

Laporan Tugas Kecil 2 Strategi Algoritma IF2211

Zayd Muhammad Kawakibi Zuhri

13520144

1. Algoritma *Divide and Conquer* Untuk Mencari *Convex Hull*

Fungsi `myConvexHull` menerima data titik-titik berupa array 2 dimensi, dengan setiap titik dalam array memiliki nilai x dan y . Karena fungsi ini mengembalikan *convex hull* dalam bentuk pasangan indeks titik-titik yang akan disambung dengan garis, kita harus menambahkan kolom indeks terlebih dahulu pada data titik-titik tadi. Setelah itu, data diurut berdasarkan nilai absis yang menaik, dengan absis yang sama diurut berdasarkan ordinat yang menaik. Dari data yang tersortir tersebut, kita ambil dua titik yaitu titik paling kiri dan paling kanan, atau titik yang di awal dan di akhir data tersortir. Menggunakan rumus matriks dalam fungsi `isAbove`, kita bagi data titik-titik yang tersisa menjadi dua, yaitu titik yang berada di atas dan bawah garis yang dibentuk dari titik terkiri ke terkanan. Hanya setelah itu kita dapat mengisi array `hull` menggunakan fungsi rekursif `makeHull`. Rekursi ini dipanggil dua kali, satu untuk kumpulan titik di atas garis awal, dan sekali lagi untuk kumpulan titik di bawah garis awal.

Fungsi rekursif `makeHull` menerima array `hull` dan kumpulan titik di atas garis yang di bentuk oleh dua titik, yaitu titik terkiri dan terkanan. Basis dari rekursi ini terjadi jika kumpulan titik tersebut kosong, yang berarti tidak ada lagi titik di atas garis yang sedang diamati. Pada basis ini kita hanya perlu menambahkan indeks dari titik terkiri dan titik terkanan sebagai suatu pasangan ke dalam array `hull`. Namun, jika terdapat titik-titik di atas garis, maka fungsi akan masuk ke bagian rekursif. Dari kumpulan titik tersebut, kita ambil titik yang terjauh dari garis dengan menggunakan fungsi `distanceFromLine`. Fungsi tersebut menerima tiga titik, dan mengembalikan jarak titik pertama ke garis yang dibentuk oleh dua titik lainnya. Setelah itu, kita dapat menghapus titik terjauh tersebut dari kumpulan titik. Sekarang, kita harus kembali membagi kumpulan titik menjadi dua, yang pertama adalah kumpulan titik yang berada di atas garis yang dibentuk dari titik terkiri ke titik terjauh, dan yang kedua adalah kumpulan titik yang berada di atas garis yang dibentuk dari titik terjauh ke titik terkanan. Di sini kita gunakan kembali fungsi `isAbove`. Pada kasus tertentu, titik terjauh tadi merupakan satu-satunya titik di atas garis yang diamati. Jika ini terjadi, maka kumpulan titik yang dibagi menjadi dua tadi keduanya akan kosong, agar masuk ke dalam kasus basis pada iterasi selanjutnya. Setelah membagi kumpulan titik menjadi dua tadi, fungsi akan direkursikan dengan memanggil `makeHull` dua kali untuk masing-masing kumpulan titik dan dua garis yang baru dibentuk. Basis rekursi memastikan garis terluar *convex hull* akan tercatat ke dalam array `hull`.

2. Source Code (Dalam bahasa Python)

```
1 # Mengembalikan nilai True jika titik p3 berada di atas garis yang dibentuk dari titik p1 ke p2
2 def isAbove(p1, p2, p3):
3     return ((p1[1]*p2[2])+(p3[1]*p1[2])+(p2[1]*p3[2])-(p3[1]*p2[2])-(p2[1]*p1[2])-(p1[1]*p3[2])) > 0
4
5 # Mengembalikan jarak titik point ke garis yang dibentuk dari titik line1 ke line2
6 def distanceFromLine(point, line1, line2):
7     p1 = np.array(line1[1:])
8     p2 = np.array(line2[1:])
9     p3 = np.array(point[1:])
10    return np.abs(np.cross(p2-p1, p1-p3)) / norm(p2-p1)
11
12 # Mengembalikan titik terjauh dari garis yang dibentuk dari titik Leftmost ke rightmost dari kumpulan points
13 def getFarthestPoint(points, Leftmost, rightmost):
14     return max(points, key=lambda point: distanceFromLine(point, Leftmost, rightmost))
```

```

16 # Fungsi rekursif untuk membentuk convex hull
17 def makeHull(hull, points, leftmost, rightmost):
18     # Basis rekursi:
19     # Jika tidak ada titik yang berada di atas garis yang dibentuk dari titik leftmost ke rightmost
20     # Tambahkan garis yang dibentuk dari titik leftmost ke rightmost ke kumpulan hull
21     if len(points) == 0:
22         hull.append([int(leftmost[0]), int(rightmost[0])])
23     # Rekursif jika ada titik selain garis yang dibentuk dari titik leftmost ke rightmost:
24     else:
25         # Dapatkan titik terjauh dari garis yang dibentuk dari titik leftmost ke rightmost
26         farthest = getFarthestPoint(points, leftmost, rightmost)
27         points = points[points[:,0] != farthest[0]]
28         # Bagi kumpulan titik menjadi 2 bagian dan abaikan sisanya
29         if (points.shape[0] == 0):
30             left = []
31             right = []
32         else:
33             left_mask = np.array([isAbove(leftmost, farthest, point) for point in points])
34             right_mask = np.array([isAbove(farthest, rightmost, point) for point in points])
35             left = points[left_mask]
36             right = points[right_mask]
37         # Lakukan rekursi untuk kedua garis yang dibentuk ke titik terjauh dengan titik-titik relevan
38         makeHull(hull, left, leftmost, farthest)
39         makeHull(hull, right, farthest, rightmost)
40
41 # Fungsi utama untuk mencari convex hull
42 def myConvexHull(data):
43     # Tambahkan index sebagai kolom pertama
44     data = np.insert(data, 0, np.arange(len(data)), axis=1)
45     # Sortir data berdasarkan absis menaik, dan ordinat menaik
46     sort = data[np.argsort(data[:,2])]
47     sort = sort[np.argsort(sort[:,1], kind='stable')]
48     # Dapatkan titik-titik paling kiri dan paling kanan
49     leftmost = sort[0]
50     rightmost = sort[-1]
51     sort = sort[1:-1]
52     hull = []
53     # Bagi kumpulan titik menjadi 2 bagian di atas dan bawah garis dari titik leftmost ke rightmost
54     above_mask = np.array([isAbove(leftmost, rightmost, sort[i]) for i in range(len(sort))])
55     below_mask = np.array([isAbove(rightmost, leftmost, sort[i]) for i in range(len(sort))])
56     above = sort[above_mask]
57     below = sort[below_mask]
58     # Mulai rekursi untuk mengisi kumpulan garis hull
59     makeHull(hull, above, leftmost, rightmost)
60     makeHull(hull, below, rightmost, leftmost)
61     # Kembalikan kumpulan garis hull
62     return hull

```

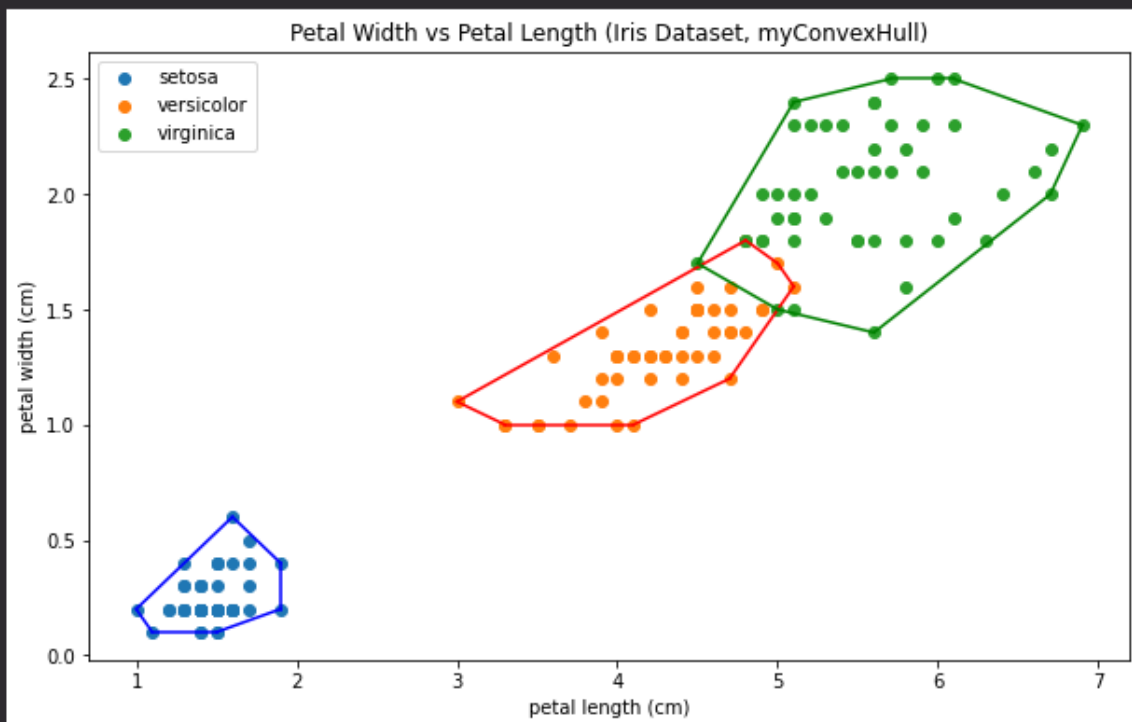
3. Contoh-Contoh Hasil myConvexHull

a. Petal Width vs Petal Length dari kolom 3 dan 4 dataset Iris

```
plt.figure(figsize = (10, 6))
colors = ['b', 'r', 'g']
plt.title('Petal Width vs Petal Length (Iris Dataset, myConvexHull)')
plt.xlabel(dataIris.feature_names[2])
plt.ylabel(dataIris.feature_names[3])
for i in range(len(dataIris.target_names)):
    bucket = dfI[dfI['Target'] == i]
    bucket = bucket.iloc[:, [2, 3]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=dataIris.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

✓ 0.2s

<matplotlib.legend.Legend at 0x1536e27ae50>

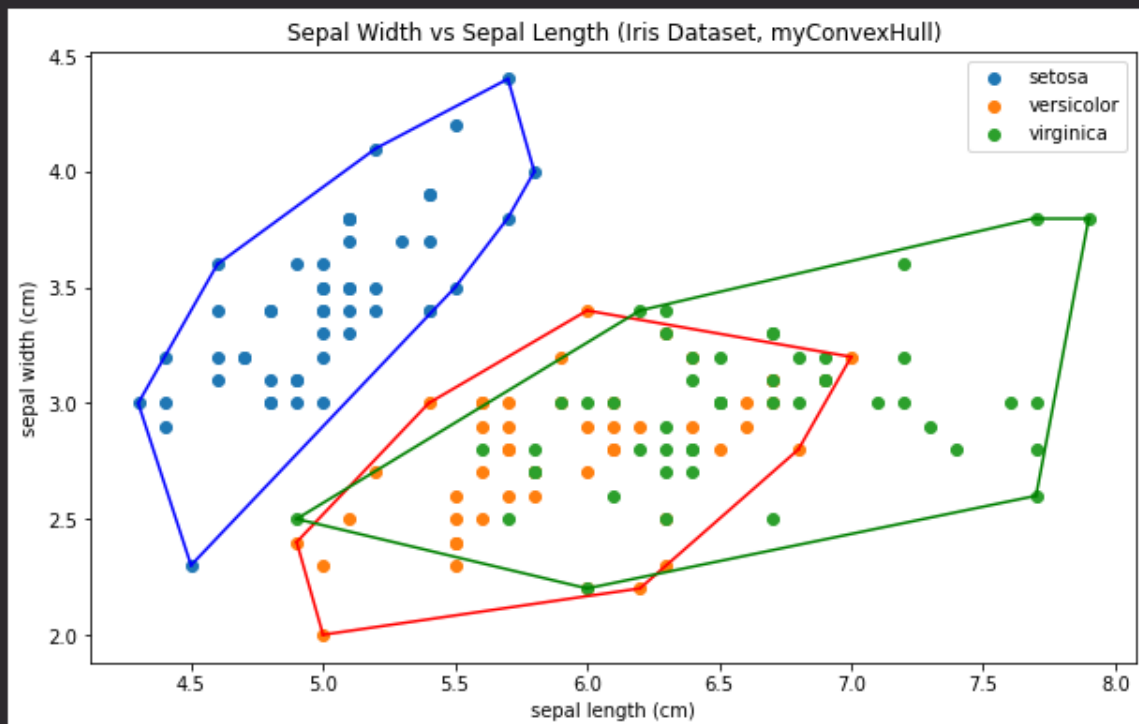


b. Sepal Width vs Sepal Length dari kolom 1 dan 2 dataset Iris

```
plt.figure(figsize = (10, 6))
colors = ['b', 'r', 'g']
plt.title('Sepal Width vs Sepal Length (Iris Dataset, myConvexHull)')
plt.xlabel(dataIris.feature_names[0])
plt.ylabel(dataIris.feature_names[1])
for i in range(len(dataIris.target_names)):
    bucket = dfI[dfI['Target'] == i]
    bucket = bucket.iloc[:, [0, 1]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=dataIris.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

✓ 0.2s

<matplotlib.legend.Legend at 0x1536e400a30>

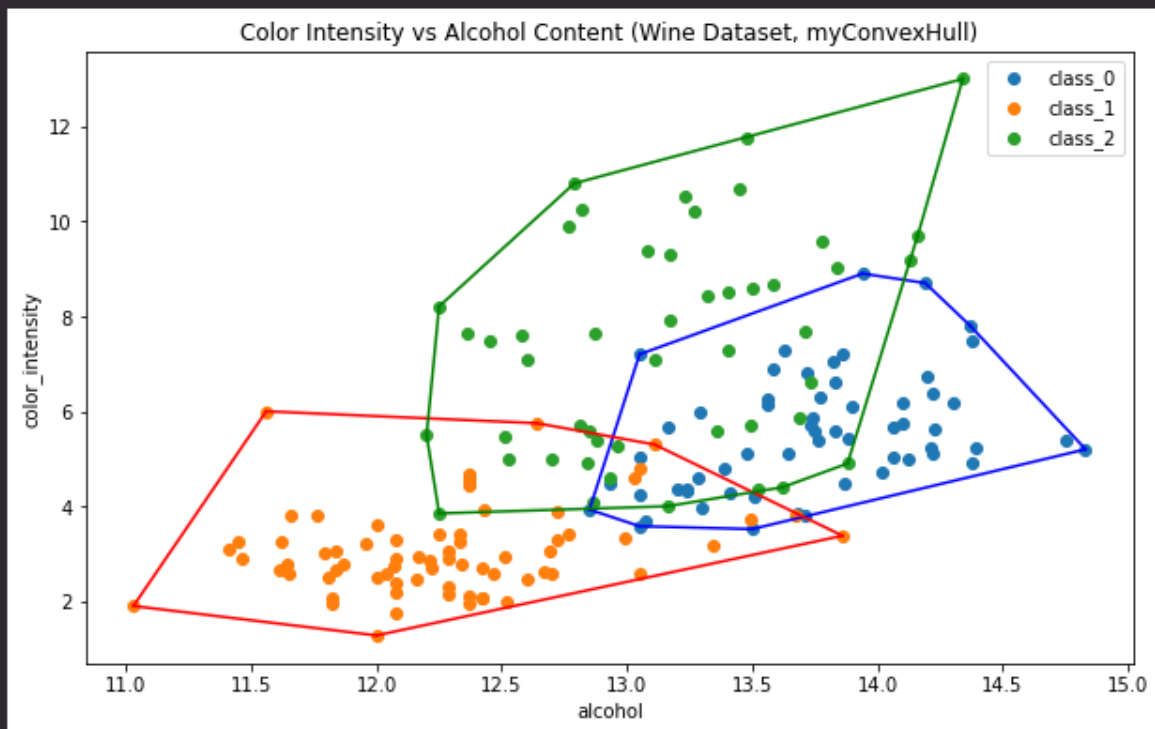


c. Color Intensity vs Alcohol Content dari kolom 1 dan 10 dataset Wine

```
plt.figure(figsize = (10, 6))
colors = ['b', 'r', 'g']
plt.title('Color Intensity vs Alcohol Content (Wine Dataset, myConvexHull)')
plt.xlabel(dataWine.feature_names[0])
plt.ylabel(dataWine.feature_names[9])
for i in range(len(dataWine.target_names)):
    bucket = dfW[dfW['Target'] == i]
    bucket = bucket.iloc[:,[0,9]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=dataWine.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

✓ 0.3s

<matplotlib.legend.Legend at 0x1536e5906d0>

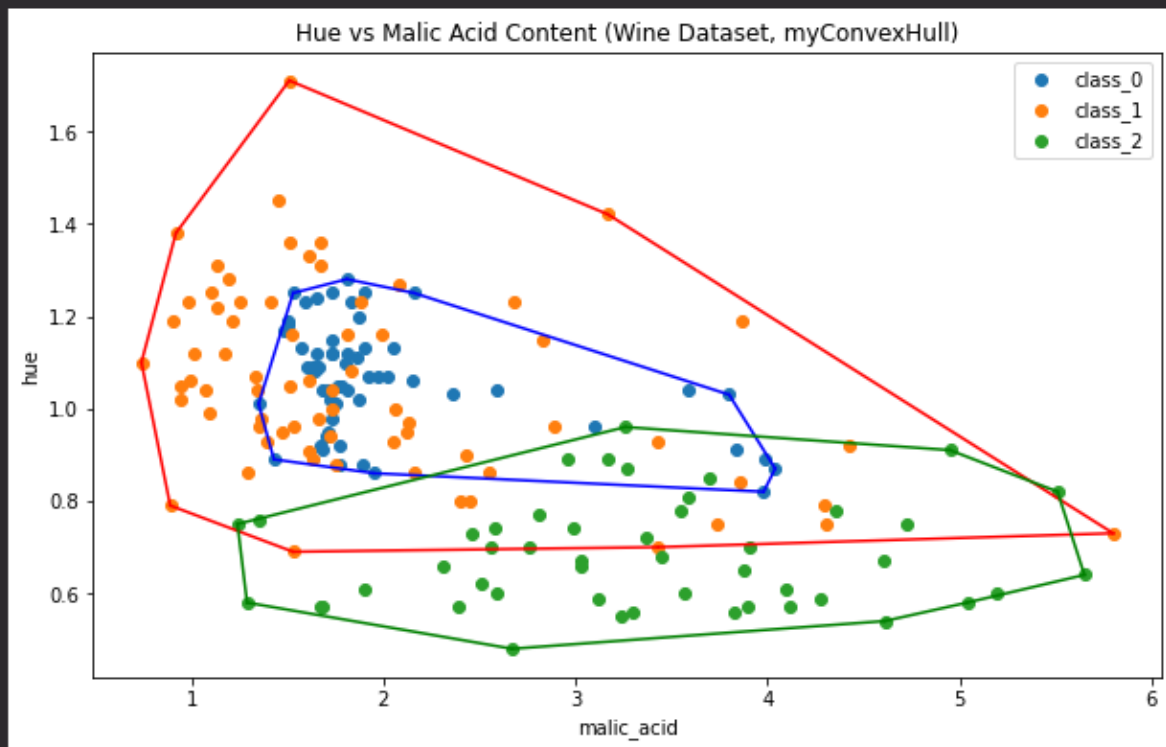


d. Hue vs Malic Acid Content dari kolom 2 dan 11 dataset Wine

```
plt.figure(figsize=(10,6))
colors = ['b', 'r', 'g']
plt.title('Hue vs Malic Acid Content (Wine Dataset, myConvexHull)')
plt.xlabel(dataWine.feature_names[1])
plt.ylabel(dataWine.feature_names[10])
for i in range(len(dataWine.target_names)):
    bucket = dfw[dfw['Target'] == i]
    bucket = bucket.iloc[:, [1,10]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=dataWine.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

✓ 0.3s

<matplotlib.legend.Legend at 0x1536e4f8370>

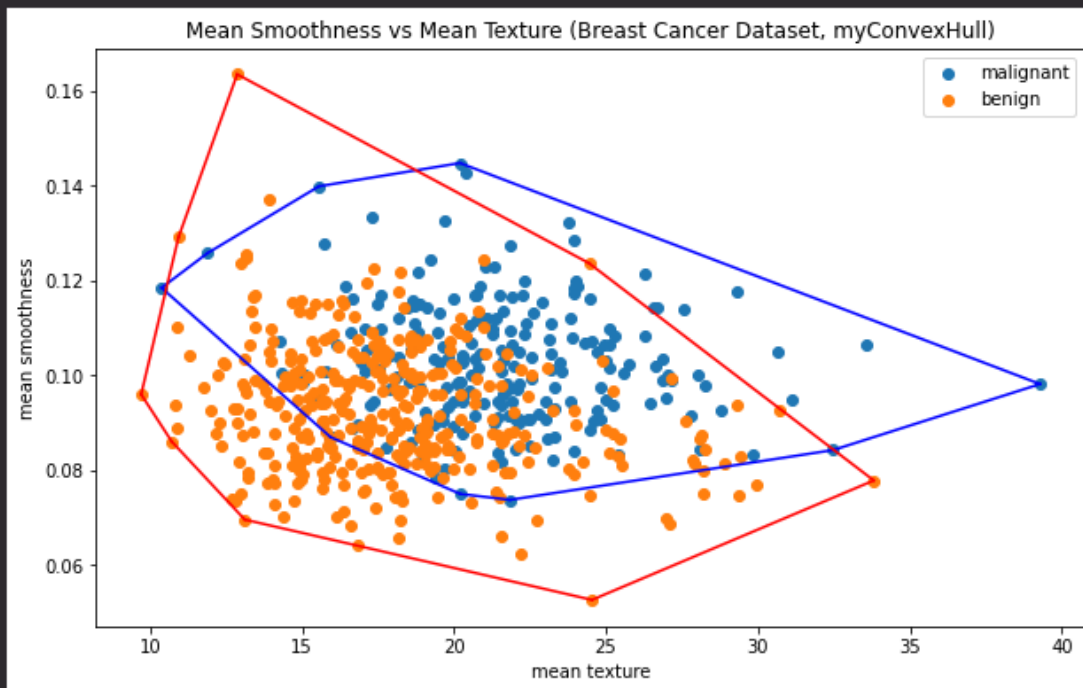


e. Mean Smoothness vs Mean Texture dari kolom 2 dan 5 dataset Breast Cancer

```
plt.figure(figsize = (10, 6))
colors = ['b', 'r', 'g']
plt.title('Mean Smoothness vs Mean Texture (Breast Cancer Dataset, myConvexHull)')
plt.xlabel(dataBreast.feature_names[1])
plt.ylabel(dataBreast.feature_names[4])
for i in range(2):
    bucket = dfB[dfB['Target'] == i]
    bucket = bucket.iloc[:, [1, 4]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=dataBreast.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

✓ 0.2s

<matplotlib.legend.Legend at 0x1536e998e20>

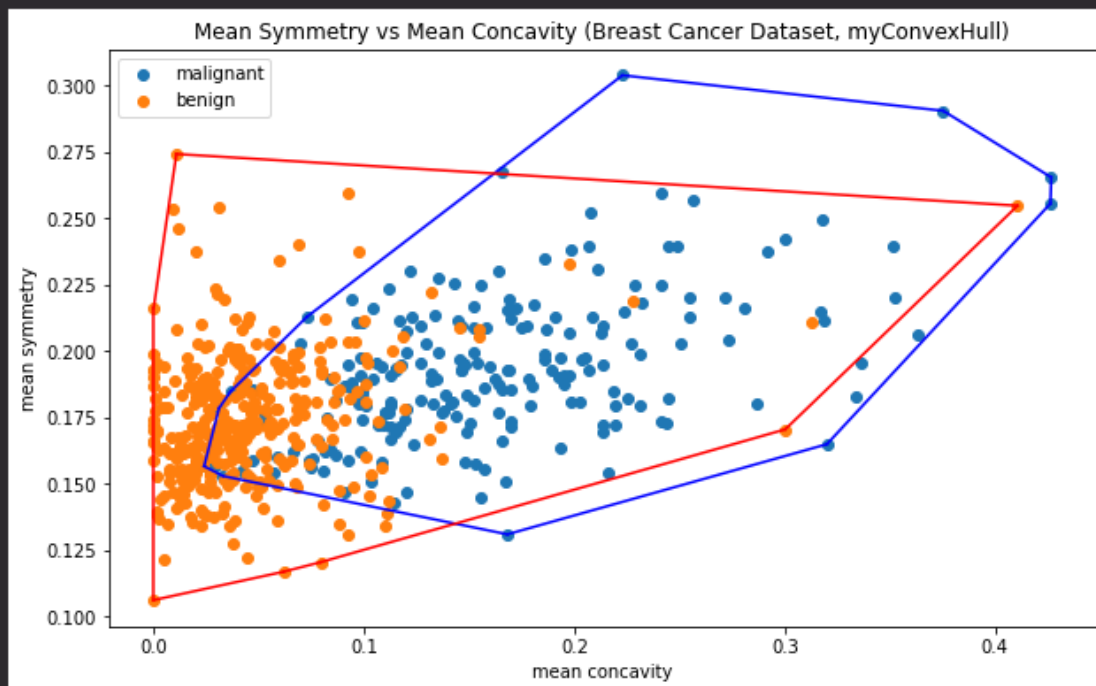


f. Mean Symmetry vs Mean Concavity dari kolom 7 dan 9 dataset Breast Cancer

```
plt.figure(figsize = (10, 6))
colors = ['b', 'r', 'g']
plt.title('Mean Symmetry vs Mean Concavity (Breast Cancer Dataset, myConvexHull)')
plt.xlabel(dataBreast.feature_names[6])
plt.ylabel(dataBreast.feature_names[8])
for i in range(2):
    bucket = dfB[dfB['Target'] == i]
    bucket = bucket.iloc[:, [6, 8]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=dataBreast.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```




✓ 0.2s

<matplotlib.legend.Legend at 0x1536ecce670>



4. Link GitHub Repository Source Code

https://github.com/zaydzuhri/Tucil2_13520144

Poin	Ya	Tidak
1. Pustaka <i>myConvexHull</i> berhasil dibuat dan tidak ada kesalahan		
2. <i>Convex hull</i> yang dihasilkan sudah benar		
3. Pustaka <i>myConvexHull</i> dapat digunakan untuk menampilkan <i>convex hull</i> setiap label dengan warna yang berbeda.		
4. Bonus: program dapat menerima input dan menuliskan output untuk dataset lainnya.	