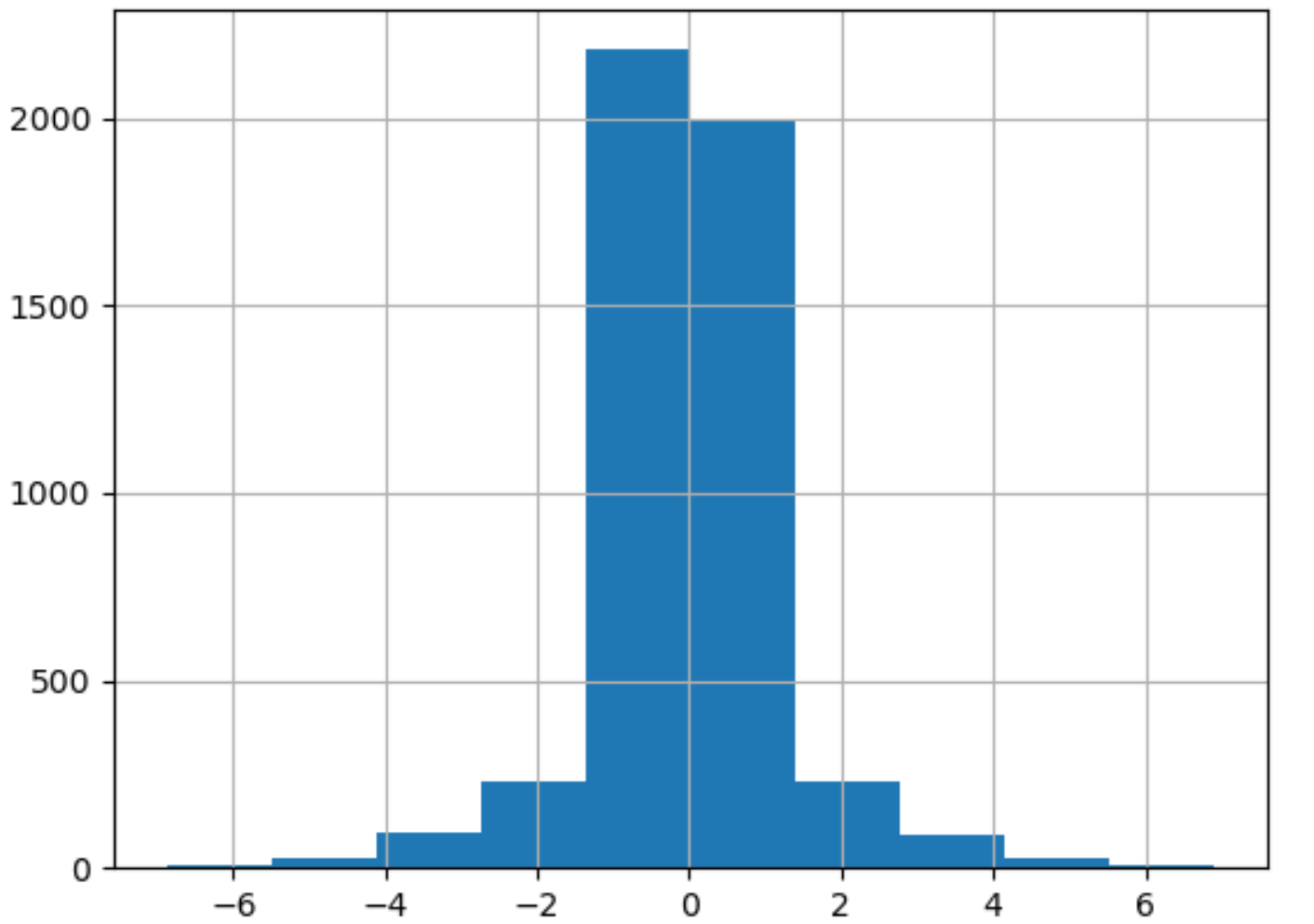
* Pie\_plots : نمودار‌های دایره ای برای داده ها می‌کشد.
  + خروجی این بخش فولدر outs\pie\_plots است که برای همه‌ی اتربیوت ها با نام خود نموداره دایره‌ای کشیده شده است.
  + مثال : نمودار دایره‌ای status\_type به صورت زیر است.



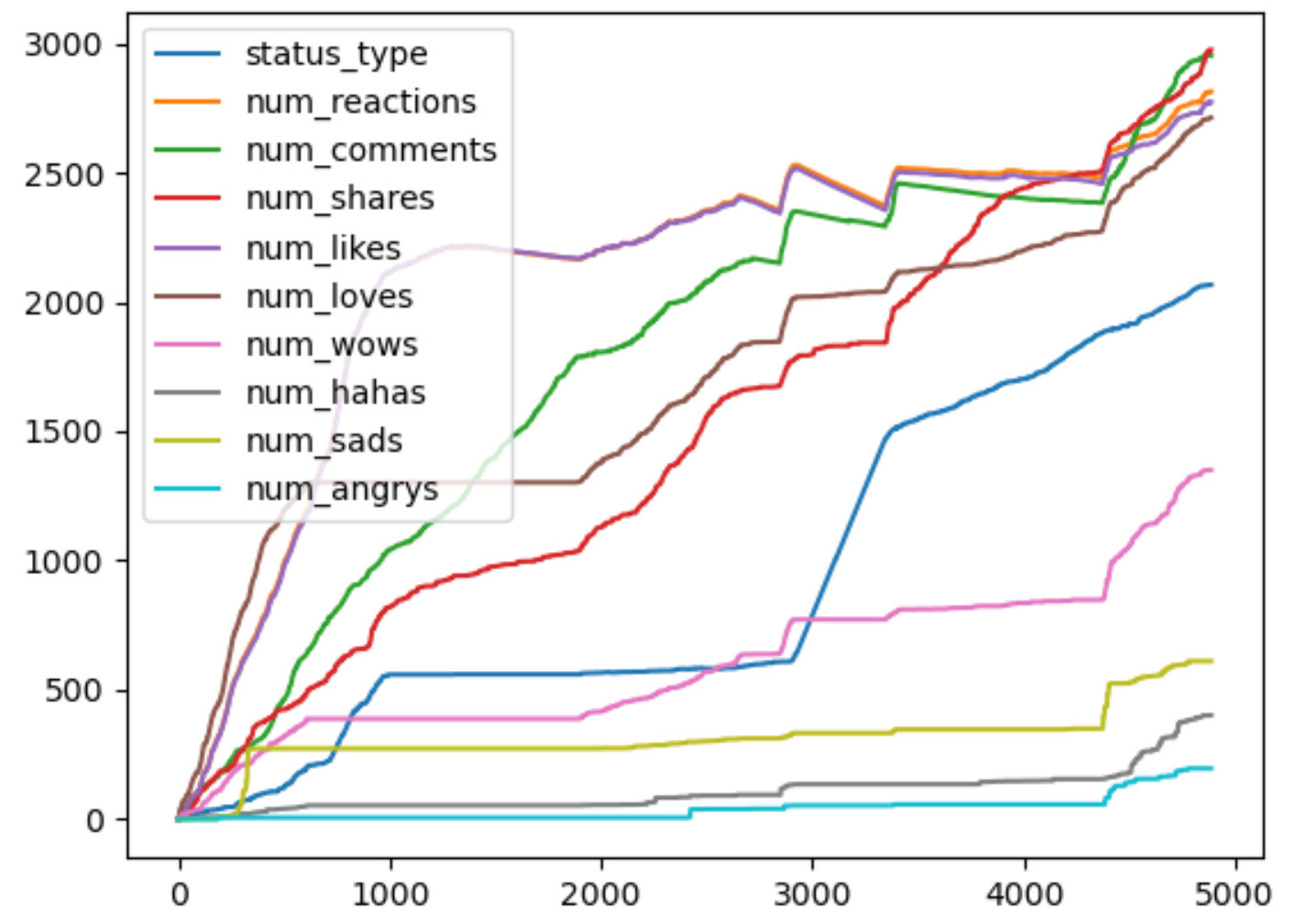
* Boxes : نمودار جعبه‌ای برای داده‌ها می‌کشد.
  + خروجی این بخش فولدر outs\boxes است که نمودار جعبه‌ای هر اتربیوت را با نام خود ذخیره کرده است.
  + مثال : برای اتربیوت num\_shares



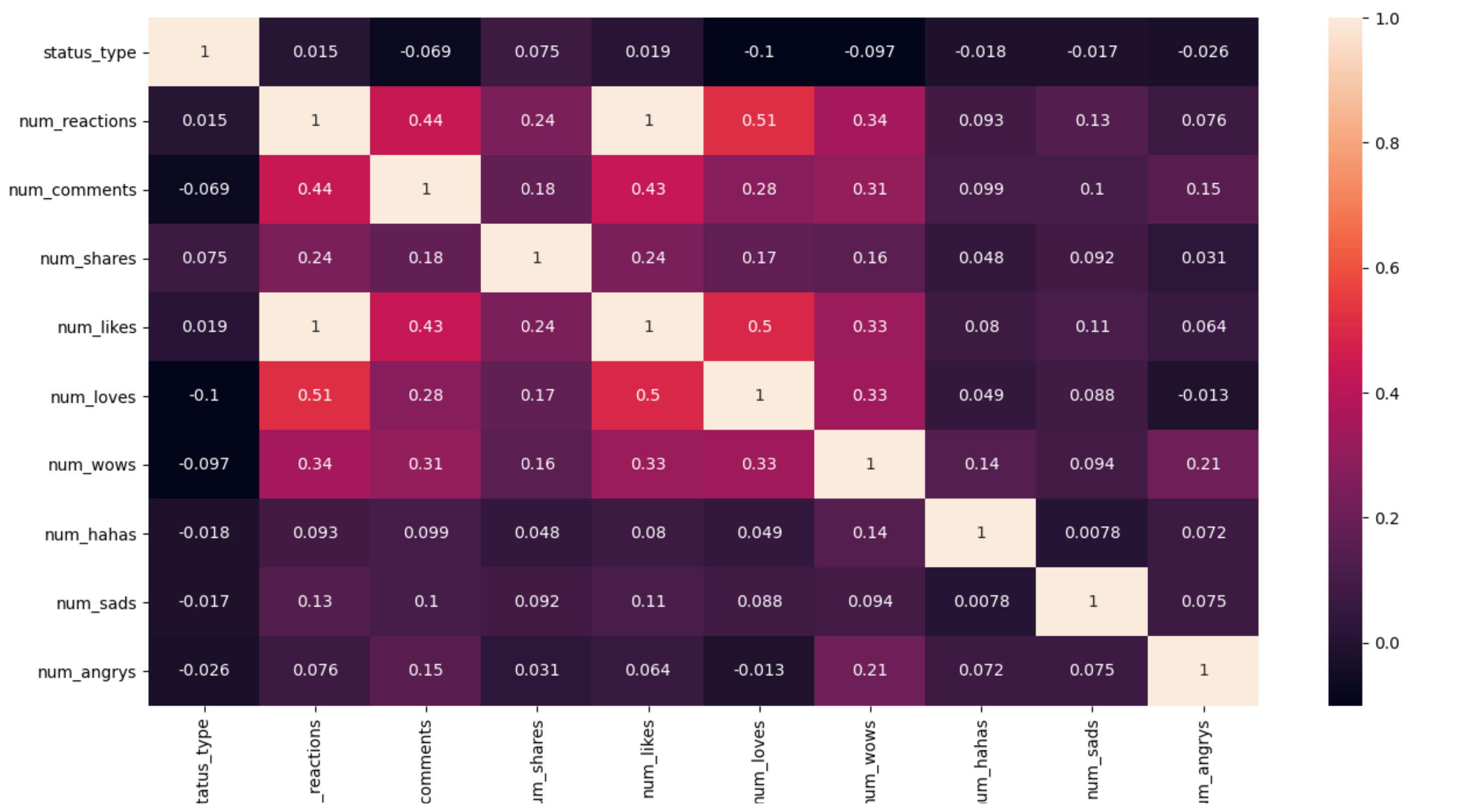
* Diff : نمودارهای هیستوگرام پراکندگی هر اتربیوت را نمایش می‌دهد.
  + خروجی این بخش outs\diff\_hists است.
  + مثال : برای اتربیوت num\_likes :



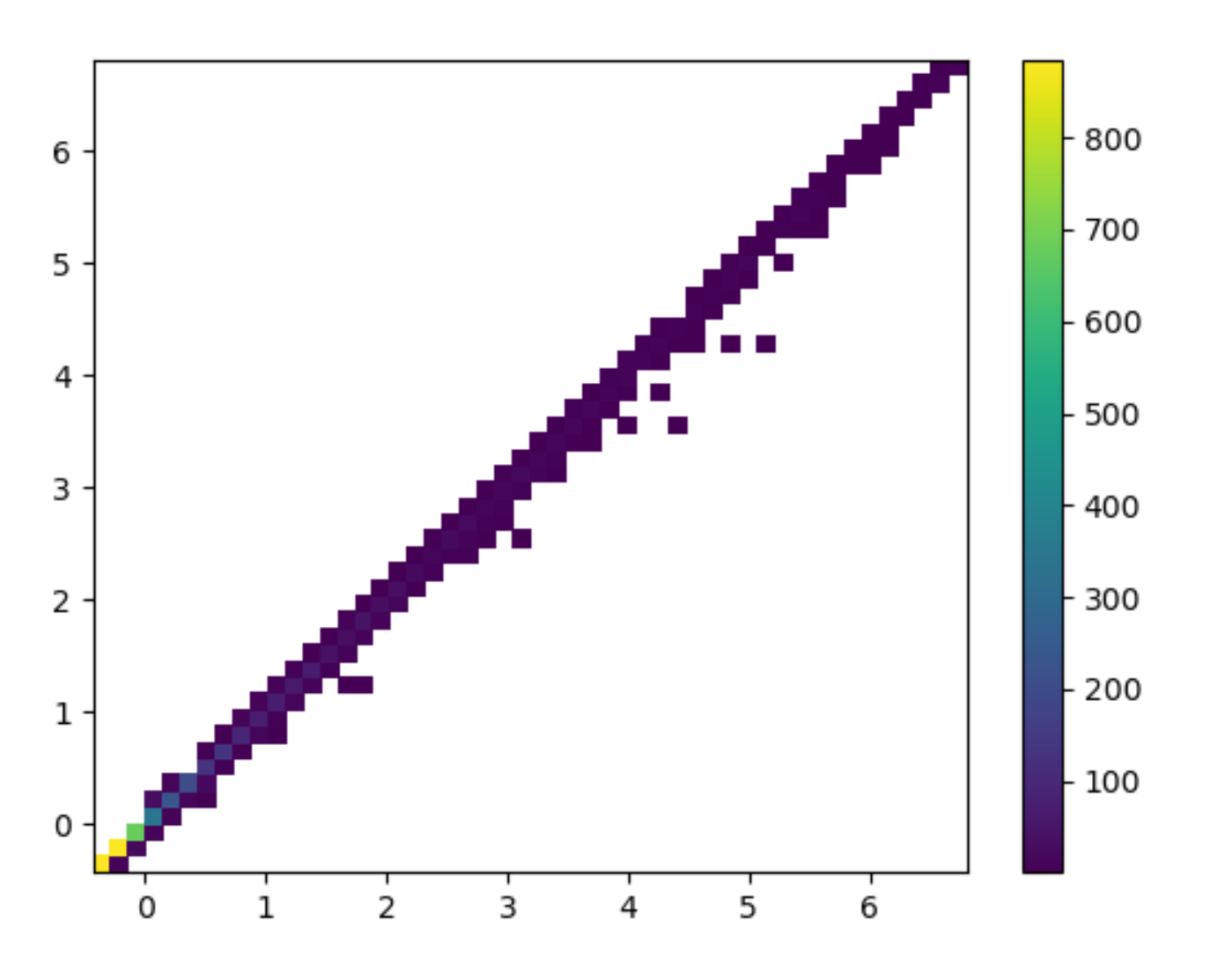
* Cumsum: نمودار cumsum برای اتربویت ها میکشد.
  + خروجی outs\cumsum.png است.

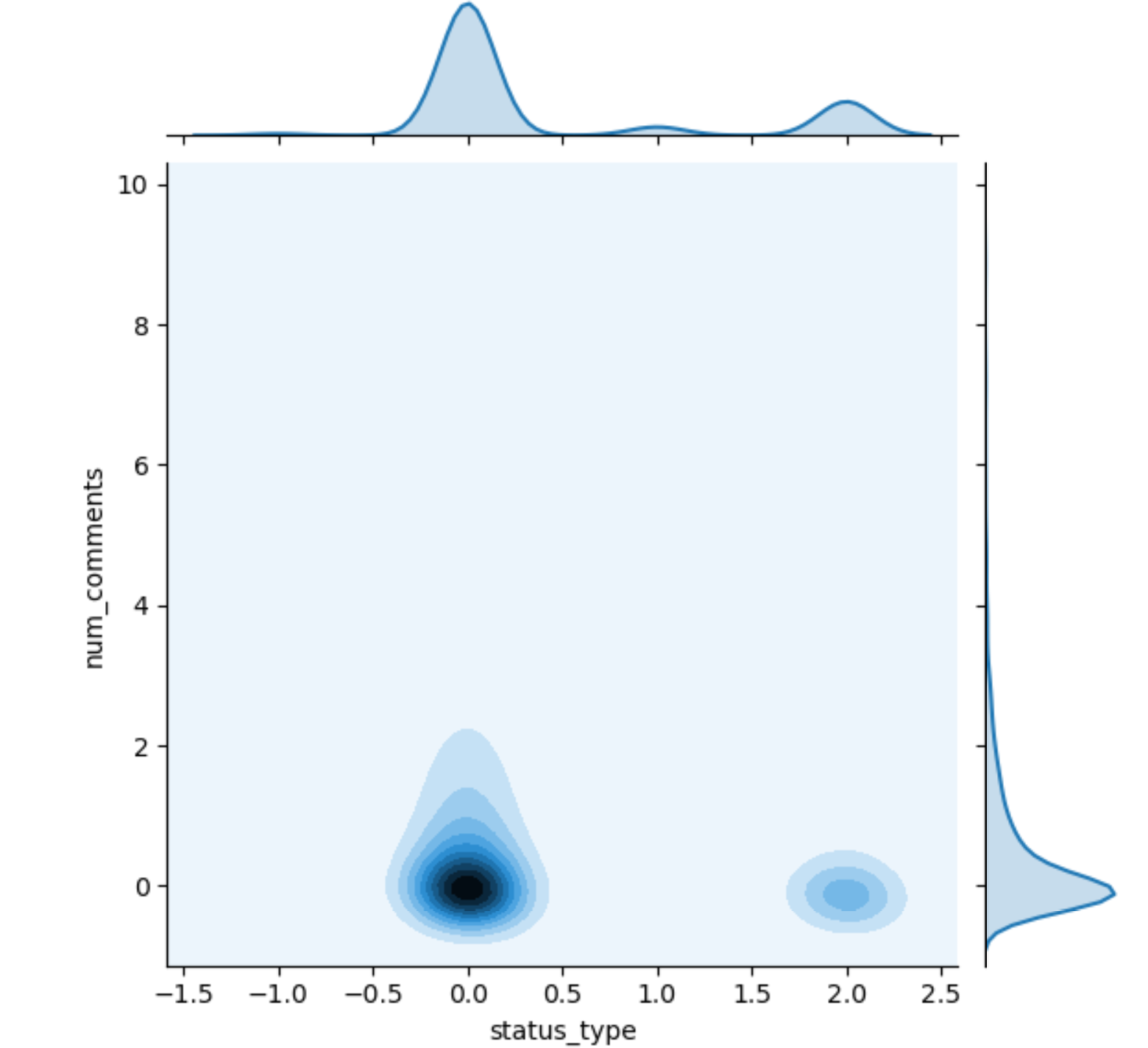


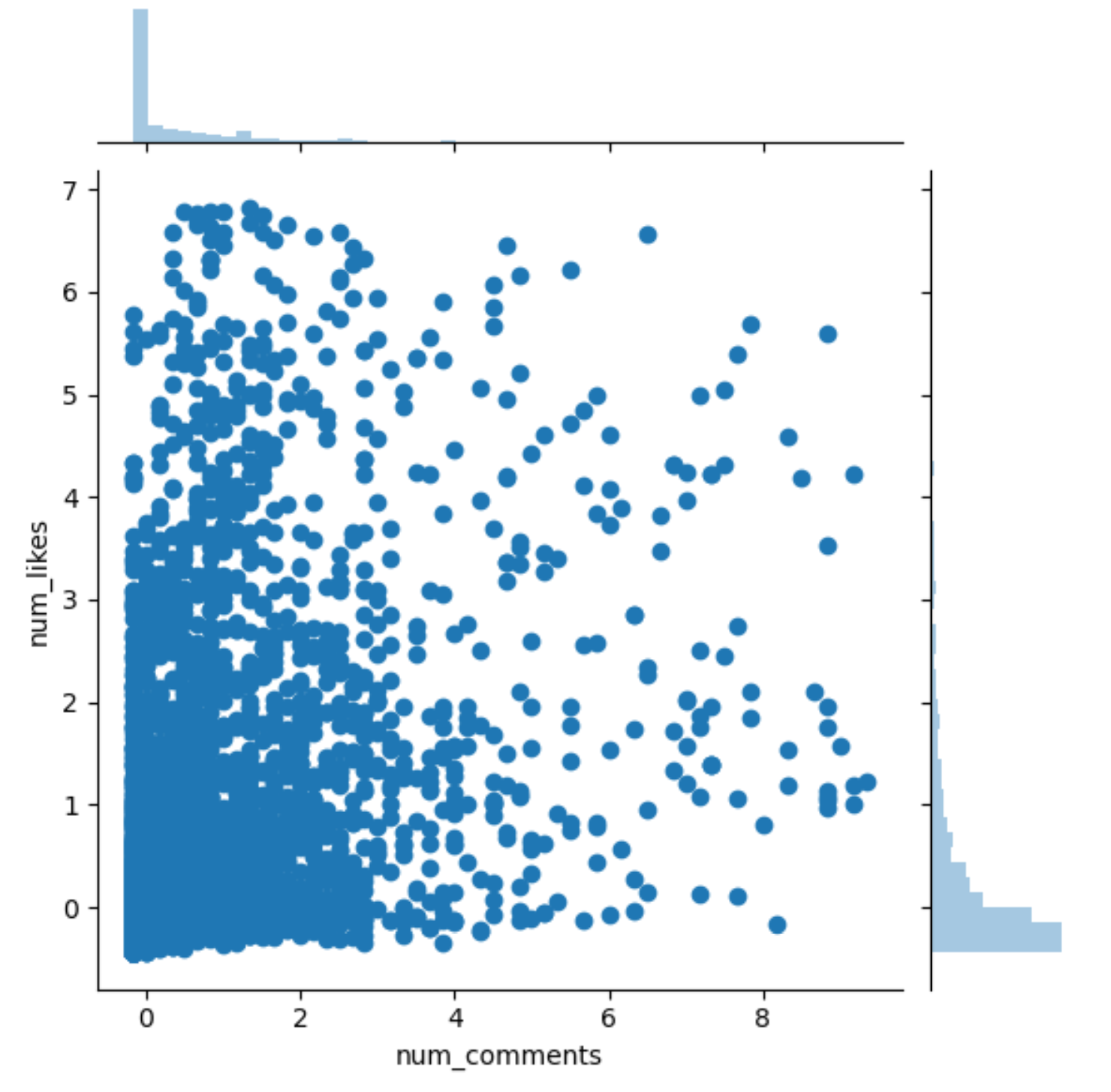
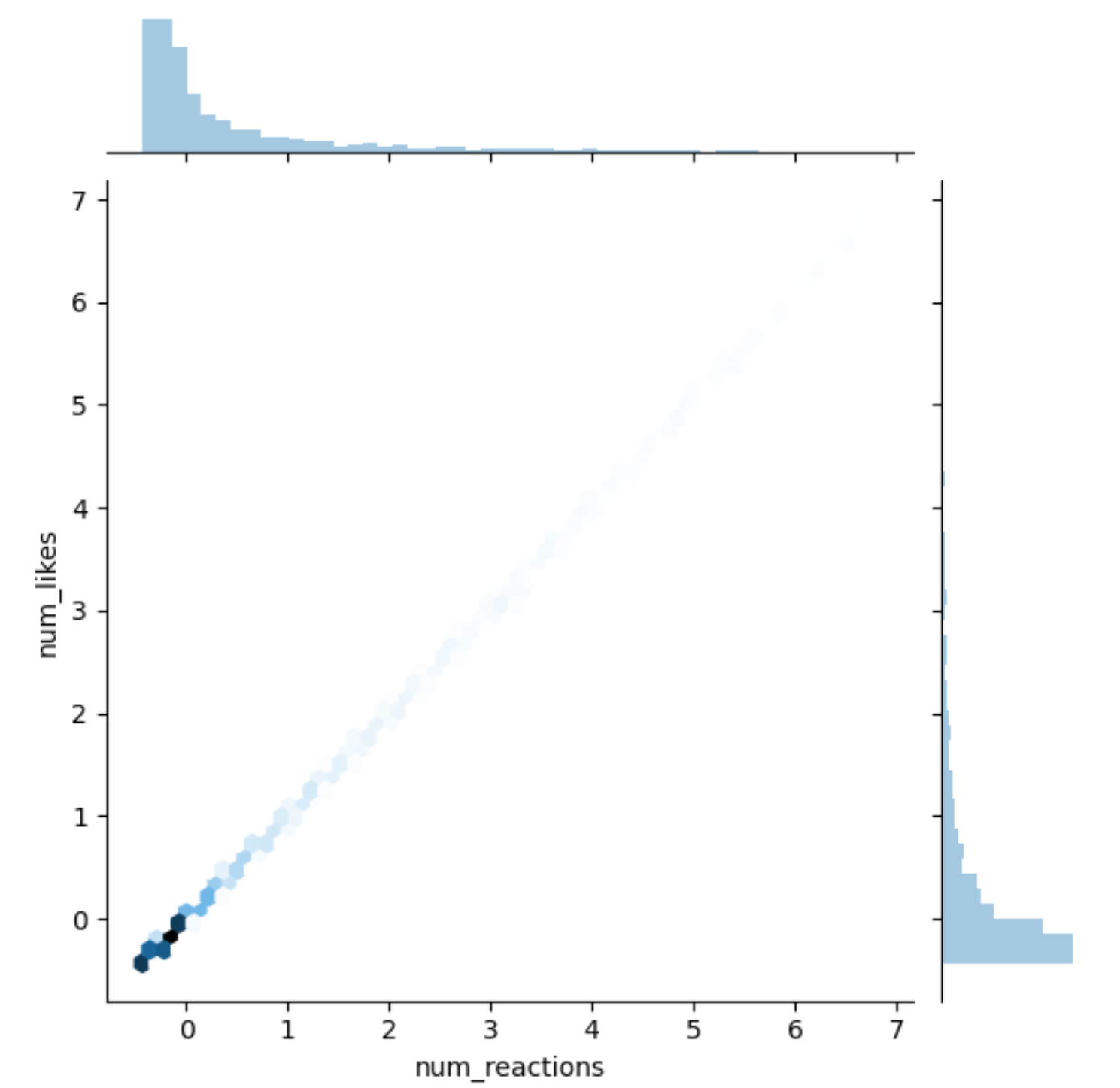
* Corr : جدول correlation داده ها را میکشد.
  + خروجی outs\corrolation.png است.



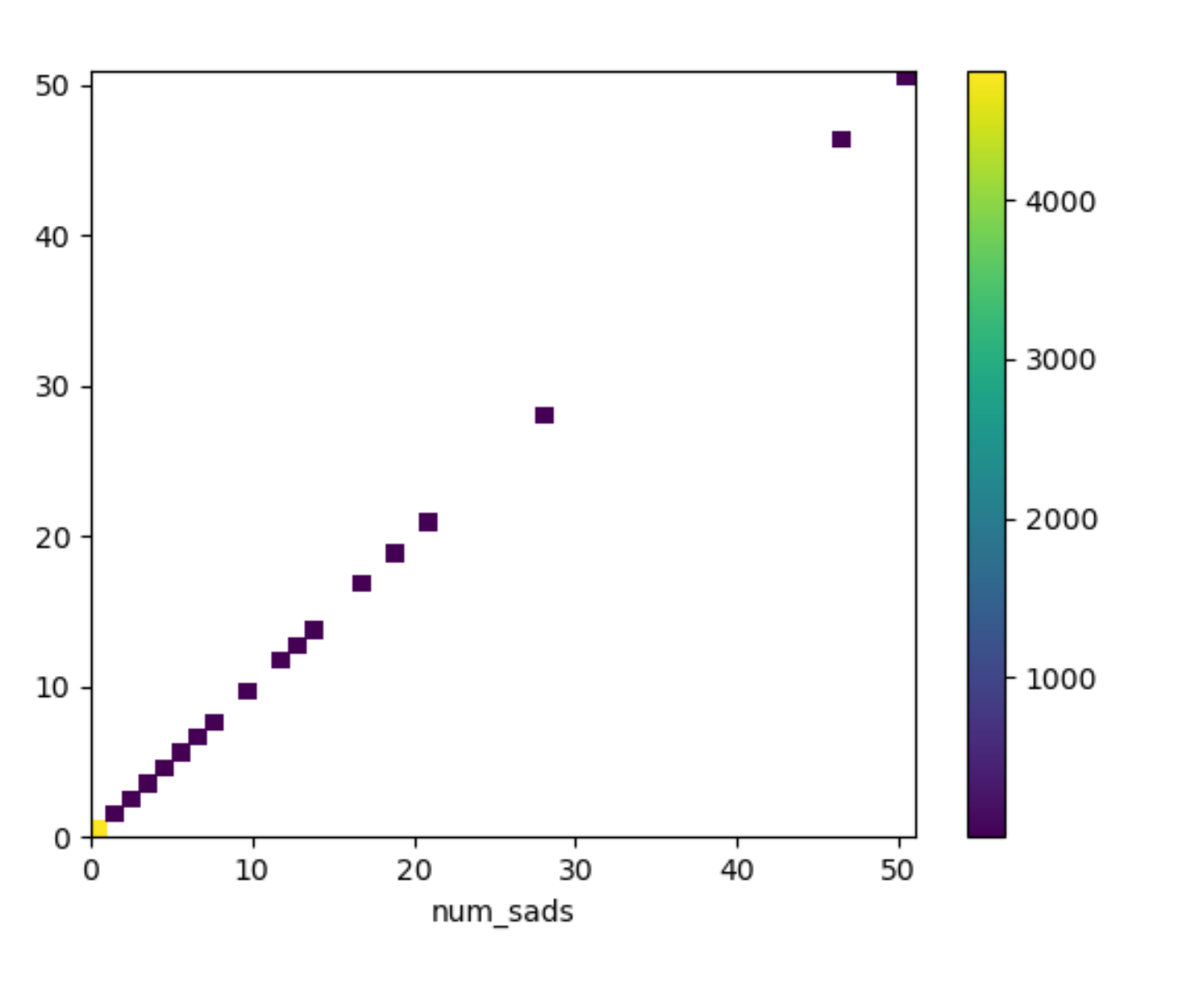
* Hist2d : به ازای همه‌ی حالت های دو به دویی داده ها نمودار نقاط را نمایش می‌دهد و در هر نقطه میزان تجمع را با رنگ مشخصی نمایش می‌دهد.
  + خروجی این بخش فولدر outs/hist2d است .
  + مثال : برای دو اتربیوت num\_reactions و num\_likes



* Density : چگالی داده‌ها را دو به دو نمایش می‌دهد.
  + خروجی آن فولدر outs\density است.
  + مثال : برای دو اتربیوت num\_comments و status\_type
  + 

* Point\_point\_plot : دو به دو اتربیوت ها را نقاطش را رسم می‌کند.
  + خروجی فولدر outs\point2point است.
  + مثال
  + 
* Hex\_bin : صورت دیگری از نمایش تراکم داده‌ها
  + خروجی فولدر outs\hex\_bin است.
  + مثال :
  + 

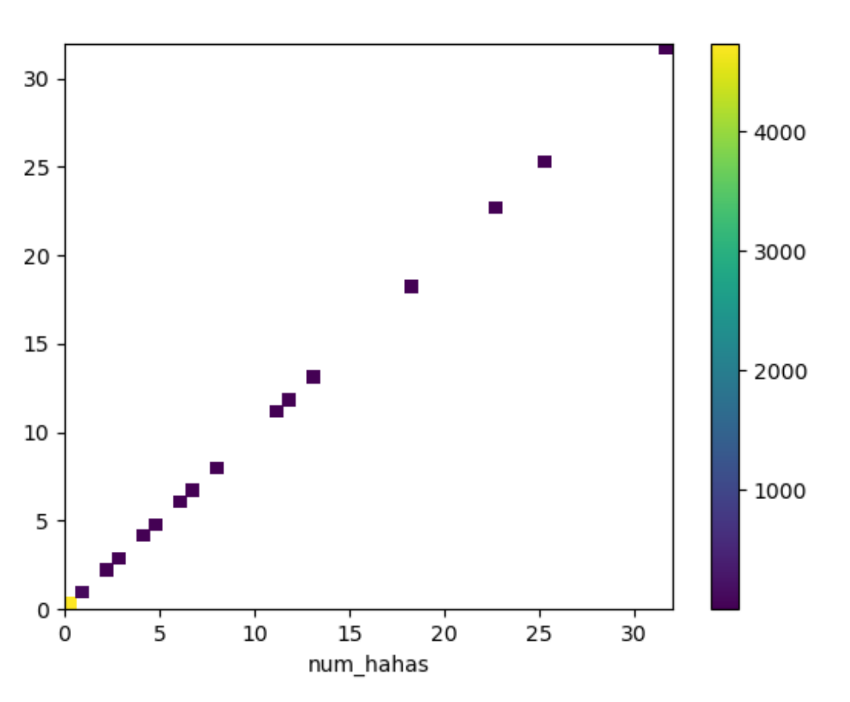
* Self\_hist2d : هر اتربیوت را با خودش نمایش می‌دهد که ببینیم چه اتربیوت هایی پراکندگی و تجمع خوبی برای کلاسترینگ دارد.
* خروجی در فولدر outs\self\_hist2d
* مثال :

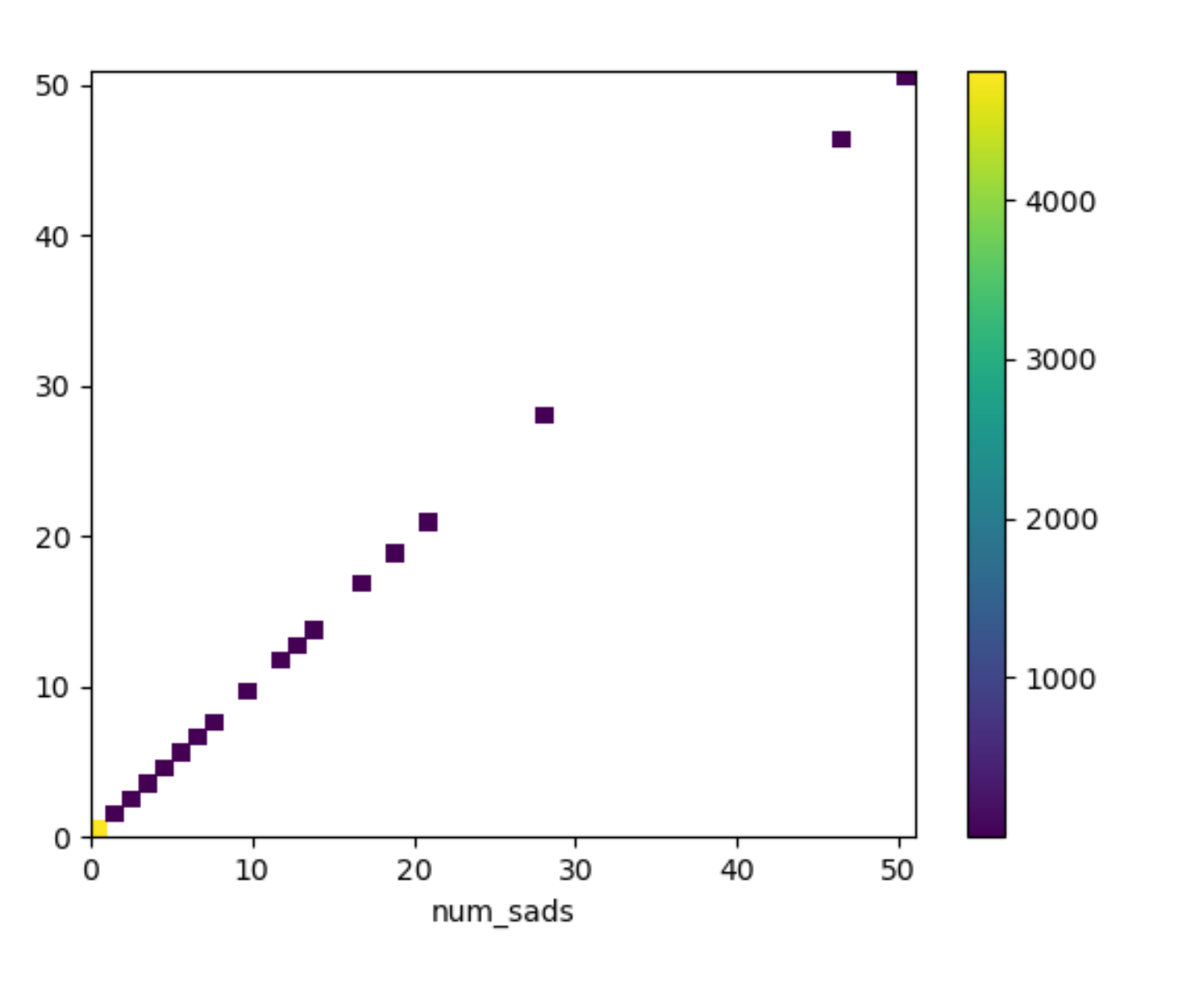


## تحلیل های این بخش

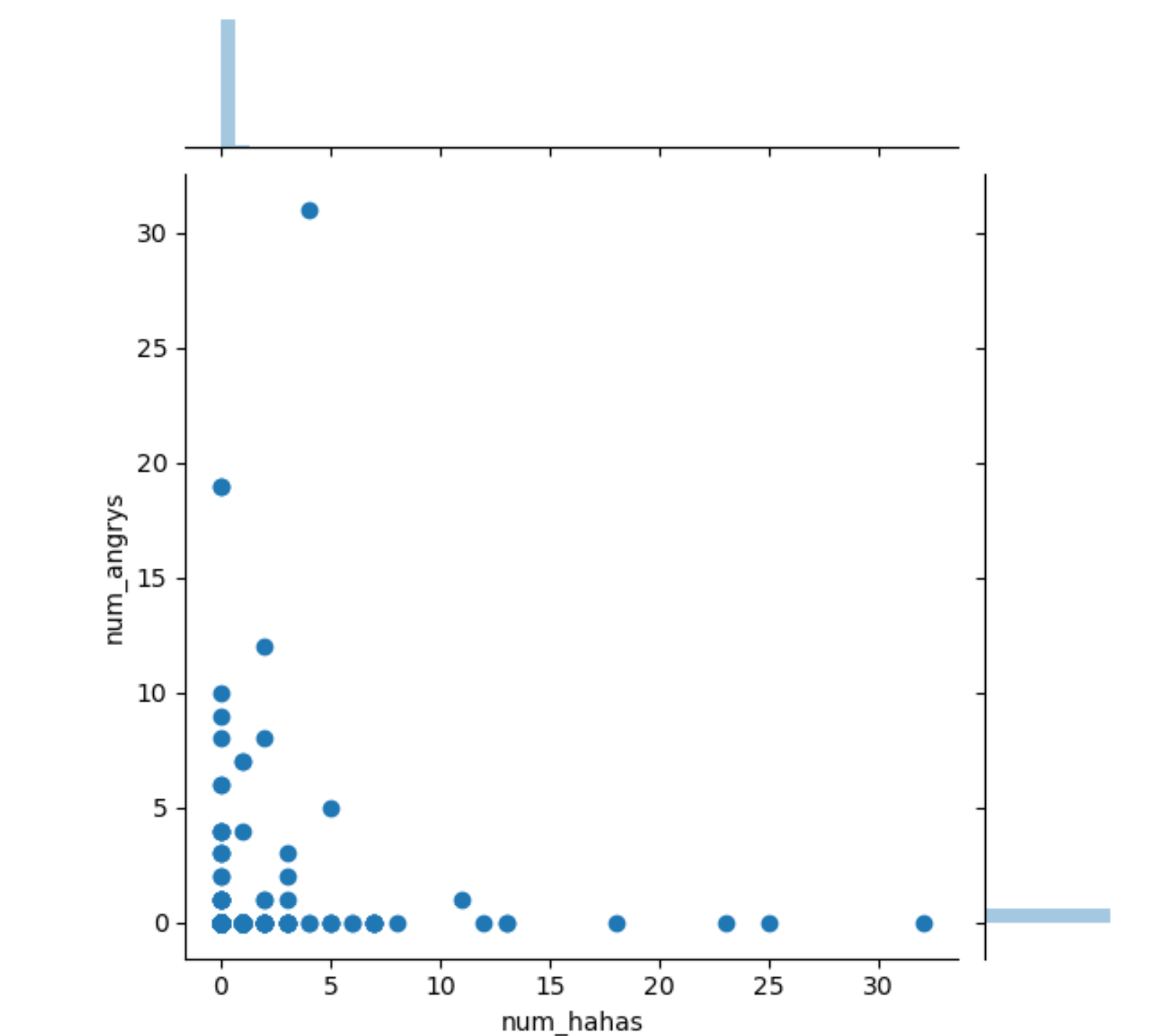
با توجه به جدول correlation رابطه‌ی بسیار بسیار قویی بین num\_likes و num\_reactions است و رفتار کاملا مشابهی دارند. و همچنین num\_likes و nume\_loves و num\_reactions رابطه‌های قویی باهم دارند.

با توجه به خروجی های self\_hist2d اتربیوت های num\_hahas و num\_sads به خوبی از هم تفکیک شده اند و تجمع در هر بخش هم زیاد است.



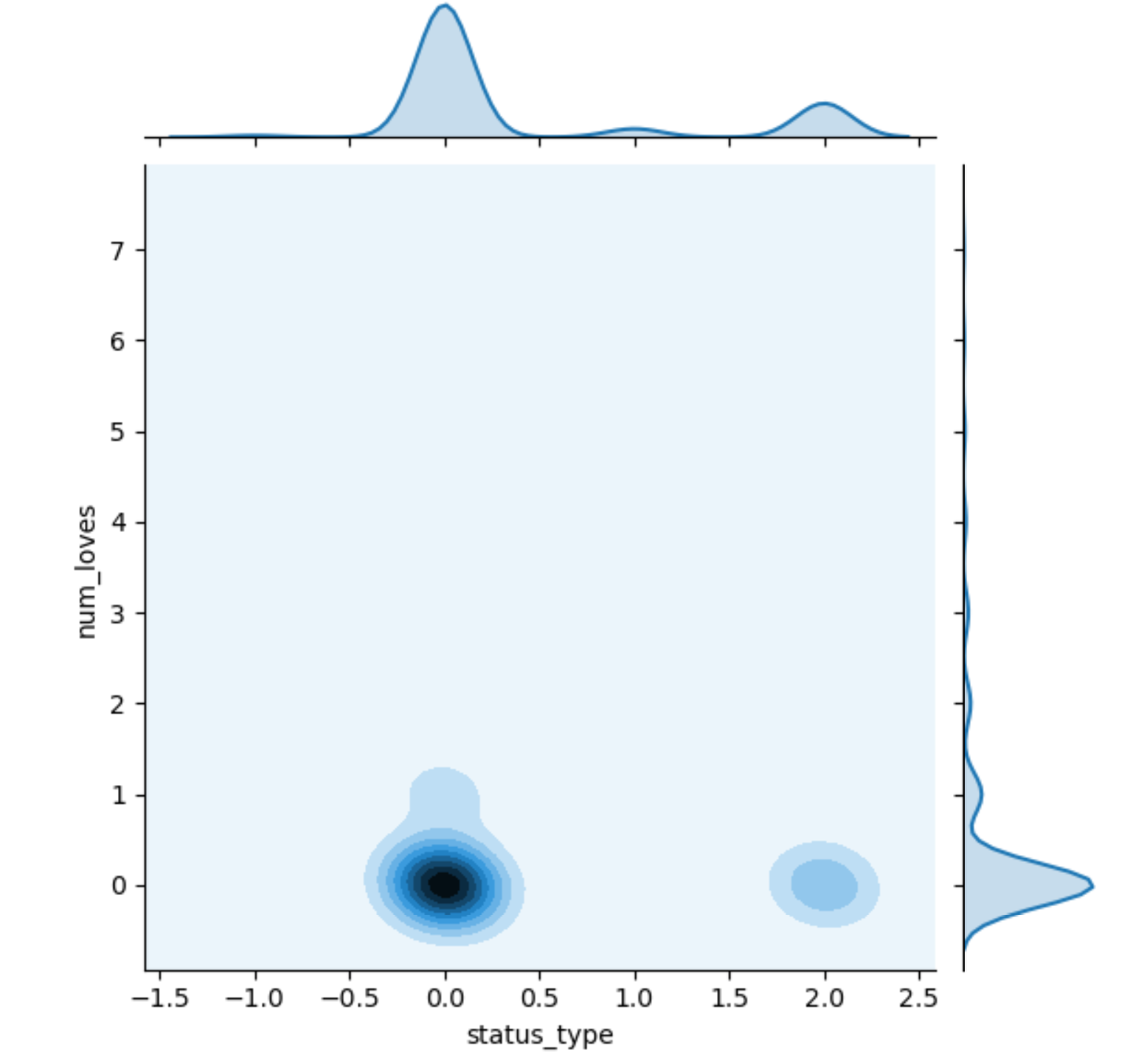


با توجه به نمودار های point to point به نظر میرسد num\_hahas و num\_angry بهترین تفکیک شدگی را دارند.



در نمودارهای چگالی نمودارهایی که نوع status\_type وجود دارند به خوبی تفکیک شده اند.

برای مثال :



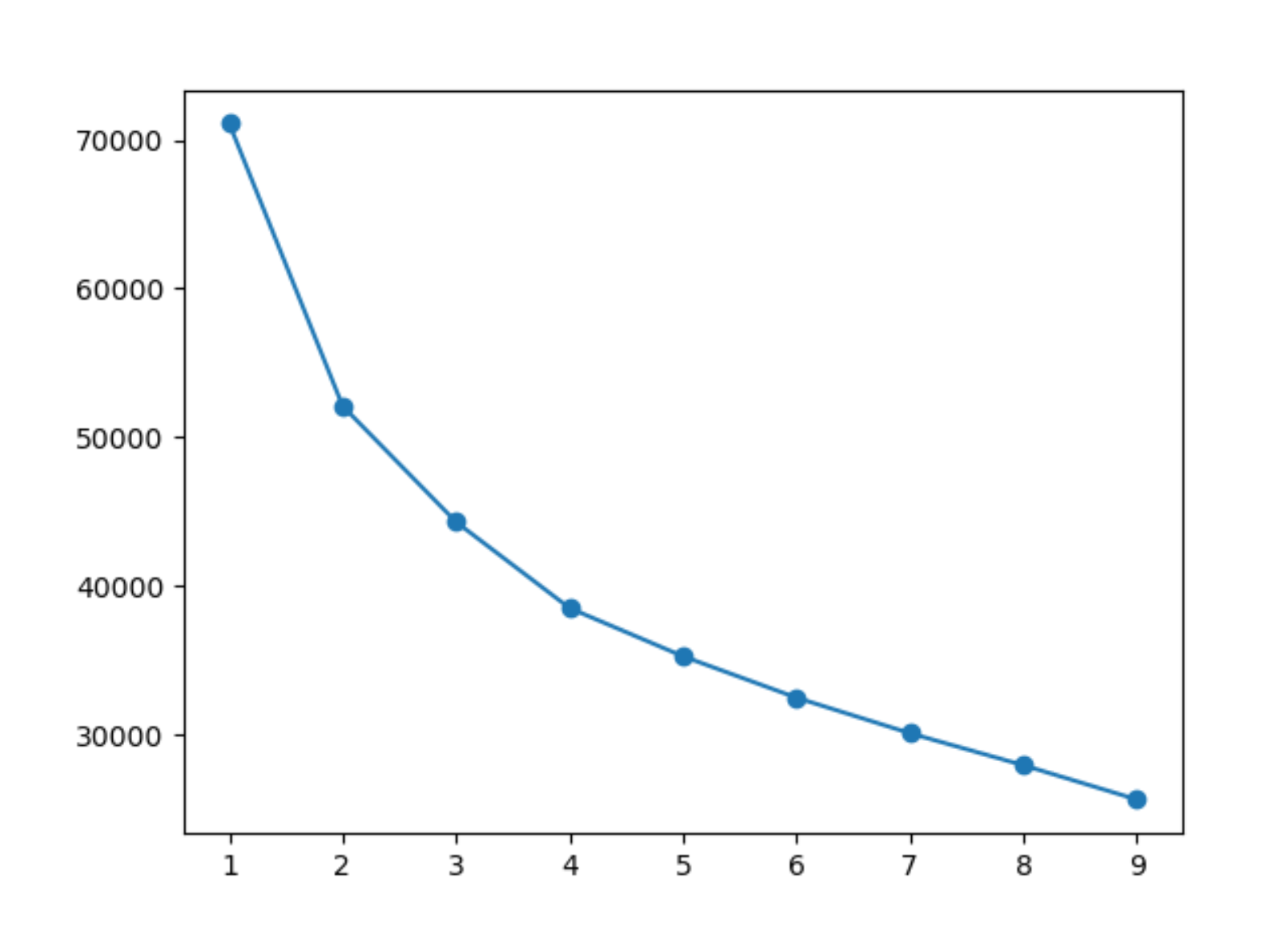
در مجموع مشاهدات در همه‌ی نمودارهای کشیده شده در همه‌ی حالات به نظر میرسد num\_hahas و num\_comment و num\_sad در مجموع بهترین اتربیوت ها برای کلاستر کردن هست.همچنین status\_type نیز به نسبت خوب تفکیک کرده است.

# بخش سوم

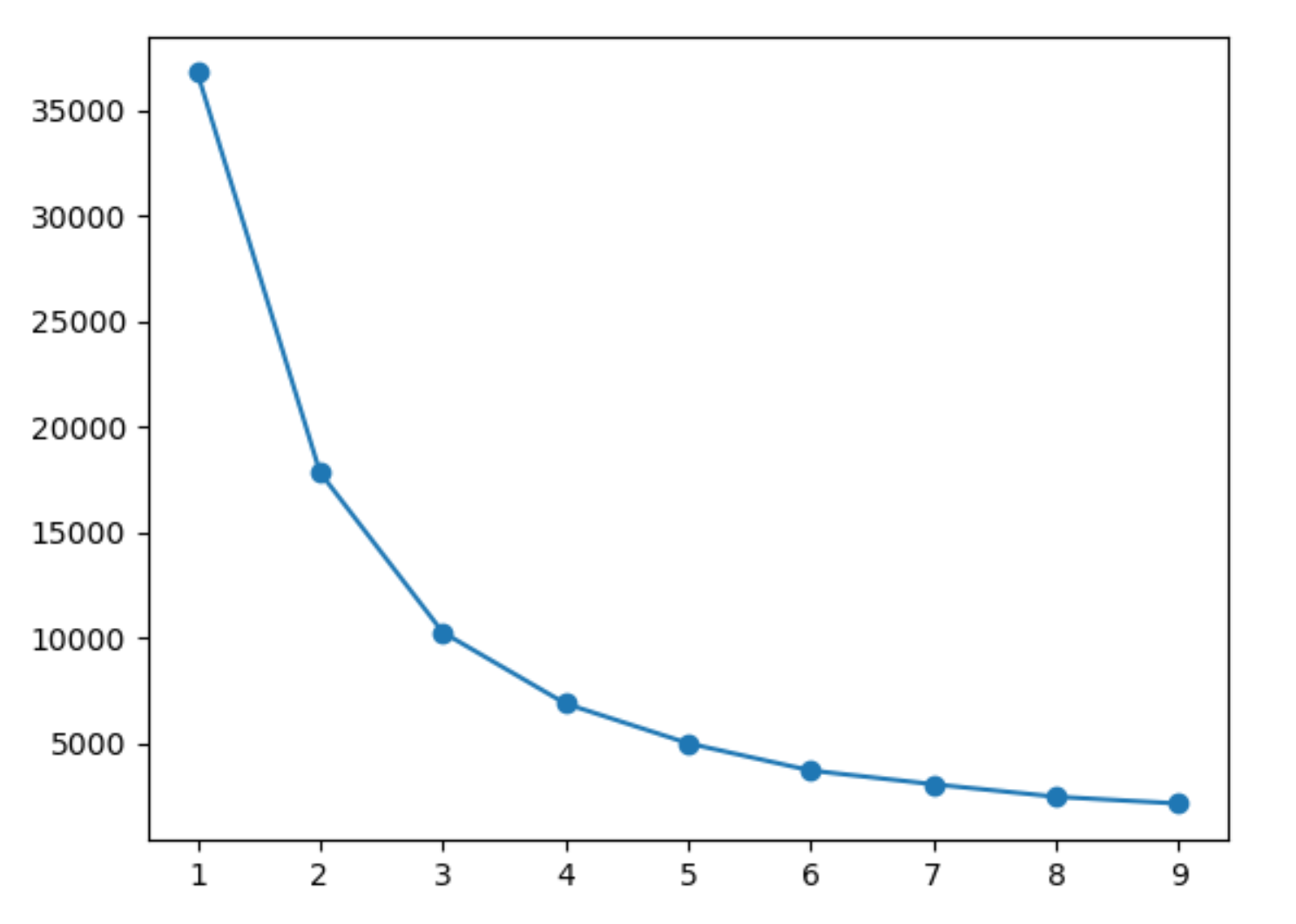
## الگوریتم k\_means

این بخش در فایل kmeans.py وجود دارد که شامل مراحل زیر است:

* با استفاده از تابع elbow\_inertia نمودار elbow آن را در دو حالت کاهش بعد و عدم کاهش بعد نمایش داده ایم :
  + نمودار elbow با کاهش بعد (فایل outs\elbow.png)



* + نمودار elbow بدون کاهش بعد (فایل outs\PCA\_elbow.png)

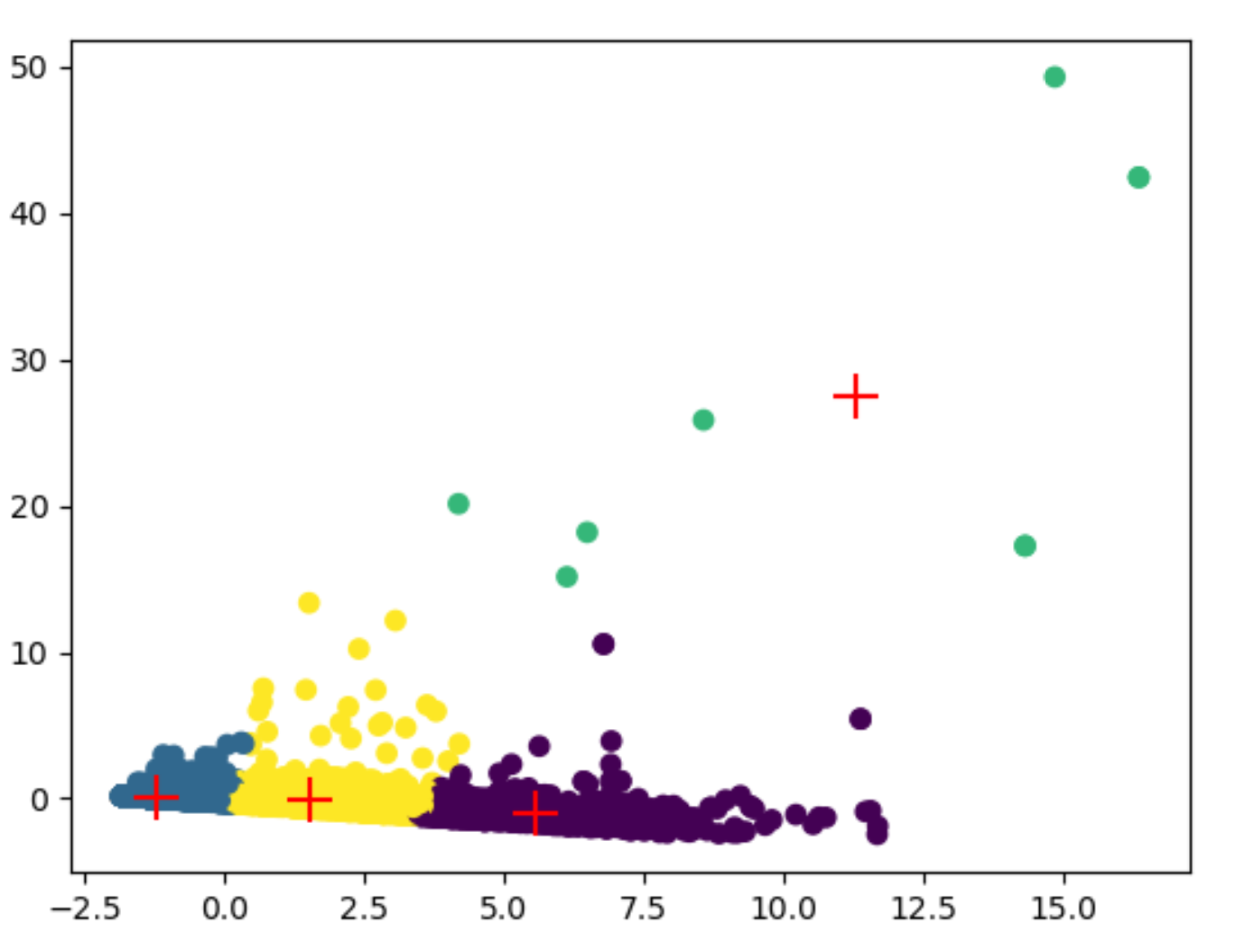


که با توجه به شکستگی در این نمودارها و تحلیل من k را برابر 4 فرض کردیم.

* سپس با استفاده از تابع PCA کاهش بعد را انجام میدهیم.
* سپس با تابع best\_k\_means الگوریتم را روی همه‌ی داده ها انجام میدهیم(بدون کاهش بعد)
* با تابع reduce\_demension\_k\_means الگوریتم روی داده های کاهش بعد شده صورت میگیرد و نمودار آن کشیده می‌شود.(با کاهش بعد)
* با استفاده از تابع k\_means\_plots نمودارهای kmeans اصلی برای هر دو تا اتربیوت نمایش داده می‌شود.
* همچنین با تابع k\_means\_each\_2\_columns الگوریتم kmeans برای هر دو تا اتربیوت ممکن حساب می‌کند و نمودار آن ها را می‌کشد.

خروجی ها :

* دقت : با استفاده از davies\_bouldin\_score :
  + 
* خروجی های نموداری :
  + Outs/reduce\_dimension\_k\_means.png: نمودار خوشه بندی با کاهش بعد



* + فولدر outs\MainKMeans : الگوریتم روی داده های اصلی بدون کاهش بعد صورت گرفته دو به دو اتربیوت ها برایش نمودار کشیده شده است.
  + فولدر outs\KMeans\_each2columns: برای هر دو به دو اتربیوت الگوریتم kmeans انجام می‌شود و نمودارش کشیده میشود.
* دیتابیس :
  + جدول kmeans\_clusters : داده‌های لیبل بندی شده‌ی اصلی بدون کاهش بعد.
  + جدول kmeans\_centers : مرکز خوشه ها در حالت بدون کاهش بعد.
  + جدول kmeans\_reduce\_dimension\_clusters : داده‌های لیبل بندی شده‌ی اصلی با کاهش بعد.
  + Kmeans\_ reduce\_dimension \_centers : مرکز خوشه ها در حالت با کاهش بعد.

## الگوریتم DBSCAN

این بخش در فایل DBScan.py وجود دارد که شامل مراحل زیر است:

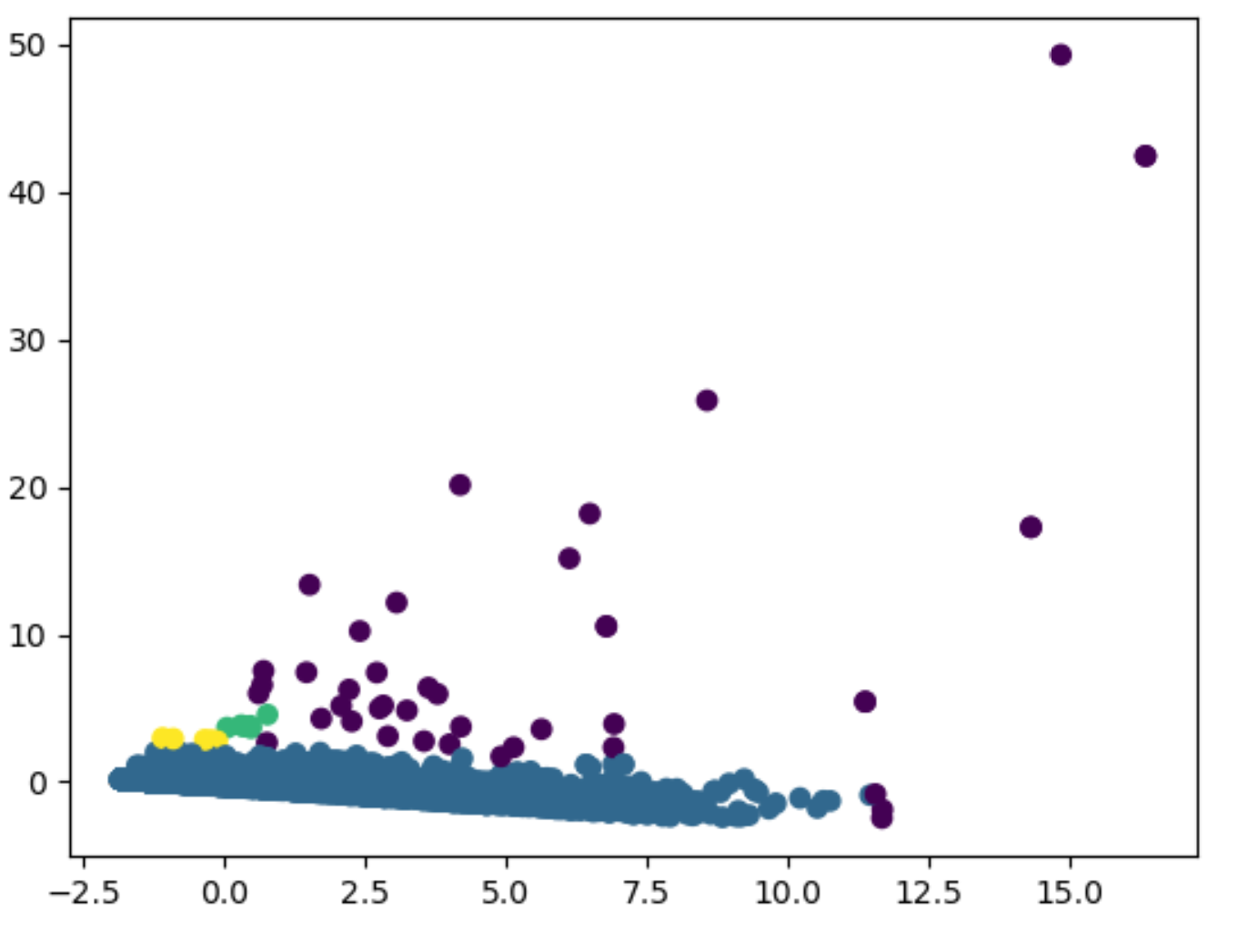
* با استفاده از تابع PCA کاهش بعد را انجام میدهیم.
* سپس با تابع best\_db\_scan الگوریتم را روی همه‌ی داده ها انجام میدهیم(بدون کاهش بعد)
* با تابع reduce\_demension\_DBSCAN الگوریتم روی داده های کاهش بعد شده صورت میگیرد و نمودار آن کشیده می‌شود.(با کاهش بعد)
* با استفاده از تابع db\_scan\_plots نمودارهای الگوریتم اصلی برای هر دو تا اتربیوت نمایش داده می‌شود.
* همچنین با تابع db\_scan\_each\_2\_columns الگوریتم برای هر دو تا اتربیوت ممکن حساب می‌کند و نمودار آن ها را می‌کشد.

خروجی ها :

* دقت : با استفاده از davies\_bouldin\_score :



* خروجی های نموداری :
  + Outs/reduce\_dimension\_DBSCAN.png: نمودار خوشه بندی با کاهش بعد

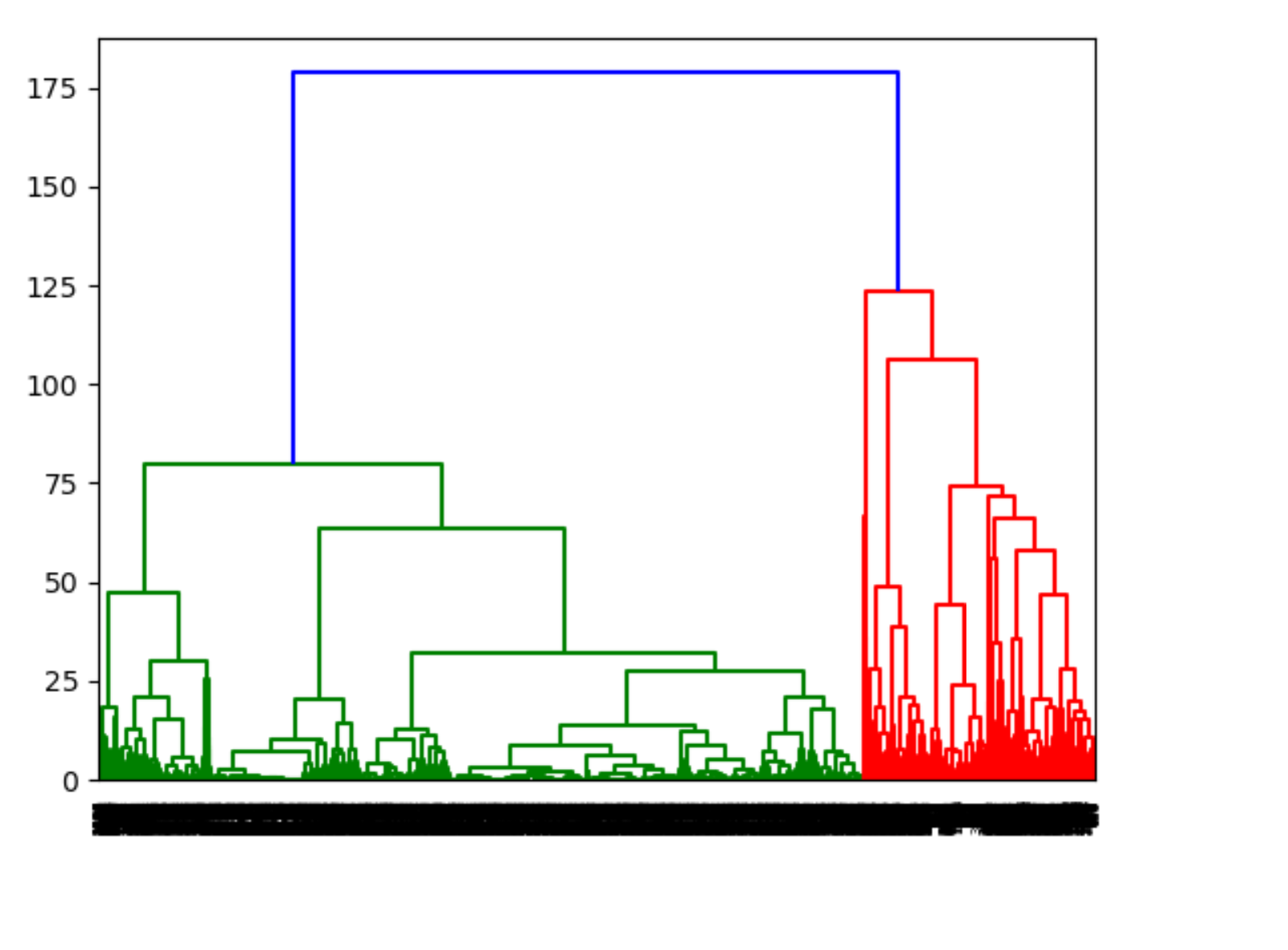


* + فولدر outs\MainDBSCAN: الگوریتم روی داده های اصلی بدون کاهش بعد صورت گرفته دو به دو اتربیوت ها برایش نمودار کشیده شده است.
  + فولدر outs\ DBSCAN\_each2columns: برای هر دو به دو اتربیوت الگوریتم kmeans انجام می‌شود و نمودارش کشیده میشود.
* دیتابیس :
  + جدول clusters\_DBSCAN: داده‌های لیبل بندی شده‌ی اصلی بدون کاهش بعد.
  + جدول reduce\_dimension\_DBSCAN\_clusters : داده‌های لیبل بندی شده‌ی اصلی با کاهش بعد.

## الگوریتم Agglomerative

این بخش در فایل agglomerative.py وجود دارد که شامل مراحل زیر است:

* با استفاده از تابع hierarchy\_plot نمودار آن را میکشیم که در outs\ hierarchy.png است.



با توجه به نمودار بالا بهترین تعداد خوشه 3 تا است ولی برای مقایسه با حالت های دیگر k را برابر 4 قرار می‌دهیم.

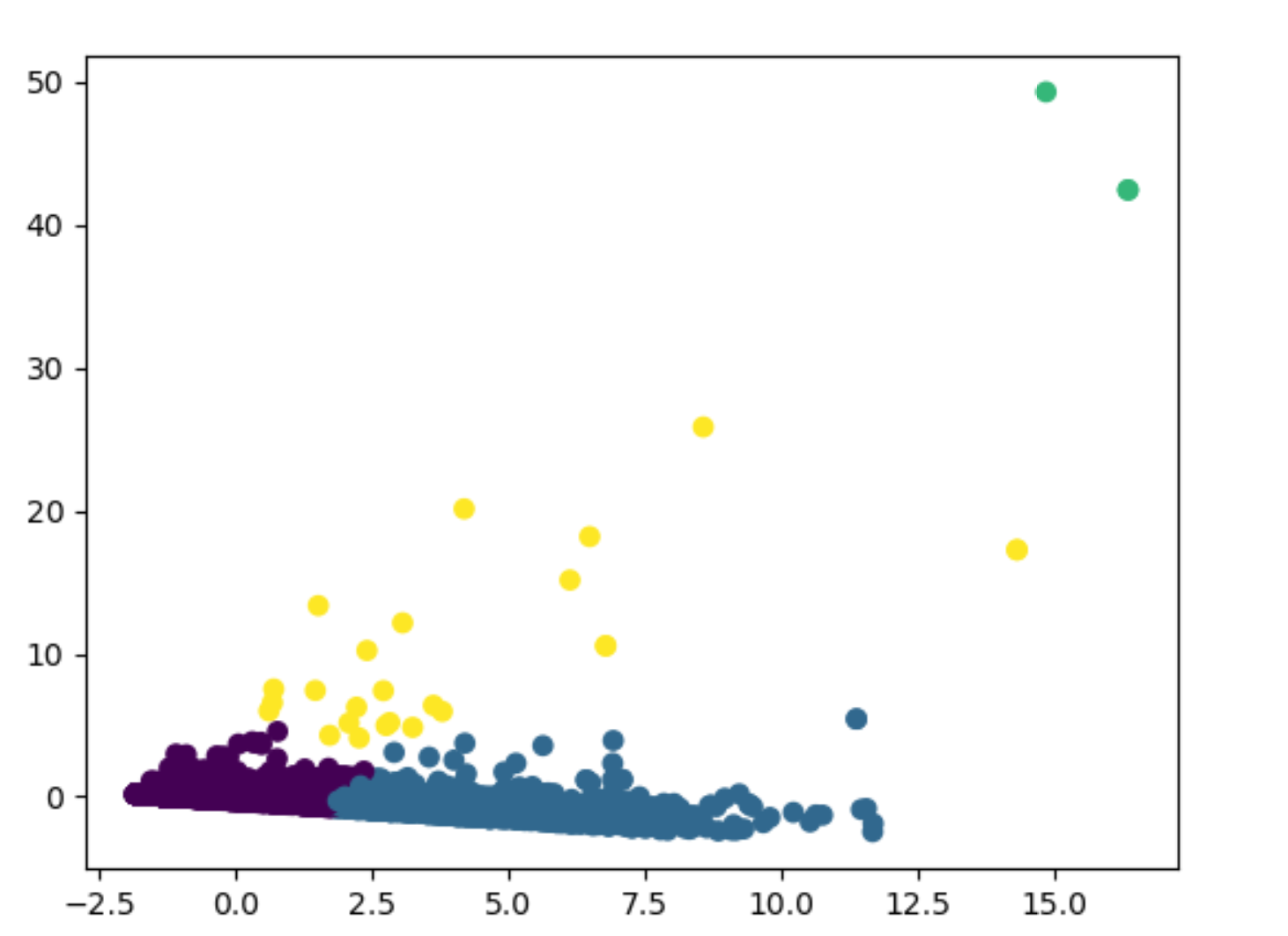
* سپس با استفاده از تابع PCA کاهش بعد را انجام میدهیم.
* با تابع best\_ agglomerative الگوریتم را روی همه‌ی داده ها انجام میدهیم(بدون کاهش بعد)
* با تابع reduce\_demension\_ agglomerative الگوریتم روی داده های کاهش بعد شده صورت میگیرد و نمودار آن کشیده می‌شود.(با کاهش بعد)
* با استفاده از تابع agglomerative \_plots نمودارهای agglomerative اصلی برای هر دو تا اتربیوت نمایش داده می‌شود.
* همچنین با تابع agglomerative \_each\_2\_columns الگوریتم agglomerative برای هر دو تا اتربیوت ممکن حساب می‌کند و نمودار آن ها را می‌کشد.

خروجی ها :

* دقت : با استفاده از davies\_bouldin\_score :



* خروجی های نموداری :
  + Outs/reduce\_dimension\_ agglomerative.png: نمودار خوشه بندی با کاهش بعد



* + فولدر outs\ agglomerative: الگوریتم روی داده های اصلی بدون کاهش بعد صورت گرفته دو به دو اتربیوت ها برایش نمودار کشیده شده است.
  + فولدر outs\ agglomerative\_each2columns: برای هر دو به دو اتربیوت الگوریتم kmeans انجام می‌شود و نمودارش کشیده میشود.
* دیتابیس :
  + جدول clusters\_ agglomerative: داده‌های لیبل بندی شده‌ی اصلی بدون کاهش بعد.
  + جدول reduce\_dimension\_ agglomerative\_clusters : داده‌های لیبل بندی شده‌ی اصلی با کاهش بعد.

## تحلیل این بخش

با توجه به معیار davies\_bouldin\_score برای سه تا الگوریتم Agglomerative بهتر از همه است چون davies\_bouldin\_score کمتری دارد (هر چه کمتر بهتر) بعد از آن الگوریتم kmeans است و بعد از آن الگوریتم DBSCAN که زیاد عملکرد آنچنان خوبی نداشته است.

با توجه به داده‌هایی که داشتیم به این نتیجه میرسیم که ساختاری سلسله مراتبی دارد و این این الگوریتم برایش بهتر است.

در kmeans و Agglomerative تقریبا نتیجه‌ی مشابهی داشته اند و نمیتوان دقیق گفت کدام بهتر است چون الگوریتم ها حریصانه هستند و قطعیت ندارند.

در kmeans ترجیح داده شده اندازه‌ی سایز خوشه ها متعادل باشند نسبت به بقیه الگوریتم ها به خصوص در DBSCAN تعادل خوبی اصلا وجود ندارد.

در الگوریتم Agglomerative سعی شده داده‌های خوشه های مختلف بیشترین تفاوت را داشته باشند.

در الگوریتم DBSCAN توجهی به شکل خوشه ها ندارد.

مزایای الگوریتم DBSCAN :

* سریع برای داده‌های با بعد کم
* یافتن خوشه‌ها برای اشکال نا منظم و کروی
* تشخیص نقاط نویز

معایب الگوریتم DBSCAN :

* نقاط مرزی که می‌توانند در دو خوشه نیز باشند، ممکن است به هریک از خوشه‌ها تعلق گیرند.
* این روش بنابه تغییرات در μ و ε رفتار غیرقابل پیش بینی می‌تواند داشته باشد

عیب روش Agglomerative :

* سرعت بسیار کند با اوردر n به توان 3