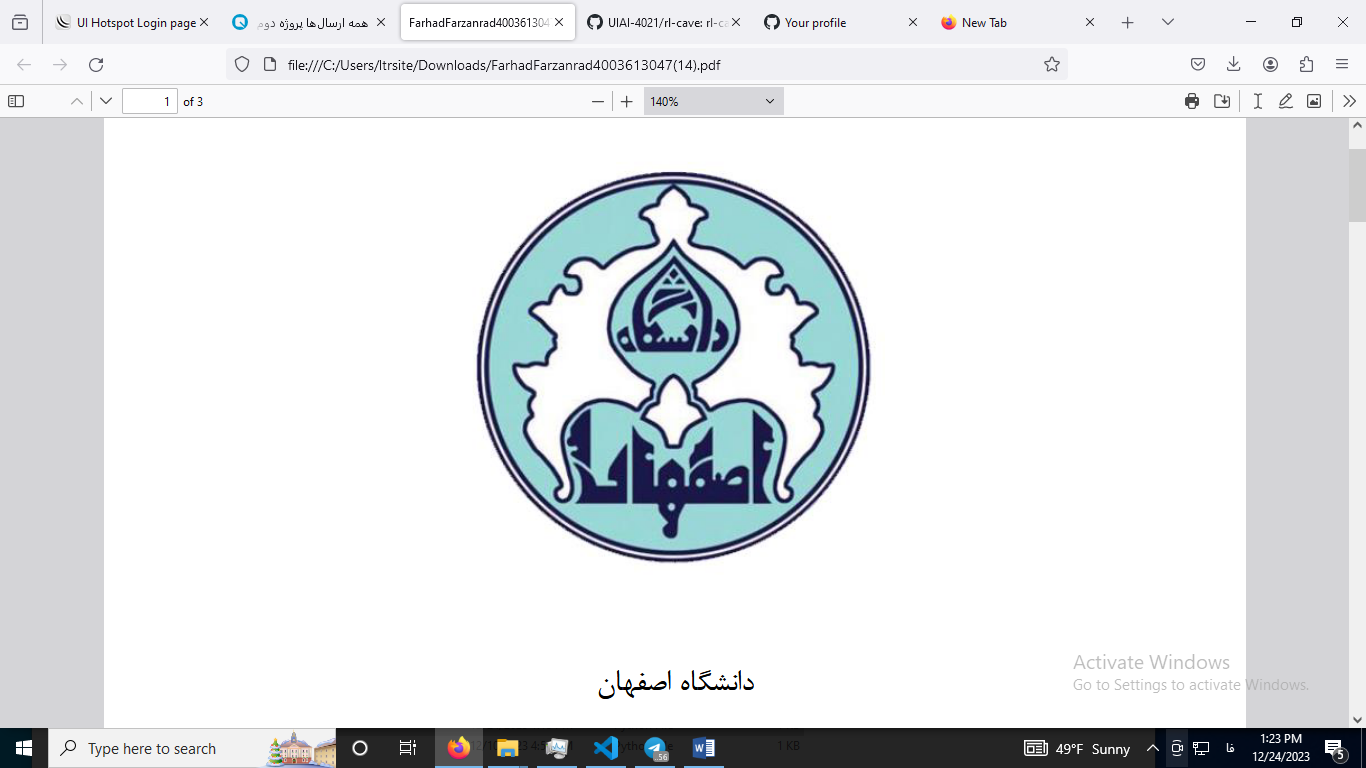
**به نام خدا**



**دانشگاه اصفهان**

**دانشکده مهندسی کامپیوتر**

**پروژه دوم درس مبانی یادگیری ماشین**

**مدل دستهبندی و خوشهبندی برای برگها**

**نام دانشجو : علی کثیری**

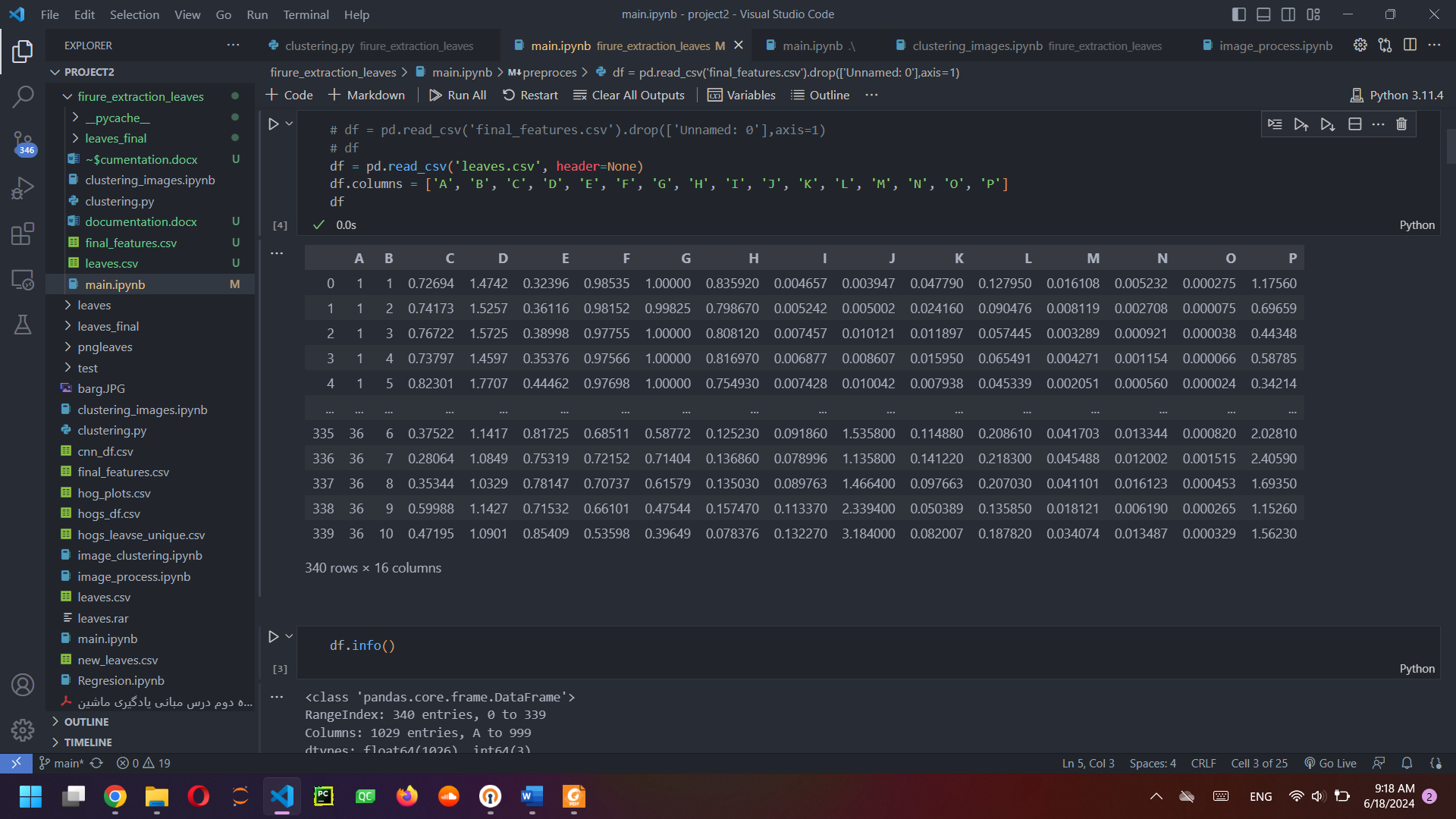
**شمار دانشجویی : ۴۰۰۳۶۱۳۰۵۱**

**نام استاد : دکتر محمد کیانی**

**بهار ۱۴۰۳**

# **پردازش اولیه**

برای استخراج ویژگی از مجموعه عکس ها و اضافه کردن آنها به دیتاست اولیه ابتدا نیاز به شناخت دیتاست اولیه داریم . ستون های دیتاست اولیه دارای هیچ نامی نبودند برای همین آنها را از A تا P نام گذاری کردم.



شکل 1:دیتاست اولیه پس از نام گذاری ستون ها

همانطور که در شکل یک میبینیم ستون A کلاس هر برگ را مشخص کرده و درواقع ستون label است. اما مشکلی که وجود دارد این است که کلاس های این ستون از ۱ تا ۳۶ نوشته شده و این درحالی است که تعداد فولدر های عکس برگ ها بیش از ۳۶ است ، پس برای همین ستون اول را چک کرده تا از تعداد دقیق فولدر هایی که برای پردازش تصویر نیاز است مطلع شویم.

پس از برسی این کار متوجه شدم که کلاس های ۱۶ تا ۲۱ دارای هیچ عضوی نیستند، پس فقط از فولدر های ۱ تا ۱۵ و ۲۲ تا ۳۶ را با استفاده از پردازش تصویر، پردازش کرده و ویژگی های جدیدی از آنها استخراج کردم.

## **حذف و تبدیل پس زمینه عکس ها به پس زمینه ای سفید**

همچنین مجموعه عکس های داده شده دارای پس زمینه ای بودند که در پردازش تصویر ، پس از تبدیل عکس ها به عکس های باینری به مشکل بر میخورد چون که در بعضی عکس ها پس زمینه هم جزئی از برگ به حساب میامد برای همین مجموعه عکس ها را با استفاده از کتاب خانه rembg و تابع remove به فرمت png در آورده و پس زمینه همه برگ هارا حذف کردم. و سپس پس زمینه ای یکسان به رنگ سفید برای همه عکس ها قرار دادم.



شکل 2:نمونه ای از مجموعه عکس ها پس از حذف و تبدیل پس زمینه عکس ها به سفید.

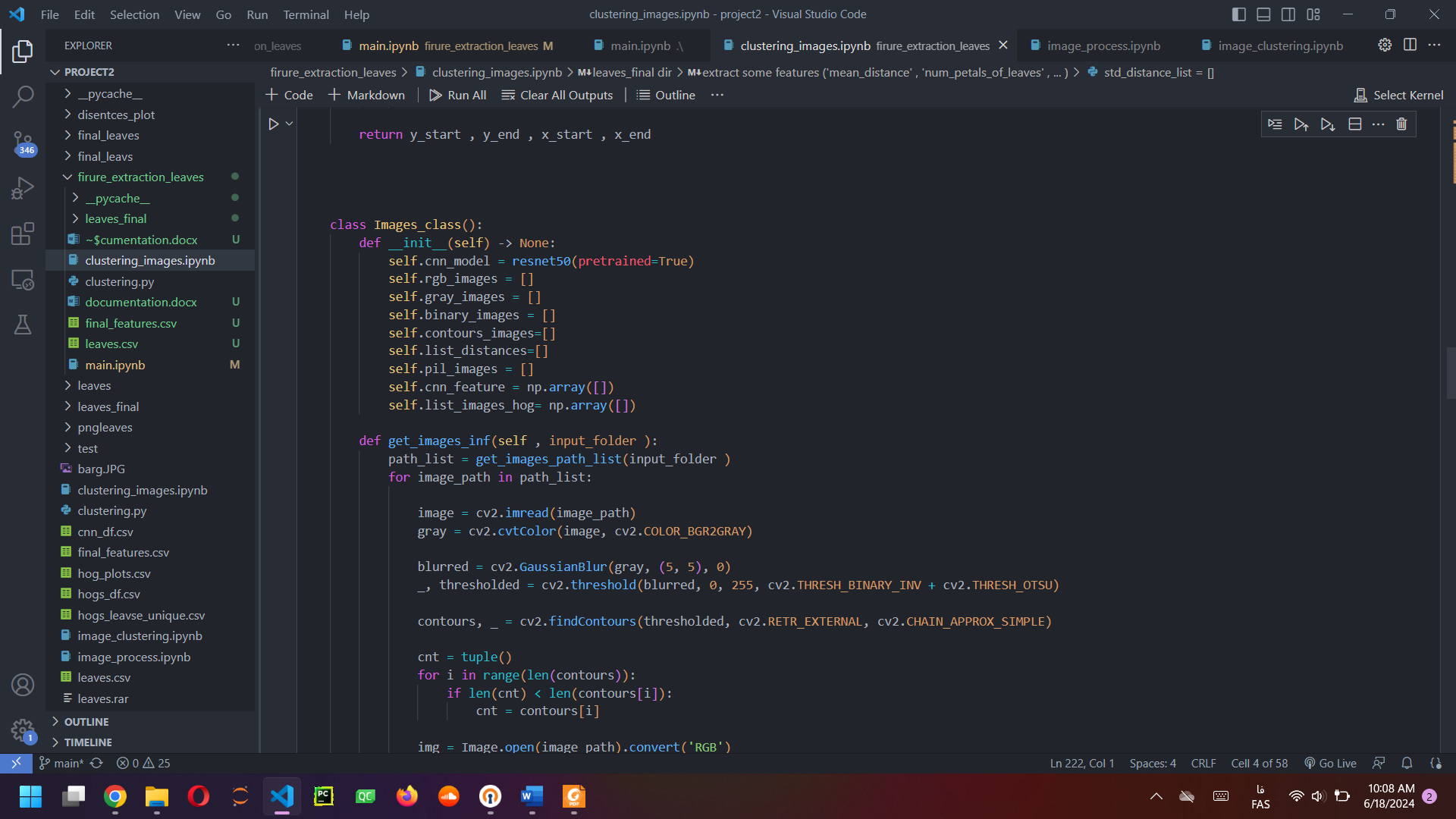
عکس بالا نمونه ای از تبدیل نهایی عکس ها را نشان میدهد ، در ادامه ویژگی های استخراج شده را برسی میکنیم.

# **استخراج ویژگی**

کد استخراج ویژگی در فایل feature\_extraction.py قرار دارد و برای این کار یک کلاس به نام Images\_class نوشتم که در آن با استفاده از کتاب خانه cv2 عکس ها را پردازش میکنم.

در ادامه توابع این کلاس را به مختصر توضیح میدهم.

* تابع سازنده کلاس



شکل 3

در این تابع متغیر ها و لیست های اولیه را مقداردهی میکنیم که کارایی هر یک به شرح زیر است:

-CNN\_MODEL: یک مدل RESNET-50 از قبل آموزش داده شده برای استخراج ویژگی.

- rgb\_images: لیست برای ذخیره تصاویر RGB.

- gray\_images: لیست برای ذخیره تصاویر سیاه و سفید.

- binary\_images: لیست برای ذخیره تصاویر باینری.

- contours\_images: لیست برای ذخیره نقاط تصاویر کانتور.

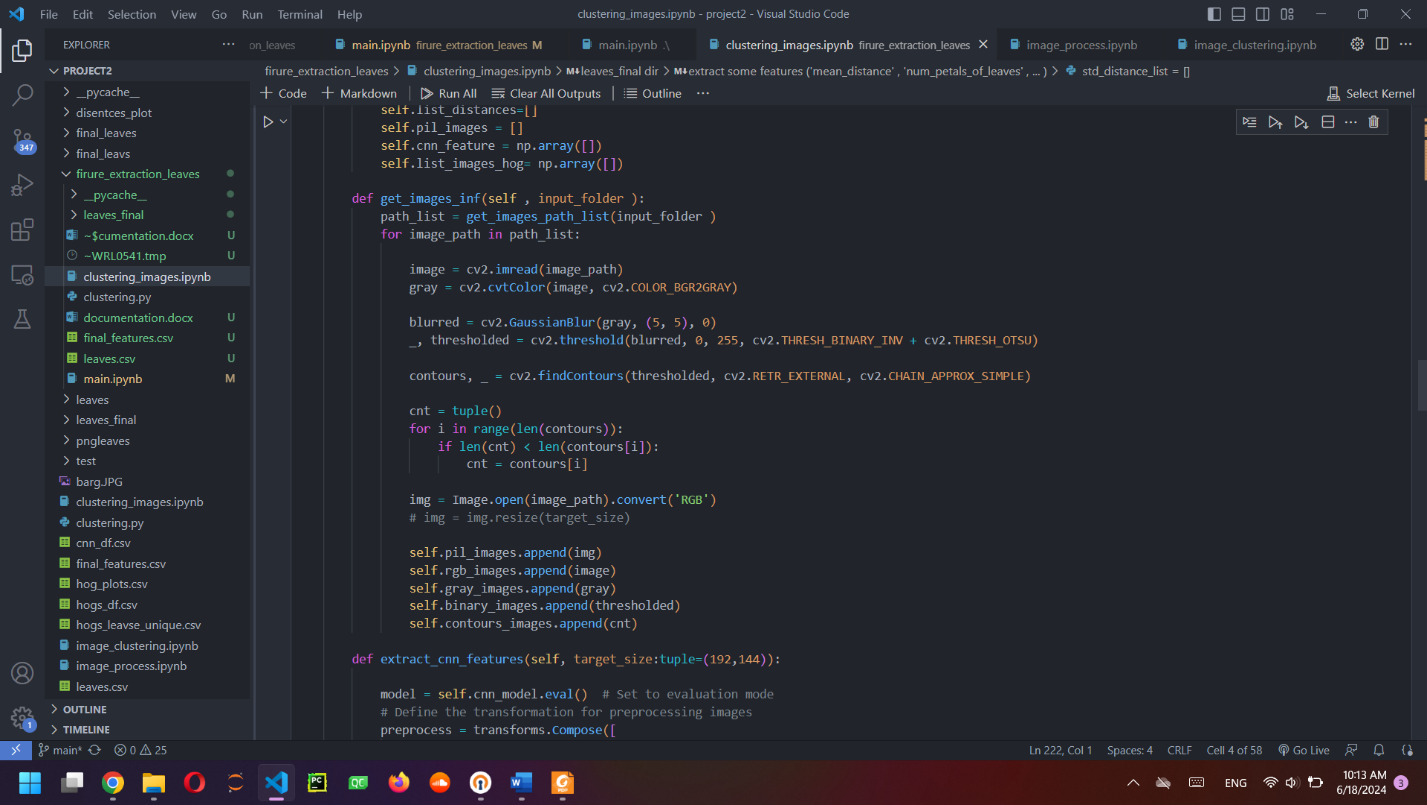
- list\_distances: لیست برای ذخیره مسافت نقاط کانتور از برگ.

- pil\_images: لیست برای ذخیره تصاویر در قالب PIL.

- cnn\_feature: آرایه ای برای ذخیره ویژگی های CNN.

- list\_images\_hog: آرایه ای برای ذخیره ویژگی hog(Histogram of Oriented Gradients).

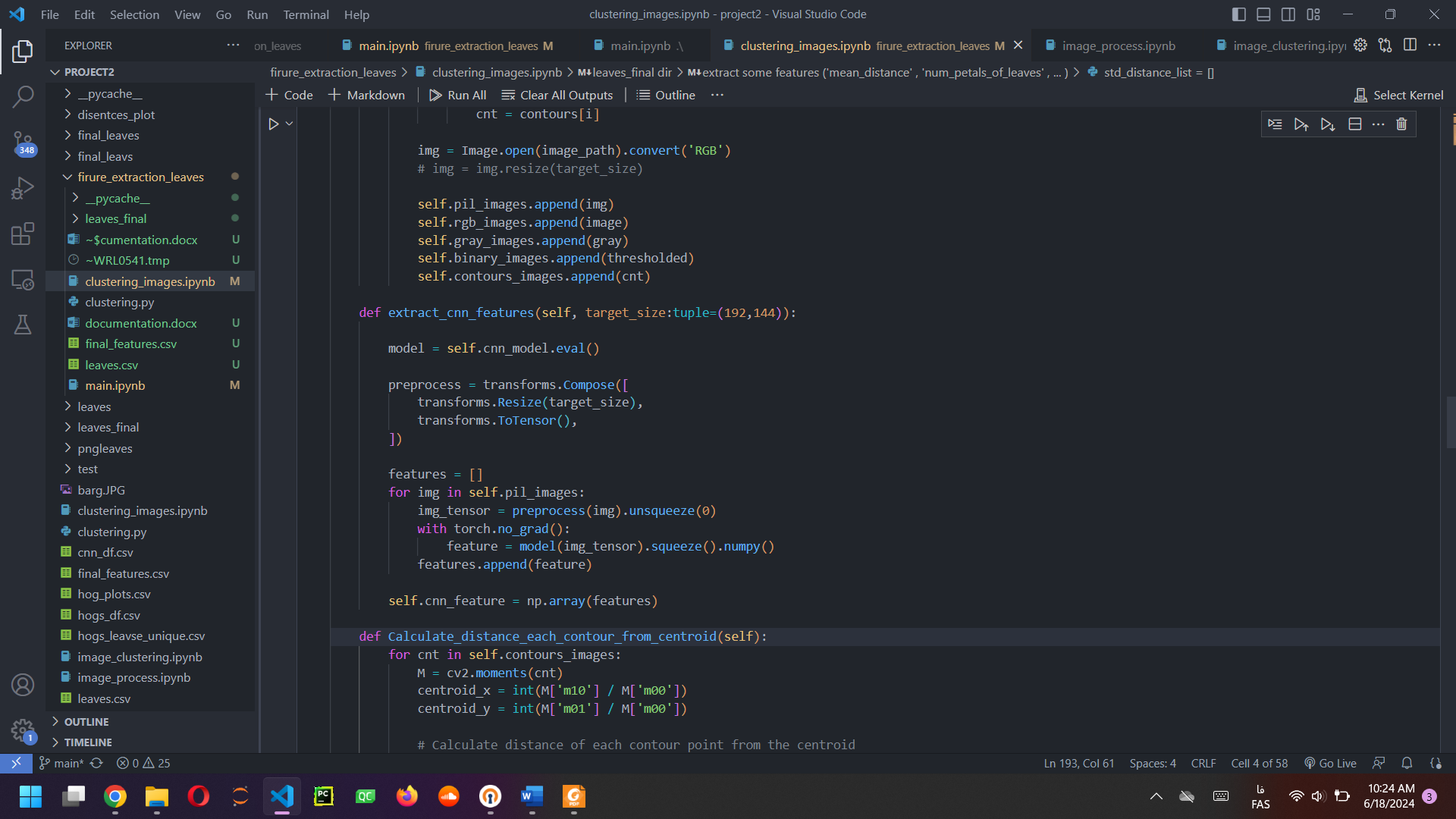
* تابع get\_images\_inf



شکل 4

این تابع تصاویر را از پوشه مشخص شده بارگیری می کند ، آنها را به قالب هایی مثل تصاویر RGB ، تصاویر سیاه و سفید و تصاویر باینری تبدیل می کند و کانتورها را استخراج و بزرگترین آن را انتخاب و ذخیره می کند.

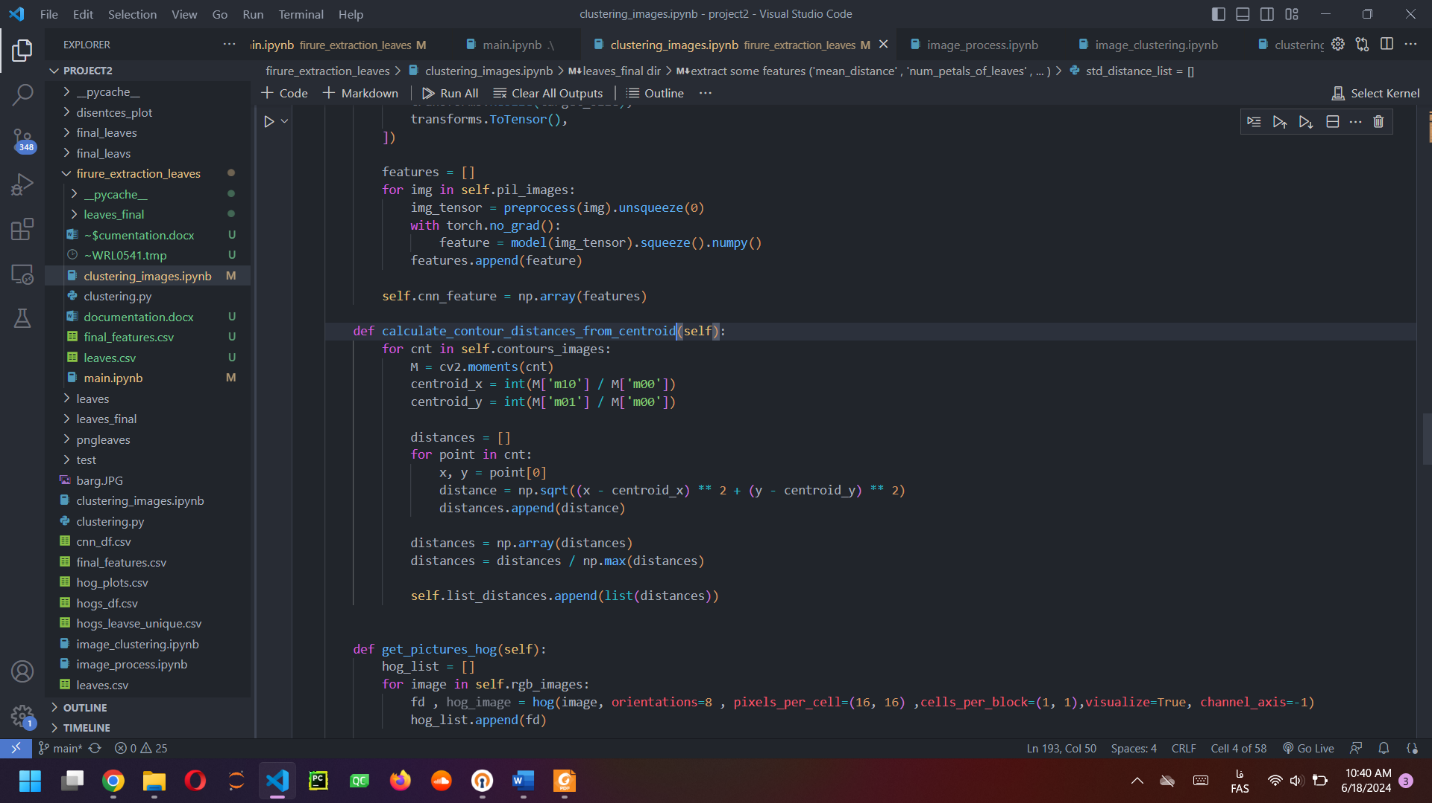
* تابع extract\_cnn\_features



شکل 5

این تابع ویژگی هایی از تصاویر با استفاده از یک مدل CNN از پیش آموزش داده شده(RESNET-50) استخراج و ذخیره میکند. ورودی targe\_size برای تبدیل کردن عکس ها به ابعاد دیگر است که در اینجا با (۱۴۴و۱۹۲) عکس را ۵ برابر کوچک تر کرده تا تعداد ویژگی های استخراجی کمتر شود.

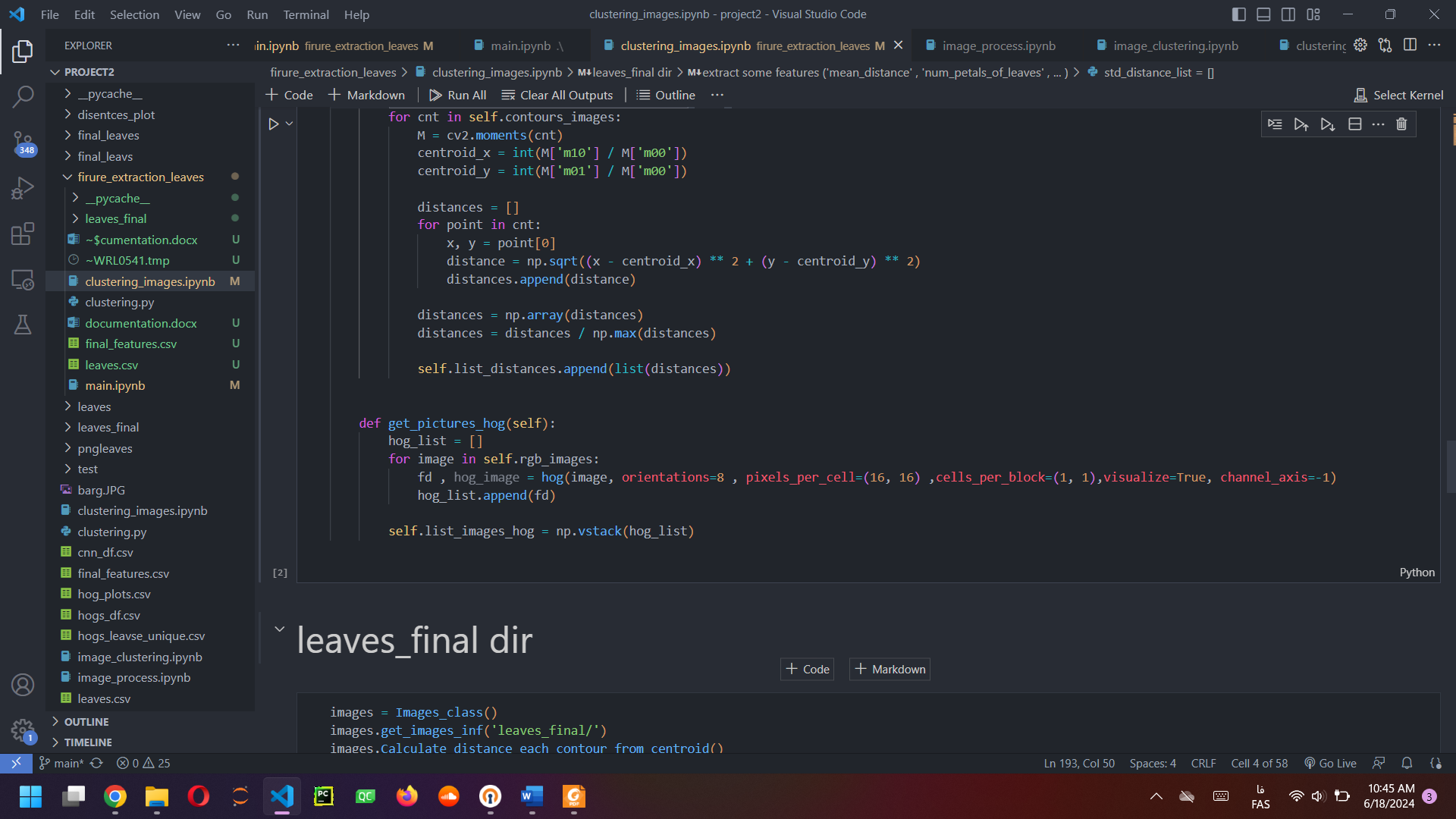
* تابع calculate\_contour\_distances\_from\_centroid



شکل 6

این تابع فاصله اقلیدسی هر نقطه کانتور را از مرکز برگ حساب کرده و در لیست distances و سپس برای نرمالایز کردن این فاصله ها ، همه را بر بیشترین فاصله تقسیم میکند و در این صورت همه فاصله ها را به عددی بین ۰ تا ۱ تبدیل و ذخیره میکند که در ادامه بیشتر به آن میپردازیم.

* تابع get\_pictures\_hog

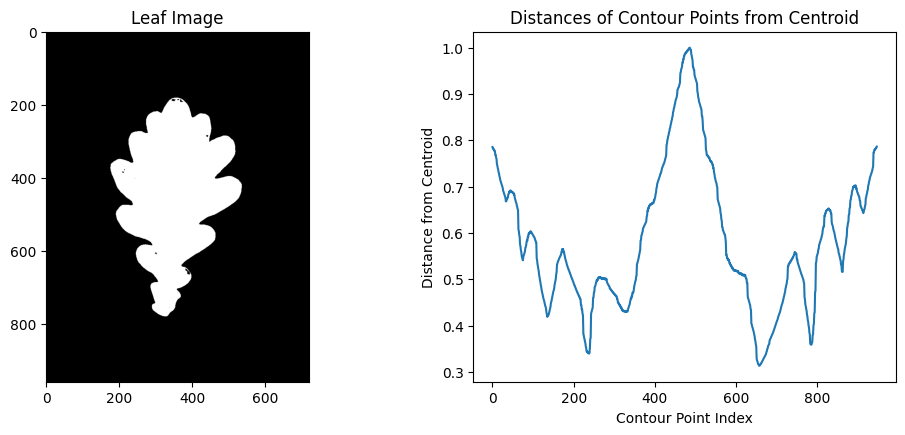


شکل 7

این تابع هم با نرخ pixels\_per\_cel = (16,16) ویژگی hog(Histogram of Oriented Gradients) را استخراج میکند. در ادامه ویژگی های استخراج شده از عکس ها را برسی میکنیم.

## **ویژگی های distances**

همانطور که در توضیح تابع calculate\_contour\_distances\_from\_centroid گفتم این تابع فاصله اقلیدسی هر نقطه کانتور را از مرکز برگ حساب کرده و در لیست distances ذخیره میکند.



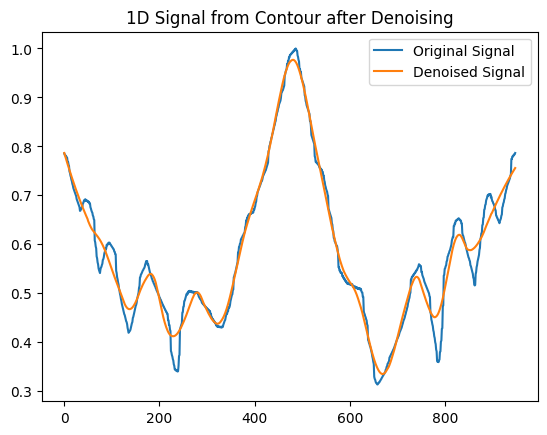
شکل 8

شکل ۸ به خوبی نمودار فواصل برگ سمت چپ را نشان میدهد. با استفاده از این فواصل میتوان برگ هایی با اشکال متفاوت به خصوص در برگ هایی که فرورفتیگی هایی بخاطر شکل و گلبرگ های خود دارند را از یک دیگر تشخیص داد. یکی از ویژگی هایی که از این مجموعه فواصل میتوان استخراج کرد تعداد گلبرگ های هر برگ است.

#### **تعداد گلبرگ های هر برگ**

همانطور که در شکل ۸ مشخص است با شمردن تعداد نقاط مینیمال محلی نمودار فواصل ، میتوان حدودی از تعداد گلبرگ های برگ بدست آورد و با استفاده از این ویژگی بین برگ های تک گلبرگی و چند گلبرگی تمایز ایجاد کرد.

تنها مشکل پیدا کردن مینیم های محلی اصلی بود که با تابع چون که مینیم های بیشتری هم در این عکس وجود دارد که با بسیار ریز و جزئی هستند. برای حل این مشکل از تابع savgol\_filter از کتاب خانه scipy.signal نمودار را از یک فیلتر عبور داده و در این صورت شکل نمودار نرم و صاف میشود و سپس با تابع find\_peaks مینیمم های محلی نمودار را بدست می آوریم.



شکل 9:نمودار فاصله ها پس از عبور از فیلتر

همانطور که در شکل ۹ مشاهده میشود نمودار فاصله ها پس از عبور از فیلتر نرم شده و به راحتی میتوان تعداد مینیمال های محلی آن را بدست آورد. در نهایت این ویژگی را با نامnum\_petals\_of\_leaves ذخیره کرم.

همچنین ویژگی های mean\_distance ، std\_distance و min\_distance را نیز به عنوان ویژگی های جدید در مجموعه داده ذخیره کردم.

## **ویژگی های ابعاد برگ ها**

#### **نسبت طول به عرض برگ**

ابتدا با استفاده از تابع find\_Length\_and\_width اطلاعات طول و عرض برگ را بدست میاوریم این تابع به ما چهار مقدار y\_start , y\_end , x\_start , x\_end را برمیگرداند که درواقع ابعاد مستطیلی است که برگ را احاطه کرده و سپس نسبت طور به عرض هر برگ را حساب و در یک لیست ذخیره میکنیم و در نهایت با نام legth\_to\_width\_ratio در مجموعه داده جدید ذخیره میشود. از مزیت های این ویژگی میتوان به تمایز برگ های باریک و بلند نسبت به برگ های کوتاه و پهن اشاره کرد.

#### **نسبت برگ مساحت برگ به پس زمینه خود**

برای بدست آوردن این ویژگی از تابع Calculate\_avg\_leaf\_to\_background\_ratio استفاده میکنیم. در این تابع نسبت تعداد پیکسل های سفید که درواقع همان برگ است را نسبت به پس زمینه خودش حساب و بر میگرداند. از آنجایی که عکس ها هرکدام با فاصله یکسانی نسبت به برگ گرفته نشده اند ، برای جلو گیری از خطای زیاد در این ویژگی بجای فرستادن آرایه کامل عکس به این تابع ، فقط عکسی با ابعاد مستطیل محیطی برگ را برای تابع ارسال میکنیم.



شکل۱۰:

همانطور که در شکل ۸ مشاهده میشود بجای فرستادن عکس سمت راست ، عکس سمت چپ را برای این تابع ازسال میکنیم تا از خطا جلوگیری شود.

#### **ویژگی Form\_Factor**

فاکتور فرم یک توصیف کننده شکل است که نشان می دهد شکل با یک دایره کامل چقدر مشابه است. این ویژگی از ۰ تا ۱ متغیر است و هرچقدر که به ۱ نزدیک تر باشد شکل به دایره نزدیک تر است.

#### **ویژگی** **Convexity**

این ویژگی برای اندازه گیری شکل محدب است که درواقع همان نسبت محیط شکل به محیط بدنه محدب آن تعریف شده است. مقدار این ویژگی بزرگتر یا مساوی ۱ است که هرچقدر این مقدار به یک نزدیک تر باشد محدب تر است.

#### **محیط هر برگ**

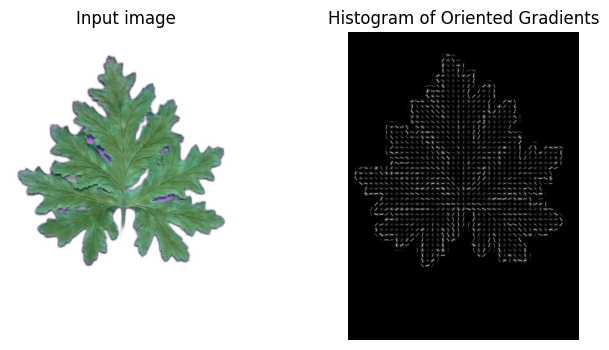
طول کل کانتور است که شکل را تعریف می کند. پس باید یک اندازه گیری ساده از مرز شکل انجام داد که با استفاده از تابع cv2.arcLength() این کار را انجام دادم.این تابع کانتور هر برگ را گرفته و محیطش را بر میگرداند.

## **ویژگی رنگ های برگ ها**

سه ویژگی red\_mean\_color و green\_mean\_color و blue\_mean\_color را نیز به سادگی با میانگین گرفتن از آرایه های rgb عکس ها بدست آورده و در مجموعه داده جدید ذخیره کردم.

## **ویژگی HOG(Histogram of Oriented Gradients)**

در این قسمت با استفاده از کتاب خانه skimage.feature و تابع hog این ویژگی را استخراج میکنیم. این تابع ورودی هایی برای تنظیم دارد که با تنظیم آنها میتوان ابعاد ویژگی های خروجی را تایین کرد چرا که در حالت دیفالت ابعاد برگشتی این تابع بسیار بزرگ است. از این ورودی ها میتوان به orientations ، cells\_per\_block و pixels\_per\_cell که به ترتیب با کاهش اولی و افزایش دو ورودی دیگر میتوان ابعاد ویژگی های برگشتی را کاهش داد.



**شکل 10**

همانظور که در شکل ۱۰ مشاهده میشود این تابع هیستوگرام گرادیان های جهت دار برگ ها را بدست میاورد که در صورت زیاد بودن تعداد فیچر ها میتوان با PCA تعداد فیچر ها را کمتر کرده و ‌ذخیره کرد.

## **استخراج ویژگی با مدل RESNET-50**

در این قسمت با استفاده از یک شبکه عصبی کانولوشنال (CNN) از پیش آموزش دیده، به ویژه ResNet-50، که یک مدل پرکاربرد برای کارهای طبقه بندی تصاویر است از تصاویر ویژگی استخراج کردم.

این روش با مقداردهی اولیه مدل ResNet-50 و تنظیم آن در حالت ارزیابی شروع می شود. ر تصویر مراحل پیش‌پردازش از پیش تعریف‌شده، از جمله تغییر اندازه و تبدیل تانسور را طی می‌کند و سپس از مدل ResNet-50 عبور می‌کند. مدل، تانسور تصویر را پردازش می‌کند و یک بردار ویژگی را نشان می‌دهد که نشان‌دهنده ویژگی‌های سطح بالایی است که در طول آموزش مدل بر روی یک مجموعه داده تصویری بزرگ آموخته شده‌اند.

این بردارهای ویژگی، که جنبه‌های اساسی تصاویر، مانند اشکال، بافت‌ها و الگوها را به تصویر می‌کشند، جمع‌آوری و در یک لیست ذخیره می‌شوند. در نهایت، لیست بردارهای ویژگی به یک آرایه NumPy تبدیل می‌شود.

در نهایت با استفاده از این مدل برای هر عکس ۱۰۰۰ ویژگی استخراج شد و به مجموعه داده اضافه شد.

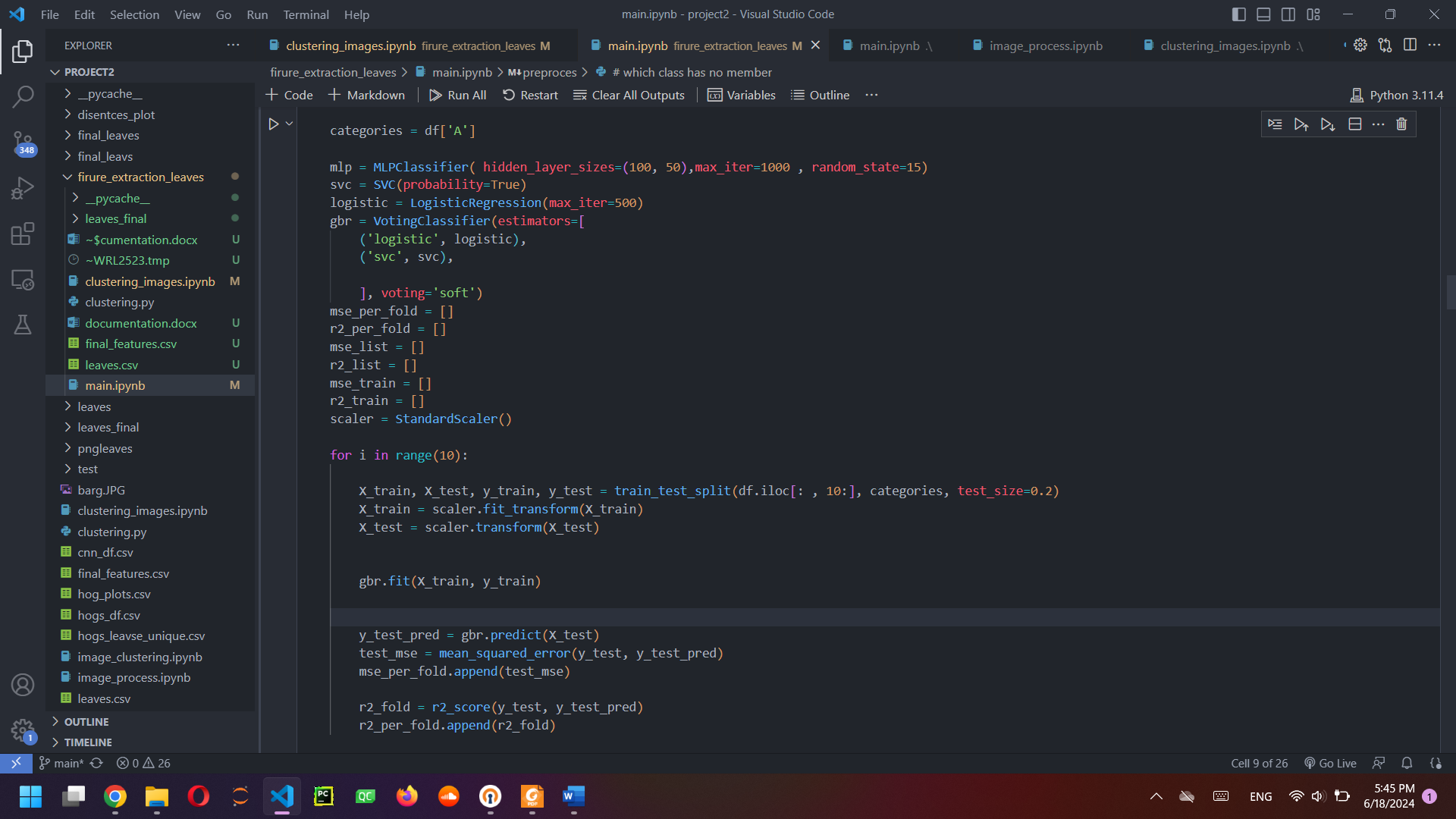
# **دسته بندی برگ ها**

## **پیش پردازش داده ها**

برای دسته بندی برگ ها ستون اول دیتاست اولیه که همان label های ما هستند را در متغیر categories و اما ستون های دیگر به تعداد ۱۵ ستون بود که با آزمایش و خطا در نهایت چهار ستون آخر این دیتاست را به اضافه ویژگی های دیگری که در بخش استخراج ویژگی به آنها اشاره شد برای یادگیری مدل استفاده کردم.

پس از انتخاب ویژگی های مناسب مرحله پیش پردازش داده دیگری نداریم چون که داده ها کامل و بدون داده های از دست رفته هستند. همچنین از آنجایی که ویژگی ها از روی پردازش تصوری بدست آمده اند ، مرحله حذف داده های پرت را نیز نادیده گرفته چون که همه اطلاعات از روی عکس ها بدست آمده و عکس ها نیز مشکلی نداشتند.

## **جدا کردن داده ها به دو بخش Train , Test و scale کردن داده های X**

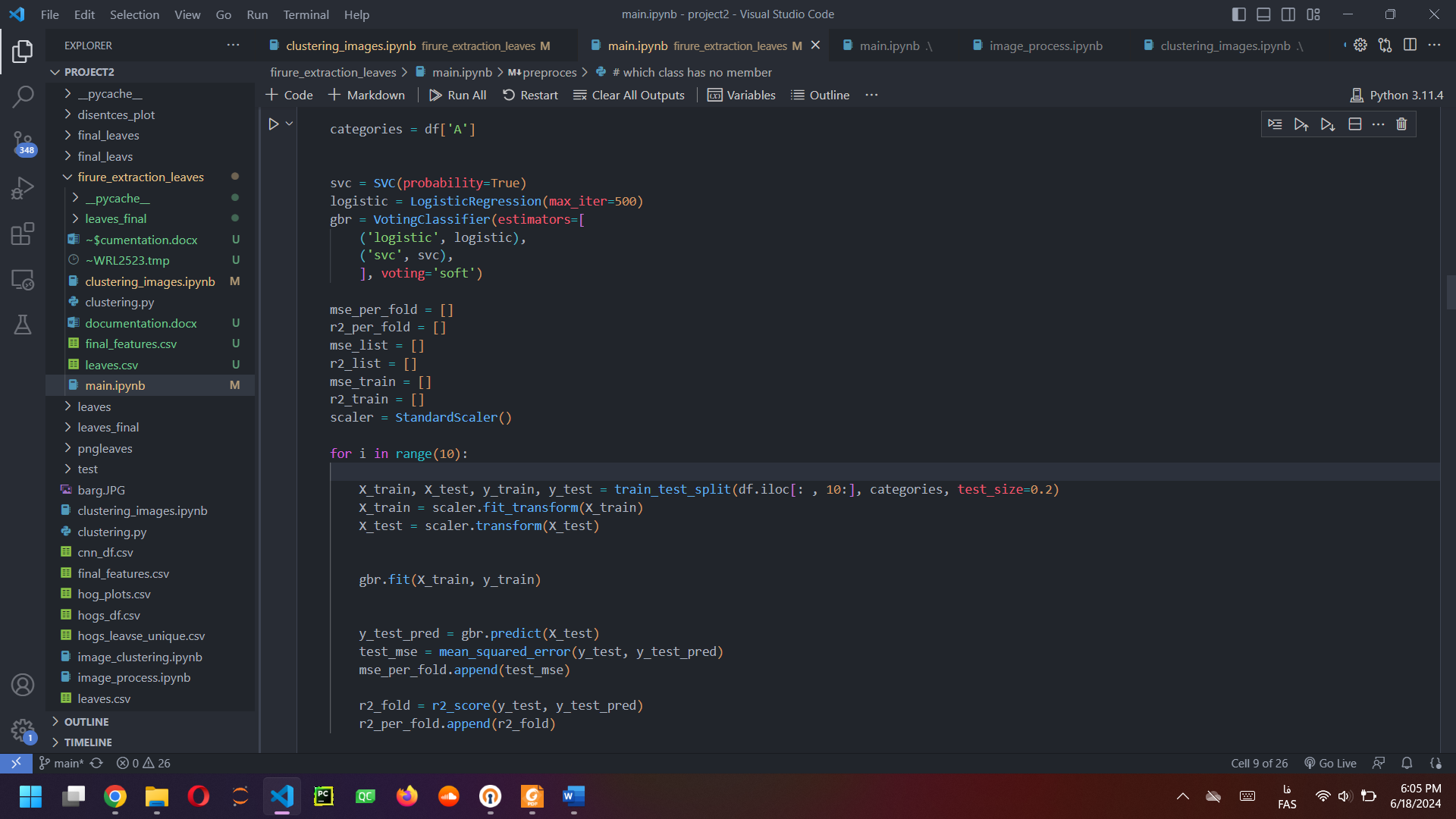


شکل 11

همانطور که در شکل ۱۰ مشخص است مجموعه داد ها را به دو بخش train , test در آورده و سپس داده های x\_train , x\_test را scale کرده تا به نتیجه بهتری برسیم. همچنین از random\_state استفاده نکرده چرا که این کد را در یک حلقه ۱۰ تایی اجرا میکنیم تا میانگینی از نتیجه نهایی مدل بدست آوریم.

## **یادگیری مدل**

از بین الگوریتم های امتحان شده برای این مجموعه داده ، در نهایت دو الگوریتم SVC , LogisticRegression دارای نتایج بهتری بودند که بر همین اساس برای یاد گیری مدل تصمیم به استفاده از VotingClassifier کردم.

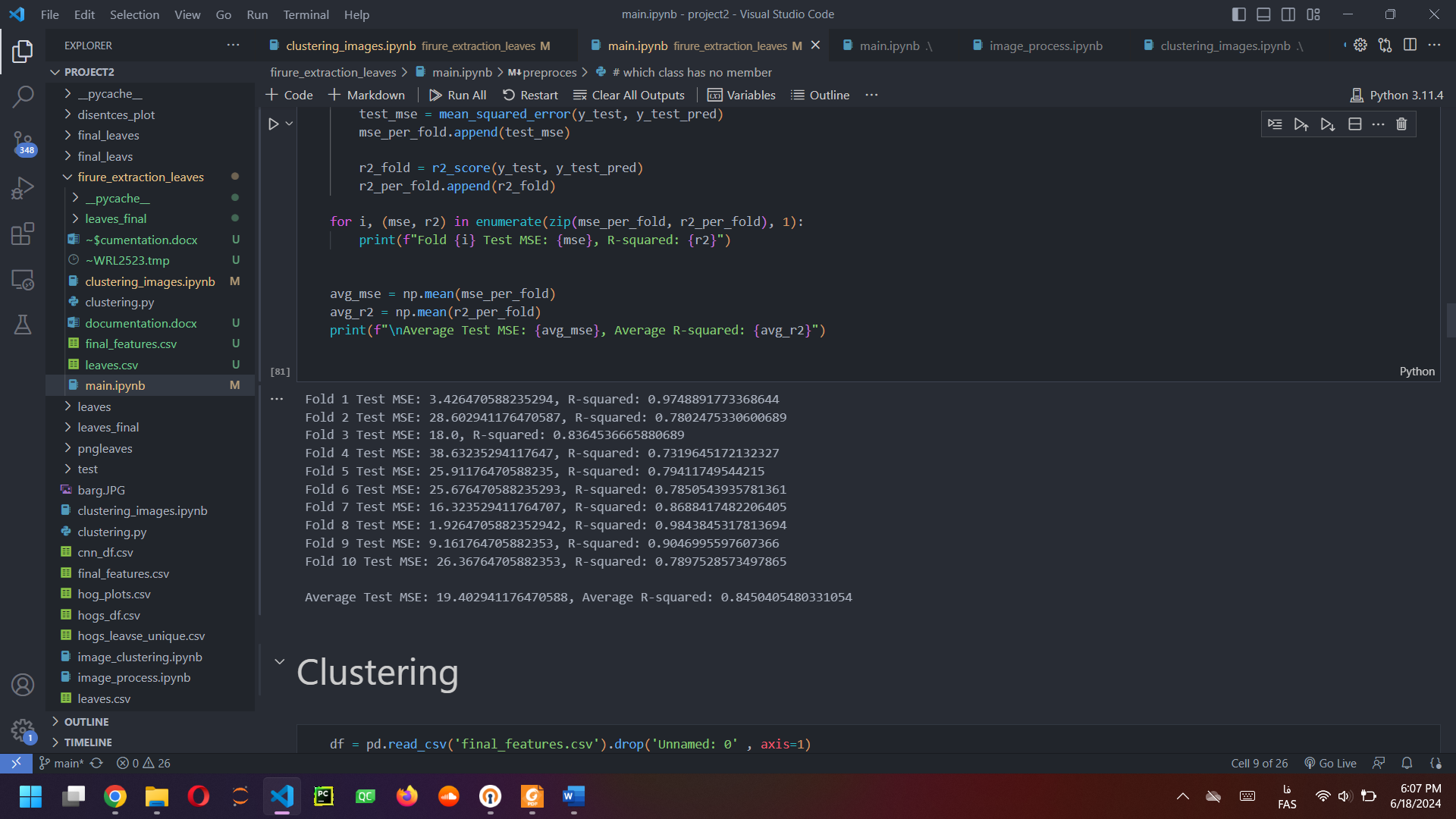


شکل 12

هماطور که در شکل ۱۲ مشاهده میشود از روش soft استفاده کردم.

## **ارزیابی مدل**

پس از انتخاب مدل به مرحله ارزیابی میرسیم که در این مرحله در یک حلقه ۱۰ تایی داده ها بدون random\_state ثابتی به دو بخش train , test تقسیم شده و توسط مدل predict میشوند و در هر دوره از حلقه دو مقدار r2\_score و MSE مدل اندازه گیری میشود و پس از ده بار تکرار میانگین این دو مقدار را گزارش میکند.



شکل 13

همانطور که در شکل ۱۳ مشاهده میشود میانگین R2\_score به ۸۵.۵ درصد رسیده است که مقدار قابل قبولی به نظر میرسد و با نگاه کردن به ۱۰ مقدار بالای آن میتوان دید که در بعضی مواقع مدل دقت خیلی بالایی نزدیک ۹۷ داشته و در بعضی مواقع تا ۷۳ هم رسیده است این اختلاف میتواند ناشی از تعداد بسیار کم سترهای داده و نسبت پایین تعداد ستر ها به تعداد کلاس ها باشد.

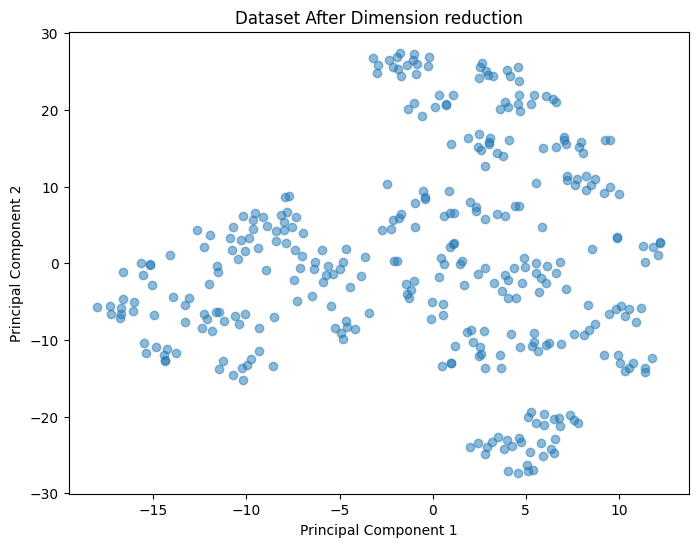
# **خوشه بندی برگ ها**

## **انتخاب ویژگی ها و نرمالایز کردن آنها**

برای انتخاب ویژگی ها از بین ۱۰۲۹ تنها ۲ ستون اول را که ربطی به ویژگی های داده ها نداشتند حذف کرده و پس از آن همه ستون های انتخابی را scale کرده تا مدل بهتر داده هارا خوشه بندی کند و تاثیر هر ویژگی به یک اندازه باشد.

## **کاهش بعد داده ها**

برای حذف ویژگی های اضافه و visualization بهتر و رسیدن به Silhouette بالاتر از کاهش بعد استفاده کردم که در بین الگوریتم های کاهش بعد الگوریتم TSNE(T-distributed Stochastic Neighbor Embedding) از بقیه الگوریتم ها بهتر عمل کرد و برای همین این الگوریتم را برای کاهش بعد داده ها استفاده کردم و تعداد ستون های داده هارا به ۲ ستون کاهش دادم.

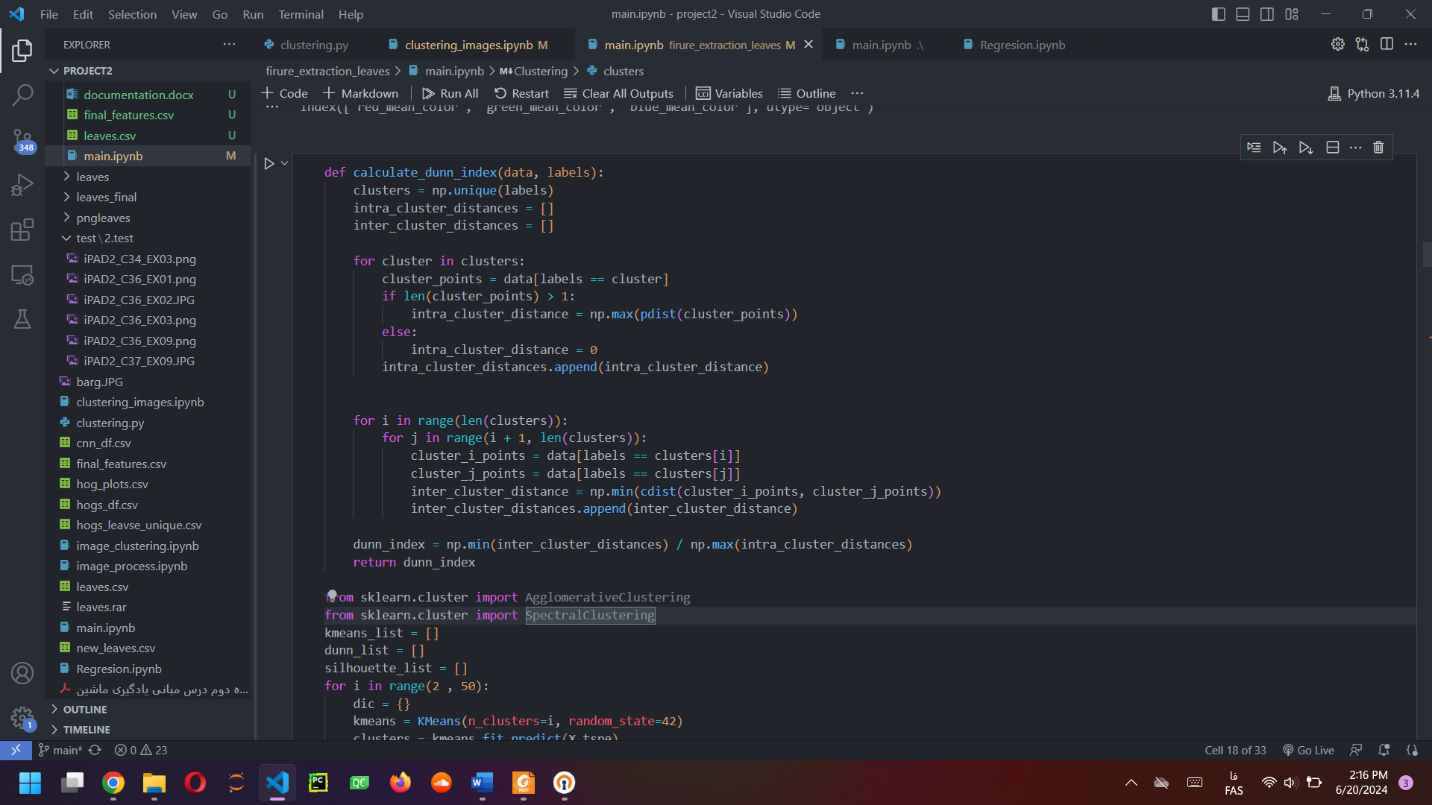


شکل 14

## **معیار های ارزیابی مدل**

برای ارزیابی خوشه بندی داده ها از دو الگوریتم معیار dunn و silhouette کمک گرفته تا با ارزیابی این دو مقدار بهترین تعداد خوشه ها و بهترین مدل خوشه بندی را پیدا کنیم.

برای محاسبه silhouette از تابع silhouette\_score در ماژول sklearn.metrics کمک گرفته و برای باتوجه به پیدا نکردن تابع آماده برای محاسبه dunn تابع calculate\_dunn\_index() را پیاده سازی کردم.



شکل 15

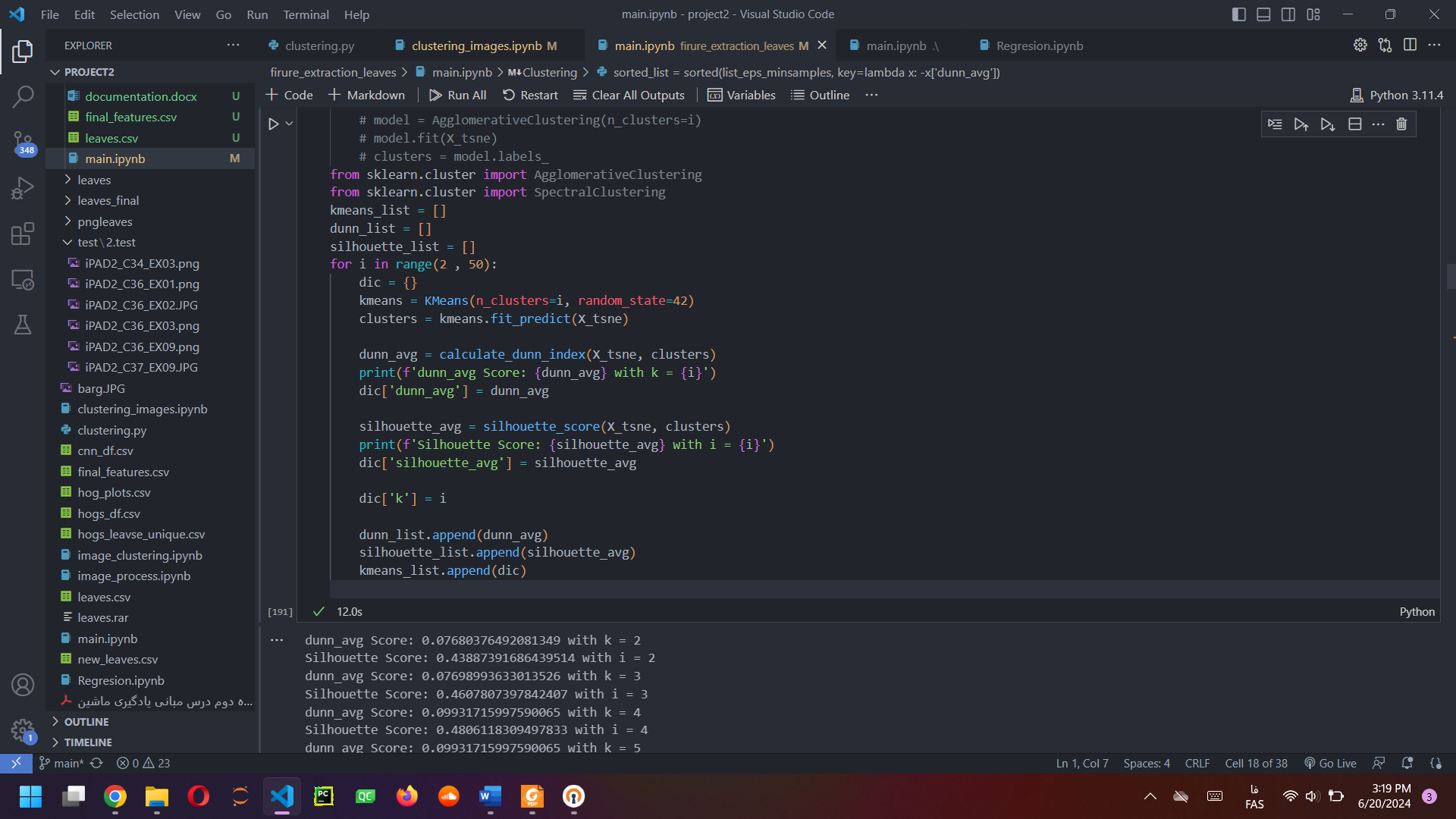
همانطور که در شکل ۱۵ مشاهده میشود این تابع از دو حلقه اصلی تشکیل شده است که جلقه اول برای محاسبه بیشترین قطر کلاستر و حلفه دوم برای محاسبه کمترین فاصله یک داده از یک کلاستر ب کلاستر دیگر است و در نهایت نسبت این دو را به عنوان خروجی بر میگرداند.

## **انتخاب مدل مناسب برای خوشه بندی**

از بین مدل های خوشه بندی مدل های K Means ، Spectral ، AgglomerativeClustering امتحان کرده و ارزیابی کردم. در ادامه برای هر مدل توضیحاتی را شرح میدهم.

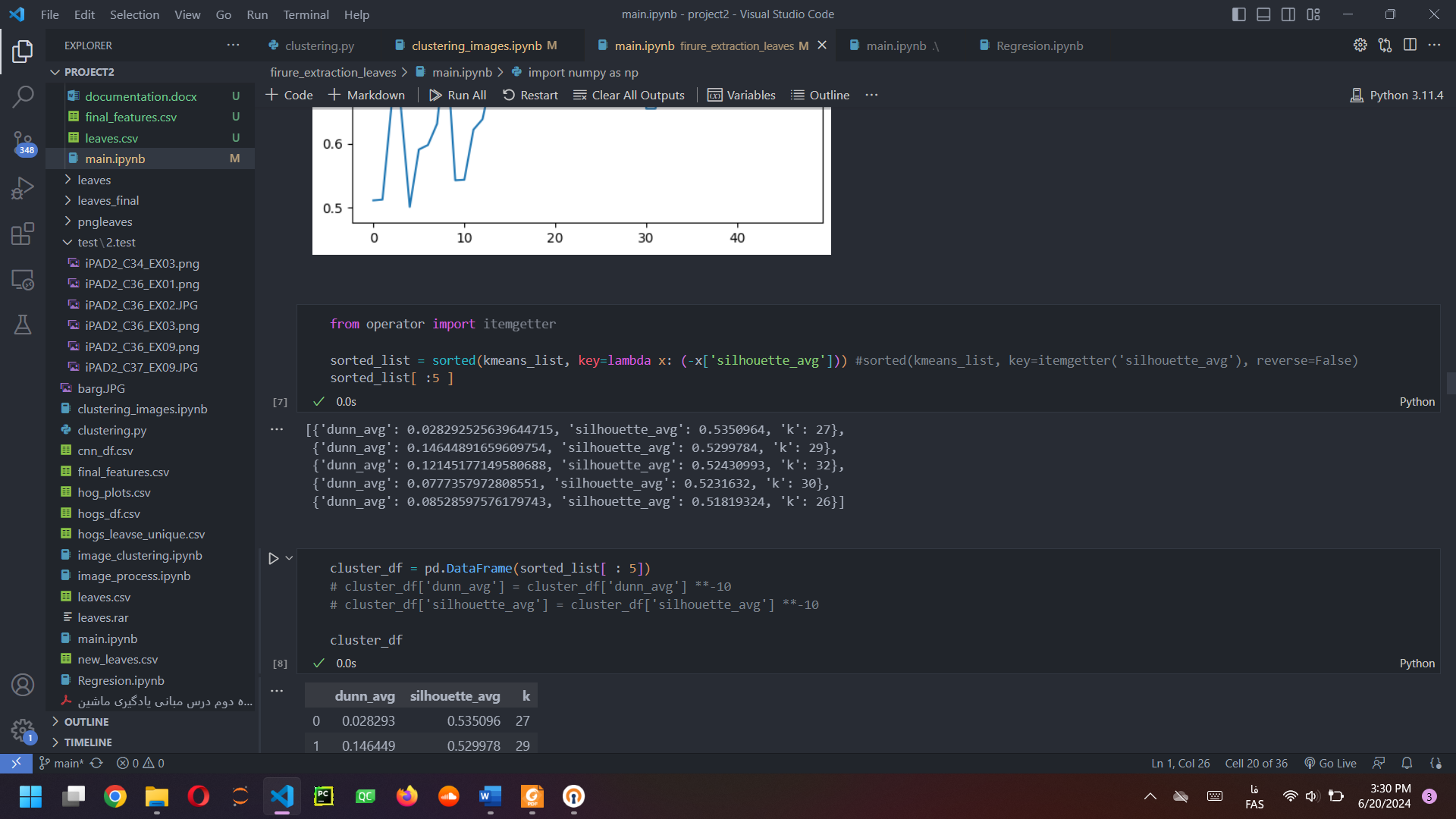
شرح سه مدل اول به این صورت است که هر کدام را در یک حلقه با K های مختلف از ۲ تا ۵۰ fit کرده و در نهایت بر اساس معیار silhouette\_score و dunn آنها را sort کرده و ۵ تا از بهترین K ها را پیدا میکنیم در نهایت بهترین مدل از بین این مدل ها را انتخاب میکنیم.

### **مدل K Means**



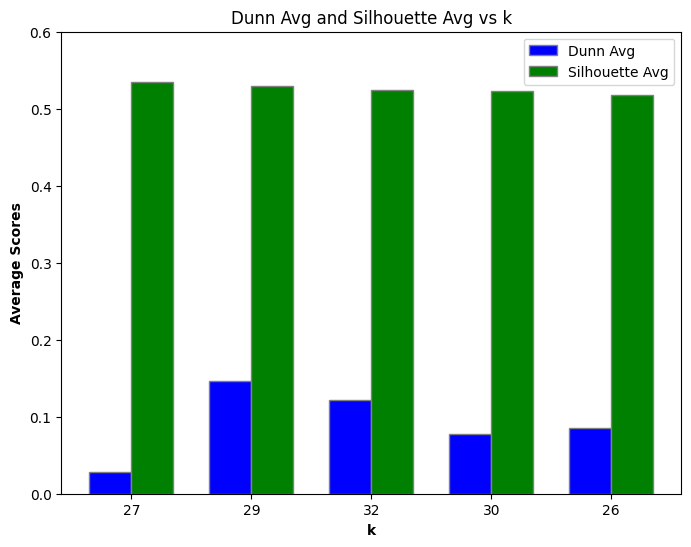
شکل 16

همانطور که در توضیح دادم و در شکل ۱۶ مشاهده میکنیم مدل K Means را در یک حلقه با K های مختلف از ۲ تا ۵۰ تست کرده و سه مقدار silhouette\_score ، dunn و K را در یک لیستی از دیکشنری ها ذخیره میکنیم.پس از اتمام این کد نوبت به sort کردن لیست دیکشنری ها بر اساس معیار silhouette میرسد تا با توجه به این معیار بهترین K را پیدا کنیم.



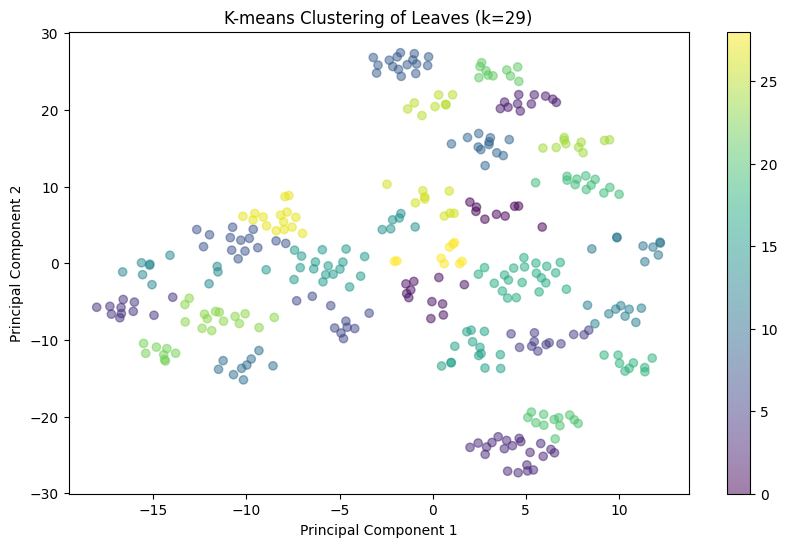
شکل 17

همانطور که در شکل ۱۷ مشاهده میشود بهترین silhouette را با K=27 مشاهده میکنیم ،‌ برای تصویر سازی بهتری این اعداد ، در شکل زیر داریم:



شکل 18

همانطور که در شکل ۱۸ مشاهده میشود نمودار های سبز برای معیار silhouette و نمودار های آبی برای معیار Dunn هستند و از چپ به راست بر اساس معیار silhouette سورت شده اند ، با توجه به این شکل میتوان دید که K=27 با اختلاف اندکی از چهار مقدار دیگر بهتر عمل کرده اما از نظر معیار Dunn عملکرد خوبی نداشته است بر همین اساس بنظر میرسد K=29 بهترین K برای این مدل است.

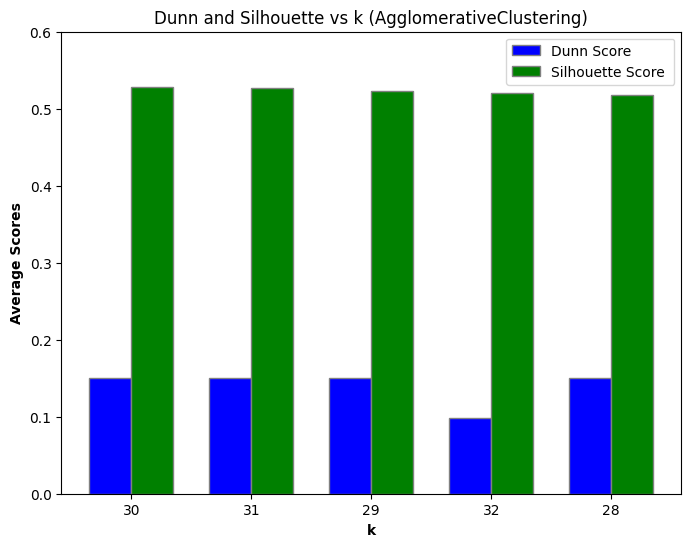
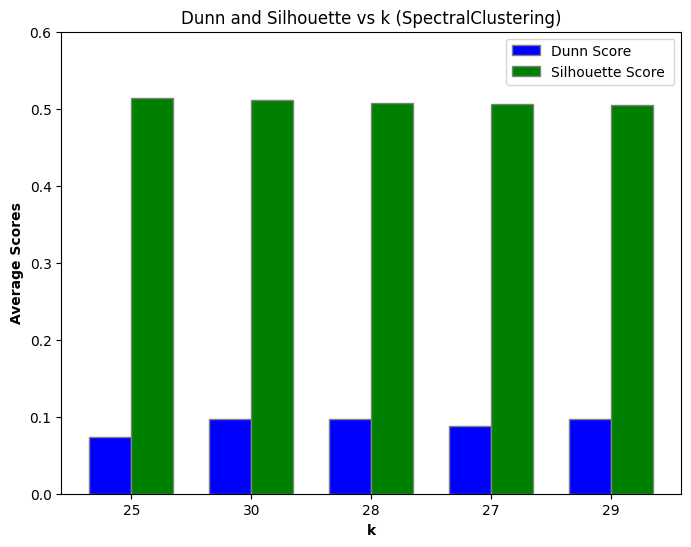


شکل 19

شکل ۱۹ نیز نمایی از مجموعه داده ها پس از خوشه بندی با الگوریتم K Means با K=29 را نشان میدهد که خوشه بندی خوبی به نظر میرسد.

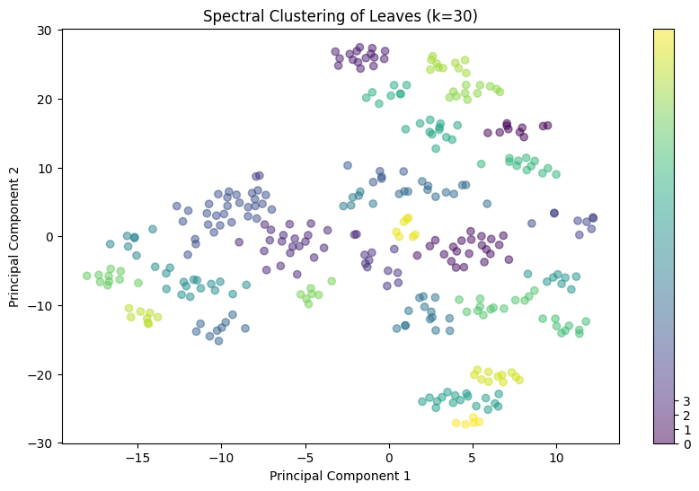
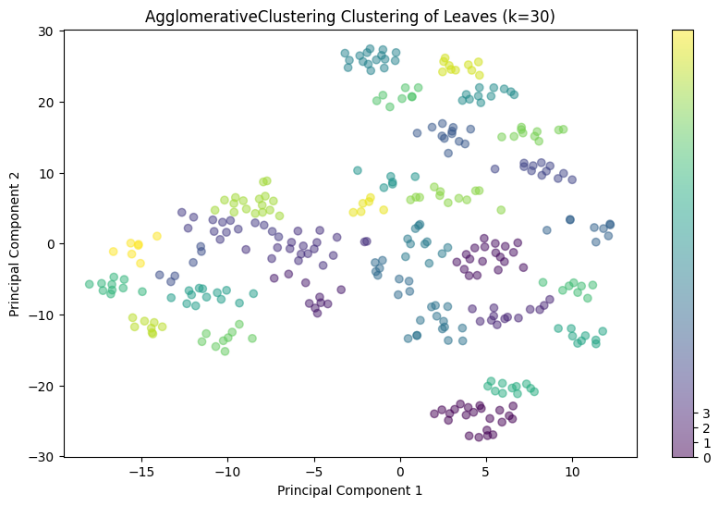
### **مدل های Spectral** **،** **AgglomerativeClustering**

با توجه به تکراری بودن مراحل در مدل های دیگر نیز ، برای جلوگیری از طولانی شدن توضیحات سه مدل دیگر را به صورت کوتاه توضیح میدهم.



شکل 20

همانطور که در شکل ۲۰ نیز مشاهده میشود برای مدل Spectral چهار مقدار ۳۰،۲۹،۲۸،۲۷ و برای مدل AgglomerativeClustering چهار مقدار ۳۰،۳۱،۲۹،۲۸ دارای مقادیر بهتری از دو معیار silhouette و Dunn هستند.

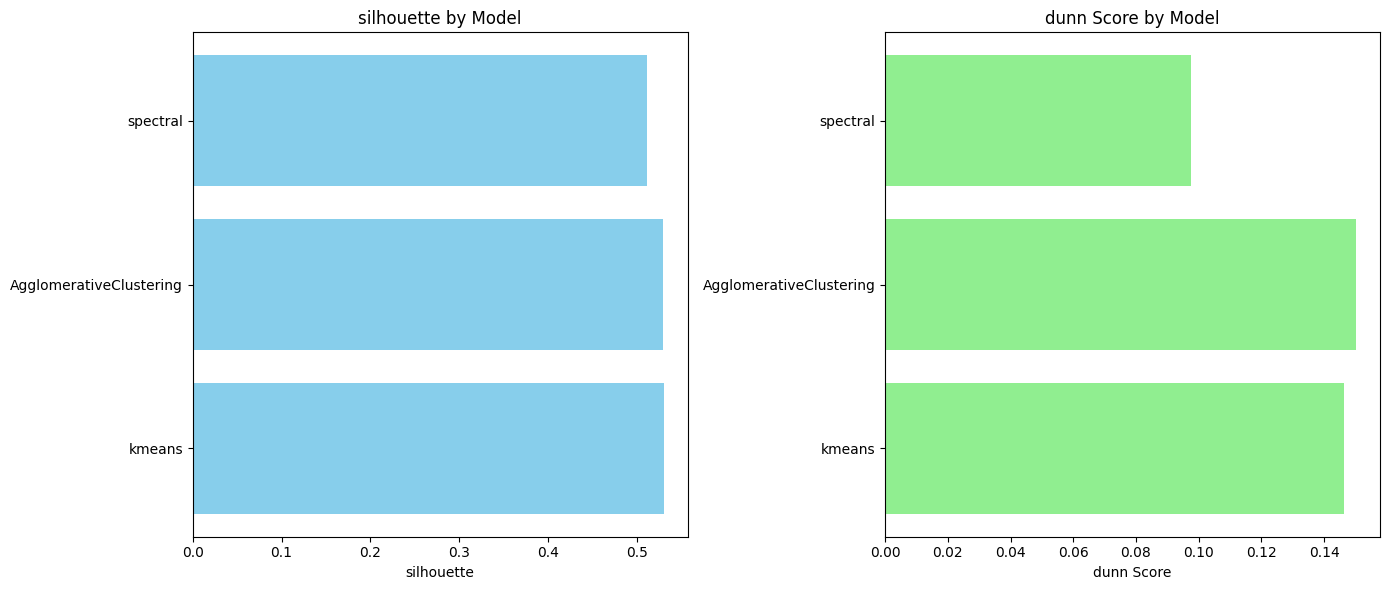


شکل 21

شکل ۲۱ نیز خوشه بندی با K=30 را با دو مدل ذکر شده نشان میدهد. در ادامه به مقایسه این سه مدل پرداخته و بهترین مدل را برای خوشه بندی انتخاب میکنیم.

## **مقایسه مدل ها و انتخاب بهترین مدل**

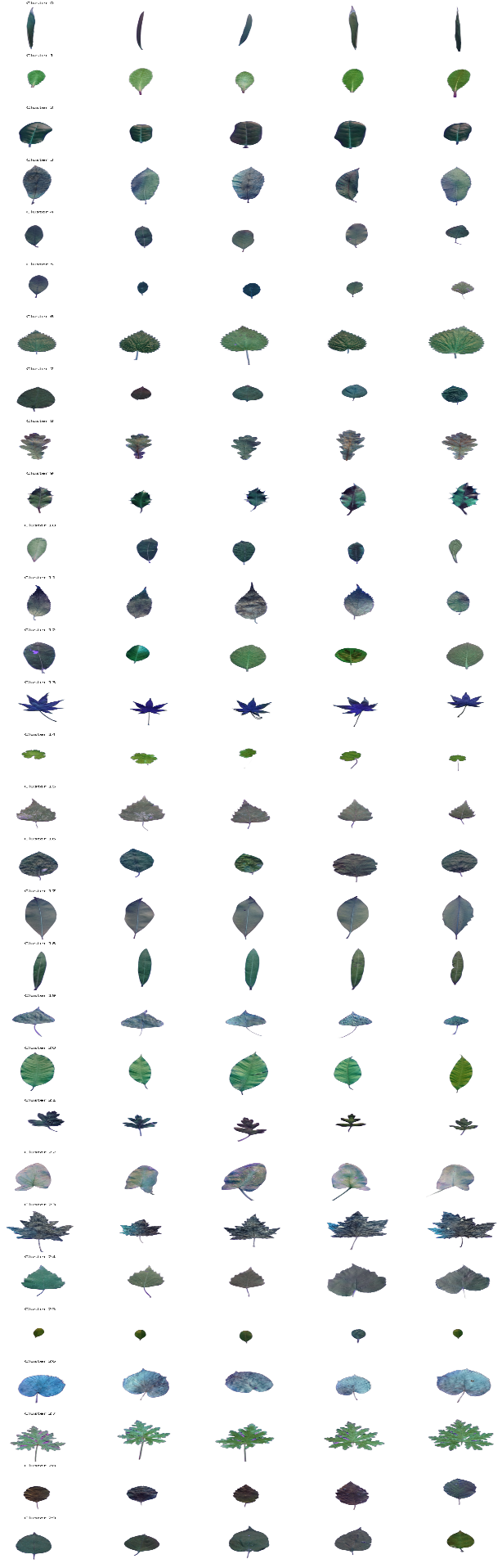
با توجه به عکس های ۲۰ و ۱۸ بهترین K ها برای هر مدل را دیدیم ، حال باید این مدل ها را با بهترین K خودشان ، با یکدیگر مقاسه کرده و مدل بهتر را برای خوشه بندی انتخاب کنیم.



شکل 22

با توجه به شکل ۲۲ میتوان به وضوح مشاهده کرد که دو مدل Kmeans و AgglomerativeClustering در معیار silhouette بهتر و تقریبا مساوی بوده و در معیار dunn نیز میبینیم که مدل AgglomerativeClustering از بقیه مدل ها بهتر بوده است ، با توجه به این دو نمودار میتوان گفت که مدل AgglomerativeClustering بهترین مدل در بین این سه مدل برای خوشه بندی است.

در ادامه با چاپ کردن ۵ عکس از هر خوشه میتوانیم بهتر و واضح تر کارایی این مدل را در خوشه بندی مشاهده کنیم.



شکل 23:چاپ ۵ عکس از هرخوشه

با توجه به شکل ۲۳ میتوان به خوبی نتیجه نهایی مدل خوشه بند را مشاهده کرد که مثال های هر خوشه از یک نوع برگ هستند.