

**LAPORAN TUGAS KECIL 2**  
**IF2211 STRATEGI ALGORITMA**

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



**Disusun oleh:**

Monica Adelia      13520096

**Program Studi Teknik Informatika**  
**Sekolah Teknik Elektro dan Informatika**  
**Institut Teknologi Bandung**  
**2022**

## 1. PENJELASAN ALGORITMA *DIVIDE AND CONQUER*

Algoritma *Divide and Conquer* adalah metode pemecahan masalah yang bekerja dengan membagi masalah menjadi beberapa upa-masalah yang lebih kecil. Kemudian tiap upa-masalah tersebut diselesaikan secara *independent* dan akhirnya solusi masing-masing upa-masalah digabungkan sehingga menjadi solusi dari masalah semula. *Divide* artinya membagi persoalan menjadi beberapa upa-masalah yang memiliki kemiripan dengan persoalan semula, namun berukuran lebih kecil. Idealnya berukuran hampir sama. Lalu, *conquer* adalah menyelesaikan masing-masing upa-masalah, baik secara langsung maupun secara rekursif. Langkah-langkah umum algoritma *divide and conquer* adalah sebagai berikut:

1. *Divide*

Permasalahan dibagi menjadi beberapa upa-masalah (berukuran lebih kecil) yang memiliki kemiripan dengan masalah semula.

2. *Conquer*

Permasalahan tiap-tiap upa, akan diselesaikan atau dipecahkan.

3. *Combine*

Setiap solusi yang masing-masing upa-masalah digabung sehingga membentuk solusi masalah semula.

Berikut ini adalah pseudocode dari algoritma *Divide and Conquer* dalam menyelesaikan permasalahan atau persoalan.

```
procedure DIVIDEandCONQUER(input  $P$  : problem,  $n$  : integer)
{ Menyelesaikan persoalan dengan algoritma divide and conquer
  Masukan: masukan yang berukuran  $n$ 
  Luaran: solusi dari persoalan semula
}
Deklarasi
   $r$  : integer

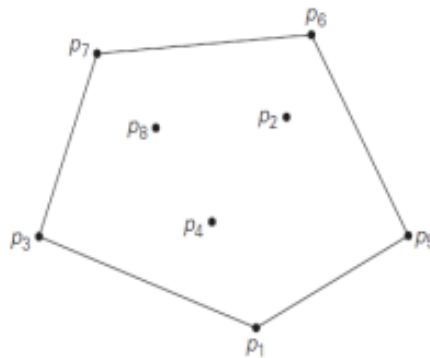
Algoritma
if  $n \leq n_0$  then {ukuran persoalan sudah cukup kecil}
  SOLVE persoalan  $P$  yang berukuran  $n$  ini
else
  DIVIDE menjadi 2 upa-persoalan,  $P_1$  dan  $P_2$ , masing-masing berukuran  $n/2$ 
  DIVIDEandCONQUER( $P_1$ ,  $n/2$ )
  DIVIDEandCONQUER( $P_2$ ,  $n/2$ )
  COMBINE solusi dari  $P_1$  dan  $P_2$ 
endif
```

Gambar 1 Pseudocode Algoritma Divide and Conquer

Sumber: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-\(2021\)-Bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf)

Algoritma *Divide and Conquer* merupakan salah satu solusi dalam penyelesaian masalah *convex hull*. Algoritma ini memiliki kompleksitas waktu yang cukup kecil dan efektif dalam menyelesaikan permasalahan ini, jika dibandingkan dengan algoritma lain. *Convex hull* merupakan persoalan klasik dalam geometri komputasional. *Convex hull* adalah poligon yang disusun dari subset titik sedemikian rupa sehingga tidak ada titik dari himpunan awal yang berada di luar poligon tersebut (semua titik berada di batas luar atau di dalam area yang dilingkupi oleh poligon tersebut).

Himpunan titik pada bidang planar disebut *convex* jika untuk sembarang dua titik pada bidang tersebut (misal  $p$  dan  $q$ ), seluruh segmen garis yang berakhir di  $p$  dan  $q$  berada pada himpunan tersebut. Untuk dua titik, maka *convex hull* berupa garis yang menghubungkan dua titik tersebut. Untuk tiga titik yang terletak pada satu garis, maka *convex hull* adalah sebuah garis yang menghubungkan dua titik terjauh. Sedangkan *convex hull* untuk tiga titik yang tidak terletak pada satu garis adalah sebuah segitiga yang menghubungkan ketiga titik tersebut. Untuk titik yang lebih banyak dan tidak terletak pada satu garis, maka *convex hull* berupa poligon *convex* dengan sisi berupa garis yang menghubungkan beberapa titik pada  $S$ .



Gambar 2 Convex Hull untuk 8 Buah Titik

Sumber: [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)

*Convex hull* dapat diselesaikan dengan *Divide and Conquer* dengan langkah-langkah sebagai berikut:

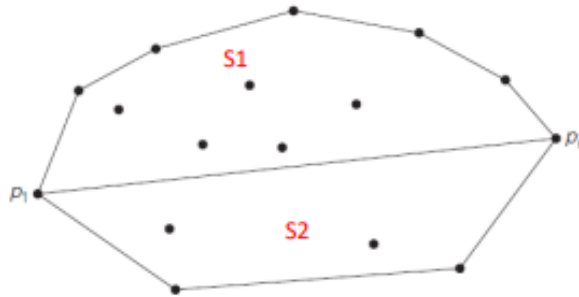
1. Mencari dua titik ekstrim yang akan membentuk *convex hull*.

Pencarian ini dengan memanfaatkan algoritma *quicksort*. Himpunan titik diurutkan berdasarkan nilai absis yang menaik dan jika ada nilai absis yang sama,

maka diurutkan dengan nilai ordinat yang menaik. Titik pertama ( $P_1$ ) dan titik terakhir ( $P_n$ ) setelah diurutkan akan menjadi titik ekstrim.

2. Membagi himpunan titik menjadi dua bagian.

Garis yang menghubungkan  $P_1$  dan  $P_n$  ( $P_1P_n$ ) membagi himpunan titik menjadi dua bagian yaitu  $s_1$  dan  $s_2$ .  $S_1$  adalah kumpulan titik di sebelah kiri atau atas garis  $P_1P_n$ , sedangkan  $s_2$  adalah kumpulan titik di sebelah kanan atau bawah garis  $P_1P_n$ . Untuk penentuan kiri atau kanan titik dari sebuah garis dapat memanfaatkan determinan. Determinan akan bernilai positif jika titik berada di sebelah kiri garis. Titik pada himpunan yang berada pada garis  $P_1P_n$  tidak mungkin membentuk *convex hull* sehingga diabaikan (tidak masuk  $s_1$  maupun  $s_2$ ).



Gambar 3 Membagi himpunan titik menjadi dua bagian

Sumber: [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)

Kumpulan titik pada  $s_1$  bisa membentuk *convex hull* bagian atas dan  $s_2$  membentuk *convex hull* bagian bawah. Lalu terapkan *Divide and Conquer*.

3. Pencarian titik *convex hull*

Untuk sebuah bagian, misal  $s_1$ , terdapat dua kemungkinan *convex hull*:

- Jika  $s_1$  merupakan himpunan kosong, maka titik  $P_1$  dan  $P_n$  menjadi pembentuk *convex hull* bagian  $s_1$
- Jika  $s_1$  bukan merupakan himpunan kosong, cari titik yang memiliki jarak terjauh dari garis  $P_1P_n$ , misal  $P_{max}$ .

4. Penentuan bagian di sebelah kiri garis

Tentukan kumpulan titik yang berada di sebelah kiri garis  $P_1P_{max}$  (menjadi bagian  $S_{1,1}$ ), dan di sebelah kanan garis  $P_{max}P_n$  (menjadi bagian  $S_{1,2}$ ). Semua titik yang berada di dalam segitiga  $P_{max}P_1P_n$  akan diabaikan dalam pemeriksaan lebih lanjut.

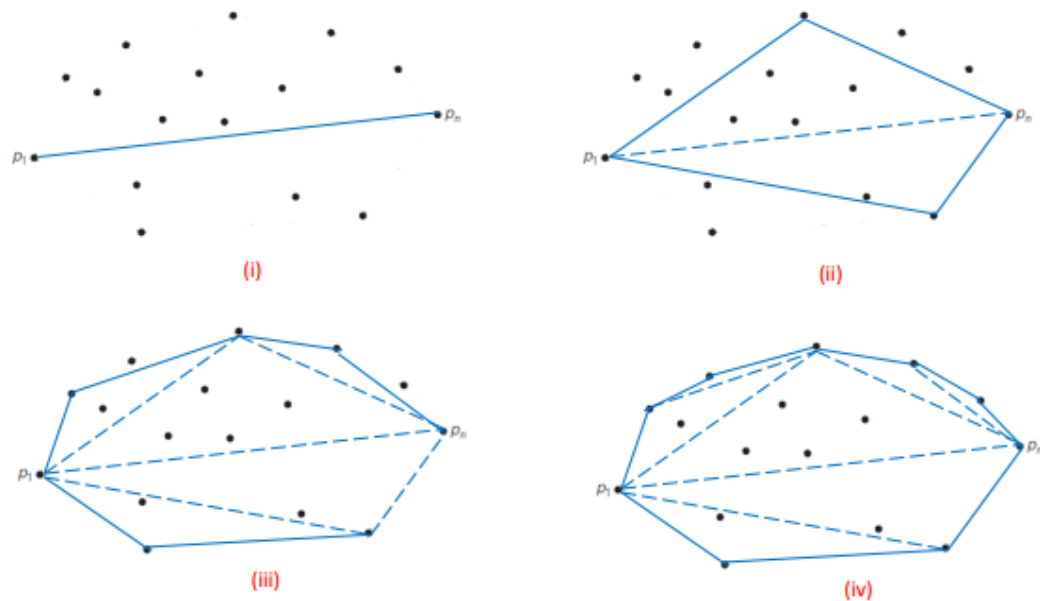
5. Rekursif

Lakukan pencarian titik *convex hull* seterusnya ke bagian yang lebih kecil lagi hingga himpunan titik kosong semua. Lakukan juga pada bagian S2 hingga bagian kiri dan kanan kosong.

6. Pengembalian pasangan titik yang dihasilkan

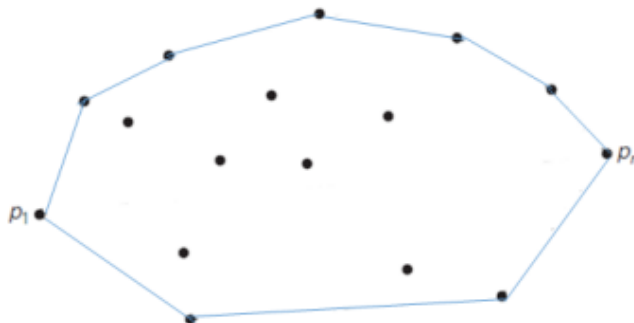
Pengembalian pasangan titik yang dihasilkan dilakukan secara rekursif dari.

Langkah-langkah dari penyelesaian *convex hull* menggunakan algoritma *divide and conquer* dapat dilihat dari gambar di bawah ini



Gambar 4 Langkah Penerapan Algoritma *Divide and Conquer* dalam Mencari *Convex Hull*

Sumber: [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)



Gambar 5 Hasil *Convex Hull* Menggunakan Algoritma *Divide and Conquer*

Sumber: [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)

Berikut ini adalah pseudocode penyelesaian *convex hull* menggunakan algoritma *divide and conquer*

```
Algorithm ConvexHull(P)
// P is a set of input points

Sort all the points in P and find two extreme points A and B
S1 ← Set of points right to the line AB
S2 ← Set of points right to the line BA
Solution ← AB followed by BA

Call FindHull(S1, A, B)
Call FindHull(S2, B, A)
```

```
Algorithm FindHull(P, A, B)

if isEmpty(P) then
    return
else
    C ← Orthogonally farthest point from AB
    Solution ← Replace AB by AC followed by CB
    Partition P - { C } in X0, X1 and X2
    Discard X0 in side triangle

    Call FindHull(X1, A, C)
    Call FindHull(X2, C, B)
end
```

Gambar 6 Pseudocode penyelesaian *convex hull* menggunakan algoritma *divide and conquer*

Sumber: <https://codecrucks.com/convex-hull-using-divide-and-conquer/>

## 2. SOURCE CODE PROGRAM

### 2.1. Program myConvexHull

```
1 import numpy as np
2
3 #my own convex hull
4 def myConvexHull(bucket):
5     solution = np.array([[0,0,0,0]])
6     sort = sorted(bucket, key=lambda x: (x[0], x[1]))
7
8     #create p1 dan pn
9     solution[0] = sort[0]
10    solution = np.vstack((solution, sort[len(sort)-1]))
11    hullSim = np.array([[5, 5]])
12    hullSim = np.delete(hullSim, 0, axis=0)
13    s1 = np.array([[1.0, 2.0]])
14    s1 = np.delete(s1, 0, axis=0)
15    s2 = np.array([[1.0, 2.0]])
16    s2 = np.delete(s2, 0, axis=0)
17
18    #membagi menjadi s1 dan s2
19    for i in range(len(bucket)):
20        if((bucket[i,0] != solution[0,0] or bucket[i,1] != solution[0,1]) or (bucket[i,0] != solution[1,0] or bucket[i,1] != solution[1,1])):
21            if((findDeter(solution[0,0], solution[0,1], solution[1,0], solution[1,1], bucket[i,0], bucket[i,1]) > 0)):
22                if((abs(findDeter(solution[0,0], solution[0,1], solution[1,0], solution[1,1], bucket[i,0], bucket[i,1]))) > 1e-12):
23                    s1 = np.vstack((s1, bucket[i]))
24                else:
25                    if((abs(findDeter(solution[0,0], solution[0,1], solution[1,0], solution[1,1], bucket[i,0], bucket[i,1]))) > 1e-12):
26                        s2 = np.vstack((s2, bucket[i]))
27
28    hullSim = np.append(hullSim, np.array(findHull(bucket, hullSim, solution, s1, solution[0], solution[1])), axis=0)
29    hullSim = np.append(hullSim, np.array(findHull(bucket, hullSim, solution, s2, solution[1], solution[0])), axis=0)
30
31    return hullSim
32
33 def findHull(bucket, hullSim, solution, S, A, B):
34     if(S.size == 0):
35         #base case
36         temp = np.array([[0, 0]])
37         temp[0][0] = findIndex(bucket, A[0], A[1])
38         temp[0][1] = findIndex(bucket, B[0], B[1])
39         return temp
40     else:
41         solution = np.array([[1.0, 2.0]])
42         solution = np.delete(solution, 0, axis=0)
43         farthest = S[0]
44         tempDistance = 0.00
45         # Find Orthogonally farthest point from AB
46         for i in range(len(S)):
47             d = np.linalg.norm(np.cross(B-A, A-S[i]))/np.linalg.norm(B-A)
48
49             if(d > tempDistance):
50                 tempDistance = d
51                 farthest = S[i]
52
53         x1 = np.array([[1.0, 2.0]])
54         x1 = np.delete(x1, 0, axis=0)
55         x2 = np.array([[1.0, 2.0]])
56         x2 = np.delete(x2, 0, axis=0)
57
58         #membuang titik dalam segitiga & membagi x1 dan x2
59         for i in range(len(S)):
60             if (pointInTriangle(A[0], A[1], B[0], B[1], farthest[0], farthest[1], S[i,0], S[i,1]) == False):
61                 if(findDeter(A[0], A[1], farthest[0], farthest[1], S[i,0], S[i,1]) > 0):
62                     x1 = np.vstack((x1, S[i]))
63                 else:
64                     x2 = np.vstack((x2, S[i]))
65
66         #rekursif
67         solution = np.append(solution, np.array(findHull(bucket, hullSim, solution, x1, A, farthest)), axis=0)
68         solution = np.append(solution, np.array(findHull(bucket, hullSim, solution, x2, farthest, B)), axis=0)
69
70     return solution
71
72 def pointInTriangle(x1, y1, x2, y2, x3, y3, x, y):
73     #output true untuk titik yang berada dalam segitiga
74     denominator = ((y2 - y3)*(x1 - x3) + (x3 - x2)*(y1 - y3))
75     a = ((y2 - y3)*(x - x3) + (x3 - x2)*(y - y3)) / denominator
76     b = ((y3 - y1)*(x - x3) + (x1 - x3)*(y - y3)) / denominator
77     c = 1 - a - b
78     return 0 <= a and a <= 1 and 0 <= b and b <= 1 and 0 <= c and c <= 1
79
80 def findDeter(a,b,c,d,x,y):
81     # output determinan dari matriks buatan
82     matriks = np.array([[a, b, 1], [c, d, 1], [0,0,1]])
83     matriks[2,0] = x
84     matriks[2,1] = y
85     d = np.linalg.det(matriks)
86     return d
87
88 def findIndex(bucket, x, y):
89     #mencari indeks dari titik untuk membuat garis
90     for i in range(len(bucket)):
91         if(bucket[i][0] == x and bucket[i][1] == y):
92             return i
```

## 2.2. Program membuat dataframe

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 from sklearn import datasets
4
5 data = datasets.load_iris()      #ubah dataset sesuai dataset yang diinginkan
6
7 df = pd.DataFrame(data.data, columns=data.feature_names)
8 df['Target'] = pd.DataFrame(data.target)
9 df.head()
```

## 2.3. Program visualisasi convexhull

```
1 #visualisasi hasil ConvexHull dataset iris: Sepal length vs sepal width
2 plt.figure(figsize = (10, 6))
3 atribut1 = 0      #ubah nilai atribut2 dengan atribut yang diinginkan
4 atribut2 = 1      #ubah nilai atribut2 dengan atribut yang diinginkan
5 colors = ['b','r','g']
6 plt.title(data.feature_names[atribut1] + ' vs ' + data.feature_names[atribut2])
7 plt.xlabel(data.feature_names[atribut1])
8 plt.ylabel(data.feature_names[atribut2])
9 for i in range(len(data.target_names)):
10     bucket = df[df['Target'] == i]
11     bucket = bucket.iloc[:,[atribut1,atribut2]].values
12     hull = myConvexHull(bucket)
13     hull = hull.astype(int)
14     plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
15     for simplex in hull:
16         plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
17 plt.legend()
```

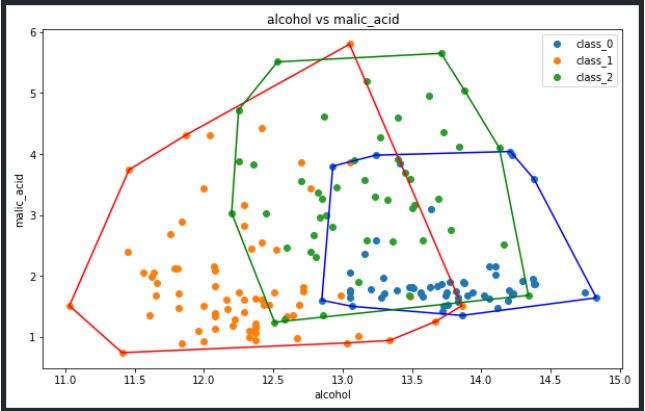
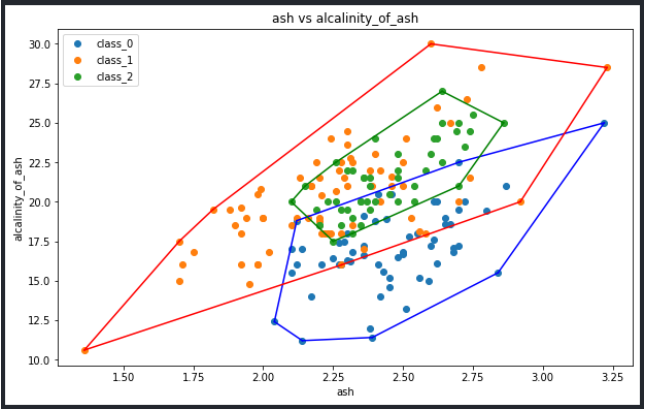


### 3. SCREENSHOT INPUT DAN OUTPUT

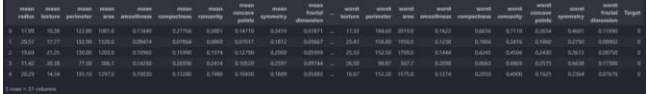
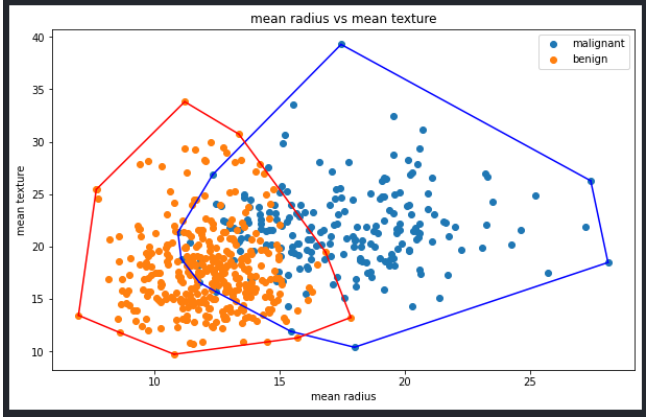
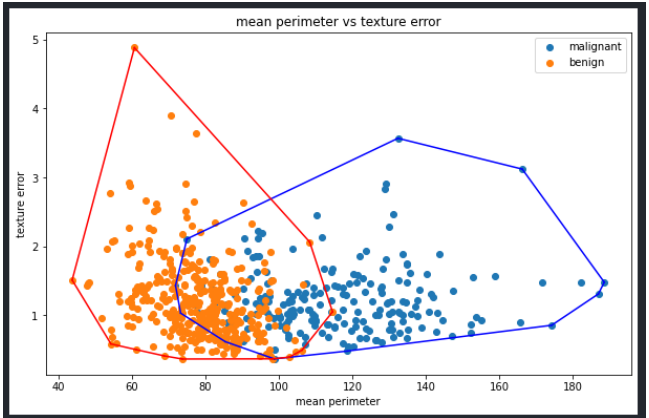
#### 3.1. Hasil Uji Dataset Iris

Dataframe Iris																																					
Input	Output																																				
<pre>#dataset iris import pandas as pd import matplotlib.pyplot as plt from sklearn import datasets  data = datasets.load_iris()  df = pd.DataFrame(data.data, columns=data.feature_names) df['Target'] = pd.DataFrame(data.target) df.head()</pre> <div>0.6s</div>	<table><thead><tr><th></th><th>sepal length (cm)</th><th>sepal width (cm)</th><th>petal length (cm)</th><th>petal width (cm)</th><th>Target</th></tr></thead><tbody><tr><td>0</td><td>5.1</td><td>3.5</td><td>1.4</td><td>0.2</td><td>0</td></tr><tr><td>1</td><td>4.9</td><td>3.0</td><td>1.4</td><td>0.2</td><td>0</td></tr><tr><td>2</td><td>4.7</td><td>3.2</td><td>1.3</td><td>0.2</td><td>0</td></tr><tr><td>3</td><td>4.6</td><td>3.1</td><td>1.5</td><td>0.2</td><td>0</td></tr><tr><td>4</td><td>5.0</td><td>3.6</td><td>1.4</td><td>0.2</td><td>0</td></tr></tbody></table>		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
	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																																
Sepal Length vs Sepal Width																																					
Input	Output																																				
<pre>#visualisasi hasil ConvexHull dataset iris: Sepal length vs sepal width plt.figure(figsize = (10, 6)) atribut1 = 0 #ubah nilai atribut2 dengan atribut yang diinginkan atribut2 = 1 #ubah nilai atribut2 dengan atribut yang diinginkan colors = ['b','r','g'] plt.title(data.feature_names[atribut1] + ' vs ' + data.feature_names[atribut2]) plt.xlabel(data.feature_names[atribut1]) plt.ylabel(data.feature_names[atribut2]) for i in range(len(data.target_names)):     bucket = df[df['Target'] == i]     bucket = bucket.iloc[:,[atribut1,atribut2]].values     hull = myConvexHull(bucket)     hull = hull.astype(int)     plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])     for simplex in hull:         plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i]) plt.legend()</pre> <div>0.2s</div>																																					
Petal Length vs Petal Width																																					
Input	Output																																				
<pre>#visualisasi hasil ConvexHull dataset iris: petal length vs petal width plt.figure(figsize = (10, 6)) atribut1 = 2 #ubah nilai atribut2 dengan atribut yang diinginkan atribut2 = 3 #ubah nilai atribut2 dengan atribut yang diinginkan colors = ['b','r','g'] plt.title(data.feature_names[atribut1] + ' vs ' + data.feature_names[atribut2]) plt.xlabel(data.feature_names[atribut1]) plt.ylabel(data.feature_names[atribut2]) for i in range(len(data.target_names)):     bucket = df[df['Target'] == i]     bucket = bucket.iloc[:,[atribut1,atribut2]].values     hull = myConvexHull(bucket)     hull = hull.astype(int)     plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])     for simplex in hull:         plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i]) plt.legend()</pre> <div>0.2s</div>																																					

### 3.2. Hasil Uji Dataset Wine

Dataframe Wine	
Input	Output
<pre>#dataset wine import pandas as pd import matplotlib.pyplot as plt from sklearn import datasets  data = datasets.load_wine()  df = pd.DataFrame(data.data, columns=data.feature_names) df['Target'] = pd.DataFrame(data.target) df.head()</pre> <p>✓ 0.6s</p>	
alcohol vs malic_acid	
Input	Output
<pre>#visualisasi hasil ConvexHull dataset wine: Malic Acid vs Alcohol plt.figure(figsize = (10, 6)) atribut1 = 0 #ubah nilai atribut2 dengan atribut yang diinginkan atribut2 = 1 #ubah nilai atribut2 dengan atribut yang diinginkan colors = ['b','r','g'] plt.title(data.feature_names[atribut1] + ' vs ' + data.feature_names[atribut2]) plt.xlabel(data.feature_names[atribut1]) plt.ylabel(data.feature_names[atribut2]) for i in range(len(data.target_names)):     bucket = df[df['Target'] == i]     bucket = bucket.iloc[:,[atribut1,atribut2]].values     hull = myConvexHull(bucket)     hull = hull.astype(int)     plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])     for simplex in hull:         plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i]) plt.legend()</pre> <p>✓ 0.2s</p>	
Ash vs alkalinity_of_ash	
Input	Output
<pre>#visualisasi hasil ConvexHull dataset wine: Ash vs Alkalinity of Ash plt.figure(figsize = (10, 6)) atribut1 = 2 #ubah nilai atribut2 dengan atribut yang diinginkan atribut2 = 3 #ubah nilai atribut2 dengan atribut yang diinginkan colors = ['b','r','g'] plt.title(data.feature_names[atribut1] + ' vs ' + data.feature_names[atribut2]) plt.xlabel(data.feature_names[atribut1]) plt.ylabel(data.feature_names[atribut2]) for i in range(len(data.target_names)):     bucket = df[df['Target'] == i]     bucket = bucket.iloc[:,[atribut1,atribut2]].values     hull = myConvexHull(bucket)     hull = hull.astype(int)     plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])     for simplex in hull:         plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i]) plt.legend()</pre> <p>✓ 0.2s</p>	

### 3.3. Hasil Uji Dataset Breast Cancer

Dataframe Breast Cancer	
Input	Output
<pre> #dataset breast_cancer import pandas as pd import matplotlib.pyplot as plt from sklearn import datasets  data = datasets.load_breast_cancer()  df = pd.DataFrame(data.data, columns=data.feature_names) df['Target'] = pd.DataFrame(data.target) df.head() </pre> <p>✓ 0.6s</p>	
mean radius vs mean texture	
Input	Output
<pre> #visualisasi hasil ConvexHull dataset breast cancer: mean radius vs mean texture plt.figure(figsize = (10, 6)) atribut1 = 0 #ubah nilai atribut1 dengan atribut yang diinginkan atribut2 = 1 #ubah nilai atribut2 dengan atribut yang diinginkan colors = ['b','r','g'] plt.title(data.feature_names[atribut1] + ' vs ' + data.feature_names[atribut2]) plt.xlabel(data.feature_names[atribut1]) plt.ylabel(data.feature_names[atribut2]) for i in range(len(data.target_names)):     bucket = df[df['Target'] == i]     bucket = bucket.iloc[:,[atribut1,atribut2]].values     hull = myConvexHull(bucket)     hull = hull.astype(int)     plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])     for simplex in hull:         plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i]) plt.legend() </pre> <p>✓ 0.2s</p>	
Mean perimeter vs texture error	
Input	Output
<pre> #visualisasi hasil ConvexHull dataset breast cancer: mean area vs mean compactness plt.figure(figsize = (10, 6)) atribut1 = 2 #ubah nilai atribut1 dengan atribut yang diinginkan atribut2 = 11 #ubah nilai atribut2 dengan atribut yang diinginkan colors = ['b','r','g'] plt.title(data.feature_names[atribut1] + ' vs ' + data.feature_names[atribut2]) plt.xlabel(data.feature_names[atribut1]) plt.ylabel(data.feature_names[atribut2]) for i in range(len(data.target_names)):     bucket = df[df['Target'] == i]     bucket = bucket.iloc[:,[atribut1,atribut2]].values     hull = myConvexHull(bucket)     hull = hull.astype(int)     plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])     for simplex in hull:         plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i]) plt.legend() </pre> <p>✓ 0.2s</p>	

#### 4. ALAMAT DRIVE KODE PROGRAM

<https://github.com/adeliaaaa/Tucil2-STIMA.git>

#### 5. CEKLIST

Poin	Ya	Tidak
1. Pustaka <i>myCovexHull</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. <b>Bonus:</b> program dapat menerima input dan menuliskan output untuk data set lainnya.	√	