

Laporan Tugas Kecil 2
IF2211 Strategi Algoritma Tahun 2021/2022



Oleh:

Addin Nabilal Huda

13520045

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2022

A. Algoritma Divide and Conquer Secara Garis Besar

Convex hull merupakan poligon yang disusun dari subset titik sedemikian sehingga untuk sembarang dua titik pada bidang tersebut (misal p dan q), seluruh segmen garis yang berakhir di p dan q berada pada himpunan tersebut. Permasalahan *convex hull* merupakan suatu permasalahan pada persoalan *computational geometry* dan dapat dikembangkan untuk persoalan animasi komputer, optimasi, dan statistik.

Pada tugas ini, algoritma *divide and conquer* digunakan untuk membuat pustaka yang dapat mengembalikan *convex hull* dari kumpulan data 2 dimensi. Ide dasar dari penerapan algoritma *divide and conquer* pada permasalahan ini adalah menggunakan algoritma *quick sort* dan mengikuti aturan berikut:

1. Untuk kumpulan titik berisi dua titik, *convex hull* berupa garis yang menghubungkan dua titik tersebut
2. Untuk kumpulan titik berupa tiga titik yang berada pada satu garis, *convex hull* berupa garis yang menghubungkan dua titik dengan jarak terjauh
3. Untuk kumpulan titik berupa tiga titik yang tidak berada pada satu garis, *convex hull* berupa segitiga yang menghubungkan ketiga titik tersebut
4. Untuk kumpulan titik dengan jumlah lebih banyak yang tidak berada pada satu garis, *convex hull* berupa poligon *convex* dengan sisi berupa garis yang menghubungkan beberapa titik

Secara garis besar, pemanfaatan algoritma *divide and conquer* mengikuti langkah sebagai berikut:

1. Program menerima masukan berupa array of point (kumpulan titik S), di mana setiap titik direpresentasikan oleh array $[x \ y]$
2. Program melakukan *sorting* pada masukan kumpulan titik berdasarkan nilai absis yang menaik, dan bila ada absis yang sama, diurutkan dengan nilai ordinat yang menaik
3. Program menyimpan titik dengan absis terkecil dan terbesar sebagai titik ekstrim yang membentuk *convex hull* untuk kumpulan titik tersebut
4. Garis yang terbentuk dari titik-titik ekstrim (misal p_1 dan p_2) membagi S menjadi dua bagian yaitu S_1 (kumpulan titik di sebelah kiri relatif terhadap garis p_1p_2) dan S_2 (kumpulan titik di sebelah kanan relatif terhadap garis p_1p_2)
5. Pada setiap sisi/bagian, dicari *convex hull* dengan terlebih dahulu mencari titik dengan jarak terjauh relatif terhadap garis p_1p_2 . Pencarian titik terjauh ini memanfaatkan rumus perhitungan determinan matriks (dilakukan oleh fungsi *findDistance*). Terdapat kemungkinan hasil perhitungan determinan sebagai berikut:
 - a. Bila hasil perhitungan determinan antara p_1 , p_2 , dan titik yang akan dicek (misalnya p_3) bernilai positif, maka titik p_3 berada di kiri garis p_1p_2
 - b. Bila hasil perhitungan determinan antara p_1 , p_2 , dan titik yang akan dicek (misalnya p_3) bernilai negatif maka titik p_3 berada di kanan garis p_1p_2

- c. Bila hasil perhitungan determinan antara p1, p2, dan titik yang akan dicek (misalnya p3) bernilai 0, maka titik p3 berada pada garis p1p2
6. Pada setiap *convex hull* pada suatu sisi/bagian (dilakukan oleh fungsi findHull), misal S1, terdapat kemungkinan sebagai berikut:
 - a. Jika tidak ada titik lain selain p1 dan p2, maka titik p1 dan p2 yang menjadi pembentuk *convex hull* bagian S1. Fungsi findHull akan mengembalikan array yang berisi [index1, index2] di mana index 1 merupakan indeks p1 pada *array of points* masukan dan index 2 merupakan indeks p2 pada *array of points* masukan
 - b. Jika S1 tidak kosong, pilih titik terjauh dari p1p2, misalnya p3 (dilakukan oleh fungsi findIndexofMaxDistance dengan memanggil fungsi findDistance dan iterasi pada setiap poin). Bila ada beberapa titik yang memiliki jarak yang sama, kembalikan titik yang membentuk sudut terbesar dengan garis p1p2. Hanya titik-titik yang berada pada sisi yang sedang ditinjau yang masuk ke dalam pencarian titik terjauh.
7. Program akan melakukan rekursi pada pemanggilan findHull untuk mencari *convex hull* pada kumpulan titik yang berada pada sebelah kiri garis p1p3 dan sebelah kanan garis p2p3 dengan. Poin 6a menjadi basis dan poin 6b menjadi rekursi dari proses rekursif ini
8. Pada setiap rekursi, program akan melakukan konkantenasi pada *array* yang berisi pasangan indeks titik-titik yang membentuk *convex hull* dari kedua sisi
9. Hasilnya adalah kumpulan pasangan indeks titik-titik yang membentuk *convex hull* dari kumpulan titik masukan

B. Kode Program

1. Fungsi findSide

Fungsi: untuk mencari letak suatu titik relatif terhadap garis yang menghubungkan titik p1 dan p2

Lokasi: side.py

```

1  # fungsi untuk mengecek sisi titik pToCheck relatif terhadap garis yang menghubungkan p1 dan p2
2  # mengembalikan -1 bila pToCheck berada di kanan garis yang menghubungkan p1 dan p2
3  # mengembalikan 1 bila pToCheck berada di kiri garis yang menghubungkan p1 dan p2
4  def findSide(p1,p2,pToCheck):
5      # menggunakan rumus determinan
6      det=p1[0]*p2[1] + pToCheck[0]*p1[1] + p2[0]*pToCheck[1] - pToCheck[0]*p2[1] - p2[0]*p1[1] - p1[0]*pToCheck[1]
7      # hasil determinan positif berarti pToCheck berada di kiri garis yang menghubungkan p1 dan p2
8      if det>0: return -1
9      # hasil determinan negatif berarti pToCheck berada di kanan garis yang menghubungkan p1 dan p2
10     elif det<0: return 1
11     # bila pToCheck berarti pToCheck berada pada garis yang menghubungkan p1 dan p2
12     else: return 0

```

2. Fungsi findDistance

Fungsi: untuk mencari jarak dari pToCheck terhadap garis yang menghubungkan p1 dan p2

Lokasi: distance.py

```
from side import findSide
from angle import findAngle

# fungsi untuk mencari jarak dari pToCheck terhadap garis yang menghubungkan p1 dan p2
def findDistance(p1,p2,pToCheck):
    return abs((pToCheck[1]-p1[1])*(p2[0]-p1[0])-(p2[1]-p1[1])*(pToCheck[0]-p1[0]))
```

3. Fungsi findIndexOfmaxDistance

Fungsi: untuk mencari indeks titik dengan jarak terjauh (dan sudut terbesar bila jarak sama) dari garis yang menghubungkan p1 dan p2

Lokasi: distance.py

```
8 # fungsi untuk mencari index pada array of point dari titik terjauh dari garis yang menghubungkan p1 dan p2
9 # pada sisi tertentu
10 def findIndexOfMaxDistance(points, p1, p2, side):
11     maxDist=0
12     maxAngle=0
13     maxDistIndex=-1
14     n=len(points)
15     # iterasi di setiap titik pada kumpulan titik
16     for i in range(n):
17         tempDist=findDistance(p1,p2,points[i])
18         tempAngle=findAngle(p1,points[i],p2)
19         # hanya memperhitungkan titik-titik pada sisi yang sedang ditinjau
20         if findSide(p1,p2,points[i])==side and tempDist>=maxDist:
21             # bila jarak sama, cari titik yang membentuk sudut terbesar dengan p1 dan p2
22             if tempDist==maxDist:
23                 if tempAngle>maxAngle:
24                     maxAngle=tempAngle
25                     maxDist=tempDist
26                     maxDistIndex=i
27             else:
28                 maxAngle=tempAngle
29                 maxDist=tempDist
30                 maxDistIndex=i
31     return maxDistIndex
```

4. Fungsi findAngle

Fungsi: untuk mencari sudut antara p1, titik yang akan dicek, dan p

Lokasi: angle.py

```

Tucil 2 > utils > angle.py > findAngle
1  from math import degrees, pow, sqrt
2  from numpy import arccos
3
4  def findAngle(p1, pToCheck, p2):
5      # square of absis distance between p1 and pToCheck
6      p1px = pow((p1[0] - pToCheck[0]), 2)
7      # square of absis distance between of p1 and p2
8      p1p2x = pow((p1[0] - p2[0]), 2)
9      # square of absis distance between of pToCheck and p2
10     pp3x = pow((pToCheck[0] - p2[0]), 2)
11
12     # square of ordinate distance between of p1 and pToCheck
13     p1py = pow((p1[1] - pToCheck[1]), 2)
14     # square of ordinate distance between of p1 and p2
15     p1p2y = pow((p1[1] - p2[1]), 2)
16     # square of coordinate distance between of pToCheck and p2
17     pp3y = pow((pToCheck[1] - p2[1]), 2)
18     if (2 * sqrt(p1px + p1py) * sqrt(pp3x + pp3y)) == 0:
19         return 0
20     # find cosine angle based on law of cosine/cosine rule
21     cosineAngle = arccos((p1px + p1py + pp3x + pp3y - p1p2x - p1p2y) / (2 * sqrt(p1px + p1py) * sqrt(pp3x + pp3y)))
22     return degrees(cosineAngle)

```

5. Fungsi findHull

Fungsi: untuk mencari titik-titik yang membentuk *convex hull* pada sisi tertentu

Lokasi: hull.py

```

1  import numpy as np
2  from side import findSide
3  from distance import findIndexOfMaxDistance
4
5  # fungsi yang menghasilkan array yang berisi hull pada sisi tertentu
6  # contoh keluaran: [[2 3] [1 4]] berarti hull terbentuk dari pasangan poin pada index 2 dan 4 serta 1 dan 4
7  def findHull(points, p1, p2, side):
8      maxDistIndex = findIndexOfMaxDistance(points, p1, p2, side)
9      if maxDistIndex == -1:
10         listOfPoints = points.tolist()
11         index1 = listOfPoints.index(p1.tolist())
12         index2 = listOfPoints.index(p2.tolist())
13         return [[index1, index2]]
14     # rekursi untuk mencari hull pada sisi di luar "segitiga" p1, p2, dan titik terjauh dengan garis yang menghubungkan
15     hull1 = findHull(points, p1, points[maxDistIndex], -findSide(p1, points[maxDistIndex], p2))
16     hull2 = findHull(points, p2, points[maxDistIndex], -findSide(p2, points[maxDistIndex], p1))
17     # gabungkan array berisi hull pada sisi kiri dan kanan di luar "segitiga" p1, p2, dan titik terjauh dengan garis
18     return np.concatenate((hull1, hull2))

```

6. Fungsi myConvexHull

Fungsi: untuk mencari *convex hull* dari kumpulan titik input

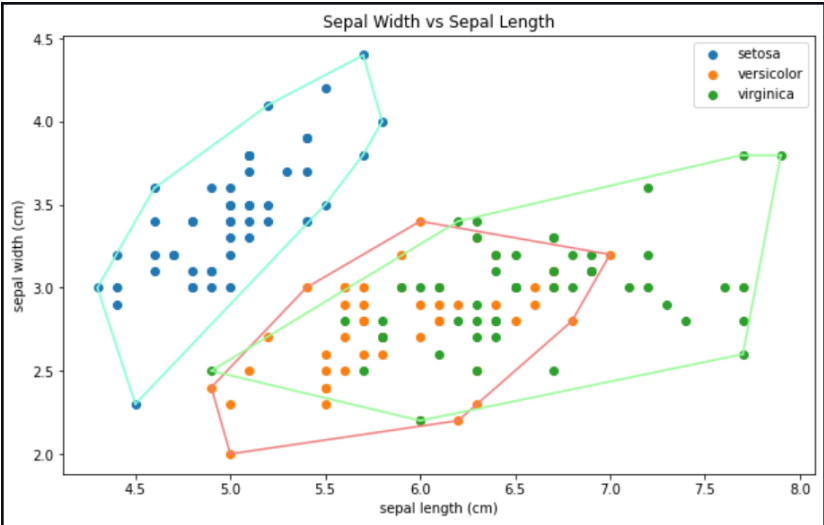
Lokasi: MyConvexHull.py

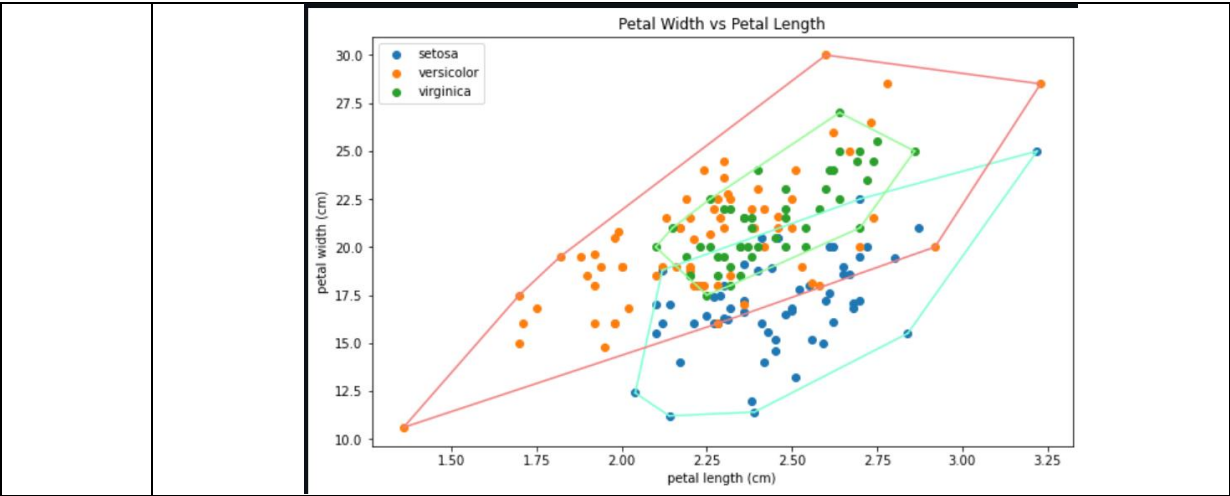
```
1 import numpy as np
2 from hull import findHull
3
4 # fungsi utama untuk mencari convex hull dari input points
5 def myConvexHull(points):
6     # buat kelas objHull dengan atribut
7     class objHull:
8         def __init__(self,simplices):
9             self.simplices=simplices
10
11     # urutkan kumpulan titik berdasarkan absis yang menaik,
12     # kemudian bila absis sama, urutkan berdasarkan ordinat yang menaik
13     pointsTemp=sorted(points, key= lambda p: (p[0],p[1]))
14     # cari titik-titik ekstrem
15     minX=0
16     maxX=len(points)-1
17     # cari hull-hull di kiri dan kanan dari garis yang menghubungkan kedua titik ekstrem
18     hullLeft=findHull(points,pointsTemp[minX],pointsTemp[maxX],1)
19     hullRight=findHull(points,pointsTemp[minX],pointsTemp[maxX],-1)
20     # gabungkan array berisi hull pada sisi kiri dan kanan di luar "segitiga" p1, p2, dan titik terjauh dengan garis p1p2
21     hull=np.concatenate((hullLeft,hullRight))
22     return objHull(hull)
```

C. Screenshot Masukan dan Keluaran Program

1. Dataset 1: Iris

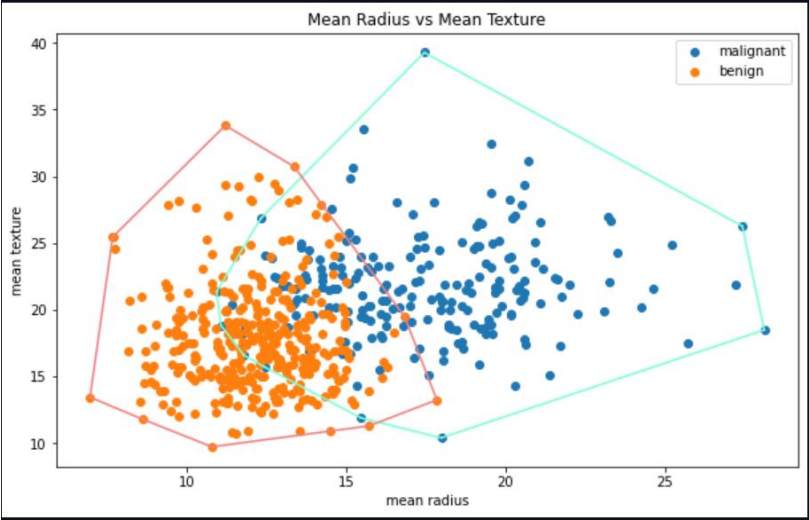
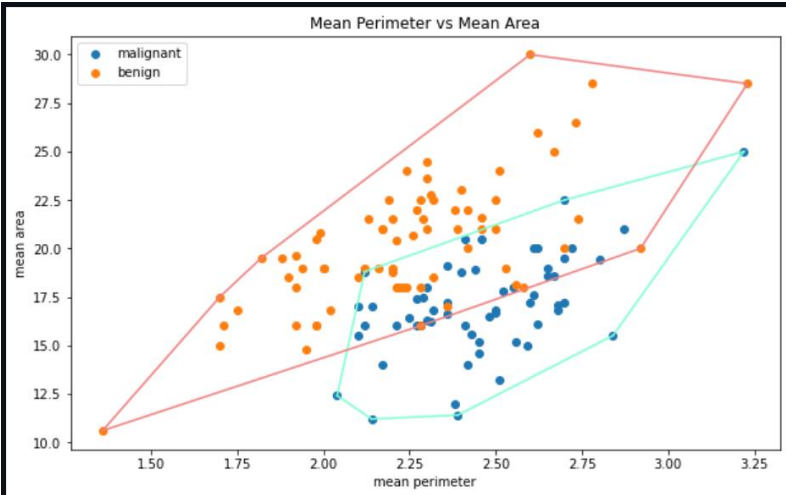
| | |
|--------------|----------|
| Data-frame | Masukan |
| | Keluaran |
| Sepal length | Masukan |

| | |
|--------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <div>vs sepal width</div> | <div>Test 1: Sepal Width vs Sepal Length</div> <div><pre># Convex hull visualization plt.figure(figsize = (10, 6)) colors = ['aquamarine','lightcoral','palegreen'] plt.title('Sepal Width vs Sepal Length') plt.xlabel(dataIris.feature_names[0]) plt.ylabel(dataIris.feature_names[1]) for i in range(len(dataIris.target_names)): bucket = df[df['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.simplices: plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i]) plt.legend()</pre></div> <div>Keluaran</div> <div></div> |
| <div>Petal length vs petal width</div> | <div>Masukan</div> <div><div>Test 2: Petal Width vs Petal Length</div><div><pre># Convex hull visualization plt.figure(figsize = (10, 6)) colors = ['aquamarine','lightcoral','palegreen'] plt.title('Petal Width vs Petal Length') plt.xlabel(dataIris.feature_names[2]) plt.ylabel(dataIris.feature_names[3]) for i in range(len(dataIris.target_names)): bucket = df[df['Target'] == i] bucket = bucket.iloc[:,[2,3]].values hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi C plt.scatter(bucket[:, 0], bucket[:, 1], label=dataIris.target_names[i]) for simplex in hull.simplices: plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i]) plt.legend()</pre></div></div> <div>Keluaran</div> |

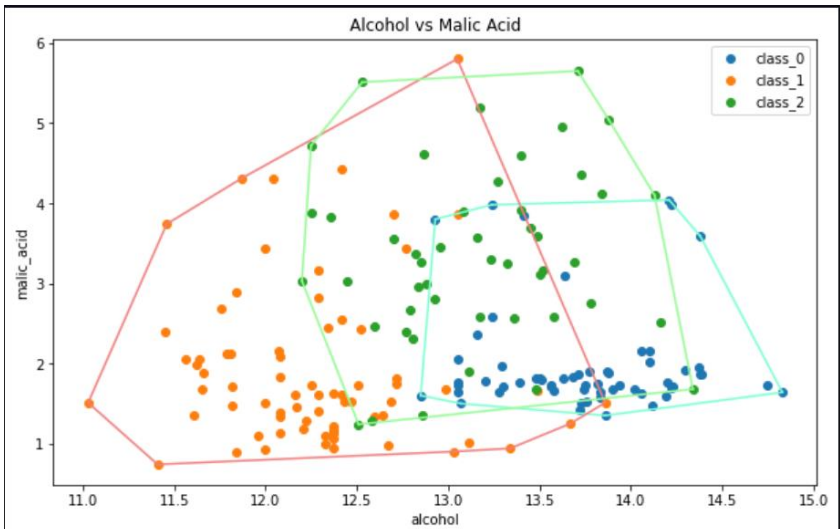


2. Dataset 2: Breast Cancer

| | Masukan | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|----------------|--------------|-----------------|------------------|-----------------|---------------------|----------------|------------------------|---------------|------------------------|-----------------|---------------|------------------|-------------------|------------------|-------------------|---|-------|-------|--------|--------|---------|---------|--------|---------|--------|---------|-----|-------|--------|--------|--------|--------|---|-------|-------|--------|--------|---------|---------|--------|---------|--------|---------|-----|-------|--------|--------|--------|--------|---|-------|-------|--------|--------|---------|---------|--------|---------|--------|---------|-----|-------|--------|--------|--------|--------|---|-------|-------|-------|-------|---------|---------|--------|---------|--------|---------|-----|-------|-------|-------|--------|--------|---|-------|-------|--------|--------|---------|---------|--------|---------|--------|---------|-----|-------|--------|--------|--------|--------|
| Datafram e | <pre># Run this cell to create breast cancer DataFrame dataBreastCancer = datasets.load_breast_cancer() #create a DataFrame df = pd.DataFrame(dataBreastCancer.data, columns=dataBreastCancer.feature_names) df['Target'] = pd.DataFrame(dataBreastCancer.target) print(df.shape) df.head()</pre> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Keluaran | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | <div>(569, 31)</div> <table><tr><th></th><th>mean radius</th><th>mean texture</th><th>mean perimeter</th><th>mean area</th><th>mean smoothness</th><th>mean compactness</th><th>mean concavity</th><th>mean concave points</th><th>mean symmetry</th><th>mean fractal dimension</th><th>...</th><th>worst texture</th><th>worst perimeter</th><th>worst area</th><th>worst smoothness</th><th>worst compactness</th></tr><tr><td>0</td><td>17.99</td><td>10.38</td><td>122.80</td><td>1001.0</td><td>0.11840</td><td>0.27760</td><td>0.3001</td><td>0.14710</td><td>0.2419</td><td>0.07871</td><td>...</td><td>17.33</td><td>184.60</td><td>2019.0</td><td>0.1622</td><td>0.1238</td></tr><tr><td>1</td><td>20.57</td><td>17.77</td><td>132.90</td><td>1326.0</td><td>0.08474</td><td>0.07864</td><td>0.0869</td><td>0.07017</td><td>0.1812</td><td>0.05667</td><td>...</td><td>23.41</td><td>158.80</td><td>1956.0</td><td>0.1238</td><td>0.1444</td></tr><tr><td>2</td><td>19.69</td><td>21.25</td><td>130.00</td><td>1203.0</td><td>0.10960</td><td>0.15990</td><td>0.1974</td><td>0.12790</td><td>0.2069</td><td>0.05999</td><td>...</td><td>25.53</td><td>152.50</td><td>1709.0</td><td>0.1444</td><td>0.2098</td></tr><tr><td>3</td><td>11.42</td><td>20.38</td><td>77.58</td><td>386.1</td><td>0.14250</td><td>0.28390</td><td>0.2414</td><td>0.10520</td><td>0.2597</td><td>0.09744</td><td>...</td><td>26.50</td><td>98.87</td><td>567.7</td><td>0.2098</td><td>0.1374</td></tr><tr><td>4</td><td>20.29</td><td>14.34</td><td>135.10</td><td>1297.0</td><td>0.10030</td><td>0.13280</td><td>0.1980</td><td>0.10430</td><td>0.1809</td><td>0.05883</td><td>...</td><td>16.67</td><td>152.20</td><td>1575.0</td><td>0.1374</td><td>0.1238</td></tr></table> <div>5 rows × 31 columns</div> | | 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 | 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 | 0.1238 | 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 | 0.1444 | 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 | 0.2098 | 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 | 0.1374 | 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 | 0.1238 |
| | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 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 | 0.1238 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 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 | 0.1444 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 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 | 0.2098 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 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 | 0.1374 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 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 | 0.1238 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Mean radius vs mean texture | Masukan | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | <pre>Test 1: Mean Radius vs Mean Texture # Convex hull visualization plt.figure(figsize = (10, 6)) colors = ['aquamarine','lightcoral'] plt.title('Mean Radius vs Mean Texture') plt.xlabel(dataBreastCancer.feature_names[0]) plt.ylabel(dataBreastCancer.feature_names[1]) for i in range(len(dataBreastCancer.target_names)): bucket = df[df['Target'] == i] bucket = bucket.iloc[:,[0,1]].values hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull plt.scatter(bucket[:, 0], bucket[:, 1], label=dataBreastCancer.target_names[i]) for simplex in hull.simplices: plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i]) plt.legend()</pre> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Keluaran | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| |  |
| | Masukan |
| Mean perimeter vs mean area | <div>Test 2: Mean Perimeter vs Mean Area</div> <pre># Convex hull visualization plt.figure(figsize = (10, 6)) colors = ['aquamarine', 'lightcoral'] plt.title('Mean Perimeter vs Mean Area') plt.xlabel(dataBreastCancer.feature_names[2]) plt.ylabel(dataBreastCancer.feature_names[3]) for i in range(len(dataBreastCancer.target_names)): bucket = df[df['Target'] == i] bucket = bucket.iloc[:, [2, 3]].values hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull plt.scatter(bucket[:, 0], bucket[:, 1], label=dataBreastCancer.target_names[i]) for simplex in hull.simplices: plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i]) plt.legend()</pre> |
| | Keluaran |
| |  |

3. Dataset 3: Wine

| Dataframe | Masukan | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|---------|-------------------|-----------|-------------------|------------|----------------------|-----------------|----------------------|-----------------|-----------------|-----|-------------|---|-------|------|------|------|-------|------|------|------|------|------|------|--|---|-------|------|------|------|-------|------|------|------|------|------|------|--|---|-------|------|------|------|-------|------|------|------|------|------|------|--|---|-------|------|------|------|-------|------|------|------|------|------|------|--|---|-------|------|------|------|-------|------|------|------|------|------|------|--|
| e | <pre># Run this cell to create wine DataFrame dataWine = datasets.load_wine() #create a DataFrame df = pd.DataFrame(dataWine.data, columns=dataWine.feature_names) df['Target'] = pd.DataFrame(dataWine.target) print(df.shape) df.head()</pre> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Keluaran | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | <pre>(178, 14)</pre> <table><thead><tr><th></th><th>alcohol</th><th>malic acid</th><th>ash</th><th>alcalinity of ash</th><th>magnesium</th><th>total phenols</th><th>flavanoids</th><th>nonflavanoid phenols</th><th>proanthocyanins</th><th>color intensity</th><th>hue</th><th>od280/od315</th></tr></thead><tbody><tr><td>0</td><td>14.23</td><td>1.71</td><td>2.43</td><td>15.6</td><td>127.0</td><td>2.80</td><td>3.06</td><td>0.28</td><td>2.29</td><td>5.64</td><td>1.04</td><td></td></tr><tr><td>1</td><td>13.20</td><td>1.78</td><td>2.14</td><td>11.2</td><td>100.0</td><td>2.65</td><td>2.76</td><td>0.26</td><td>1.28</td><td>4.38</td><td>1.05</td><td></td></tr><tr><td>2</td><td>13.16</td><td>2.36</td><td>2.67</td><td>18.6</td><td>101.0</td><td>2.80</td><td>3.24</td><td>0.30</td><td>2.81</td><td>5.68</td><td>1.03</td><td></td></tr><tr><td>3</td><td>14.37</td><td>1.95</td><td>2.50</td><td>16.8</td><td>113.0</td><td>3.85</td><td>3.49</td><td>0.24</td><td>2.18</td><td>7.80</td><td>0.86</td><td></td></tr><tr><td>4</td><td>13.24</td><td>2.59</td><td>2.87</td><td>21.0</td><td>118.0</td><td>2.80</td><td>2.69</td><td>0.39</td><td>1.82</td><td>4.32</td><td>1.04</td><td></td></tr></tbody></table> | | alcohol | malic acid | ash | alcalinity of ash | magnesium | total phenols | flavanoids | nonflavanoid phenols | proanthocyanins | color intensity | hue | od280/od315 | 0 | 14.23 | 1.71 | 2.43 | 15.6 | 127.0 | 2.80 | 3.06 | 0.28 | 2.29 | 5.64 | 1.04 | | 1 | 13.20 | 1.78 | 2.14 | 11.2 | 100.0 | 2.65 | 2.76 | 0.26 | 1.28 | 4.38 | 1.05 | | 2 | 13.16 | 2.36 | 2.67 | 18.6 | 101.0 | 2.80 | 3.24 | 0.30 | 2.81 | 5.68 | 1.03 | | 3 | 14.37 | 1.95 | 2.50 | 16.8 | 113.0 | 3.85 | 3.49 | 0.24 | 2.18 | 7.80 | 0.86 | | 4 | 13.24 | 2.59 | 2.87 | 21.0 | 118.0 | 2.80 | 2.69 | 0.39 | 1.82 | 4.32 | 1.04 | |
| | alcohol | malic acid | ash | alcalinity of ash | magnesium | total phenols | flavanoids | nonflavanoid phenols | proanthocyanins | color intensity | hue | od280/od315 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | 14.23 | 1.71 | 2.43 | 15.6 | 127.0 | 2.80 | 3.06 | 0.28 | 2.29 | 5.64 | 1.04 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 13.20 | 1.78 | 2.14 | 11.2 | 100.0 | 2.65 | 2.76 | 0.26 | 1.28 | 4.38 | 1.05 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | 13.16 | 2.36 | 2.67 | 18.6 | 101.0 | 2.80 | 3.24 | 0.30 | 2.81 | 5.68 | 1.03 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | 14.37 | 1.95 | 2.50 | 16.8 | 113.0 | 3.85 | 3.49 | 0.24 | 2.18 | 7.80 | 0.86 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | 13.24 | 2.59 | 2.87 | 21.0 | 118.0 | 2.80 | 2.69 | 0.39 | 1.82 | 4.32 | 1.04 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Masukan | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | <pre>Test 1: Alcohol vs Malic Acid # Convex hull visualization plt.figure(figsize = (10, 6)) colors = ['aquamarine','lightcoral','palegreen'] plt.title('Alcohol vs Malic Acid') plt.xlabel(dataWine.feature_names[0]) plt.ylabel(dataWine.feature_names[1]) for i in range(len(dataWine.target_names)): bucket = df[df['Target'] == i] bucket = bucket.iloc[:,[0,1]].values hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer plt.scatter(bucket[:, 0], bucket[:, 1], label=dataWine.target_names[i]) for simplex in hull.simplices: plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i]) plt.legend()</pre> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Alcohol vs malic acid | Keluaran | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Masukan | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |



D. Alamat Source Code Program

<https://github.com/addinnabilal/Convex-Hull>

E. Checklist

| Poin | Ya | Tidak |
|---------------------------------------------------------------------------------------------------------------|----|-------|
| 1. Pustaka myConvexHull berhasil dibuat dan tidak ada kesalahan | ✓ | |
| 2. Convex hull yang dihasilkan sudah benar | ✓ | |
| 3. Pustaka myConvexHull dapat digunakan untuk menampilkan convex hull setiap label dengan warna yang berbeda. | ✓ | |
| 4. Bonus: program dapat menerima input dan | ✓ | |

| | | |
|------------------------------------------|--|--|
| menuliskan output untuk dataset lainnya. | | |
|------------------------------------------|--|--|

F. Referensi

1. Munir, Rinaldi. 2022. Algoritma Divide and Conquer Bagian 4. Diakses pada 28 Februari 2022 dari [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-\(2022\)-Bagian4.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf)
2. Muthukrishnan. 2019. Using the law of cosines and vector dot product formula to find the angle between three points. Diakses pada 28 Februari 2022 dari <https://muthu.co/using-the-law-of-cosines-and-vector-dot-product-formula-to-find-the-angle-between-three-points/>