به نام خدا

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

پروژه درس یادگیری ماشین

فردین آیار

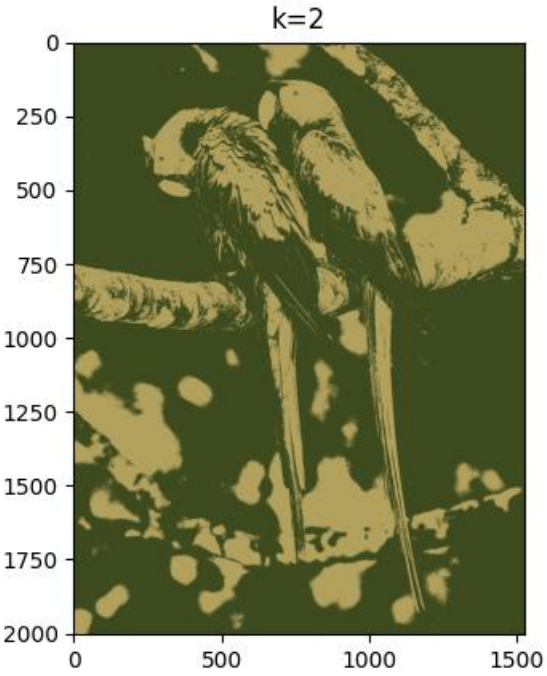
شماره دانشجویی: ۹۹۱۳۱۰۴۰

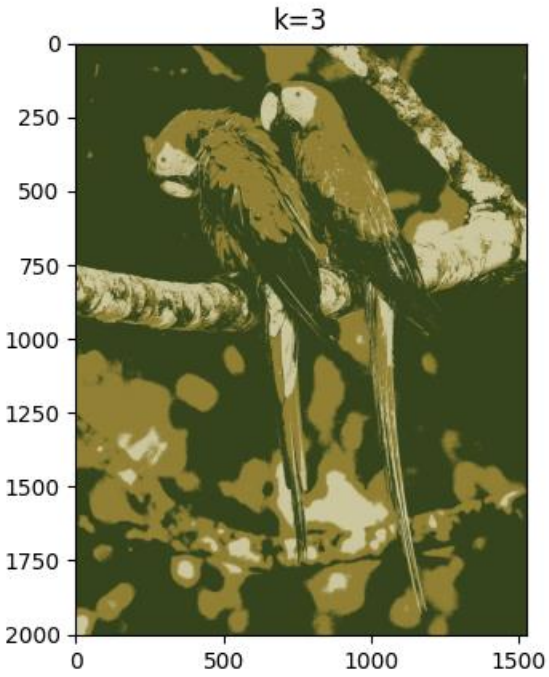
استاد: دکتر ناظرفرد

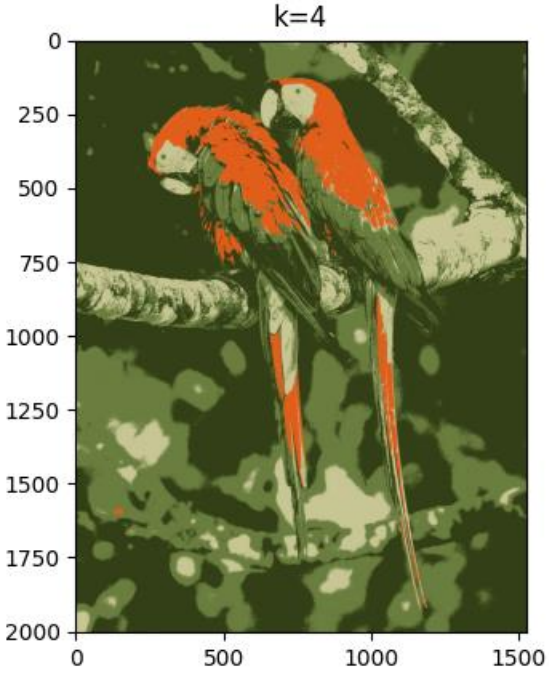
دانشکده کامپیوتر

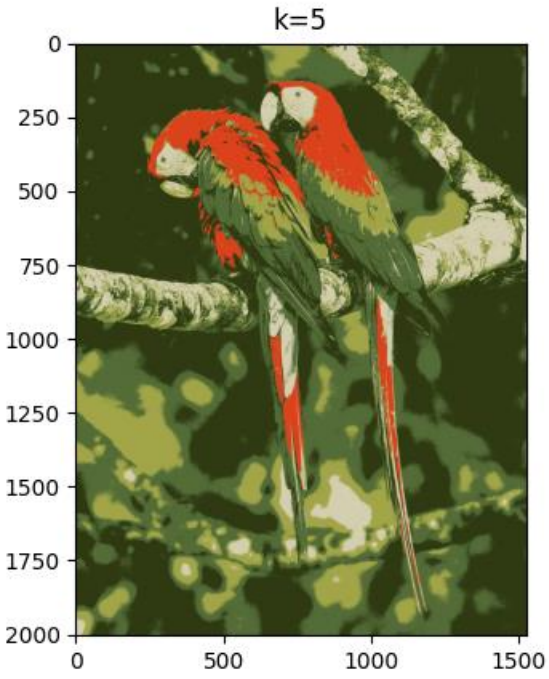
۱)

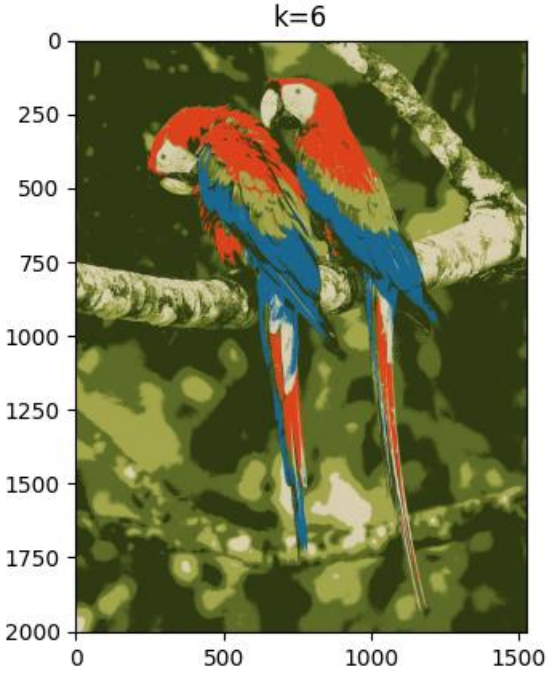
۱–۱) کد مربوط به این قسمت در فایل 1-1.py قرار دارد. برای این کار از کتابخانه OpenCV و الگوریتم K-means موجود در آن استفاده شده است. نتایج به شرح زیر است. برای تصویر parrots.jpg:

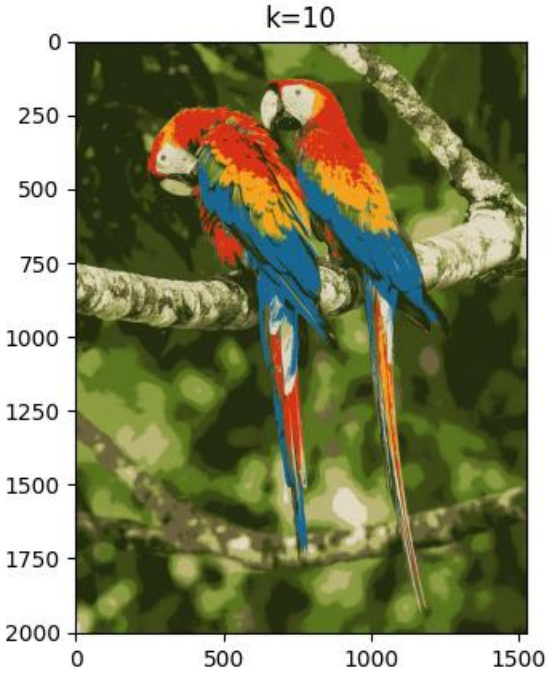


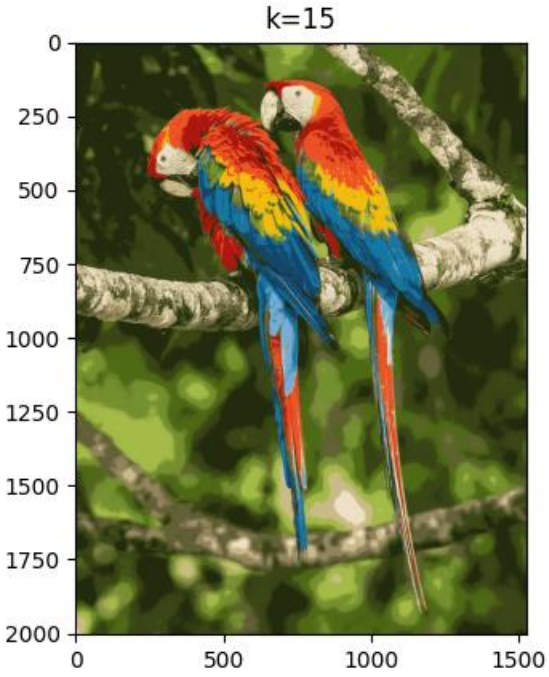

k=3
0
250
500
750
1000
1250
1500
1750
2000
0
500
1000
1500

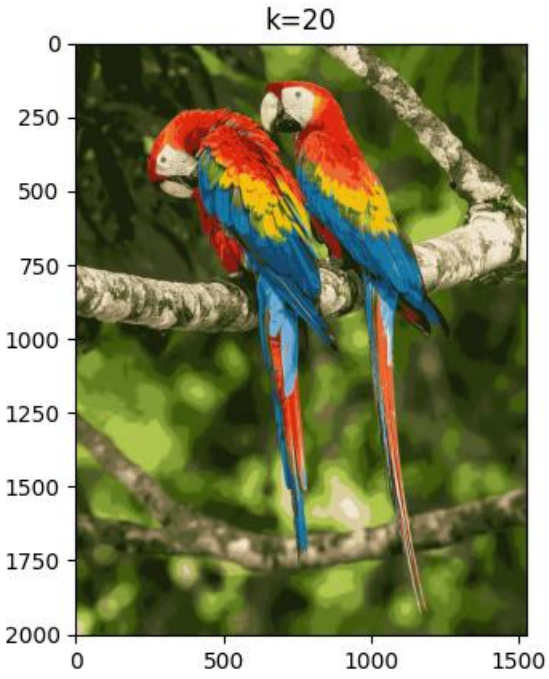

k=4
0
250
500
750
1000
1250
1500
1750
2000
0
500
1000
1500


k=5
0
250
500
750
1000
1250
1500
1750
2000
0
500
1000
1500

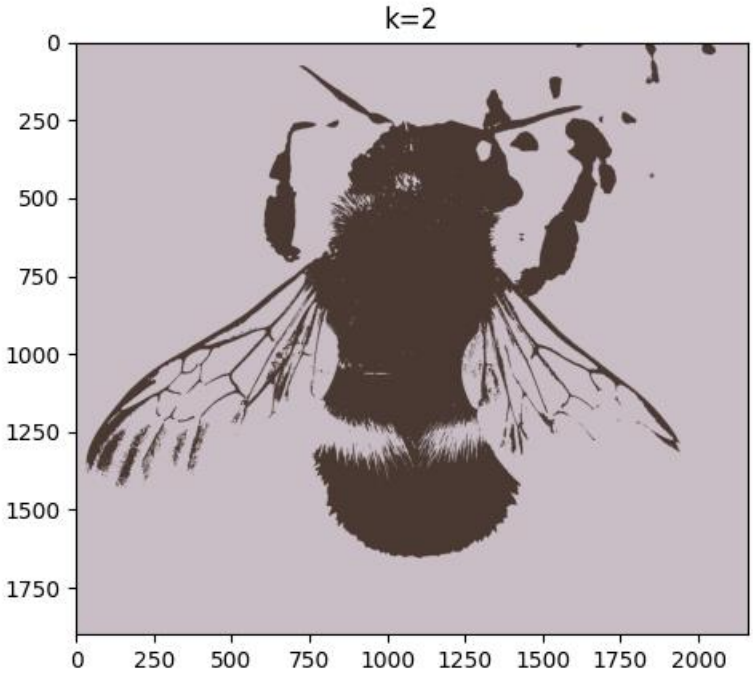

k=6
0
250
500
750
1000
1250
1500
1750
2000
0
500
1000
1500

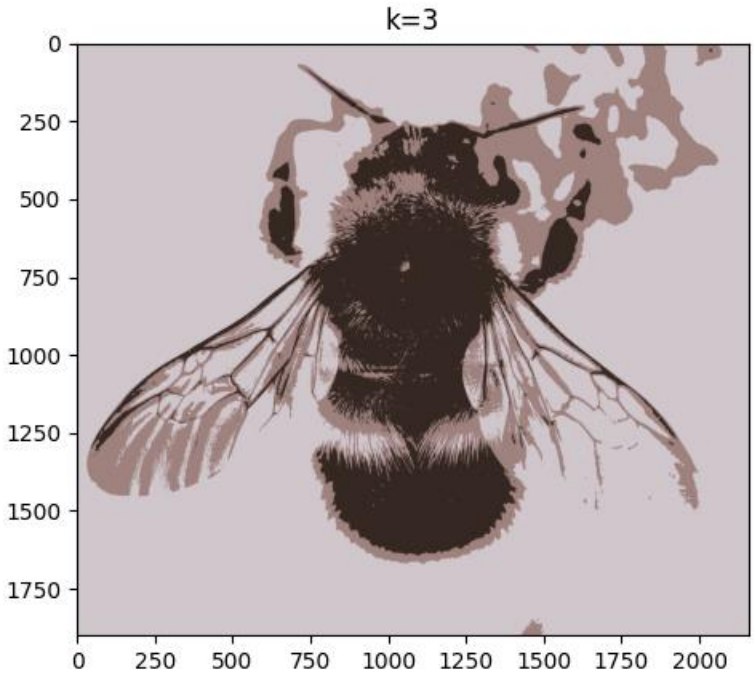

k=10
0
250
500
750
1000
1250
1500
1750
2000
0
500
1000
1500

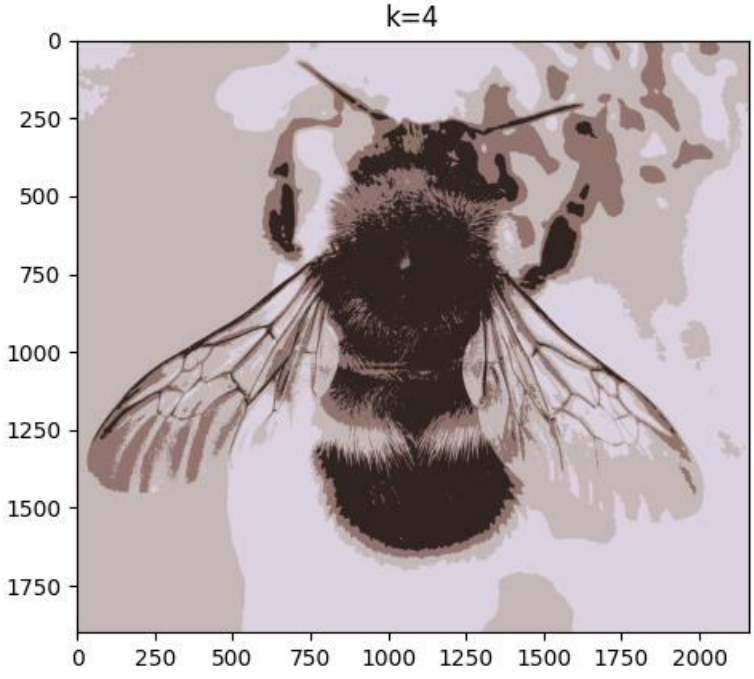

k=15
0
250
500
750
1000
1250
1500
1750
2000
0
500
1000
1500

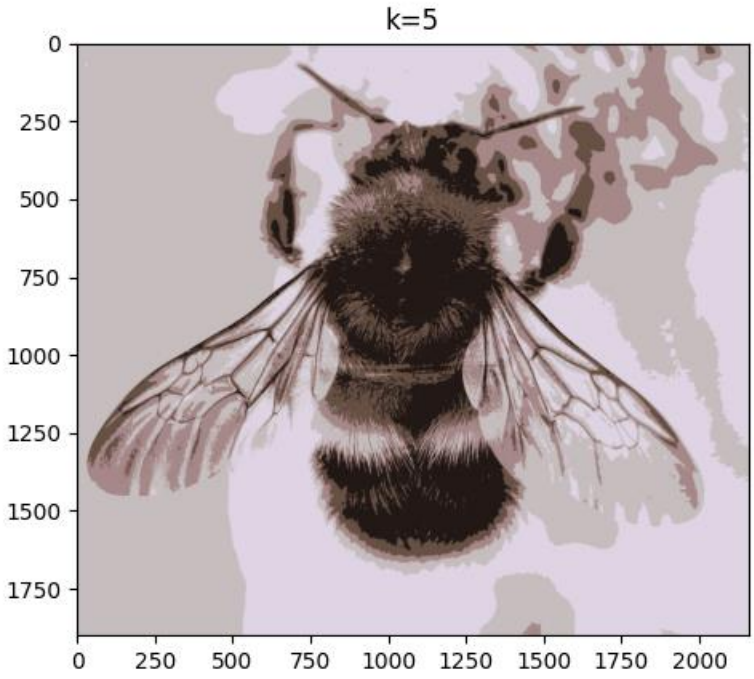


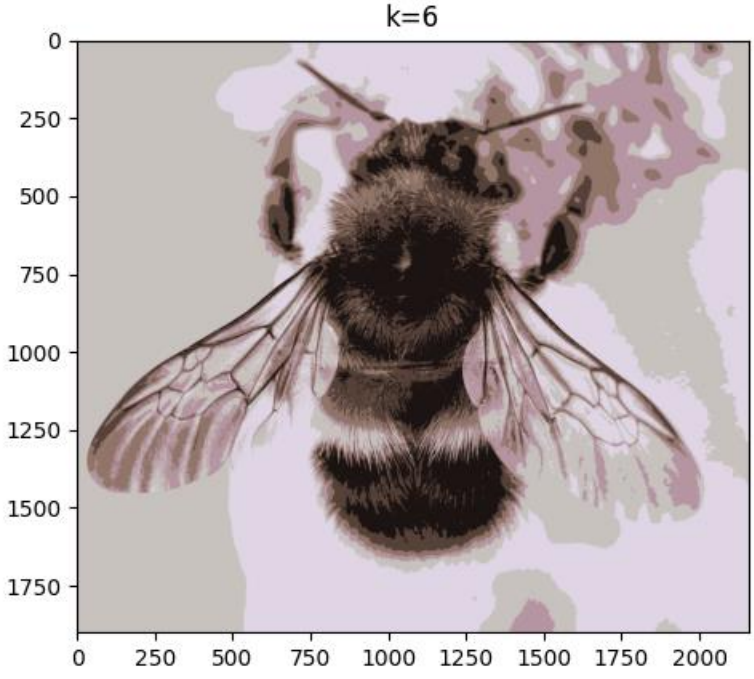
برای تصویر bee:

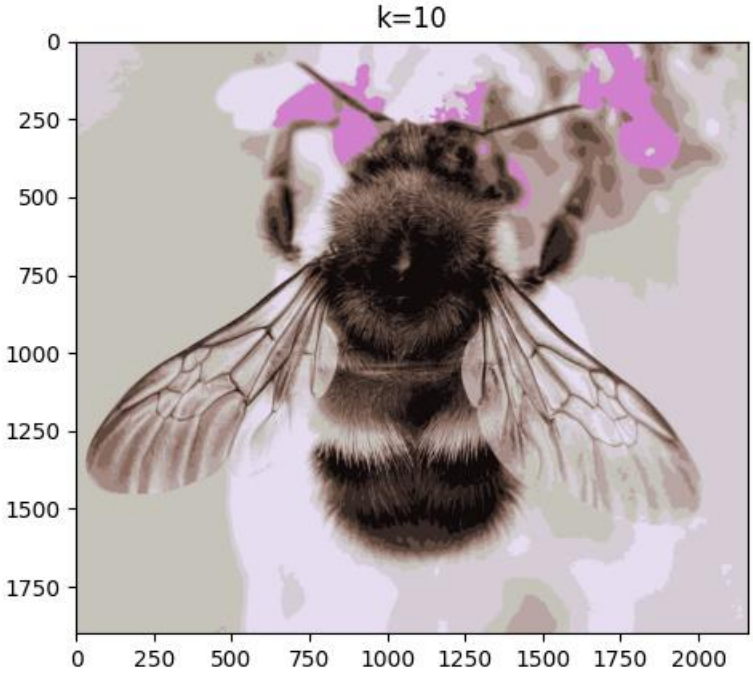


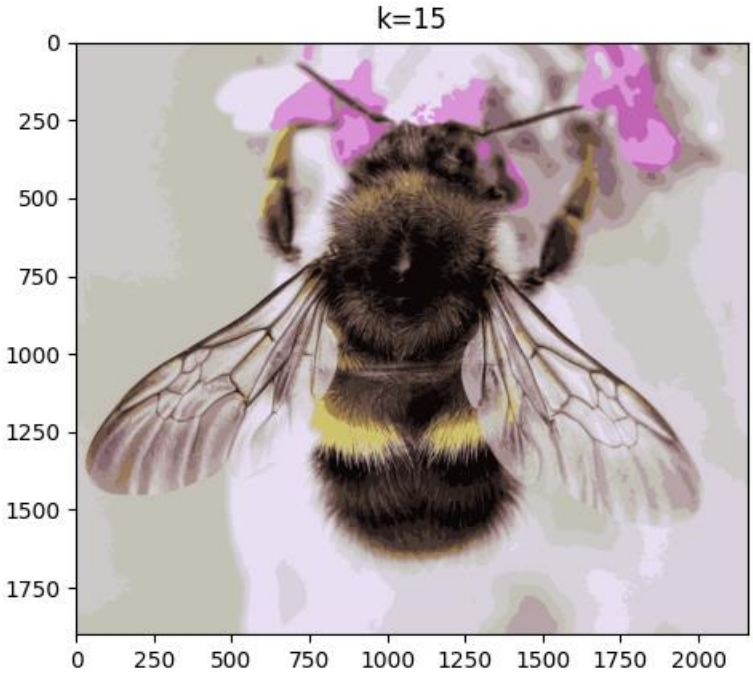

k=3
0
250
500
750
1000
1250
1500
1750
0
250
500
750
1000
1250
1500
1750
2000

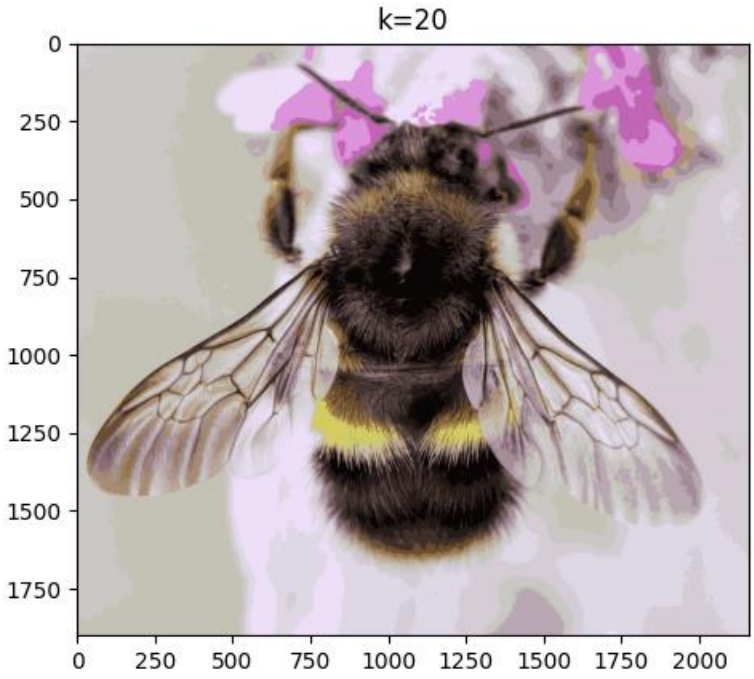

k=4
0
250
500
750
1000
1250
1500
1750
0
250
500
750
1000
1250
1500
1750
2000


k=5
0
250
500
750
1000
1250
1500
1750
0
250
500
750
1000
1250
1500
1750
2000


k=6
0
250
500
750
1000
1250
1500
1750
0
250
500
750
1000
1250
1500
1750
2000

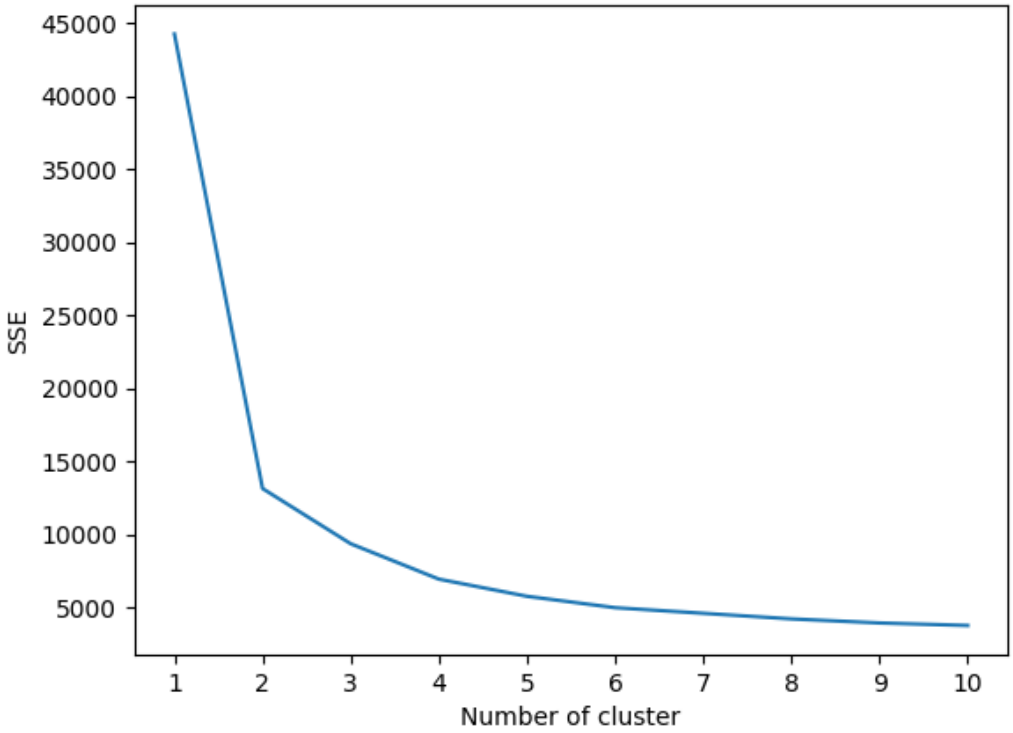

k=10
0
250
500
750
1000
1250
1500
1750
0
250
500
750
1000
1250
1500
1750
2000


k=15
0
250
500
750
1000
1250
1500
1750
0
250
500
750
1000
1250
1500
1750
2000



۱–۲–الف) در روش elbow پس از رسم نمودار هزینه برحسب تعداد مراکز،آخرین نقطه‌ای از نمودار، که در آن تغییر شدید در شیب نمودار هزینه مشاهده می‌شود، به عنوان تعداد بهینه مراکز انتخاب می‌شود. معمولا پس از نقطه انتخابی، کاهش هزینه به اندازه‌ای کم است که می‌توان از آن صرف‌نظر کرد.

۱–۲–ب) کد مربوط به این قسمت در فایل 1-2.py قراردارد. از کتابخانه scikit-learn و معیار هزینه SSE استفاده می‌کنیم. همچنین ستون‌های مرتبط با ID، به علت بی ارتباط بودن حذف شده‌اند. نمودار هزینه برحسب تعداد مراکز به این صورت است:



با توجه به نمودار فوق و با استفاده از تعریفی که در بخش قبل ارائه شد، عدد ۲ برای تعداد مراکز مناسب به نظر می‌رسد.

۱-۲-ج) کد مربوط به این قسمت در فایل 1-2.py قراردارد. پس از اعمال k-means و ذخیره برچسب داده‌ها(شماره کلاستری که داده به آن تعلق دارد)، ابتدا ماتریس‌درهم‌ریختگی را محاسبه می‌کنیم. سپس معیار purity به این صورت محاسبه می‌شود:

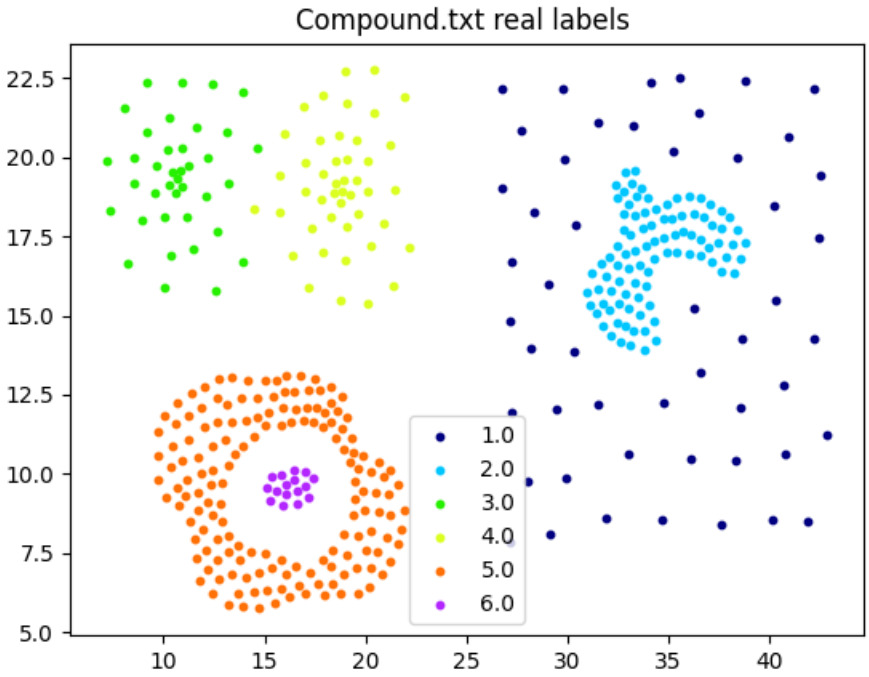
$$purity = \frac{\sum_{c \in columns(conf\_matrix)} max(c)}{number\ of\ data}$$

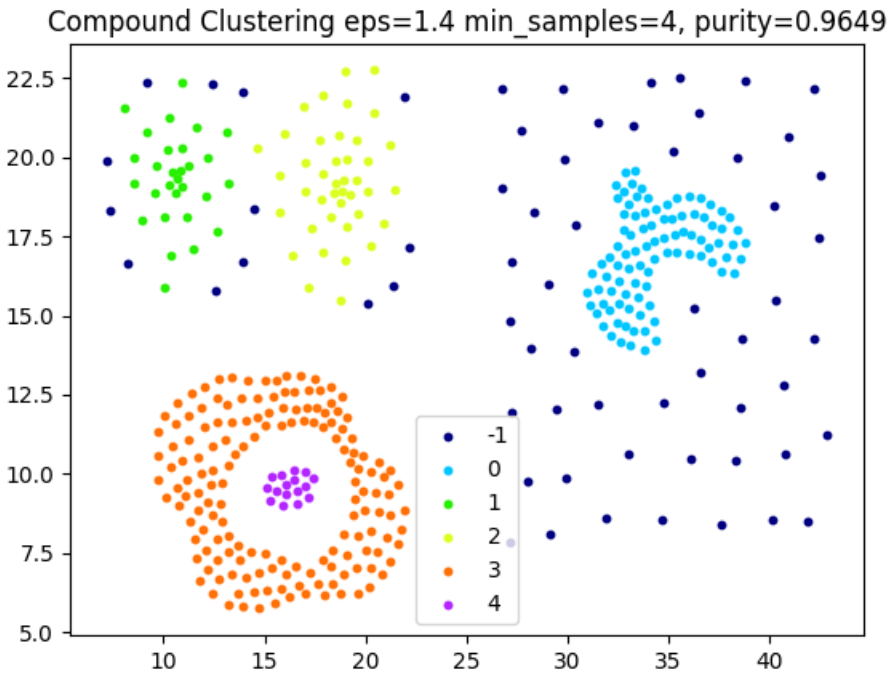
در واقع می‌توان فرض می‌کنیم هر کلاستر مربوط به کلاسی است که بیشترین تعداد از داده‌های آن کلاس در آن کلاستر وجود دارد. بنابراین برای محاسبه معیار purity، بیشترین عددهای هر ستون را با هم جمع می‌کنیم و بر تعداد کل داده‌ها تقسیم می‌کنیم. با این روش معیار purity برای k=2 برابر ۰.۸۹۳۲ بدست می‌آید.

۲)

کد مربوط به این سوال در فایل 2.py قرار دارد. پارامترهای مناسب برای هر دیتاست با سعی و خطا انتخاب شده‌اند. برای محاسبه معیار purity از روشی که در سوال قبل شرح داده‌شد، استفاده می‌شود. برای هر دیتاست ابتدا برچسب‌های واقعی و سپس خروجی خوشه‌بندی رسم می‌شود.

دیتاست Compound:

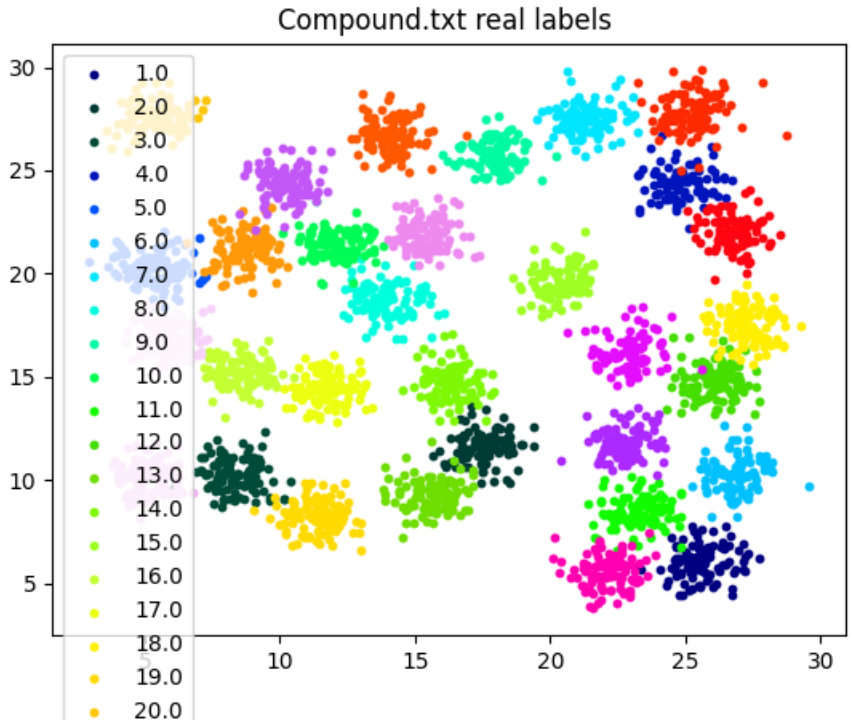

Compound.txt real labels
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
10
15
20
25
30
35
40
1.0
2.0
3.0
4.0
5.0
6.0

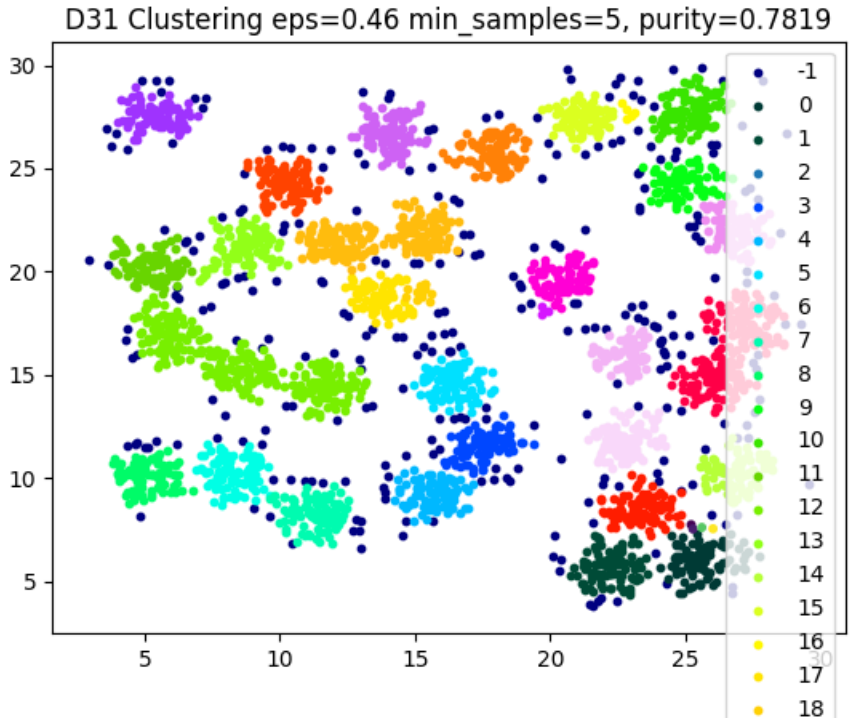

Compound Clustering eps=1.4 min_samples=4, purity=0.9649
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
10
15
20
25
30
35
40
-1
0
1
2
3
4

همانطور که مشاهده می‌شود معیار purity مقدار بالایی دارد و تنها یک کلاس اشتباها نویز(کلاس ۱-) تشخیص داده شده‌اند. اما از آن‌جایی که کل داده-های آن در کلاس نویز قرار دارند شاید بتوان آن را مورد قبول دانست. الگوریتم DBSCAN روی این دیتاست، به دلیل نزدیک بودن نقاط اکثر کلاسترها، عملکرد مطلوبی دارد.

دیتاست D31:

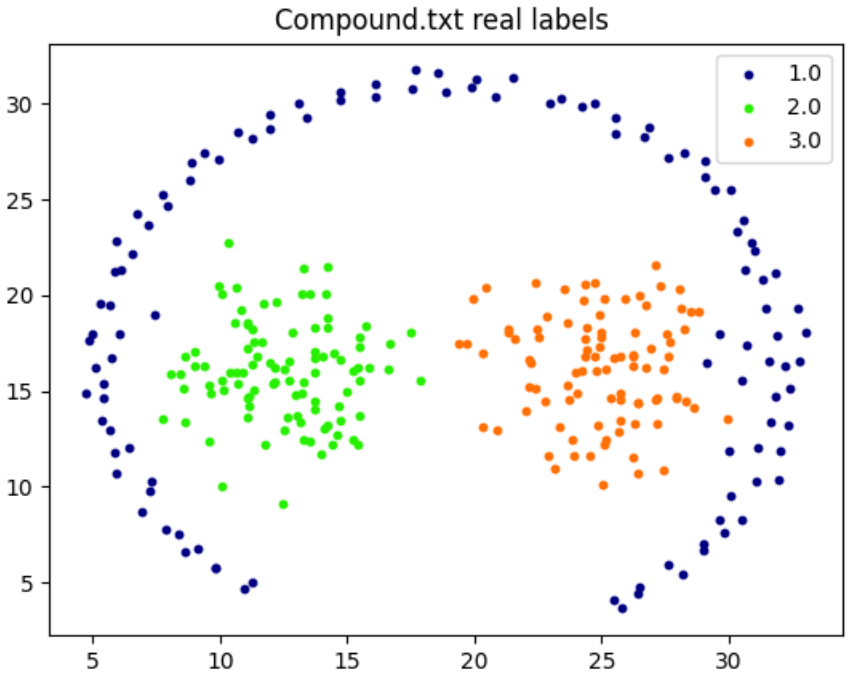
به دلیل زیاد بودن تعداد کلاس‌ها، نمایش رنگ بعضی از کلاسترها بسیار شبیه به هم است.

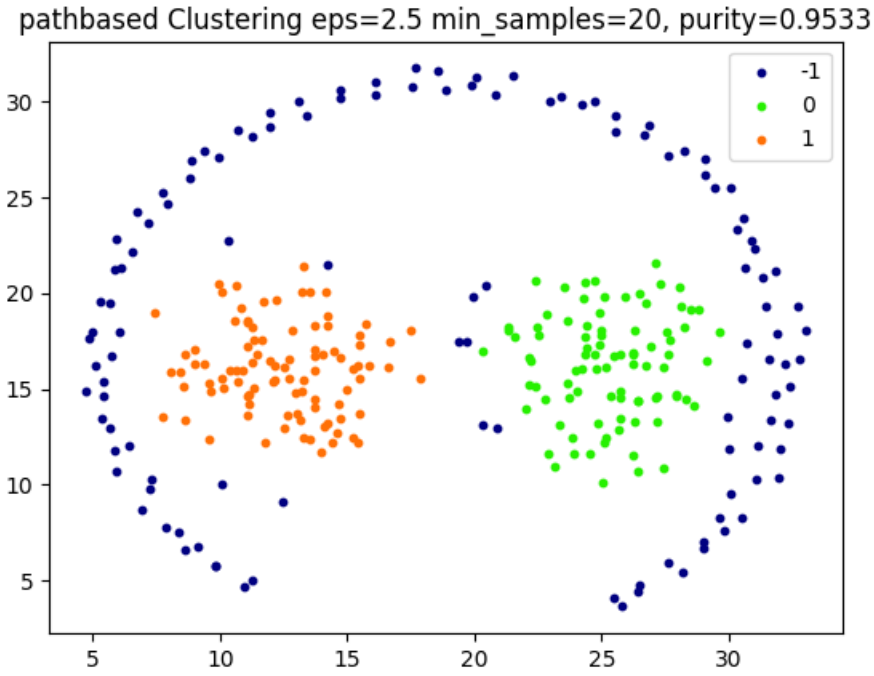




در این دیتاست، فاصله برخی نقاط کلاسترها از هم زیاد است و کلاسترها به هم نزدیک هستند؛ بنابراین نقاط زیادی به عنوان نویز شناخته شده است. لازم به ذکر است پارامترها به گونه‌ای تنظیم شده‌اند که تعداد کلاسترها دقیقا برابر با تعداد کلاس‌ها شوند.

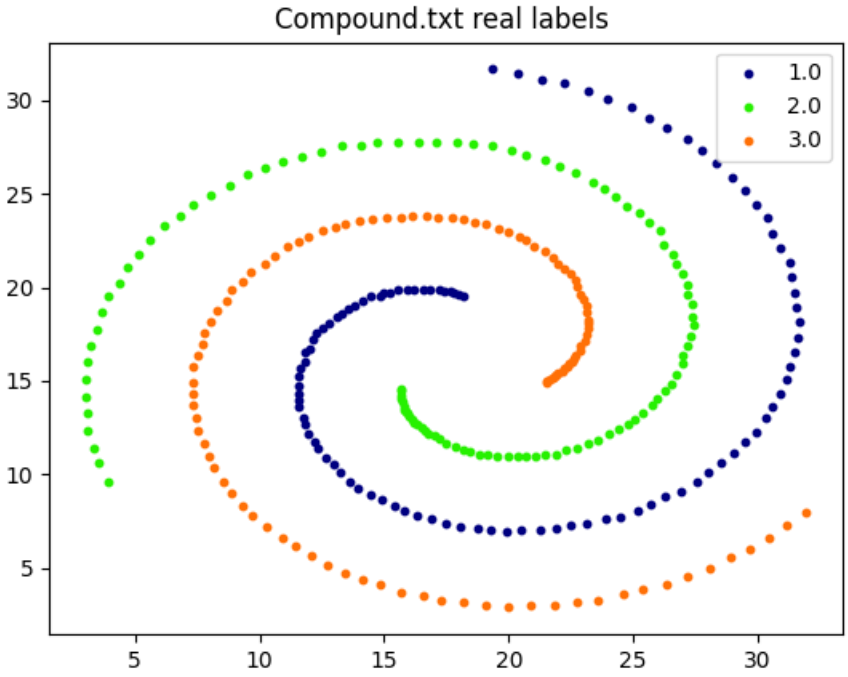
دیتاست pathbased:

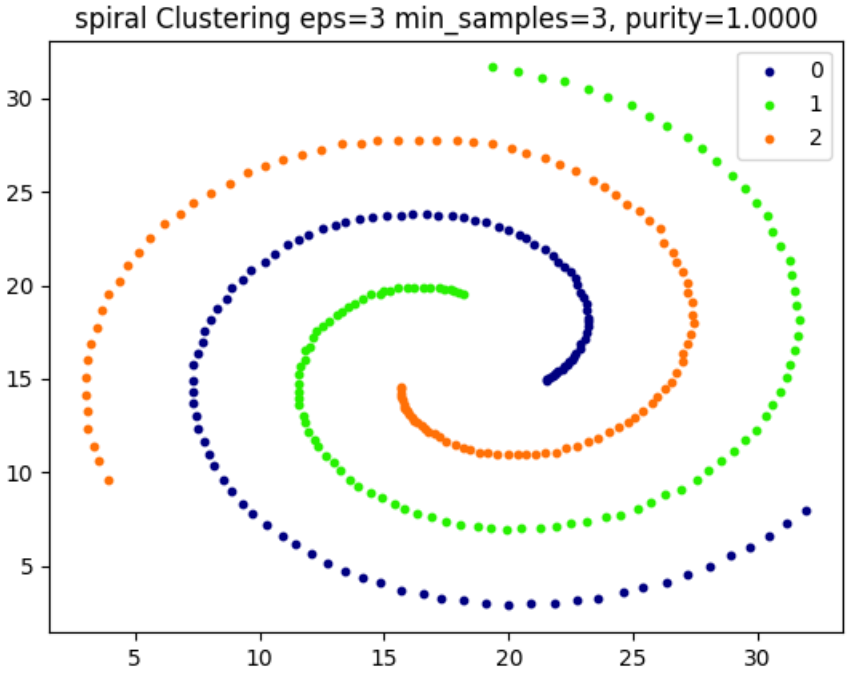




در اینجا نیز به نظر می‌رسد عملکرد DBSCAN مناسب بوده است. اگرچه داده‌های کلاس ۱ نویز در نظر گرفته شده‌اند و در نتیجه ممکن است در صورت اضافه شدن داده‌های نویز دیگر عملکرد آن قابل قبول نباشد.

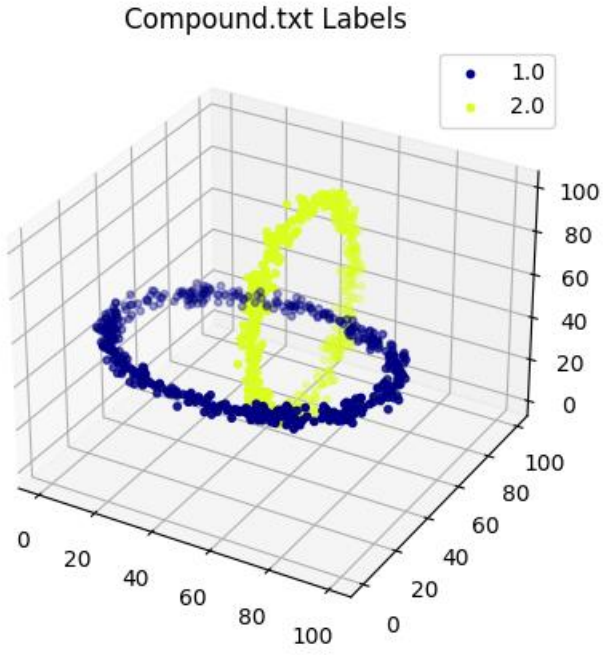
دیتاست spiral:

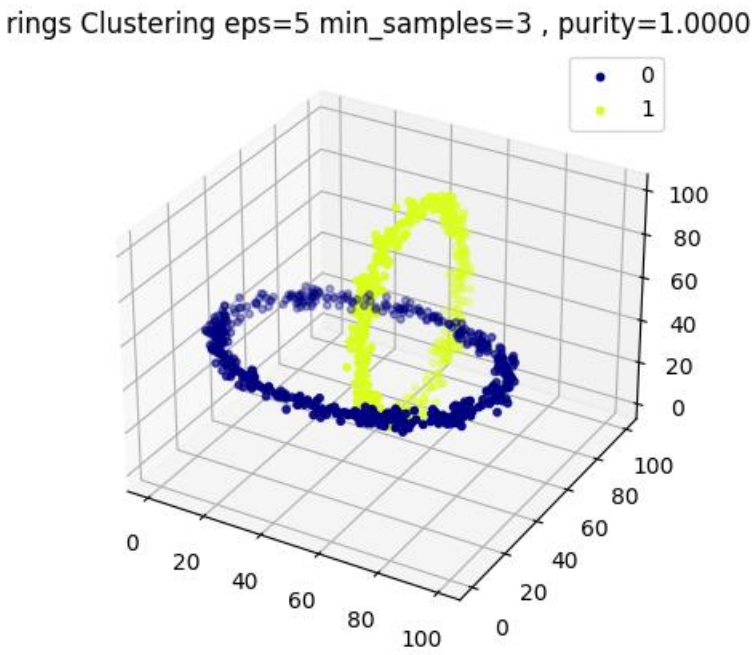




داده‌های کلاس‌ها در این دیتاست، به هم نزدیک هستند و از کلاس‌های دیگر فاصله دارند؛ در نتیجه الگوریتم DBSCAN در اینجا عملکرد بی‌نقصی داشته‌است.

دیتاست rings:





در این دیتاست نیز به علت فاصله مناسب دو کلاس، عملکرد الگوریتم بی نقص بوده‌است. در مجموع از همه دیتاست‌های قبلی می‌توان نتیجه گرفت که الگوریتم DBSCAN، در دیتاست‌هایی که فاصله کلاس‌ها از هم زیاد است، و نمونه‌های درون کلاس در فاصله کمی نسبت به هم قرار دارند، صرفا نظر از شکل کلاس‌ها، می‌تواند عملکرد مطلوبی داشته‌باشد.

۳)

کد مربوط به این سوال در فایل 3.py قرار دارد.

الف) شرط خاتمه را برابر شدن مقادیر V در دو تکرار متوالی در نظر می‌گیریم. البته این شرط اندکی سخت‌گیرانه است و در محیط‌های پیچیده تر می‌تواند زمان خاتمه را بیشتر کند. خروجی پس از ۰.۱۲۹ ثانیه در شکل زیر ارائه شده است. خانه های H با −۱ و خانه G با ۱۱۱ نشان داده شده‌اند.

```
iter: 177
      0     1     2     3     4
0  Down  Left    Up  Left    -1
1  Down  Left    -1    -1  Down
2    Up    Up  Down  Down  Right
3    -1    -1  Left    -1  Right
4  Down  Down  Down  Down   111
          0         1         2         3         4
0  0.007058  0.007575  0.003550  0.001403  0.000000
1  0.010278  0.012103  0.000000  0.000000  0.105758
2  0.013893  0.024864  0.061759  0.093292  0.267506
3  0.000000  0.000000  0.099817  0.000000  0.570874
4  0.101285  0.154906  0.290536  0.579979  0.000000
```

ب) شرط خاتمه طبق الگوریتم policy-iteration، پایدار شدن سیاست بهینه می‌باشد. خروجی پس از ۰.۰۰۸ به این صورت است:

```
iter: 10
       0     1     2     3     4
0  Right  Left    Up  Left    -1
1   Down  Left    -1    -1  Down
2     Up    Up  Down  Down  Right
3     -1    -1  Left    -1  Right
4   Down  Down  Down  Down   111
          0         1         2         3         4
0  0.001451  0.001973  0.000501  0.000099  0.000000
1  0.003688  0.005598  0.000000  0.000000  0.101713
2  0.006783  0.017127  0.052750  0.087651  0.261785
3  0.000000  0.000000  0.088087  0.000000  0.566829
4  0.066773  0.123787  0.266563  0.566814  0.000000
```

سیاست بهینه تنها در خانه شروع با الگوریتم value iteration تفاوت دارد. دلیل این است که برای خانه شروع، حرکت‌های Right و Down معادل هستند. همانطور که مشاهده می‌شود، الگوریتم Value iteration در زمان بسیار کمتری به سیاست بهینه رسیده است.

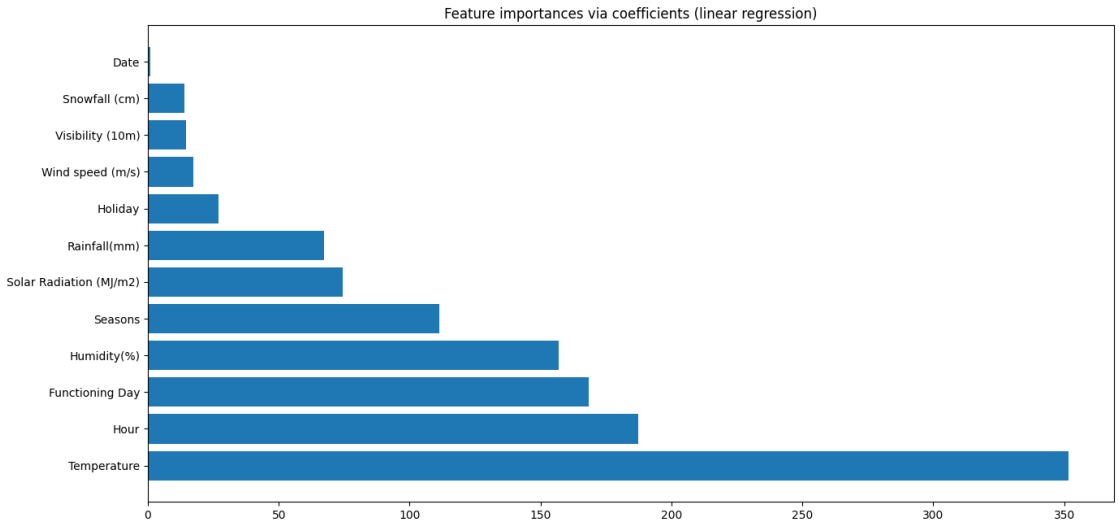
۴)

کد مربوط به این سوال در فایل 4.py قراردارد.

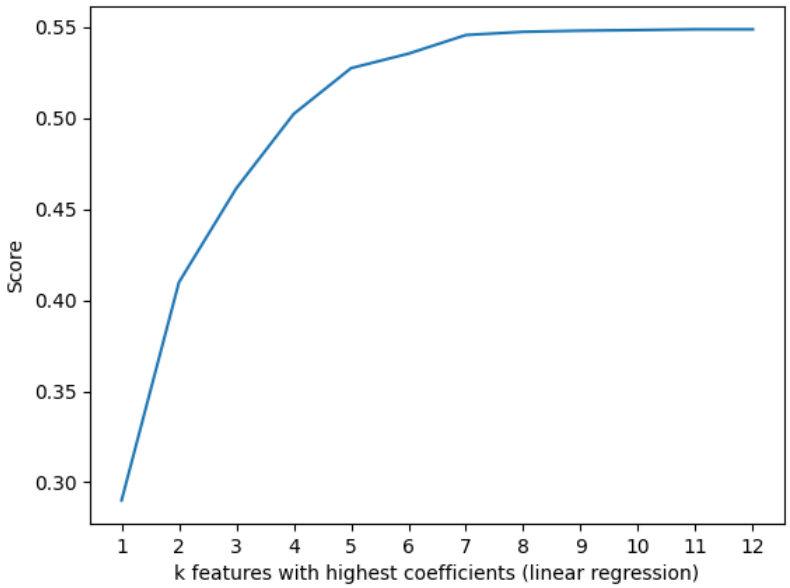
الف) ابتدا ویژگی‌های غیر عددی مانند seasons، holiday و ... را به ویژگی‌های nominal عددی تبدیل می‌کنیم. سپس کل داده‌ها را نرمال می‌کنیم. ویژگی‌هایی که همبستگی آن‌ها بیشتر از ۰.۵ است را می‌یابیم. سپس امتیاز مدل رگرسیون خطی را پس از حذف هر یک حساب می‌کنیم. همانطور که در خروجی زیر مشاهده می‌شود حذف هر یک از ویژگی‌ها باعث کاهش اندکی در امتیاز می‌شود که می‌توان از آن صرفنظر کرد. در نهایت با توجه به خروجی‌ها Dew point temperature را حذف می‌کنیم. زیرا با هر دو ویژگی دیگر، همبستگی بالایی دارد.

```
features with correlation>0.55:
[('Temperature', 'Dew point temperature'), ('Humidity(%)', 'Dew point temperature'), ('Dew point temperature', 'Temperature'), ('Dew point temperature', 'Humidity(%)')]
Score of model after removing features
None                    0.549310
Temperature             0.548140
Humidity(%)             0.543486
Dew point temperature   0.548727
dtype: float64
```

برای حذف ویژگی‌های بیشتر از ضرایب رگرسیون خطی استفاده می‌کنیم. ضرایب برای ویژگی‌های باقی مانده به این صورت است:



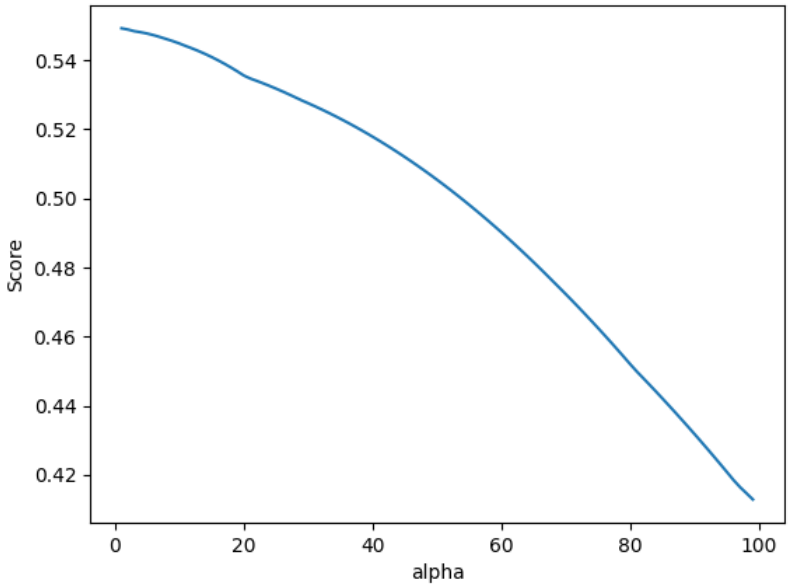
حال نمودار امتیاز مدل رگرسیون خطی به ازای k ویژگی با بیشترین ضرایب را رسم می‌کنیم:



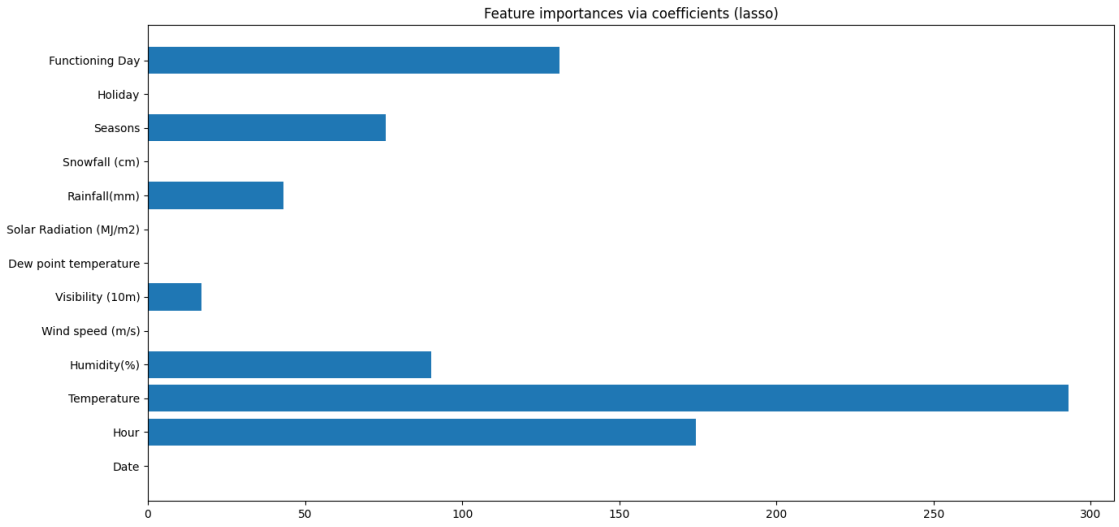
با توجه به نمودار فوق، اگرچه با افزایش k، امتیاز مدل افزایش یافته، اما به نظر می‌رسد انتخاب تنها ۷ ویژگی برتر کافی می‌باشد. در نهایت ویژگی‌های زیر انتخاب شده‌اند. مجدداً تاکید می‌شود که حذف هیچ ویژگی‌ای باعث افزایش دقت نشده و در صورت سختگیری زیاد در امتیاز مدل، شاید بهتر باشد که هیچ ویژگی‌ای حذف نشود.

```
k selected features: Index(['Temperature', 'Hour', 'Functioning Day', 'Humidity(%)', 'Seasons',
       'Solar Radiation (MJ/m2)', 'Rainfall(mm)'],
      dtype='object')
```

ب) برای این قسمت ابتدا نمودار امتیاز بر حسب آلفا را رسم می‌کنیم:



با توجه به نمودار فوق و به صورت تقریبی به نظر می‌رسد آلفای ۳۰، امتیاز مطلوبی دارد و در عین حال ویژگی‌های بیشتری را حذف می‌کند. با انتخاب مقدار ۳۰ برای آلفا، ویژگی‌های انتخابی و ضرایب آن‌ها به صورت زیر است:



در مقایسه با قسمت ب، در این قسمت به جای ویژگی Solar Radiation، ویژگی Visibility انتخاب شده‌است.

ج) به وضوح می‌توان دید که روش حذف ویژگی در اینجا عملکرد مطلوبی ندارد و برای افزایش دقت، باید از دسته‌بندهای غیر خطی استفاده کرد.

۵)

کد مربوط به این سوال در فایل 5.py قرار دارد. لازم به ذکر است داده‌ها به نسبت ۳۰ و ۷۰ به مجموعه‌های آزمون و آموزش تقسیم شده‌اند.

الف) برای رسیدگی به مقادیر گم شده روش‌های مختلفی وجود دارد. از روش‌های ساده مانند استفاده از میانگین تا روش‌های پیچیده‌تر مانند استفاده از KNN. در این سوال برای ویژگی‌های پیوسته از میانگین و ویژگی‌های گسسته از مد استفاده می‌کنیم. این دو روش در عین سادگی عملکرد مطلوبی در اکثر موارد دارند. برای پر کردن مقادیر گم شده داده‌های آزمون، از داده‌های آموزش استفاده می‌کنیم.

ب) برای این‌کار از دسته بند پایه SVM با پنالتی L1 استفاده می‌کنیم. استفاده از پنالتی L1 باعث می‌شود اکثر ضرایب در مدل نهایی نزدیک به صفر باشند. در نهایت از SelectFromModel کتابخانه scikit-learn برای انتخاب ویژگی‌هایی با ضریب‌هایی که بیشتر از 1e-5 هستند استفاده می‌کنیم. ویژگی های انتخابی به شرح زیر است.(با استفاده از داده‌های آموزش)

```
Index(['age', 'creatinine_phosphokinase', 'ejection_fraction', 'serum_sodium',
       'time'],
      dtype='object')
```

ج) از مدل‌های زیر استفاده می‌کنیم:

Model	*Classifier 1*	*Classifier 2*	*Classifier 3*	*Voting*
1	LogisticRegression	DecisionTreeClassifier	GaussianNB	Hard
2	KNeighborsClassifier(k=10)	RandomForestClassifier	AdaBoostClassifier	Soft
3	KNeighborsClassifier(k=5)	KNeighborsClassifier(k=7)	KNeighborsClassifier(k=9)	Hard

در مواردی که پارامترهای دسته‌بندهای پایه ذکر نشده است، از مقادیر پیش‌فرض کتابخانه scikit-learn استفاده شده‌است. دقت مدل‌ها به این صورت است:

```
Accuracy of voting model 1: 0.7888888888888889
Accuracy of voting model 2: 0.8333333333333334
Accuracy of voting model 3: 0.8
```

نتایج فوق به دلیل شافل کردن داده‌ها برای ساخت مجموعه آموزش و آزمون، ممکن است در اجراهای بعدی تغییر کند. به نظر می‌رسد حذف بعضی از ویژگی‌های اضاف باعث شده همه مدل‌ها دقت بالایی داشته باشند. نتایج دسته‌بندهای پایه مدل ۲ به این صورت است.

```
Accuracy of KNeighborsClassifier: 0.7888888888888889
Accuracy of RandomForestClassifier: 0.8111111111111111
Accuracy of AdaBoostClassifier: 0.7555555555555555
```

همانطور که مشاهده می‌شود، دسته بندهای پایه مدل ۲ عملکرد ضعیف‌تری از خود آن دارند. بنابراین می‌توان نتیجه گرفت استفاده از voting در این مسئله مفید بوده‌است.