Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Semester 2 2021/2022

Implementasi Convex Hull untuk Visualisasi Tes Linear Separability Dataset dengan Algoritma Divide and Conquer

Hafidz Nur Rahman Ghozali

13520117

Sekolah Teknik Elektro dan Informatika

Teknik Informatika

2022

1. Algoritma *Divide and Conquer*

   Algoritma *Divide and Conquer* adalah algoritma untuk menyelesaikan suatu permasalahan dengan membaginya menjadi upa-persoalan yang memiliki karakteristik yang sama dengan persoalan semula *(Divide)*. Upa-persoalan kemudian diselesaikan secara sendiri-sendiri *(Conquer)* atau dipecah kembali menjadi upa-persoalan yang lebih kecil. Upa-persoalan yang telah diselesaikan kemudian digabungkan kembali sehinggan menghasilkan solusi permasalahan yang utuh *(Combine)*.

   Penulis menggunakan algoritma *Divide and Conquer* dalam implementasi *Convex Hull* pada program yang dibuat. Berikut adalah gambaran penyelesaian *Convex Hull*:

   a. Pengurutan data

      Sebelum melakukan *Divide and Conquer*, program akan mengurutkan data titik-titik secara menaik berdasarkan koordinat x dan y. Di dalam program ini, penulis menggunakan algoritma Quicksort dengan kompleksitas rata-rata sebesar $O(n \log n)$.

   b. Pembagian daerah persoalan

      Langkah pertama dalam persoalan *convex hull* adalah mencari titik terkiri dan terkanan. Kedua titik kemudian dihubungkan menjadi garis. Titik-titik di atas garis tersebut dipisahkan untuk diproses secara terpisah dengan titik-titik yang di bawah garis.

   c. Pengecekan daerah persoalan dan rekursi

      Apabila daerah persoalan hanya berisi 2 data titik, program akan langsung mengembalikan data titik tersebut. Apabila masih cukup banyak data titik yang ada, buat garis dari titik terkiri (a) dan terkanan (b). Kemudian pilih satu titik dengan jarak terjauh dari garis yang menghubungkan titik a dan titik b. Himpunan titik-titik di dalam segitiga a,b,c tidak diperhitungkan untuk menghasilkan solusi. Setelah itu, lakukan pemanggilan rekursi fungsi *convex hull* untuk himpunan dari titik a ke titik c dan titik c ke titik b.

   d. Penggabungan solusi

      Hasil dari pemanggilan fungsi *convex hull* harus digabungkan untuk menghasilkan solusi persoalan secara utuh. Pada program ini, penulis menggabungkan dengan mengurutkan titik-titik solusi secara melingkar searah dengan jarum jam. Pada penggabungan ini juga menghindarkan adanya duplikasi titik yang muncul pada himpunan solusi.

2. *Source Program Library myConvexHull*

Fungsi pembantu untuk melakukan pengurutan dengan algoritma Quicksort

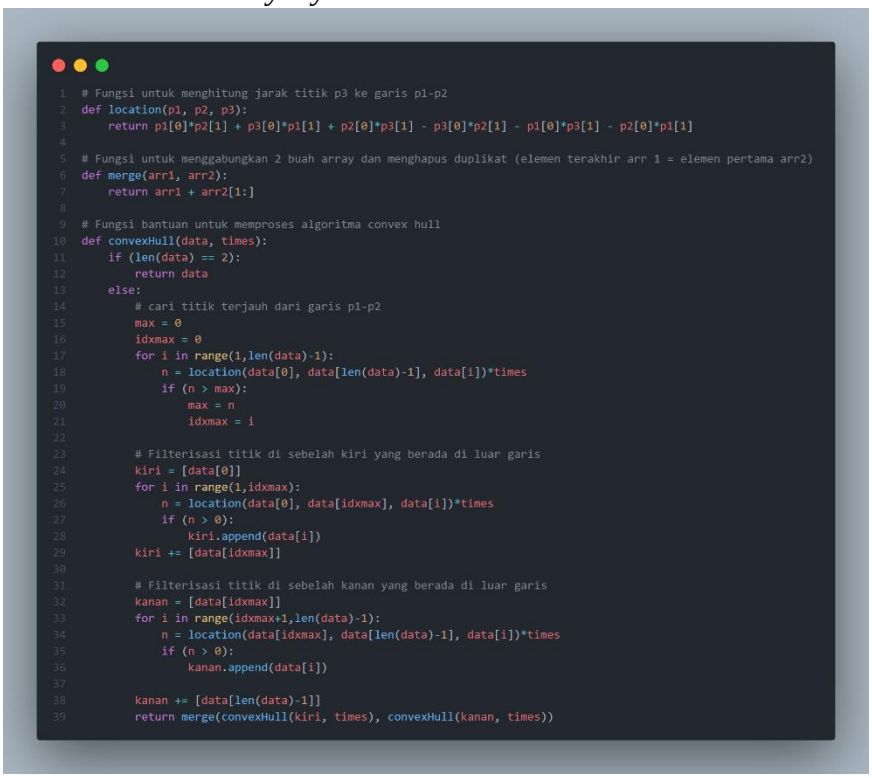
```python
# Fungsi untuk mempartisi array
def partition(arr, low, high):
    i = (low-1)             # index elemen pertama
    pivot = arr[high]       # pivot

    for j in range(low, high):
        # penukaran elemen sesuai dengan urutan menaik
        if (arr[j][0] < pivot[0]):
            i += 1
            arr[i][0], arr[j][0] = arr[j][0], arr[i][0]
            arr[i][1], arr[j][1] = arr[j][1], arr[i][1]
        elif (arr[j][0] == pivot[0] and arr[j][1] < pivot[1]):
                i += 1
                arr[i][0], arr[j][0] = arr[j][0], arr[i][0]
                arr[i][1], arr[j][1] = arr[j][1], arr[i][1]

    arr[i+1][0], arr[high][0] = arr[high][0], arr[i+1][0]
    arr[i+1][1], arr[high][1] = arr[high][1], arr[i+1][1]
    return (i+1)

# Fungsi untuk mengurutkan array dengan algoritma QuickSort
def quickSort(arr, low, high):
    if len(arr) == 1:
        return arr
    if low < high:
        pi = partition(arr, low, high)
        quickSort(arr, low, pi-1)
        quickSort(arr, pi+1, high)
```

Fungsi pembantu untuk *library myConvexHull*



```python
# Fungsi untuk menghitung jarak titik p3 ke garis p1-p2
def location(p1, p2, p3):
    return p1[0]*p2[1] + p3[0]*p1[1] + p2[0]*p3[1] - p3[0]*p2[1] - p1[0]*p3[1] - p2[0]*p1[1]

# Fungsi untuk menggabungkan 2 buah array dan menghapus duplikat (elemen terakhir arr 1 = elemen pertama arr2)
def merge(arr1, arr2):
    return arr1 + arr2[1:]

# Fungsi bantuan untuk memproses algoritma convex hull
def convexHull(data, times):
    if (len(data) == 2):
        return data
    else:
        # cari titik terjauh dari garis p1-p2
        max = 0
        idxmax = 0
        for i in range(1,len(data)-1):
            n = location(data[0], data[len(data)-1], data[i])*times
            if (n > max):
                max = n
                idxmax = i

        # Filterisasi titik di sebelah kiri yang berada di luar garis
        kiri = [data[0]]
        for i in range(1,idxmax):
            n = location(data[0], data[idxmax], data[i])*times
            if (n > 0):
                kiri.append(data[i])
        kiri += [data[idxmax]]

        # Filterisasi titik di sebelah kanan yang berada di luar garis
        kanan = [data[idxmax]]
        for i in range(idxmax+1,len(data)-1):
            n = location(data[idxmax], data[len(data)-1], data[i])*times
            if (n > 0):
                kanan.append(data[i])

        kanan += [data[len(data)-1]]
        return merge(convexHull(kiri, times), convexHull(kanan, times))
```

Fungsi utama *myConvexHull*

```python
# Fungsi awal dalam algoritma convex hull
def myConvexHull(data):
    # urutkan data dalam keadaan menaik
    quickSort(data, 0, len(data)-1)
    if (len(data) == 2):
        # base case
        return data
    else:
        atas = [data[0]]
        bawah = [data[0]]
        # pisahkan titik titik di atas dan bawah garis ujung kiri dan kanan
        for i in range(1,len(data)-1):
            n = location(data[0], data[len(data)-1], data[i])
            if (n > 0):
                atas.append(data[i])
            elif (n < 0):
                bawah.append(data[i])

        atas += [data[len(data)-1]]
        bawah += [data[len(data)-1]]
        # mengembalikan gabungan dari convex hull atas dan bawah
        return merge(convexHull(atas, 1), convexHull(bawah, -1)[::-1])
```

3. Eksperimen

3.1 Dataset Iris

```python
data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
df.head()
```

✓ 0.7s

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

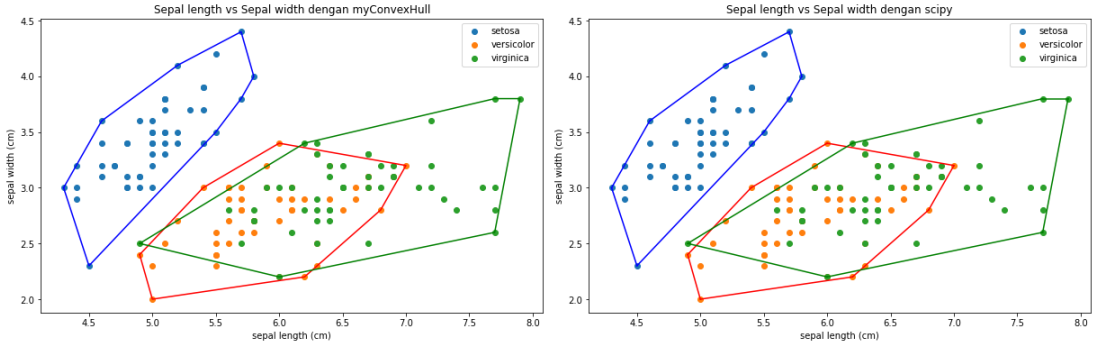
3.2.1 Sepal Length dan Sepal Width

Input :

```python
data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
df.head()

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Sepal length vs Sepal width dengan myConvexHull')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    hull = myConvexHull(bucket)            # Pemanggilan fungsi Convex Hull
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    hull = np.transpose(hull)
    plt.plot(hull[0], hull[1], color=colors[i])
plt.legend()
```

Output:



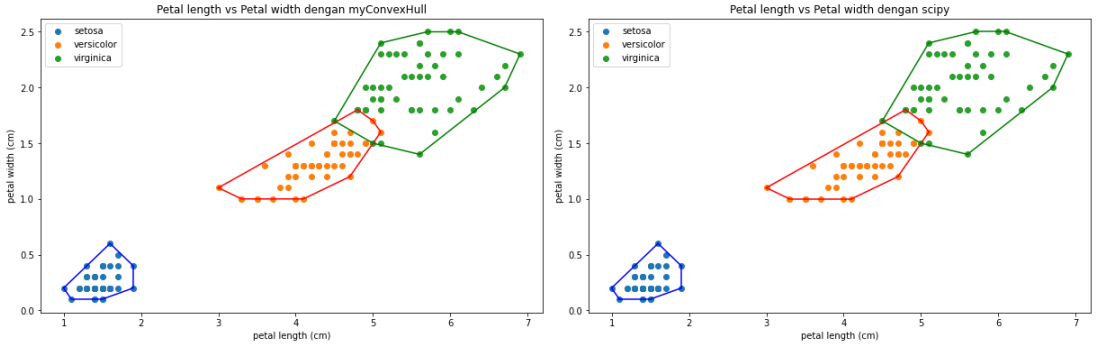
3.2.2 Petal Length dan Petal Width

Input:

```
data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
df.head()

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Petal length vs Petal width dengan myConvexHull')
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[2,3]].values
    hull = myConvexHull(bucket)            # Pemanggilan fungsi Convex Hull
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    hull = np.transpose(hull)
    plt.plot(hull[0], hull[1], color=colors[i])
plt.legend()
```

Output:



3.2 Dataset Wine

```
data = datasets.load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
df.head()
```

✓ 0.6s

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline	Target
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065.0	0
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050.0	0
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185.0	0
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480.0	0
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735.0	0

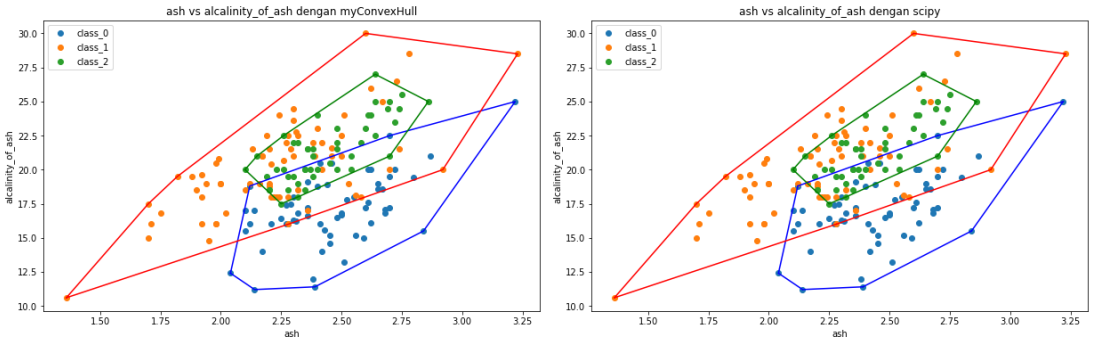
3.2.1 Ash dan Alcalinity of Ash

Input:

```
data = datasets.load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
df.head()
#print(df)

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(data.feature_names[2] + ' vs ' + data.feature_names[3] + ' dengan myConvexHull')
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[2,3]].values
    hull = myConvexHull(bucket)             # Pemanggilan fungsi Convex Hull
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    hull = np.transpose(hull)
    plt.plot(hull[0], hull[1], color=colors[i])
plt.legend()
```

Output:



3.2.2 Total Phenols dan Nonflavanoid Phenols

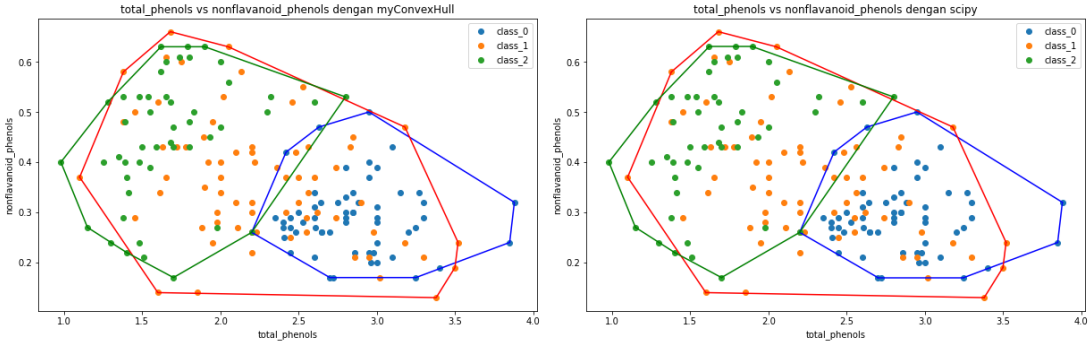
Input:

```
data = datasets.load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
df.head()
#print(df)

#visualisasi hasil ConvexHull

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(data.feature_names[5] + ' vs ' + data.feature_names[7] + ' dengan myConvexHull')
plt.xlabel(data.feature_names[5])
plt.ylabel(data.feature_names[7])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[5,7]].values
    hull = myConvexHull(bucket)             # Pemanggilan fungsi Convex Hull
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    hull = np.transpose(hull)
    plt.plot(hull[0], hull[1], color=colors[i])
plt.legend()
```

Output:



3.3 Dataset Breast Cancer

```
data = datasets.load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
df.head()
```

[32] ✓ 0.9s

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	Target
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890	0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902	0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758	0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300	0
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678	0

5 rows × 31 columns

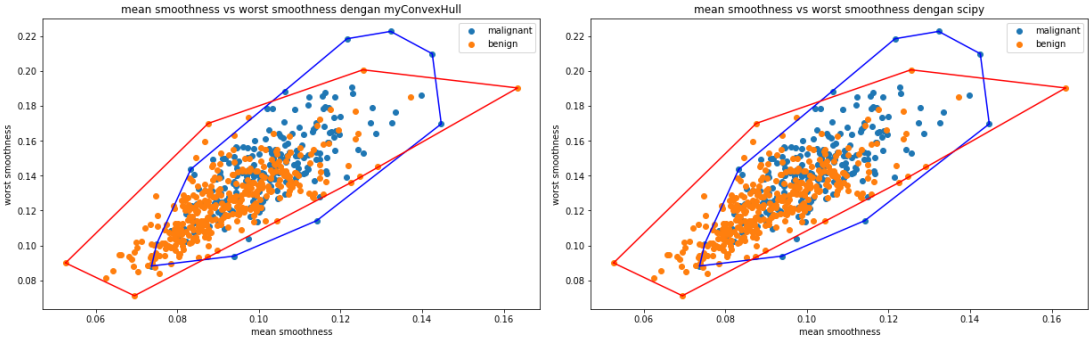
3.3.1 Mean Smoothness dan Worst Smoothness

Input:

```
data = datasets.load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
df.head()

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(data.feature_names[4] + ' vs ' + data.feature_names[24] + ' dengan myConvexHull')
plt.xlabel(data.feature_names[4])
plt.ylabel(data.feature_names[24])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[4,24]].values
    hull = myConvexHull(bucket)             # Pemanggilan fungsi Convex Hull
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    hull = np.transpose(hull)
    plt.plot(hull[0], hull[1], color=colors[i])
plt.legend()
```

Output:



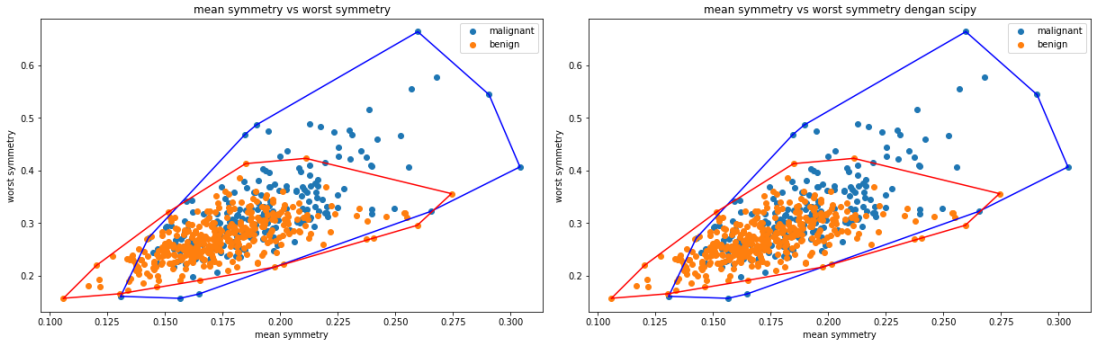
3.3.2 Mean Perimeter dan Worst Perimeter

Input:

```
data = datasets.load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
df.head()

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(data.feature_names[8] + " vs " + data.feature_names[28])
plt.xlabel(data.feature_names[8])
plt.ylabel(data.feature_names[28])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[8,28]].values
    hull = myConvexHull(bucket)            # Pemanggilan fungsi Convex Hull
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    hull = np.transpose(hull)
    plt.plot(hull[0], hull[1], color=colors[i])
plt.legend()
```

Output:



4. Alamat Kode Program

   Link Github :

   https://github.com/hafidznrg/Stima-ConvexHull

Poin	Ya	Tidak
1. Pustaka *myConvexHull* berhasil dibuat dan tidak ada kesalahan	✓	
2. *Convex Hull* yang dihasilkan sudah benar	✓	
3. Pustakan *myConvexHull* dapat digunakan untuk menampilkan *convex hull* setiap label dengan warna yang berbeda	✓	
4. Bonus : Program dapat menerima input dan menuliskan outpu untuk dataset lainnya	✓	