دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مدیریت، علم و فناوری

# پروژه ۵ درس داده کاوی

# کاهش ابعاد و خوشه بندی بر روی مقالات کنفرانسNIPS

نگارش
**فاطمه سادات علیخانی**

استاد درس
**دکتر مهدی قطعی**

فهرست

# 1 مقدمه

در این گزارش سعی شده است که ابتدا سعی شده با pca ابعاد دیتاست را کاهش داده  این الگوریتم سعی میکند واریانس هر ستون بالا باشد و ستون های مختلف با هم همبستگی نداشته باشند که این قضیه باعث میشود که تعداد ستون های تولید شده بسیار کم شود. بعد از آن از ستون هایی که با pca به دست آمده اند عملیات خوشه بندی را انجام می‌دهیم.

از طریق الگوریتم های خوشه بندی میتوانیم دیتاها را هر یک در یک کلاس قرار دهیم و دیتاهای شبیه به هم را پیدا کنیم. الگوریتم های مختلف clustering مانند ,k-means,k-medoid

Agglomerativeو birch  بر روی دیتاست  NIPS Conference Papers 1987-2015 بررسی شوند.
از بین این الگوریتم ها الگوریتم birch میتوان گفت برای داده هایی با ابعاد بالا مناسب است.

# dataset 2

در این پروژه از دیتاست NIPS Conference Papers 1987-2015 استفاده شده است، که شامل ۱۱۴۶۳ کلمه استفاده شده و تعداد تکرار آنها در مقالات مختلف ارائه شده در کنفرانس NIPS است،

این کنفرانس در مورد سیستم‌های پردازش اطلاعات عصبی ( Neural Information Processing Systems) یک کنفرانس یادگیری ماشین و علوم اعصاب محاسباتی است که هر سال در ماه دسامبر برگزار می‌شود و شامل ۱۱۴۶۳ سطر و ۵۸۱۲ ستون است. هر ستون نشان دهنده یک مقاله در کنفرانس های NIPS است و هر سطر نشان دهنده یک کلمه است.

این دیتاست نشان می‌دهد در هر مقاله چندبار از هر کلمه استفاده شده است. قسمتی از داده‌ها به شکل زیر است:

```
df = pd.read_csv( "NIPS_1987-2015.csv")
print(df.shape)
df
```

```
(11463, 5812)
```

| | Unnamed: 0 | 1987_1 | 1987_2 | 1987_3 | 1987_4 | 1987_5 | 1987_6 | 1987_7 | 1987_8 | 1987_9 | ... | 2015_394 | 2015_395 | 2015_396 | 2015_397 | 2015_398 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | abalone | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | |
| 1 | abbeel | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | |
| 2 | abbott | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | |
| 3 | abbreviate | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | |
| 4 | abbreviated | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 11458 | zoo | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | |
| 11459 | zoom | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | |

در ستون ها نام مقالات به صورت i d آنها در کنار سال انتشارشان آمده است اولین ستون نشان دهنده این است که هر در هر ردیف تعداد تکرار چه کلمه‌ای آمده.

# ۳ پیش پردازش داده‌ها

## ۳.۱ ترانهاده کردن دیتاست

با توجه به این که میخواهیم عملیات خوشه بندی را بر روی این دیتاست انجام دهیم، با استفاده از کتابخانه pandas ماتریس خوانده شده را transpose کرده به طوری که کلمات در ستون ها قرار بگیرند و در سطرها نام مقالات قرار بگیرد. این کار برای عملیات خوشه بندی پرمعناتر است که بر اساس کلمات استفاده شده در مقالات، مقالات را دسته بندی کنیم.

و دیتا به صورت زیر تبدیل میشود و تعداد ستون ها ۱۱۴۶۳ و تعداد سطر ها ۵۸۱۱ سطر میشود.

| Unnamed: 0 | abalone | abbeel | abbott | abbreviate | abbreviated | abc | abeles | abernethy | abilistic | abilities | ... | zhou | zhu | zien | zilberstein | zones | zoo |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1987_1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 |
| 1987_2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 |
| 1987_3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | ... | 0 | 0 | 0 | 0 | 0 | 0 |
| 1987_4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 |
| 1987_5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2015_399 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 |
| 2015_400 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 |
| 2015_401 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 |
| 2015_402 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 |
| 2015_403 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 |

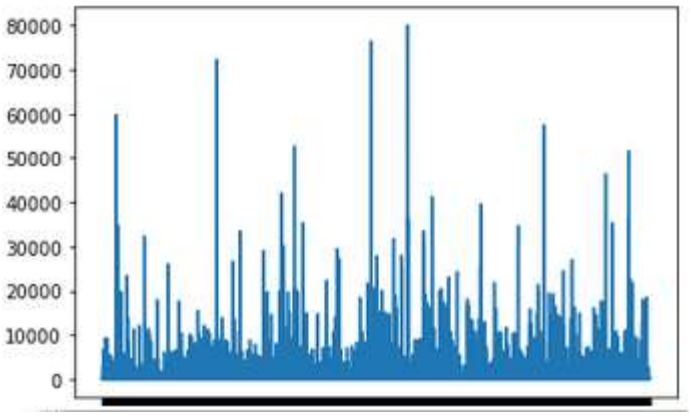
5811 rows × 11463 columns

## ۳.۲ بررسی داده‌های Null

داده null ایی در این دیتاست وجود ندارد.

```
df.isnull().sum().sum()
```

```
0
```




به طور کلی نمودار تعداد تکرار کلمات مختلف به صورت رو به رو است. همانطور که مشاهده میشود بعضی از کلمات به تعداد زیادی حتی تا ۷۰۰۰۰ هزار مرتبه هم تکرار شده اند و بعضی کمتر از بقیه.

### ۳.۳ بررسی Variance

اگر واریانس یک ستون برابر مقدار بسیار کمی باشد، باشد به این معنی است که تمام دیتاهای آن ستون مشابه هم هستند و وجود آن ستون در ارزیابی ما تاثیر ندارد.

در این جا ستون هایی که واریانس آنها کمتر از یک صدم است را حذف می‌کنیم. در مجموع از ۱۱۴۶۳ ستون ۱۵۰ ستون هستند.

```
low_variance = []
for i in range ( 0 , len(variances)) :
    if ( arr[i] < 0.01) :
        low_variance.append(i)
        print("variance :" , round(variances[i], 6 ) , df.columns[i] )

df.drop(df.columns[low_variance], axis=1 , inplace = True)
```

```
variance : 0.010602 abilistic
variance : 0.009555 abound
variance : 0.01094 activa
variance : 0.009048 advan
variance : 0.009907 adversely
variance : 0.010575 alfred
```

### ۳.۴ بررسی correlation

از معیار correlation استفاده شده هر چه قدرمطلق correlation بین دو ستون بیشتر باشد به این معنی است که این دو ستون بیشتر با هم مشابه هستند و میشود از آنها به جای یکدیگر استفاده کرد، یکی از ستون‌هایی که همبستگی آنها بیشتر از ۰,۸ است را حذف می‌کنیم در مجموع ۲۹۸ ستون حذف شده است.

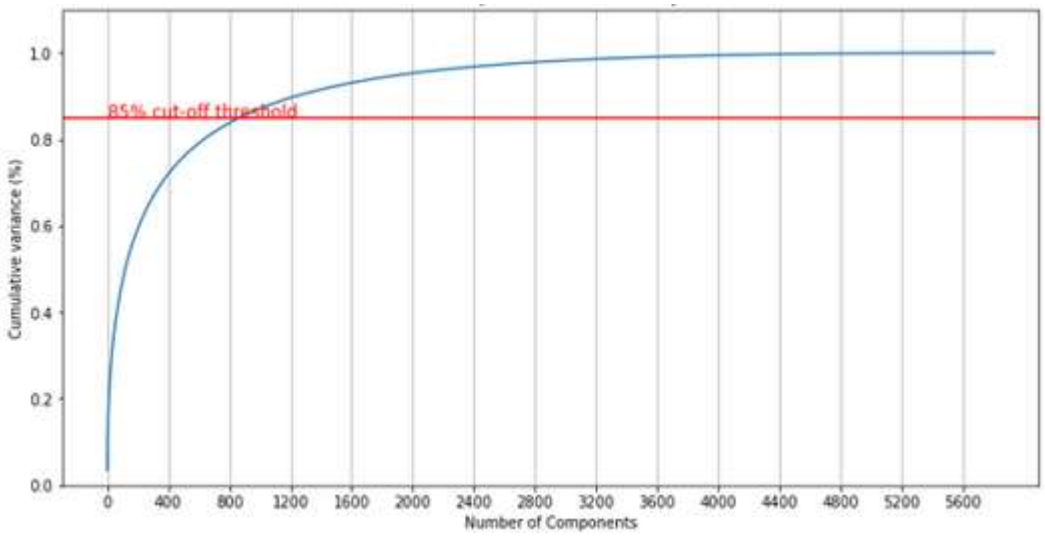
قسمتی از ستون های حذف شده به شکل زیر است

```
drop_col = []
for i in range(0, len(upper_corr)) :
    if ( upper_corr[i]> 0.8) :
        drop_col.append(df.columns[i])
print("drop col :" , drop_col)
df.drop(drop_col, axis = 1 , inplace = True)
```

```
drop col : ['absorption', 'abstractions', 'advertisements', 'advertisers', 'amino', 'analogy', 'ancestral', 'ann
otators', 'arms', 'articulatory', 'artists', 'atoms', 'auctions', 'axonal', 'barhen', 'biasing', 'biclusters', '
bidder', 'bidders', 'blockmodels', 'bob', 'borrowed', 'borrowing', 'buyers', 'captchas', 'captions', 'cesa', 'ch
orales', 'clamping', 'coeffi', 'community', 'consciousness', 'contacts', 'conventions', 'copulas', 'critic', 'cu
boids', 'curio', 'deblurring', 'diabetes', 'diamond', 'disks', 'disparity', 'divide', 'documents', 'dqn', 'dueli
ng', 'echoes', 'ego', 'electrostatic', 'els', 'englewood', 'equalizer', 'exercise', 'exercises', 'exogenous', 'f
acebook', 'faults', 'filterboost', 'fingerprints', 'fisherface', 'fouling', 'fragmentation', 'fragments', 'games
', 'ganglia', 'glimpses', 'granule', 'grasping', 'haplotypes', 'hasselmo', 'hawkes', 'hearer', 'hermite', 'hiera
r', 'hints', 'hips', 'holdout', 'homeostasis', 'honor', 'huxley', 'hyperedges', 'hypergraph', 'hypergraphs', 'il
luminant', 'impressions', 'indian', 'inh', 'inheritance', 'inhibitory', 'insects', 'instability', 'instructions
', 'international', 'interventions', 'investors', 'invite', 'ipsilateral', 'isomorphic', 'japanese', 'jerusalem
', 'kingmans', 'knots', 'kong', 'labelers', 'lacoste', 'landauer', 'las', 'lawmakers', 'leapfrog', 'leibler', 'l
esions', 'listen', 'magdon', 'mains', 'male', 'males', 'managers', 'markets', 'maxw', 'mcg', 'mellon', 'melody',
```

### PCA 3.5

برای کاهش ابعاد از تکنیک PCA استفاده می‌کنیم که در آن یک سری ستون جدید که ستون‌های آن با یکدیگر همبستگی ندارند (uncorrelated) و واریانس هر ستون از آن مقدارتقریبا زیادی است، تولید می‌شود. نشان می‌دهند با هر تعداد ستون مقدار variance داده ها چقدر است و مقدار ۸۵ درصد را انتخاب کرده که تعداد ۸۰۰ ستون جدید تولید میشود که از این به بعد با این ستون‌ها الگوریتم‌ها را اجرا کنیم.



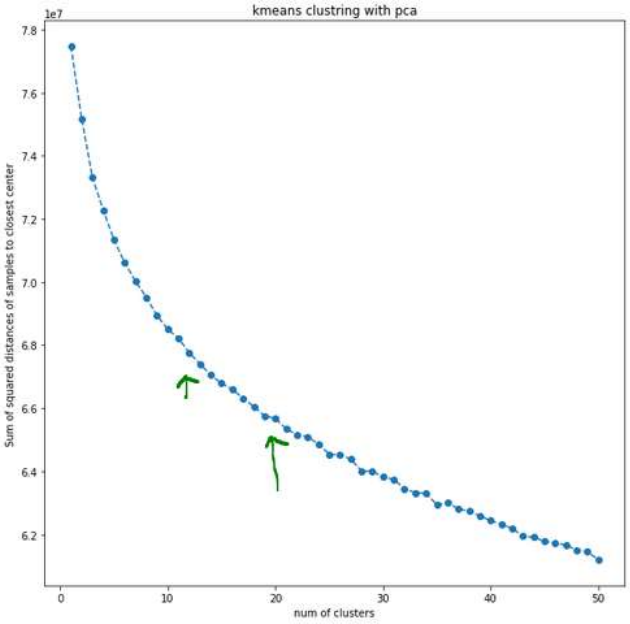

## ۴ الگوریتم kmeans

### 4.1 توضیح الگوریتم

روش k-means به این صورت است که در ابتدا k نقطه در دیتاست مشخص میکنیم و سپس بقیه‌ی نقاط دیتا ست را با توجه به اینکه به کدام یک از k نقطه‌ی اولیه نزدیک‌تر هستند دسته‌بندی می‌کند و k دسته به وجود می‌آید میانگین این k دسته را پیدا کرده و الگوریتم را دوباره با این k نقطه جدید ادامه می‌دهیم. این کار را تا یک تعداد دور مشخص ادامه می‌دهیم.

## ۴.۲ انتخاب k مناسب

عملیات خوشه بندی را با kهای مختلف از ۱ تا ۵۰ امتحان کردیم و نمودار جمع فاصله های مربعات نمونه های مختلف تا نزدیک ترین مرکز را می‌کشیم.( Sum of squared distances of samples to their closest cluster center) با توجه به elbow method بهتر است kایی را انتخاب کنیم که در خم نمودار قرار دارد.به این دلیل اینکار را انجام میدهیم که:




۱. تا از اینکه هر نمونه از مرکز کلاستر خودش بسیاردور باشد جلوگیری کنیم، زیرا این نشان دهنده این است که نمونه های درون یک کلاستر با هم شباهت زیادی ندارند.

۲. تعداد کلاسترها خیلی زیاد شود، این موضوع نیز باعث میشود فاصله کلاسترهای مختلف از هم کم شود.

نقاطی که در شکل علامت زده شده اند میتوانند تعداد کلاسترهای مناسبی برای این مساله باشند.

همینطور برای این مقادیر k ،Silhouette score را محاسبه کرده این مقدار از طریق فرمول زیر بدست می آید :

Silhouette Score = (b-a)/max(a,b)

که در آن a : میانگین فاصله‌های بین نمونه‌های یک کلاستر است.( هر چه این مقدار کمتر باشد بهتر است)
b : میانگین فاصله‌های بین کلاسترها است.( هر چه این مقدار بیشتر باشد به این معنی است که کلاسترهای بهتری تولید شده است.) در کل هر چه مقدار Silhouette score به یک نزدیک‌تر و مثبت باشد، به معنی این است که کلاسترهای بهتری انتخاب شده است.

این مقدار را برای k=7و k=20 که در شکل قبل گفته محاسبه کرده و نتیجه به صورت زیر می‌شود:

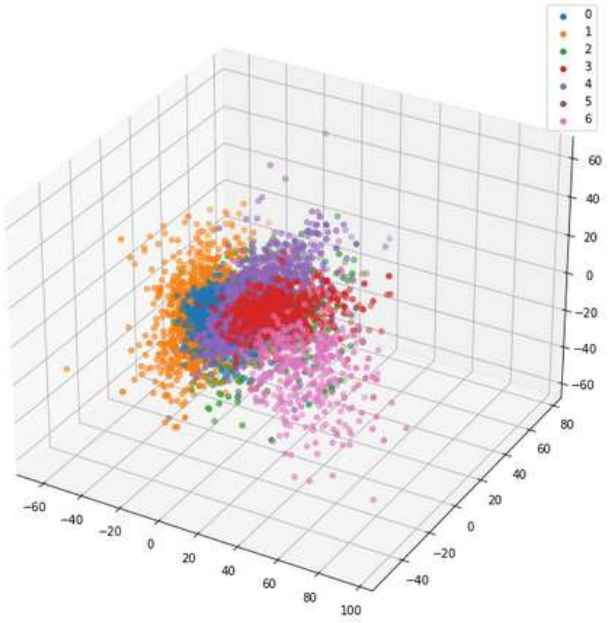
```
For n_clusters = 7 The average silhouette_score is : 0.024637454696582795
For n_clusters = 20 The average silhouette_score is : -0.0203430046808322
```

و نتیجه می‌شود که تعداد کلاسترهای 7 بهتر از ۲۰ است و الگوریتم را با این تعداد کلاستر اجرا می‌کنیم.

## ۴.۳ اجرای الگوریتم

برای پیاده سازی الگوریتم از متد KMeans در کتابخانه sklearn استفاده می‌کنیم، تعداد cluster ها را با توجه به قسمت قبل برابر ۷ در نظر میگیریم و مشخص شده عملیات kmeans ۲۰۰ بار تکرار شود و k-means++ مشخص می‌کند که در ابتدا طبق چه متدی اولین مرکزهای کلاستر را انتخاب شود.

```
best_kmeans = KMeans(n_clusters = 7, init = "k-means++" , max_iter = 200)
label = best_kmeans.fit_predict(pca_data)
```



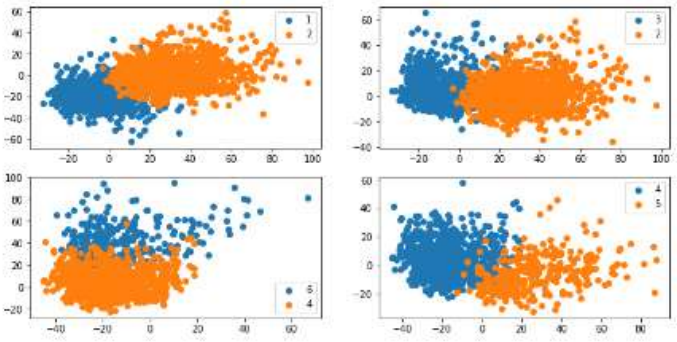

نتیجه کلاسترینگ را با ستون های ۳و۵و۸ از ستون های pca به صورت مقابل است :

که کلاس‌ها را میتوان در شکل از هم تشخیص داد و اینکه دیتاهای کلاسهای مختلف از هم فاصله دارند و همگی یکجا قرار گرفته اند، البته باید توجه داشت که تعداد کل ستون ۸۰۰ است و نمی‌شود به طور دقیق کلاستر ها را با سه ستون نشان داد.

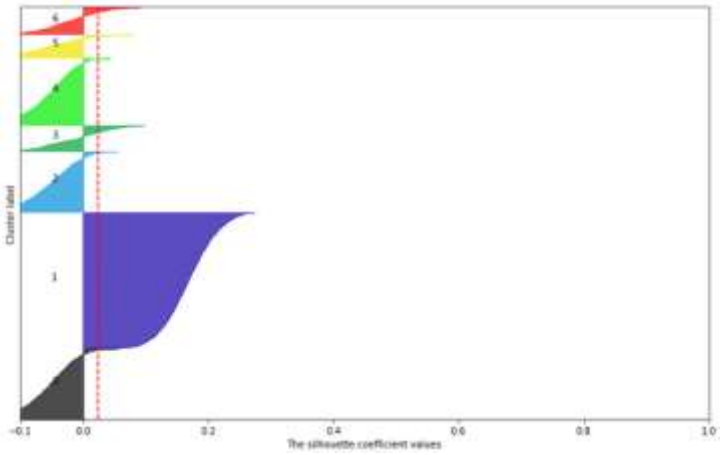
```
dict1 = df[label==1].sum().to_dict()
sorted(dict1.items(), key=operator.itemgetter(1), reverse=True)
```

```
[('graph', 5194),
 ('graphs', 2204),
 ('algorithm', 1915),
 ('set', 1743),
 ('model', 1442),
 ('learning', 1172),
 ('data', 1118),
 ('matrix', 1118),
 ('function', 1111),
 ('random', 1109),
 ('number', 1098),
 ('edges', 1094),
 ('edge', 1091),
 ('problem', 1066),
 ('nodes', 1005),
 ('given', 973),
 ('models', 963),
 ('using', 934),
 ('theorem', 909),
 ('figure', 878),
 ('vertices', 866),
```

کلمه هایی بیشترین تکرار را در مقالات کلاستر یک داشتند به صورت رو به رو است، همانطور که مشاهده میشود، کلماتی که بیشتر استفاده شده به هم مربوط هستند و این نشان دهنده شباهت مقاله‌های موجود در کلاستر یک است. میتوان گفت در مقالات این دسته بیشتر از گراف ها ماتریس ها استفاده شده است به دلیل تکرار زیاد کلمات graph , graphs, edge, nodes



شکل رو به رو خوشه های مختلف را دو به دو بررسی کرده خوشه‌ها تقریبا از هم فاصله دارند، البته تعداد ستون هایی که در pca کاهش دادیم ۸۰۰ ستون بود و این شکل با ستون های ۱ و ۳ از pca کشیده شده است.



نمودار Silhouette score برای هر خوشه به صورت رو به رو است.

این نمودار نشان می‌دهد که این خوشه بندی بیشتر داده ها در دسته ۱ قرار گرفته‌اند و این دسته است که Silhouette score آن مقدار قابل قبولی است و بیشتر از صفر است اما بقیه کلاسترها اصلا مقدار مناسبی ندارد و ب خوبی کلاستر بندی نشده اند یا کلاسترهای مختلف بسیار به هم شبیه‌اند و یا داده‌های درون یک کلاستر با هم تفاوت زیادی دارند.

( البته نمودار های دیگر نیز کشیده شد و این بهترین نتیجه بود!!)

# ۵ الگوریتم KMedoids

## 5.1 توضیح الگوریتم

روش کار این الگوریتم مانند الگوریتم kmeans است با این تفاوت که k نقطه ایی که به عنوان نماینده هر کلاستر مشخص میشود از خود داده ها است و سپس عملیات تکرار برای پیدا کردن کلاسترهایی که درون آنها میزان عدم شباهت کم باشد انجام میشود.

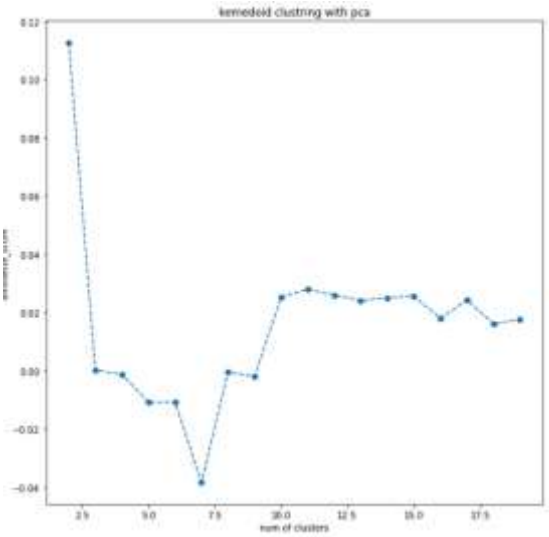
## ۵.۲ انتخاب k مناسب

برای پیدا کردن k مناسب از Silhouette score که در قسمت قبل هم گفته شد استفاده میکنیم این مقدار از فرمول

رو به رو بدست می آید: $Silhouette\ Score = (b-a)/max(a,b)$

که در آن a : میانگین فاصله‌های بین نمونه‌های یک کلاستر است.( هر چه این مقدار کمتر باشد بهتر است)

b : میانگین فاصله‌های بین کلاسترها است.( هر چه این مقدار بیشتر باشد به این معنی است که کلاسترهای بهتری تولید شده است.) در کل هر چه مقدار Silhouette score به یک نزدیک‌تر و مثبت باشد، به معنی این است که کلاسترهای بهتری انتخاب شده است.




این مقدار را برای k های بین ۲ تا ۲۰ برای الگوریتم k-medoid رسم می‌کنیم و نمودار آن به صورت شکل مقابل می‌شود.
همانطورکه در شکل مشخص است مقدار Silhouette score برای داده های ۱۰ تا ۱۵ تقریبا از بقیه مقادیر بهتر است همینطور تعداد ۲ کلاستر.

مقدار و ۱۴ را برای تعداد کلاستر انتخاب در این قسمت انتخاب می‌کنیم.

## ۵.۳ اجرای الگوریتم

برای پیاده سازی این الگوریتم از k-medoid از کتابخانه sklearn_extra استفاده میکنیم در آن تعداد کلاستر را برابر ۱۴ قرار می‌دهیم.

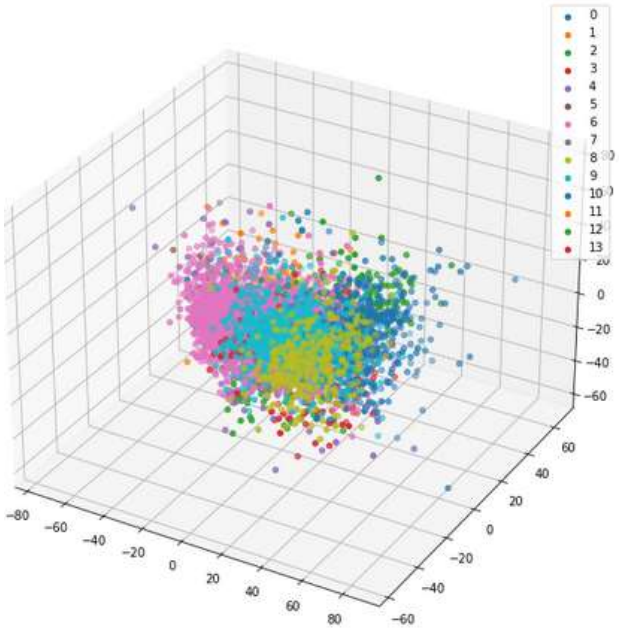
```
best_kmedoids = KMedoids(n_clusters=14,init = 'k-medoids++').fit(pca_data)
best_kmedoids.labels_
```

```
array([ 6,  4,  6, ..., 12, 10, 10], dtype=int64)
```

همانند قسمت قبل کلماتی که در مقاله های کلاستر ۶ بیشتر تکرار شده‌اند مشاهده میشود که کلماتی که بیشتر استفاده شده به هم مربوط هستند. میتوان گفت در مقالات این دسته  در مورد شبکه های عصبی بوده به دلیل وجود کلمات neural, network, model , training

```
dict1 = df[best_kmedoids.labels_==6].sum().to_dict()
sorted(dict1.items(), key=operator.itemgetter(1), reverse=True)
```

```
[('learning', 26600),
 ('data', 26308),
 ('model', 24191),
 ('set', 23103),
 ('function', 21703),
 ('network', 20645),
 ('using', 20623),
 ('figure', 20303),
 ('algorithm', 20271),
 ('time', 19101),
 ('neural', 17795),
 ('number', 16401),
 ('problem', 15618),
 ('used', 15549),
 ('training', 15164),
```

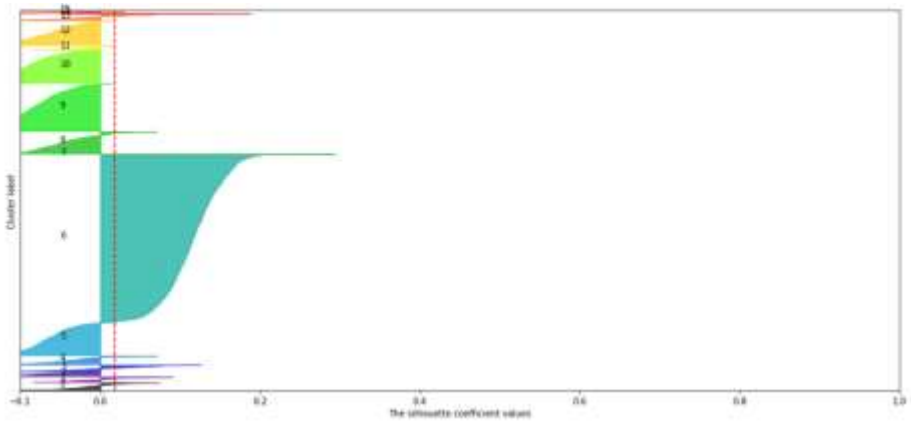



شکل کلی نمودار با استفاده از ستون های 4, ,1 , 0 از نمودار pca

همانطور که در شکل مشاهده می‌کنیم بعضی از کلاستر ها که قابل تشخیص هستند یک مرکزیت خاص دارند و به خوبی دسته بندی شده اند.

نمودار Silhouette score برای این خوشه بندی به صورت روبه رو است ، این نمودار نشان می‌دهد که این خوشه بندی بیشتر داده ها در دسته ۶ قرار گرفته‌اند و این دسته است که Silhouette score آن مقدار قابل قبولی است و برای بقیه کلاسترها مقدار مناسبی ندارد. و یا مانند آخرین دسته حجم داده‌های دسته بندی شده در آن بسیار کم است. و یا در دسته آهر تعدادی از نمونه های یک کلاستر نسبت به هم خوب هستند و در مورد تعدادی از آنها اینطور نیست.

که این به این معنی است که داده های درون یک خوشه شباهت کمی نسبت به هم دارند و به خوشه های دیگر نزدیک هستند.



# 6 Agglomerative

## ۶.۱ توضیح الگوریتم

این الگوریتم با دو الگوریتم قبل متفاوت است، یک الگوریتم پایین به بالا است در ابتدا هر نمونه را یک کلاستر در نظر میگیرد و سپس بر اساس شباهت، کلاسترها آنهایی که با هم شبیه‌ند دوبه دو به یک کلاستر تبدیل میشود و این عملیات تا انتها ادامه دارد.

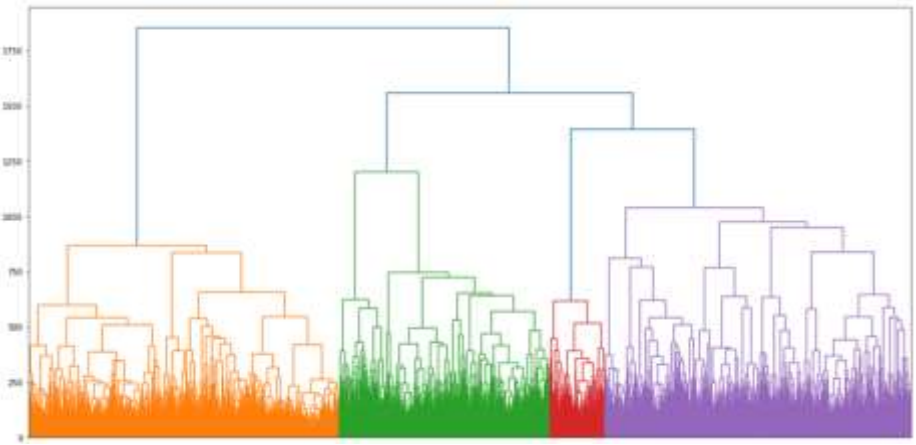
## ۶.۲ پیدا کردن تعداد کلاستر مناسب و پیاده سازی

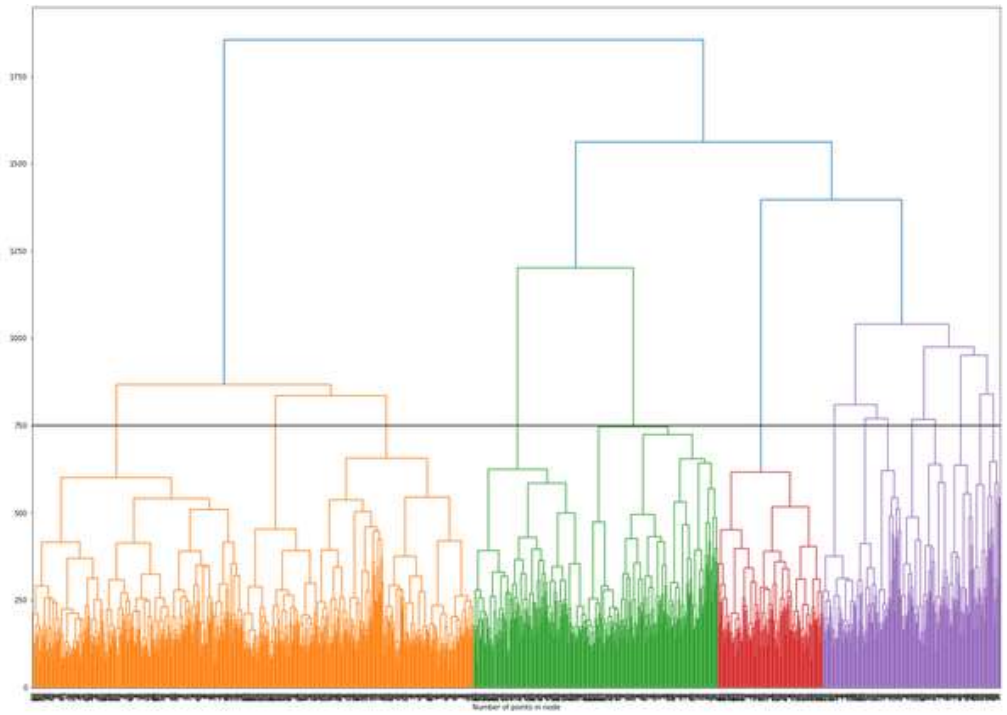
در ابتدا یک مدل از agglomerative به شکل زیر میسازیم با استفاده از کتابخانه Sklearn:

```
agglomerative_model = AgglomerativeClustering(distance_threshold=0.1, n_clusters=None, linkage = 'average')
ClusteringModel = agglomerative_model.fit(pca_data)
```

این مدل فاصله‌ها را بر اساس فاصله اقلیدسی بدست می آورد و برای بدست آوردن شباهت بین دو کلاستر average فاصله بین آنها را حساب میکند.

حال برای تعیین کلاستر مناسب از نمودار dendrogram استفاده میکنیم که وضعیت یکی شدن کلاستر ها را به طور دقیق نشان میدهد، هرچه ارتفاع یک خط در این نمودار بیشتر باشد آن دو کلاستر از هم دورتر هستند مثلا در قسمت نارنجی کلاستری که در انتها تشکیل شده به کلاستر سمت راست بیشتر شبیه است تا کلاستر سمت چپ به همین دلیل اگر نیاز به قطع کرد نمودار است بهتر است نمودار در جایی قطع شود که ارتفاع نمودار دندوگرام آن بیشتر است.

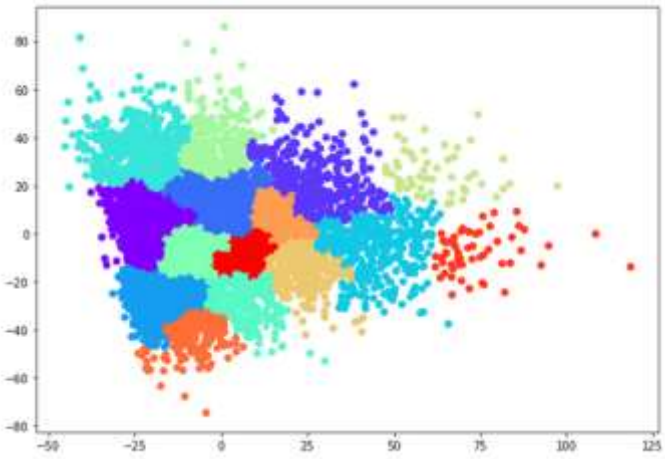




با توجه به قسمت بالا اگر از خط مشخص شده نمودار قطع شود تعداد ۱۵ کلاستر تولید میشود و داده‌هایی که با هم تفاوت زیاد دارند ادغام نمی‌شوند مثلا دو قسمت سبز رنگ یا (قسمت قرمز با بنفش ) و سمت چپ‌ترین کلاستر های رنگ بنفش این ها با هم فاصله زیادی دارند و بهتر است در یک دسته قرار نگیرند.

پس در نهایت تعداد داده ها را به تعداد ۱۵ کلاستر تقسیم بندی می‌کنیم

```
AgglomerativeCluster  = AgglomerativeClustering(n_clusters=15, affinity='euclidean',
AgglomerativeCluster.fit_predict(pca_data)
```



کلاسترها به صورت کلی به صورت رو به رو تقسیم بندی میشوند( البته این شکل مدل بر روی دو بعد fit شده است)

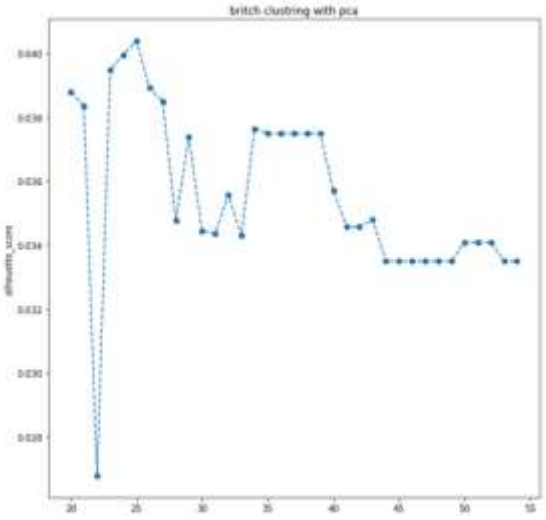
- همینطور مقدار score silhouette آن برابر ۰,۲ است.

```
For n_clusters = 15 The average silhouette_score is : 0.02447622758796164
```

# 7 الگوریتم brich

## ۷.۱ توضیح الگوریتم

در این الگوریتم سعی میشود مقدار حافظه کمتری نسبت به الگوریتم سلسله مراتبی استفاده شود در ابتدا یک cf-tree ساخته می‌شود این الگوریتم یک مقدار branching factor دارد که بیشترین تعداد زیرشاخه ها را مشخص میکند و یک مقدار threshold.




## 7.2 پیدا کردن پارامترهای مناسب

Branching factor های مختلف را برای این تابع محاسبه کرده و نمودار silhouette آن را رسم میکنیم در مقدار ۲۵ بیشترین مقدار دقت را دراد پس مقدار factor branching را برابر ۲۵ قرار میدهیم.

## 7.3 پیاده سازی الگوریتم

این الگوریتم نیز مانند الگوریتم های قبل از کتابخانه sklearn استفاده میکنیم. و مقدار branching factor ۲۵ را برای آن انتخاب میکنیم که با این branching factor ۴۸ کلاستر تولید میشود.

```
[('algorithm', 50),
 ('matrix', 38),
 ('recovery', 37),
 ('dataset', 34),
 ('set', 34),
 ('associative', 32),
 ('sparse', 31),
 ('vectors', 27),
 ('basis', 26),
 ('vector', 26),
 ('message', 25),
 ('phase', 25),
 ('memory', 23),
```

کلماتی که در یک کلاستر از این دیتاست استفاده شده است به صورت زیر است :

همانطور که مشخص است این کلمات تعداد آنها نسبت به کلاستر هایی که در بخش‌های قبل مشاهده کردیم بسیار کمتر هستند و همینطور نمیتوان از این کلاستر نتیجه خاصی گرفت،

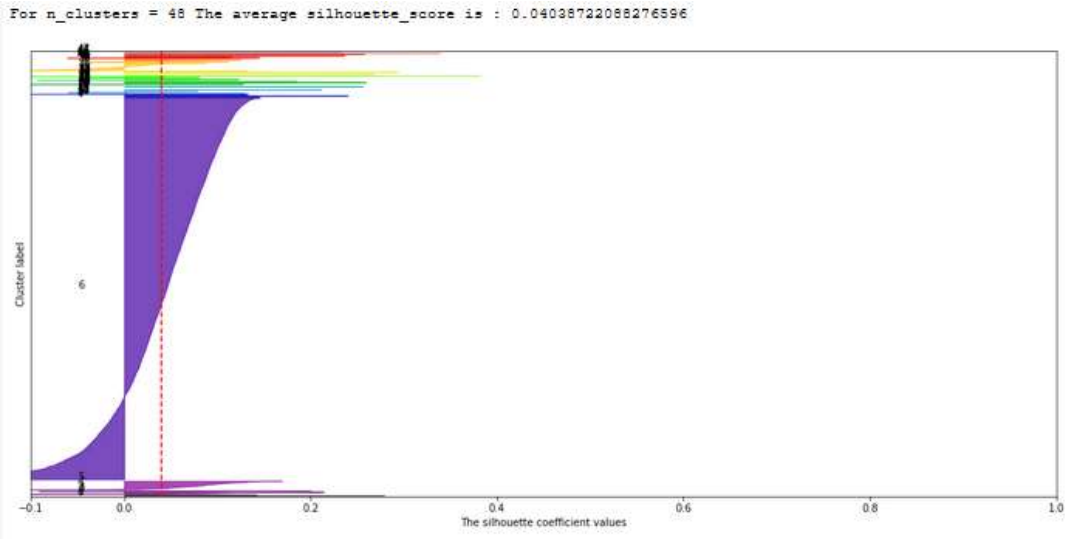
با توجه به اینکه ۴۸ کلاستر داریم ممکن است یک کلاستر بسیار کوچک باشد و خوب کلاستر بندی نشده باشد.

```
[('model', 76175),
 ('learning', 69440),
 ('data', 67646),
 ('set', 51622),
 ('algorithm', 48475),
 ('using', 47871),
 ('function', 47817),
 ('time', 42221),
 ('figure', 39895),
 ('number', 37556),
 ('models', 34335),
 ('problem', 33847),
 ('used', 33761),
 ('training', 33713),
 ('network', 32288),
 ('given', 32235),
 ('results', 31676),
 ('also', 31487),
 ('distribution', 31162),
 ('neural', 30041),
```

اما کلاستر دیگری از آن به این صورت است که تعداد کلمات زیادی در آن به کار رفته شده است مثلا کلمه ’model‘ ۷۶۱۷۵ دفعه تکرار شده است و کلمات مورد استفاده در آن بیشتر از کلاستر قبل به هم مرتبط هستند.

این موضوع میتوان نتیجه گیری کرد که اندازه همه کلاستر ها به یک اندازه نیست بعضی بسیار بزرگ و بعضی بسیار کوچک هستند.

این موضوع را میتوان از نمودار silhouette نیز به وضوح مشاهده کرد که یک کلاستر بسیار بزرگ و بقیه کوچک هستند.




مقدار silhouette به صورت میانگین در این الگوریتم ۰,۴ است که از تمامی مقادیر silhouette در الگوریتم های دیگر بزرگ تر است.

همینطور زمان اجرایی این الگوریتم از زمان اجرایی بقیه الگوریتم ها به خصوص الگوریتم سلسله مراتبی بهتر بوده است.

## نتیجه گیری

در این گزارش الگوریتم های مختلف را روی دیتاست خود بررسی کردیم، الگوریتم brich از لحاظ زمانی و همینطور میانگین silhouette از بقیه الگوریتم‌ها نتیجه بهتری داشت، الگوریتم agglomerative

بیشترین زمان اجرا را داشت و این موضوع مشخص است زیرا تمام عنصر ها را بررسی میکند.

در نهایت الگوریتم های k-means و k-medoid که الگوریتم k-medoid عملکرد بهتری نسبت به k-means داشت، اما برای همه الگوریتم ها حتی با وجود اینکه بهترین تعداد کلاستر را انتخاب کردیم مقادیر silhouette قابل قبول و خوب نبوند.

# 8 منابع

(BIRCH Clustering Algorithm Example In Python, 2019)

(Are the clusters good?, 2020)

(K-Means Clustering — One rule to group them all, 2020)

(clustring, 2019)

(K-medoids Clustering, 2018)