گزارش فاز اول پروژه هوش محاسباتی

بخش اول(استخراج بخش‌های ارزشمند):

الف) استخراج ویژگی‌های رنگی و مکانی:

در این بخش در هر تصویر برای هر پیکسل پنج ویژگی رنگ، شدت رنگ، روشنایی و مختصات طول و عرض هر پیکسل با فاصله از مرکز استخراج شد.

```
height, width, _ = img.shape
img = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)

x, y = np.meshgrid(np.abs(np.arange(width) - np.full(width, width / 2)),
                   np.abs(np.arange(height) - np.full(height, height / 2)))
x = x.flatten()
y = y.flatten()


h = img[:, :, 0]
s = img[:, :, 1]
r = img[:, :, 2]
h_color = h.reshape((-1, 1))
s_color = s.reshape((-1, 1))
r_color = r.reshape((-1, 1))
```

ب) تغییر اهمیت ویژگی های رنگی و مکانی با استفاده از الگوریتم K-means:

با استفاده از K-meansبا قرار دادن پارامتر$k = 5$:

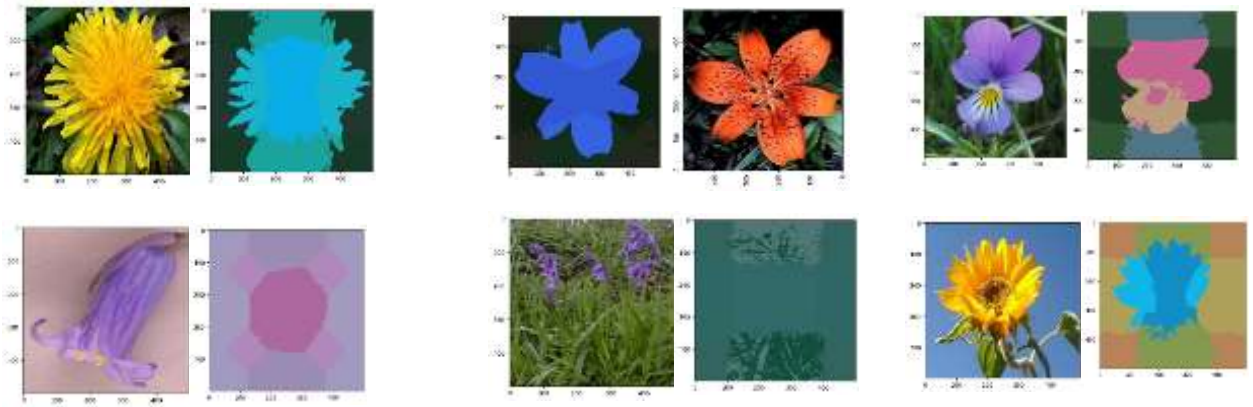
ویژگی‌های استخراج شده در فاز قبل را با نسبت‌های گوناگون روی تعدادی داده‌ی رندوم آزمایش و نتایج حاصل از بررسی در هر مرحله را به صورت کیفی مقایسه کردیم.

حالت 1)

```
Z = np.column_stack((h_color, s_color, r_color, z_coordinates, z_coordinates))
```

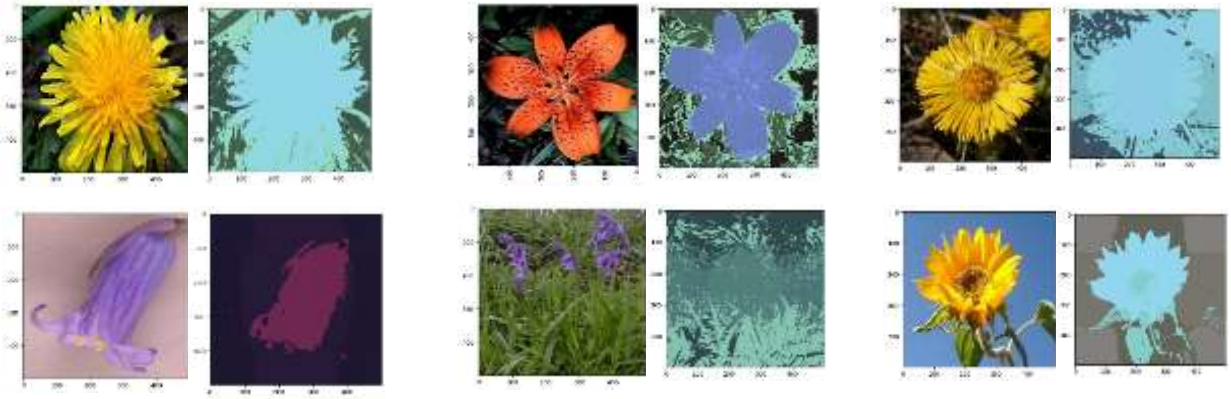
حالت2)

```
Z = np.column_stack((h_color, s_color, r_color, z_coordinates))
```



حالت3)

```
Z = np.column_stack((h_color, s_color, r_color, h_color, s_color, r_color, z_coordinates))
```



همان‌طور که مشاهده می‌شود با هر تغییر نسبت ویژگی‌ها خوشه بندی برای تعدادی تصویر بهتر و برای تعدادی تصویر بدتر می‌شود و در نهایت پس از چندین آزمایش با نسبت‌های متخلف بهترین نسبت ویژگی‌ها برای این خوشه بندی به این صورت می‌باشد:

```
Z = np.column_stack((2*h_color,  2*h_color, h_color,
                     1.5*s_color, 1.5*s_color,
                     1.5*r_color, r_color, z_coordinates))
```

ج) بررسی الگوریتم‌های مختلف و پارامترهای مختلف آن‌ها:

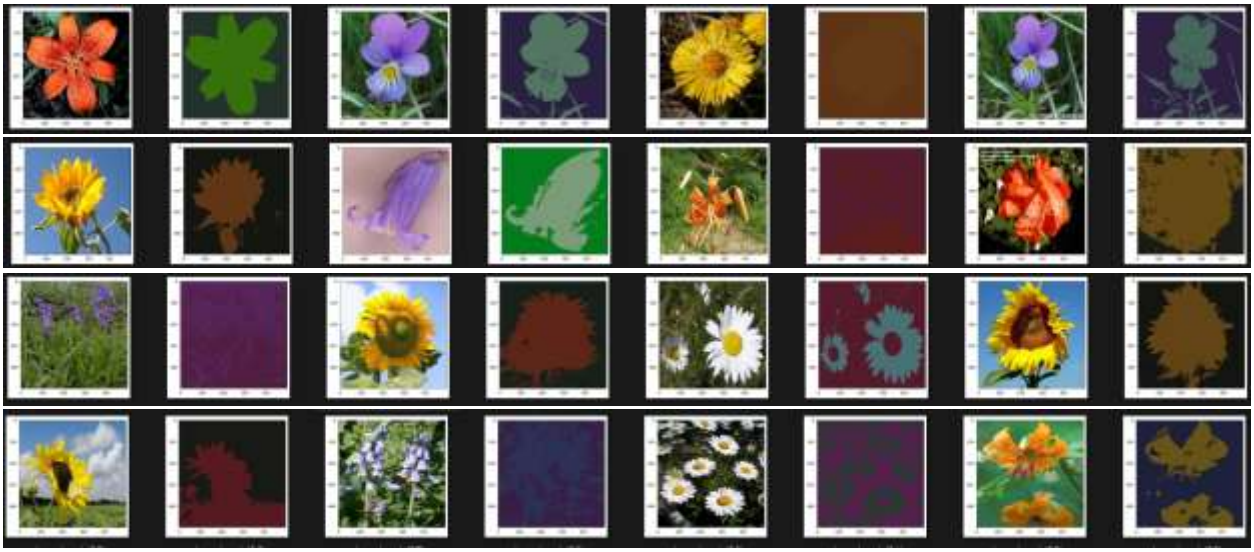
پس از به دست آوردن نسبت مناسب برای فیچرهای استخراج شده، به دنبال الگوریتم مناسب برای خوشه‌بندی این دیتاست می گردیم.

K-means

با مقادیر مختلف k به بررسی نحوه عملکرد این الگوریتم ، طبق معیارهای بیان‌شده در صورت پروژه می پردازیم.

دو نمونه از محدوده بالا و پایین مقادیر:

K = 2:



این مقدار برای تعدادی از تصاویر به خوبی عمل می‌کند و توانایی جداکردن کامل گل از پشت‌زمینه را دارد. اما برای بعضی تصاویر پشت‌زمینه با گل ترکیب می‌شود و توانایی تشخیص گل را نداریم. همچنین نواحی استخراج‌شده برای هر گل هم محدود به یک ناحیه است.

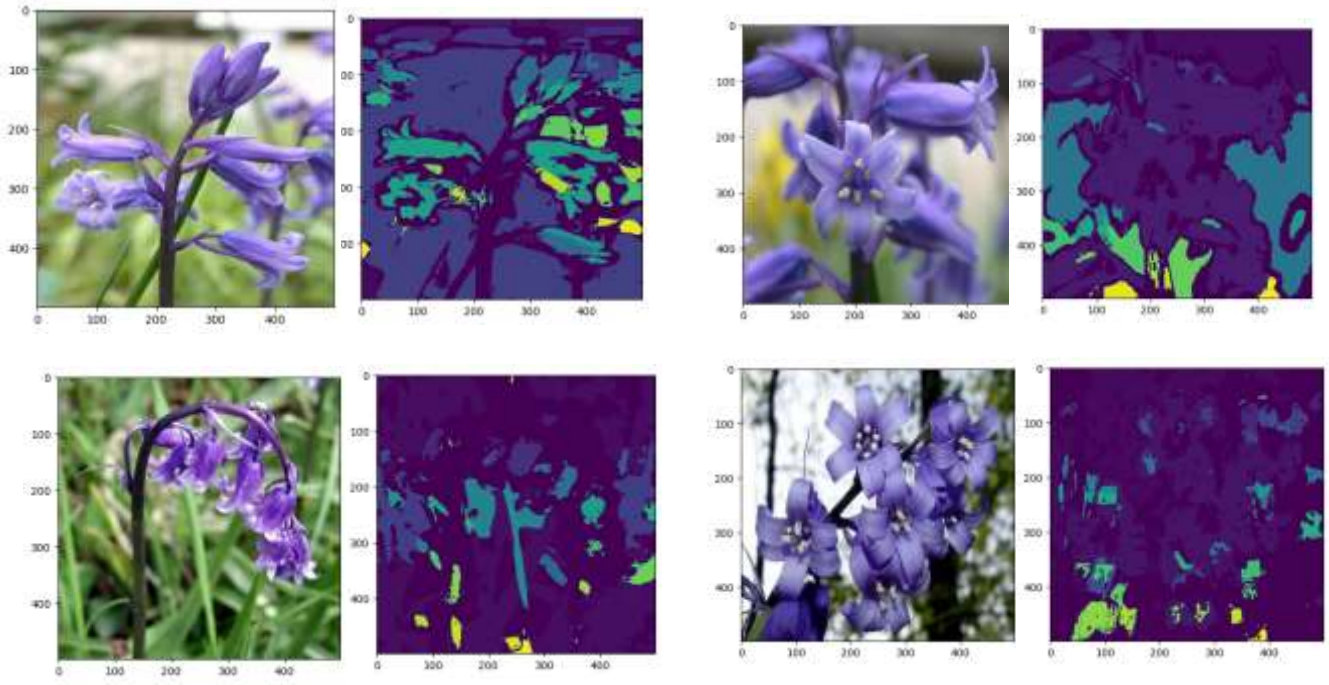
K = 10:

این مقدار نیز با وجود درنظر گرفتن نواحی بیشتری برای هر گل، نواحی بیشتری را به پشت‌زمینه اختصاص می‌دهد که مطلوب ما نیست. همچنین باعث ایجاد نواحی بسیار کوچک می‌شود و پراکندگی پیکسل‌های یک ناحیه زیاد می‌شود.

با تست مقادیر مختلف مقدار بهینه برای این پارامتر 5 می‌باشد که به صورت کلی برای تعداد بیشتری از تصاویر به خوبی خوشه‌بندی می‌کند.

DBSCAN

این الگوریتم با مقادیر متفاوت برای شعاع همسایگی و تعداد مینیمم همسایه‌ها تست کردیم. در نهایت بهترین خروجی خوشه‌بندی با پارامترهای شعاع همسایگی برابر با 60 و مینیمم همسایه‌ها برابر با 80 به دست آمد که تعدادی از خروجی‌های به این صورت هستند:



در نهایت با مقایسه بهترین خروجی‌های این دو الگوریتم با پارامترهای مشخص‌شده و با ویژگی‌های ثابت در هر دو حالت به این نتیجه رسیدیم که الگوریتم کی مینز به صورت کلی برای بیشتر تصاویر خروجی بهتری نسب به الگوریتمDBSCANدارد.

د) معیارهای کمی برای ارزیابی مدل:

واریانس رنگی هر خوشه: هر چه واریانس رنگی هر خوشه کمتر باشد، به این معنا است که در خوشه بندی رنگ‌های مشابه در یک خوشه قرار گرفته‌اند و نشان‌دهند خوشه بندی بهتری است.

Davies-Bouldin Index: اگر لیبل‌های داده‌ها شناخته نشده باشند، این شاخص را می‌توان برای ارزیابی مدل استفاده کرد که مقدار پایین‌تر مربوط به مدلی با جداسازی بهتر بین خوشه‌ها است. این شاخص میزان شباهت متوسط بین خوشه‌ها را اندازه‌گیری می‌کند، که شباهت به عنوان نسبت فاصله بین خوشه‌ها به اندازه خود خوشه‌ها تعریف می‌شود. فرمول آن به این صورت محاسبه می‌شود:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

که در آن

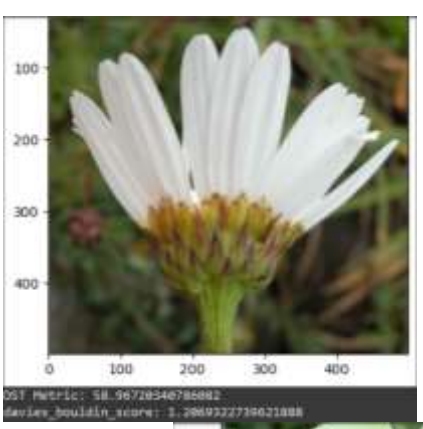
$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{ij}$$

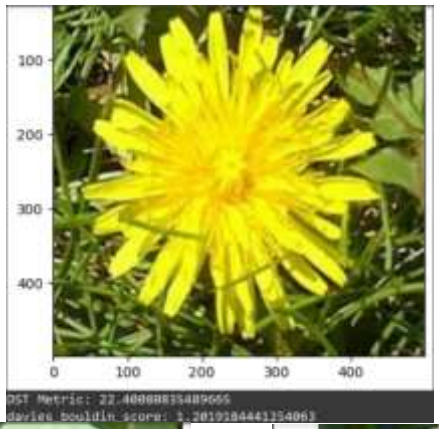
مزیت‌های این شاخص این است که به هر دو جداسازی و تراکم خوشه‌ها حساس است، محاسبه آن هزینه محاسباتی زیادی ندارد، و نسبتاً در برابر خروجی‌های غیرعادی مقاوم است.

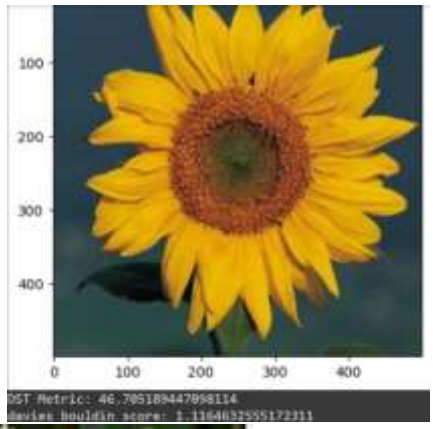
Calinski-Harabasz Index : یک معیار برای ارزیابی جداسازی و تراکم خوشه‌ها در یک الگوریتم خوشه‌بندی است. این شاخص با محاسبه نسبت واریانس بین خوشه‌ها به واریانس درون خوشه‌ای برای همه خوشه‌ها محاسبه می‌شود. این شاخص میزان تفاوت متوسط بین خوشه‌ها را اندازه‌گیری می‌کند، که تفاوت به عنوان نسبت واریانس بین خوشه‌ها به واریانس درون خوشه‌ای تعریف می‌شود. مقدار بالاتر آن نشان‌دهنده تفاوت بیشتر خوشه‌ها است، در حالی که مقدار پایین‌تر نشان‌دهنده شباهت بیشتر خوشه‌ها است.

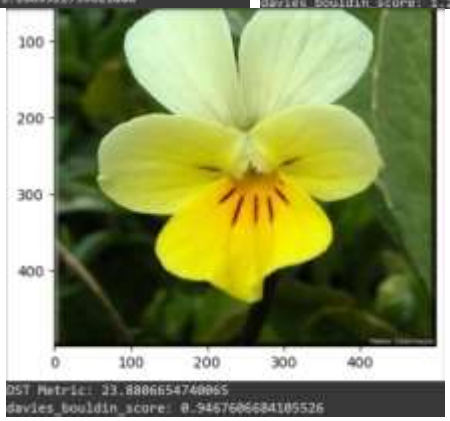
حال دو معیار واریانس رنگی هر خوشه وDavies-Bouldin Index را برای بهترین مدل خود یعنی الگوریتم کی‌مینز با$k=5$و نسبت ویژگی‌های ذکرشده را بر روی چند تصویر به صورت رندوم اعمال می‌کنیم که خروجی به این صورت می‌باشد:

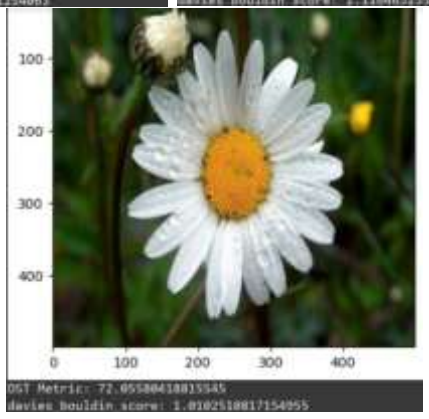
```python
# Extract cluster colors
res = center[label.flatten()]
res2 = res.reshape((height, width, 9))[:, :, :3]

# Calculate DST metric for clustering performance
labels = label.reshape(res2.shape[:2])
STD = 0
for cluster_id in range(K):
  h = (img[:, :, 0][labels == cluster_id])
  STD += np.std(h)
print("DST Metric:", STD)

 # Calculate Davies-Bouldin Index
print("davies_bouldin_score:", metrics.davies_bouldin_score(Z, label))
```











بخش دوم(استخراج ویژگی از نواحی):

الف) ویژگی‌های رنگی و آماری هر ناحیه:

پس از استخراج نواحی از هر تصویر برای هر ناحیه ویژگی‌های آماری رنگی از جمله میانگین، واریانس، کشیدگی و چولگی (درجه تقارن در توزیع مقادیر رنگ) برای هر کدام از کانال‌هایh, S, Vمحاسبه شد.

```
for cluster_id in range(K):
# Extract features for the current cluster
    h = (img[:, :, 0][label == cluster_id])/ 180
    s = (img[:, :, 1][label == cluster_id]) / 255
    r = (img[:, :, 2][label == cluster_id]) / 255
    filtered_label = label[label == cluster_id]
    # Check if the cluster is empty
    if len(h) > 1:
        # Calculate the color mean and std for h, s, and r
        center_cluster = center[filtered_label]
        cluster_mean_h = np.mean(h)
        cluster_std_h = np.std(h)
        cluster_mean_s = np.mean(s)
        cluster_std_s = np.std(s)
        cluster_mean_r = np.mean(r)
        cluster_std_r = np.std(r)
        # Check if the mean and standard deviation are valid (not NaN)
        if not np.isnan(cluster_mean_h) and not np.isnan(cluster_std_h):
            # Calculate cluster skewness and kurtosis and spatial features
            cluster_skew_h, cluster_kurt_h = cv2.meanStdDev(h)
            cluster_skew_s, cluster_kurt_s = cv2.meanStdDev(s)
            cluster_skew_r, cluster_kurt_r = cv2.meanStdDev(r)
```

همچنین ویژگی‌های مکانی از جمله تعداد کل پیکسل‌های موجود در خوشه، طول، عرض، ارتفاع و مرز کوچکترین مستطیلی که خوشه را احاطه می‌کند، و زاویه محور اصلی خوشه نسبت به محور افقی نیز برای هر ناحیه تصاویر استخراج شدند و تمام ویژگی‌های رنگی و آماری هر ناحیه در یک بردار ذخیره شدند.

```
            pixel = len(h)
            bounding_box = cv2.boundingRect(center_cluster)
            bounding_box_width = bounding_box[2]
            bounding_box_height = bounding_box[3]
            area = bounding_box_width * bounding_box_height
            perimeter = 2 * (bounding_box_width + bounding_box_height)
            moments = cv2.moments(center_cluster)
            mu20 = moments['m20']
            mu02 = moments['m02']
            orientation = np.arctan2(moments['mu11'], moments['mu20'])


            x = [cluster_mean_h, cluster_std_h, cluster_skew_h[0][0], cluster_kurt_h[0][0],
                     cluster_mean_s, cluster_std_s, cluster_skew_s[0][0], cluster_kurt_s[0][0],
                     cluster_mean_r, cluster_std_r, cluster_skew_r[0][0], cluster_kurt_r[0][0],
                     pixel, bounding_box_width, bounding_box_height, area, perimeter, orientation]
            region_vectors.append(x)
            cluster_features.append(x)
        else:
            cluster_skew_h = np.nan
            cluster_kurt_h = np.nan
image_features.append(cluster_features)
```

ب) ساخت بردار ویژگی‌های تصاویر:

حال تمامی بردارهای ساخته‌شده برای هرناحیه از هر تصویر را به الگوریتم خوشه بندی k-means می‌دهیم.

در نهایت به ازای هر تصویر یک بردار به اندازه تعداد خوشه های موردنظر خواهیم داشت که فراوانی بردارهای ناحیه‌های آن تصویر در در خوشه‌ها محاسبه خواهد شد و به ازای هر تصویر هیستوگرام ویژگی‌های موجود را خواهیم داشت.

مقدار پارامتر k نیز پس از تعیین عملکرد خوب نسبت ویژگی‌های هر ناحیه تعیین می‌شود.

```
# Cluster all the region vectors
kmeans = KMeans(n_clusters=44)
kmeans.fit(region_vectors)
cluster_labels = kmeans.labels_

# Find the cluster of each vector
cluster_members = []
for i in range(kmeans.n_clusters):
    cluster_members.append(region_vectors[cluster_labels == i])

# Construct a histogram of the features for each image
image_vectors = []
for i, image in enumerate(image_features):
    image_vectors.append([0] * kmeans.n_clusters)
    for im_vector in image:
      for j, cluster in enumerate(cluster_members):
        for cl_vector in cluster:
          if np.all(im_vector == cl_vector):
            image_vectors[i][j] += 1
```

سپس این بردارها به همراه لیبل‌های هر تصویر را به الگوریتم طبقه‌بندی داده‌شده می‌دهیم.

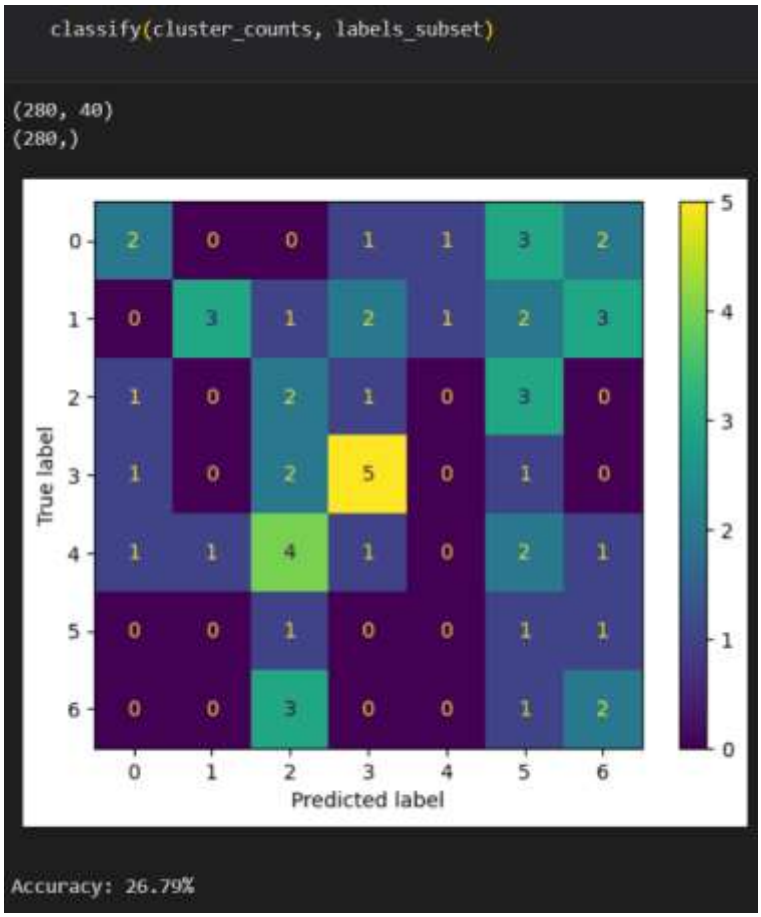
ج) تغییر ویژگی‌های وابسته به شکل ناحیه نسبت به رنگ:

به علت زمان‌بر بودن کل فرآیند برای همه تصاویر، 280تا داده به صورت رندوم از دیتاست انتخاب می‌شوند.

```
random_indices = np.random.choice(num_images, 280, replace=False)

# Convert random_indices to a list before indexing
random_indices = list(random_indices)
images_subset = data[random_indices]
labels_subset = []

# Iterate over the random indices and extract the corresponding labels
for index in random_indices:
    labels_subset.append(labels[index])
print(len(images_subset))
```

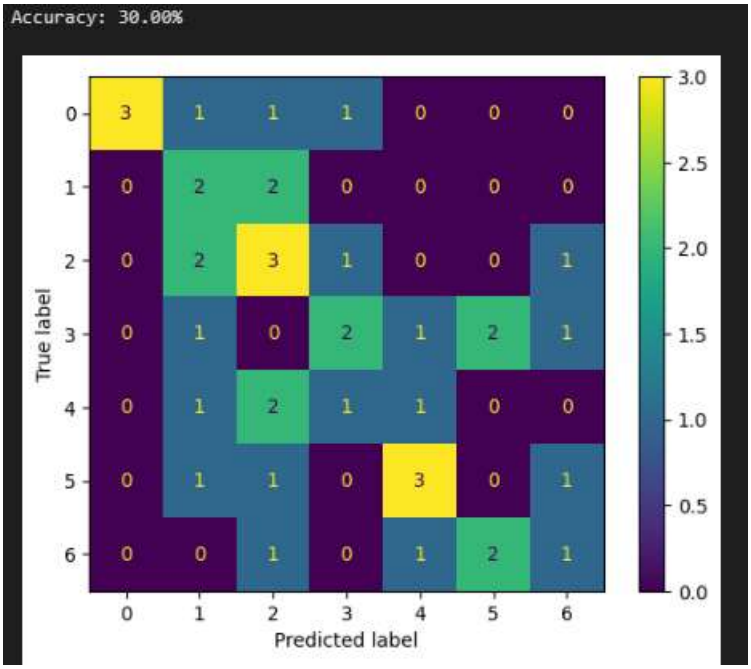
ابتدا تمام ویژگی‌های استخراج‌شده را به نسبت مساوی قرار دادیم و با دادن بردارهای هر تصویر به الگوریتم طبقه‌بندی، بهترین دقت به ازای kمختلف تست‌شده به این صورت بود:

که با توجه به ماتریس گمراهی می‌توان گفت فقط یک کلاس درست تشخیص داده می‌شود.



سپس ضریب تکرار ویژگی‌های رنگی مربوط به کانال h را به این صورت افزایش دادیم و خروجی به ازای kهای متفاوت تست شد و بهترین عمکلرد به این صورت بود:

```
x = [cluster_mean_h, cluster_mean_h,
                     cluster_std_h, cluster_std_h,
                     cluster_skew_h[0][0], cluster_kurt_h[0][0], cluster_skew_h[0][0], cluster_kurt_h[0][0],
                     cluster_mean_s, cluster_std_s,
                     cluster_mean_r, cluster_std_r,
    pixel, bounding_box_width, bounding_box_height, area, perimeter, orientation]
```

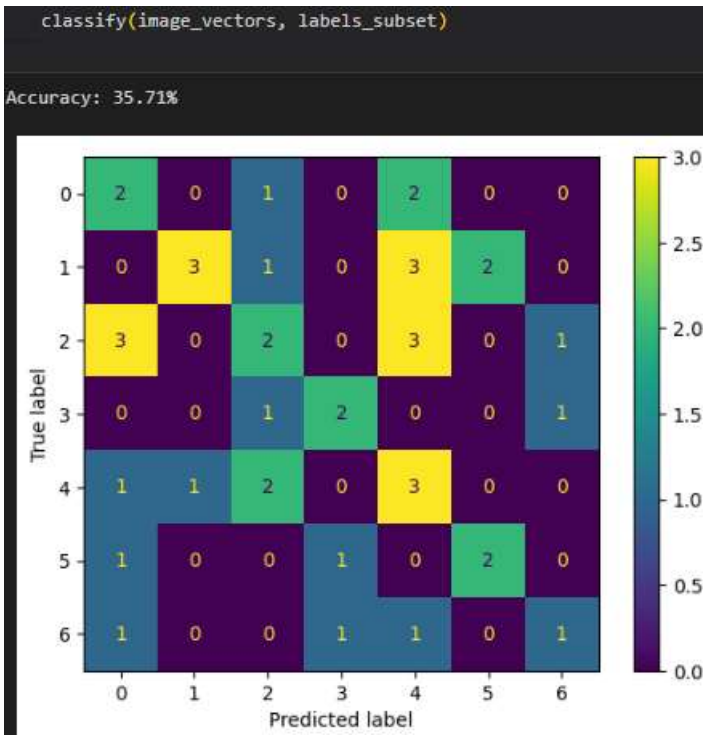


با افزایش مجدد تعداد تکرار ویژگی‌های آماری مربوط به رنگ برای هر 3 کانال و تست مجدد با مقادیر مختلف

K باز هم بهبود چندانی صورت نگرفت. یک نمونه از نسبت‌های متفاوت تست شده به این صورت می‌باشد:

```
x = [cluster_mean_h, cluster_mean_h, cluster_mean_h, cluster_mean_h, cluster_mean_h,
                cluster_std_h, cluster_std_h, cluster_std_h, cluster_std_h,
                cluster_skew_h[0][0], cluster_kurt_h[0][0], cluster_skew_h[0][0], cluster_kurt_h[0][0],
                cluster_mean_s, cluster_std_s, cluster_mean_s, cluster_std_s,
                cluster_skew_s[0][0], cluster_kurt_s[0][0], cluster_skew_r[0][0], cluster_kurt_r[0][0],
                 cluster_mean_r, cluster_std_r,
     pixel, bounding_box_width, bounding_box_height, area, perimeter]
```

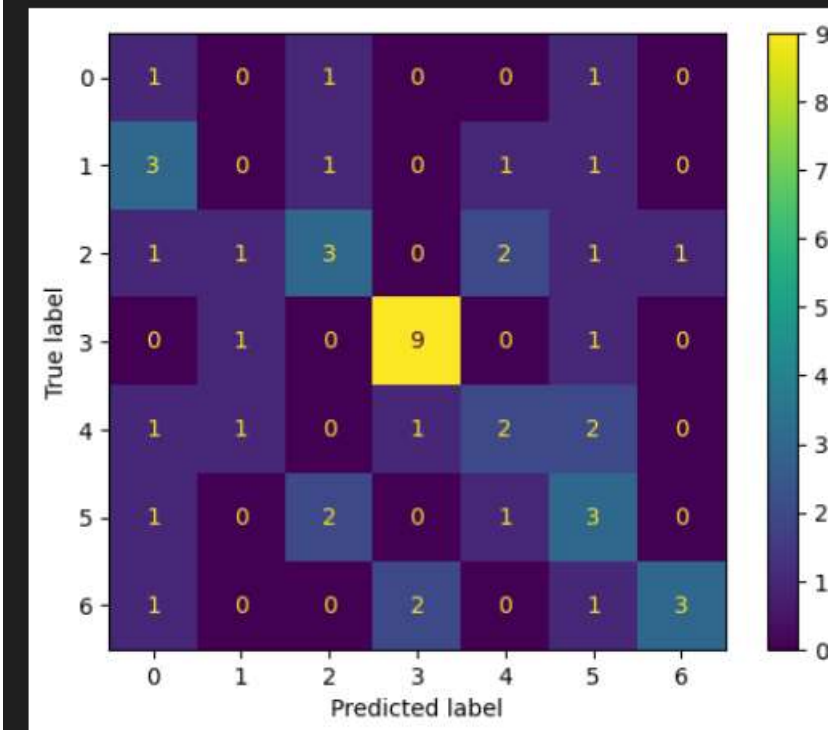
k = 55



سپس یک حالت دیگری که انجام شد این بود که فقط واریانس و میانگین کانالh به عنوان ویژگی استخراج‌شده مورداستفاده قرار گرفتند و باkهای متفاوت تست شدند و بهترین دقت به این صورت بود.

```
x = [cluster_mean_h, cluster_std_h]
region_vectors.append(x)
classify(cluster_counts, labels_subset)
```

```
(250, 33)
(250,)
Accuracy: 42.00%
```



سپس با اضافه کردن ویژگی‌های آماری رنگی دیگر از جمله کشیدگی و چولگی به این نتیجه رسیدیم که ویژگی‌های مکانی استخراج‌شده احتمالا باعث پایین آمدن دقت الگوریتم طبقه‌بندی می‌شدند و درنهایت تغییر نسبت‌های ویژگی‌های رنگی، بهترین دقت مشاهده‌شده، مقدار 73.81 بود که نسبت ویژگی‌های آن به این صورت می‌باشد:

```
x = [cluster_mean_h, cluster_mean_h, cluster_mean_h,
                     cluster_std_h, cluster_std_h,
                     cluster_skew_h[0][0], cluster_kurt_h[0][0],
                     cluster_mean_s, cluster_std_s,
                     cluster_mean_r, cluster_std_r]
```

k= 70

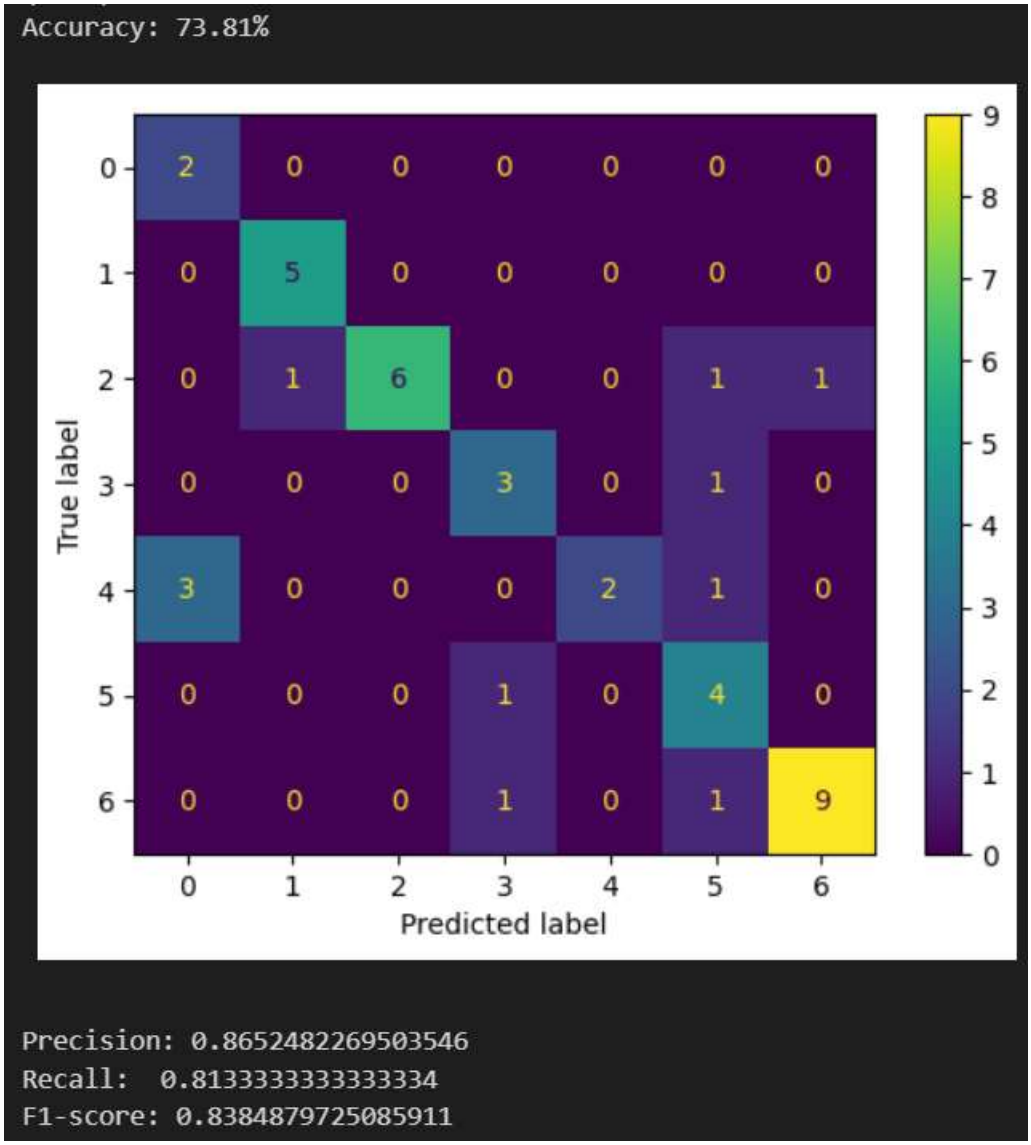
د) محاسبه معیارهای دقت و ترسیم ماتریس گمراهی برای بهترین مدل:

پس از دستیابی به بهترین مدل، معیارهای recall, precision, f1score طبق تابع زیر با دریافت مقدار واقعی لیبل و مقدار محاسبه‌شده توسط الگوریتم طبقه‌بندی برای داده‌های تست محاسبه شدند و ماتریس گمراهی آن نیز رسم شد:

```python
def accuracy_metrics(y_true, y_pred):
    tp = np.sum(y_true & y_pred)
    fp = np.sum(y_pred & ~np.array(y_true))
    precision = tp / (tp + fp)
    tp = np.sum(y_true & y_pred)
    fn = np.sum(y_true & ~y_pred)
    recall = tp / (tp + fn)
    f1 = 2 * precision * recall / (precision + recall)
    return precision, recall, f1
```

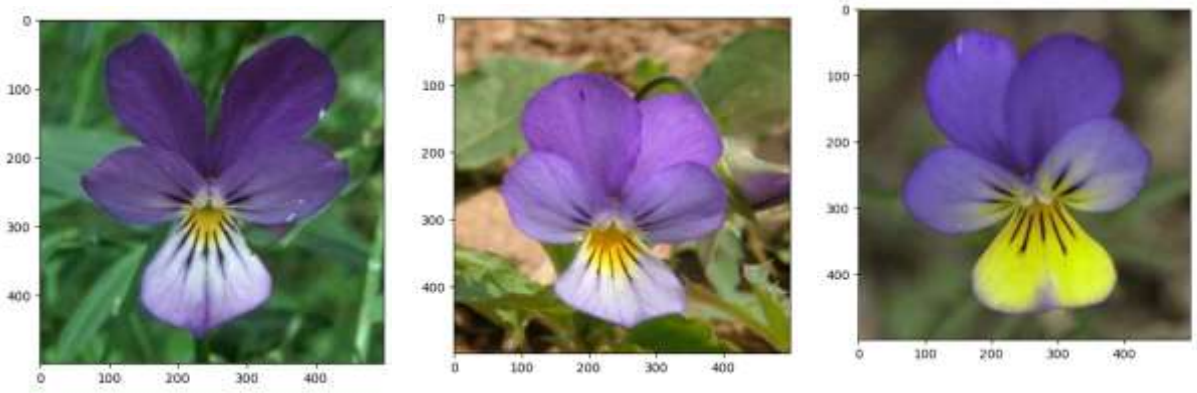
```python
def classify(datapoints, labels):
  test_size = 0.2
  X_train, X_test, y_train, y_test = train_test_split(datapoints, labels, test_size=test_size, random_state=42)

  clf = RandomForestClassifier(n_estimators=100, random_state=42)

  clf.fit(X_train, y_train)

  y_pred = clf.predict(X_test)

  accuracy = accuracy_score(y_test, y_pred)
  print(f"Accuracy: {accuracy * 100:.2f}%")
  cm = confusion_matrix(y_test, y_pred, labels=clf.classes_)
  disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=clf.classes_)
  disp.plot()
  plt.show()
  precision, recall, f1_score = accuracy_metrics(y_test, y_pred)
  print(f"Precision:", precision)
  print(f"Recall: ", recall)
  print(f"F1-score:", f1_score)
```

خروجی معیارهای دقت و ماتریس گمراهی برای بهترین مدل به این صورت است:



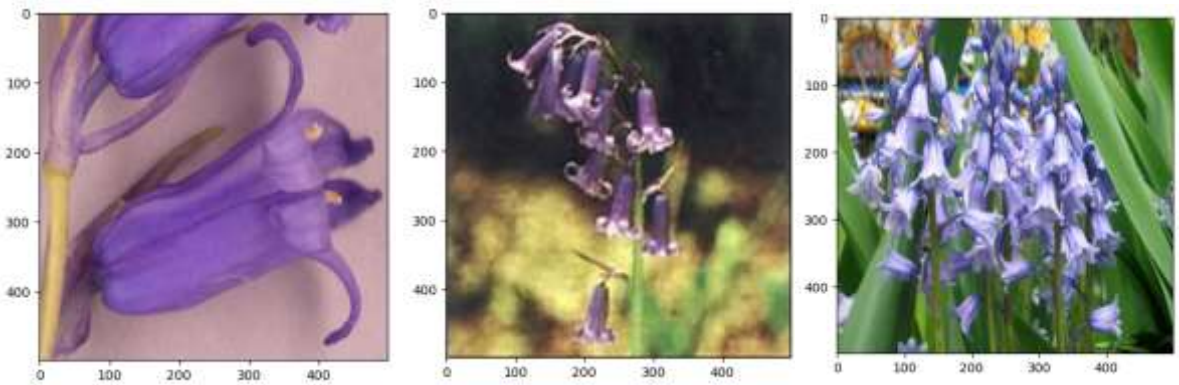
بیشترین اشتباهات رخ‌داده در دو خانه‌ای از ماتریس می‌باشد که لیبل‌های واقعی برابر 4 و لیبل تشخیص داده‌شده برابر 0 هست. حال به کمک کد زیر این گل‌های اشتباه تشخیص داده‌شده که تعدادشان برابر 3 هست را پلات می‌کنیم:

```
mistaken = []

for i in range(len(yt)):
  if(yt[i] == 4 and yp[i] == 0):
    mistaken.append(x[i])

for i  in range(len(mistaken)):
  for j, vector in enumerate(image_vectors):
    if mistaken[i] == vector:
      print("image:", labels_subset[j])
      img = images_subset[j]
      img1 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
      plt.imshow(img1)
      plt.imshow(img1)
      plt.show()
```



3 گل اشتباه تشخیص‌داده شده با لیبل کلاس 4

حال برای بررسی علت این تشخیص اشتباه تعدادی از گل‌های کلاس 0 را نیز پلات می‌کنیم.



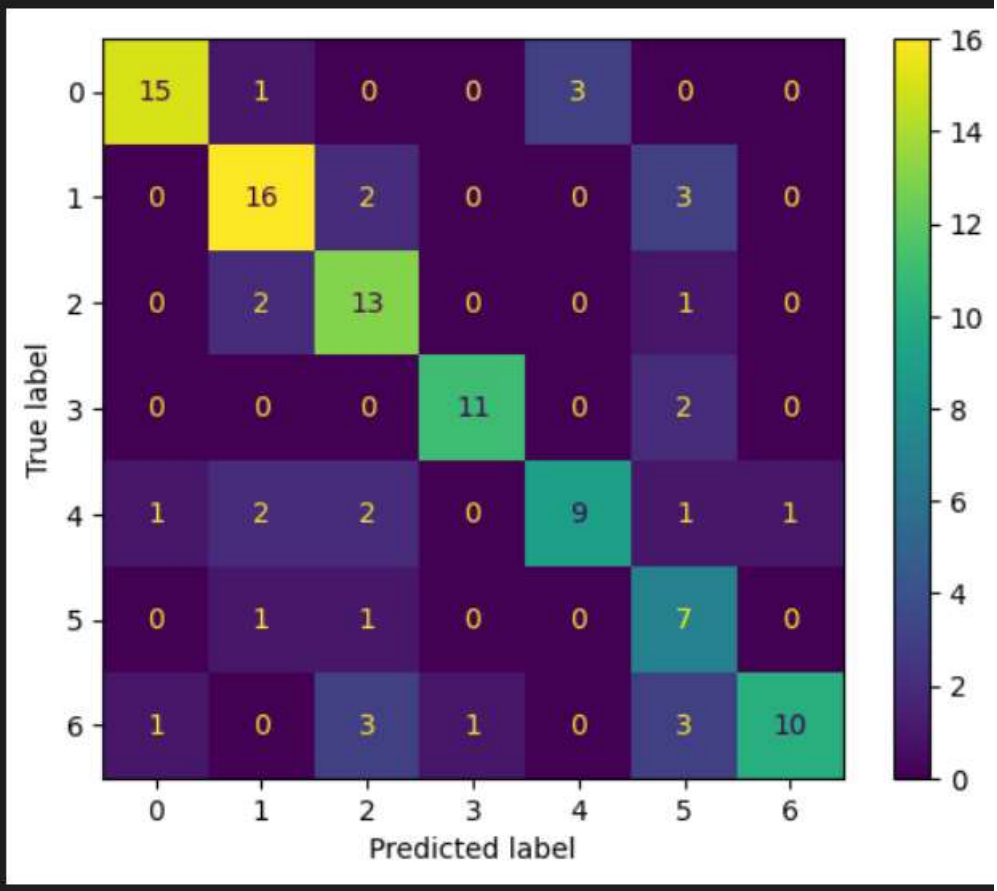
تعدادی از گل‌ها با لیبل کلاس 0

همان‌طور که مشاهده می‌شود بیشتر این اشتباهات به این علت می‌باشد که رنگ گل‌های اشتباه تشخیص داده‌شده همه بنفش هستند و با توجه به اینکه در بهترین مدل فقط ویژگی‌های استخراج‌شده از رنگ را داریم و ویژگی‌های مکانی استخراج‌شده در مراحل قبل نیز باعث دقت پایین مدل می‌شدند باید به دنبال ویژگی‌های مکانی بهتری بگردیم که با توجه به تفاوت در شکل گلبرگ‌های2 کلاس بتواند بهتر میان این کلاس‌ها تمایز قائل شود. همچنین می‌توان با توجه به اینکه شدت رنگ در بیشتر گل‌های کلاس در دو کلاس با هم متفاوت است، اگر ضریب ویژگی‌های آماری رنگی مربوط به کانال saturation که مربوط به شدت رنگ می‌باشد را افزایش دهیم، مدل بتواند تمایز بهتری میان این دو کلاس قائل شود.

راهکار دیگر برای رفع مشکل این است که از آن‌جا که تفاوت رنگی میان گل‌های لیبل 4 وجود دارد با افزایش تعداد داده‌ها مدل نمونه‌های بیشتری از رنگ‌های بنفش، سفید و زرد دریافت کند و بهتر این کلاس را تشخیص دهد.

درنهایت بهترین مدل بر روی کل دیتاست نیز آزمایش شد که نتیجه آن به این صورت می‌باشد:

```
Accuracy: 72.32%
```




```
Precision: 0.802675585284281
Recall:  0.7766990291262136
F1-score: 0.7894736842105263
```