

**Laporan Tugas Kecil IF2211 Strategi Algoritma
Implementasi *Convex Hull* untuk Visualisasi Tes Linear
Separability Dataset dengan Algoritma *Divide and Conquer***

Disusun oleh
Samuel Christopher - 13520075



Teknik Informatika

Institut Teknologi Bandung

Bandung

2022

I. Algoritma *Divide and Conquer*

Algoritma Divide and Conquer merupakan algoritma yang sangat populer di dunia Ilmu Komputer. *Divide and Conquer* merupakan algoritma yang berprinsip memecah-mecah permasalahan yang terlalu besar menjadi beberapa bagian kecil sehingga lebih mudah untuk diselesaikan. Langkah-langkah umum algoritma Divide and Conquer :

- Divide : Membagi masalah menjadi beberapa upa-masalah yang memiliki kemiripan dengan masalah semula namun berukuran lebih kecil(idealnya berukuran hampir sama).
- Conquer : Memecahkan(menyelesaikan) masing-masing upa-masalah(secara rekursif).
- Combine : Menggabungkan solusi masing-masing upa-masalah sehingga membentuk solusi masalah semula.

Objek masalah yang dibagi adalah masukan (input) atau instances yang berukuran n: tabel(larik), matriks, dan sebagainya, bergantung pada masalahnya. Tiap-tiap upa-masalah mempunyai karakteristik yang sama (the same type) dengan karakteristik masalah asal, sehingga metode Divide and Conquer lebih natural diungkapkan dalam skema rekursif. Pada tugas kecil kali ini, kita akan memanfaatkan algoritma *Divide and Conquer* untuk mencari sebuah visualisasi dari *Convex Hull*.

Garis besar program:

1. Pengguna memasukkan masukan berupa pilihan dataset yang ingin divisualisasikan
2. Pengguna memasukkan masukan berupa pilihan kolom dari dataset yang nantinya akan divisualisasikan
3. Program akan memanggil dan mengkonversi kolom menjadi *list dataset* x dan y
4. Program akan memanggil fungsi *myConvexHull* yang menerima masukan berupa *list* dua dimensi yang berisi data x dan y
5. Fungsi *myConvexHull* akan mengembalikan keluaran berupa *list* titik yang akan membuat *convex hull*
6. Program akan memvisualisasikan *List* titik pembuat *convex hull* dengan memanfaatkan library *matplotlib*
7. Program berhasil memvisualisasikan *convex hull* pada dataset yang sudah dipilih pengguna

Garis besar algoritma:

1. Menerima masukan berupa sebuah dataset yang dapat diimplementasikan pada ruang Euclidean berderajat 2
2. Mencari titik paling kiri dan paling kanan dari dataset tersebut
3. Membuat garis dari titik paling kiri dan paling kanan dari dataset tersebut
4. Memecah dataset menjadi 2 bagian yaitu di atas garis dan di bawah garis
5. Setelah memecah dataset menjadi 2 bagian, maka akan melakukan rekursi untuk tiap bagian (bagian atas akan mencari titik terjauh di atas garis, dan bagian bawah akan mencari titik terjauh di bawah garis).

Garis besar rekursi:

1. Memasukkan dataset dan menentukan bagian yang akan dicari (atas atau bawah garis)
2. Menentukan garis dari titik paling kiri dan paling kanan dari dataset lalu dimasukan kedalam variabel *leftmost* dan *rightmost*
3. Membuat garis antara *leftmost* dan *rightmost* dan dimasukkan ke dalam variable *line*
4. Mencari titik terjauh dari *line* dari dataset
5. Jika sudah tidak ada lagi titik terjauh dari sebuah garis (di atas maupun di bawah), maka artinya garis tersebut adalah sebuah garis yang akan membentuk convex hull
6. Ulangi tahap dari 1-5 dengan parameter:
 - a. Rekursi untuk bagian atas pertama
Leftmost <- *leftmost*
Rightmost <- *furthestPointAbove*
Points <- dataset
Flag <- True
 - b. Rekursi untuk bagian atas kedua
Leftmost <- *furthestPointAbove*
Rightmost <- *rightmost*
Points <- dataset
Flag <- True
 - c. Rekursi untuk bagian bawah pertama
Leftmost <- *furthestPointAbove*
Rightmost <- *rightmost*
Points <- dataset
Flag <- False
 - d. Rekursi untuk bagian bawah kedua
Leftmost <- *furthestPointAbove*
Rightmost <- *rightmost*
Points <- dataset
Flag <- False
7. Setelah mencari seluruh garis yang membentuk convex hull, maka garis-garis tersebut akan dimasukkan ke dalam array lalu akan divisualisasikan menggunakan matplotlib

II. Kode program

1. *Import semua dependencies*

Prerequisite

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
from sklearn import datasets
import matplotlib.pyplot as plt
```

Python

2. Memilih *dataset*

Choose dataset

```
listDataset = [["iris", datasets.load_iris()], ["wine", datasets.load_wine()], ["breast cancer", datasets.load_breast_cancer()]]

print("List of dataset: ")
for i in range(len(listDataset)):
    print(f"{i+1} : {listDataset[i][0]}")

choice = int(input("Choose your dataset: "))

while choice > len(listDataset) or choice < 1:
    print("please choose your dataset correctly")
    choice = int(input("Choose your dataset: "))

print("You have selected dataset:", listDataset[choice-1][0])

data = listDataset[choice-1][1]
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
```

Python

3. Memilih 2 kolom untuk diklasifikasi

Choose column to be classified to E² Plane

```
print("Choose two columns to be classified:")
for i,v in enumerate(df.columns[0:len(df.columns)-1]):
    print(f"{i+1} : {v}")

firstColumnChoice = int(input("Choose first column: "))

while firstColumnChoice > len(df.columns)-1 or firstColumnChoice < 1:
    print("please choose your first column correctly")
    firstColumnChoice = int(input("Choose first column: "))

secondColumnChoice = int(input("Choose second column: "))

while secondColumnChoice > len(df.columns)-1 or secondColumnChoice < 1 or secondColumnChoice == firstColumnChoice:
    print("please choose your second column correctly")
    secondColumnChoice = int(input("Choose second column: "))

print("You have chosen first column:", df.columns[firstColumnChoice-1], "and second column:", df.columns[secondColumnChoice-1])
```

Python

4. Fungsi pembantu

```
1 # mencari titik terjauh dari line, dengan tipe "above" atau "below", yang berarti di atas garis atau di bawah garis
2 def searchForTheFurthestPointOf(type, line, points):
3
4     # index titik terjauh pada points (2d array yang berisi kumpulan semua data)
5     furthestPoint = 0
6
7     # menghitung jarak terjauh dari titik ke garis
8     furthestDistance = -9999999
9
10    # untuk memenuhi parameter yang dibutuhkan oleh fungsi searchForArray
11    lines = line
12
13    # untuk mempermudah dalam pencarian x1, x2, y1, dan y2
14    # yang nantinya akan digunakan untuk menghitung jarak terjauh
15    line = [points[line[0]], points[line[1]]]
16    x1 = line[0][0]
17    x2 = line[1][0]
18    y1 = line[0][1]
19    y2 = line[1][1]
20    a = y2 - y1
21    b = x1 - x2
22    c = x2*y1 - x1*y2
23
24    if type == "Above":
25        # mencari seluruh titik yang berada di atas garis
26        arrayAbove = searchForArray("Above", lines, points)
27
28        # jika tidak ada titik yang berada di atas garis, maka return None
29        if arrayAbove == []:
30            return None
31
32        # jika ada, maka mencari titik dengan jarak terjauh dari garis
33        for i in arrayAbove:
34            x = points[i][0]
35            y = points[i][1]
36            distance = abs(a*x + b*y + c) / math.sqrt(a**2 + b**2)
37            if distance > furthestDistance:
38                furthestDistance = distance
39                furthestPoint = i
40
41    if type == "Below":
42        # mencari seluruh titik yang berada di bawah garis
43        arrayBelow = searchForArray("Below", lines, points)
44
45        # jika tidak ada titik yang berada di bawah garis, maka return None
46        if arrayBelow == []:
47            return None
48
49        # jika ada, maka mencari titik dengan jarak terjauh dari garis
50        for i in arrayBelow:
51            x = points[i][0]
52            y = points[i][1]
53            distance = abs(a*x + b*y + c) / math.sqrt(a**2 + b**2)
54            if distance > furthestDistance:
55                furthestDistance = distance
56                furthestPoint = i
57
58    return furthestPoint
59
```


5. Fungsi utama pembentuk *convex hull*

```
1 # rekursi convexhull
2 def dngConvexHull(points, leftmost, rightmost, flag):
3     # list of hull points
4     hull = []
5
6     # garis yang terbuat dari titik paling kiri dan kanan dari set of points yang diberikan
7     line = [leftmost, rightmost]
8
9     # untuk menentukan orientasi points yang ingin dipilih
10    # apakah di atas atau di bawah garis
11    # flag == True berarti di atas garis
12
13    if flag:
14        # mencari titik terjauh di atas garis
15        furthestPointAbove = searchForTheFurthestPointOf("Above", line, points)
16
17        # jika tidak ada titik terjauh di atas garis, maka return titik paling kiri dan paling kanan pembentuk garis
18        if furthestPointAbove == None:
19            return [leftmost, rightmost]
20
21        # return convexhull dari titik kiri pembuat garis dan titik terjauh dari garis dan
22        # titik terjauh di atas garis dan titik kanan dari pembuat garis
23        return dngConvexHull(points, leftmost, furthestPointAbove, True) + dngConvexHull(points, furthestPointAbove, rightmost, True)
24
25    else:
26        # mencari titik terjauh di atas garis
27        furthestPointBelow = searchForTheFurthestPointOf("Below", line, points)
28
29        # jika tidak ada titik terjauh di bawah garis, maka return titik paling kiri dan paling kanan pembentuk garis
30        if furthestPointBelow == None:
31            return [leftmost, rightmost]
32
33        # return convexhull dari titik kiri pembuat garis dan titik terjauh dari garis dan
34        # titik terjauh di atas garis dan titik kanan dari pembuat garis
35        return dngConvexHull(points, leftmost, furthestPointBelow, False) + dngConvexHull(points, furthestPointBelow, rightmost, False)
36
37    # menambahkan hasil kedalam hull untuk di return
38    hull.append(leftmost)
39    hull.append(rightmost)
40    return hull
41
42
43 def myConvexHull(points):
44     hull = []
45     mn = [float('inf'), 0]
46     mx = [-999999, 0]
47
48     # mencari titik paling kiri dan kanan yang nantinya akan digunakan untuk menjadi garis
49     # yang akan dimanfaatkan untuk titik2 yang berada di atas dan di bawah garis
50     leftmost = 0
51     rightmost = 0
52     for i in range(len(buckets)):
53         if buckets[i][0] < mn[0]:
54             mn[0] = buckets[i][0]
55             mn[1] = buckets[i][1]
56             leftmost = i
57         if buckets[i][0] > mx[0]:
58             mx[0] = buckets[i][0]
59             mx[1] = buckets[i][1]
60             rightmost = i
61
62     # mencari seluruh convexhull yang berada di atas garis yang dibuat oleh leftmost dan rightmost
63     up = dngConvexHull(points, leftmost, rightmost, True)
64
65     # mencari seluruh convexhull yang berada di bawah garis yang dibuat oleh leftmost dan rightmost
66     down = dngConvexHull(points, leftmost, rightmost, False)
67
68     # keduanya akan ditambahkan kedalam hull dan akan di return
69     # mengapa down harus dibalik?
70     # karena orientasi dari down adalah kiri ke kanan, dan orientasi dari up adalah kiri ke kanan juga
71     # maka akan terjadi kesalahan yaitu terhubungnya titik terjauh paling kiri dan titik terjauh paling kanan
72     # misal:
73     # up -> kanan ke kiri -> mencapai ujung kanan convexhull
74     # down -> kanan ke kiri -> mencapai ujung kanan convexhull
75     # karena up dan down dihubungkan saat up mencapai ujung kanan dan down ada di ujung kiri, maka akan ada garis yang menghubungkan
76     # ujung kanan dan ujung kiri convexhull
77
78     # maka down haruslah diputar agar sesuai orientasi yaitu
79     # up -> kanan ke kiri -> mencapai ujung kanan convexhull
80     # down -> kiri ke kanan -> kembali lagi ke ujung kiri convexhull
81     # sehingga orientasi yang terjadi benar
82     hull = up + down[::-1]
83     return hull
```

6. Visualisasi *convex hull*

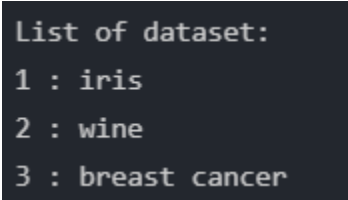
Main visualization program

```
plt.figure(figsize = (10, 6))
colors = ['c','m','y','k']
plt.title(f"Convex Hull of {df.columns[firstColumnChoice-1]} vs {df.columns[secondColumnChoice-1]}")
plt.xlabel(data.feature_names[firstColumnChoice-1])
plt.ylabel(data.feature_names[secondColumnChoice-1])
for j in range(len(data.target_names)):
    bucket = df[df['Target'] == j]
    buckets = bucket.iloc[:,[firstColumnChoice-1,secondColumnChoice-1]].values
    plt.scatter(buckets[:,0], buckets[:,1], label = data.target_names[j], color = colors[j])
    hull = list(myConvexHull(buckets))
    for i in range(len(hull)-1):
        x = [buckets[hull[i]][0], buckets[hull[i+1]][0]]
        y = [buckets[hull[i]][1], buckets[hull[i+1]][1]]
        plt.plot(x, y, colors[j])
plt.legend()
```

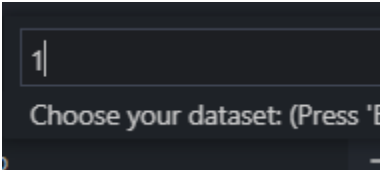
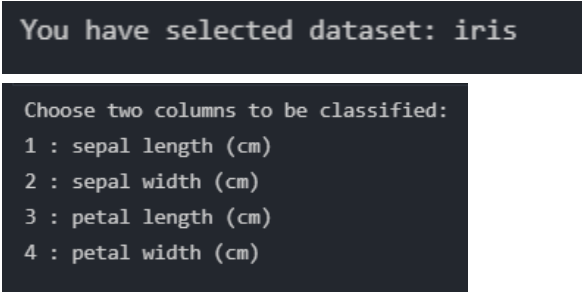
Python

III. *Screenshot* input-output program

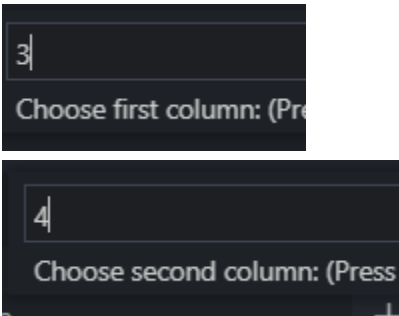
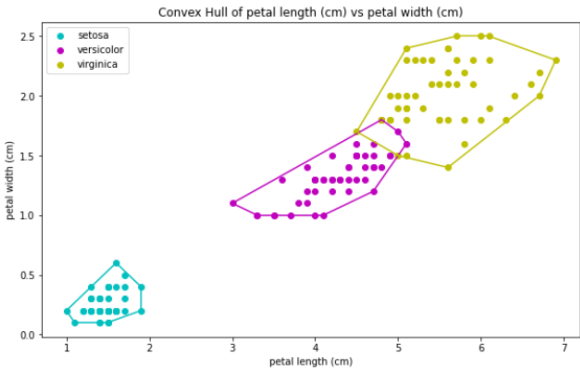
Program Pertama

Input/Output	Screenshot	Keterangan
Output		Pada saat program pertama kali jalan, ditampilkan dataset yang tersedia untuk dipilih

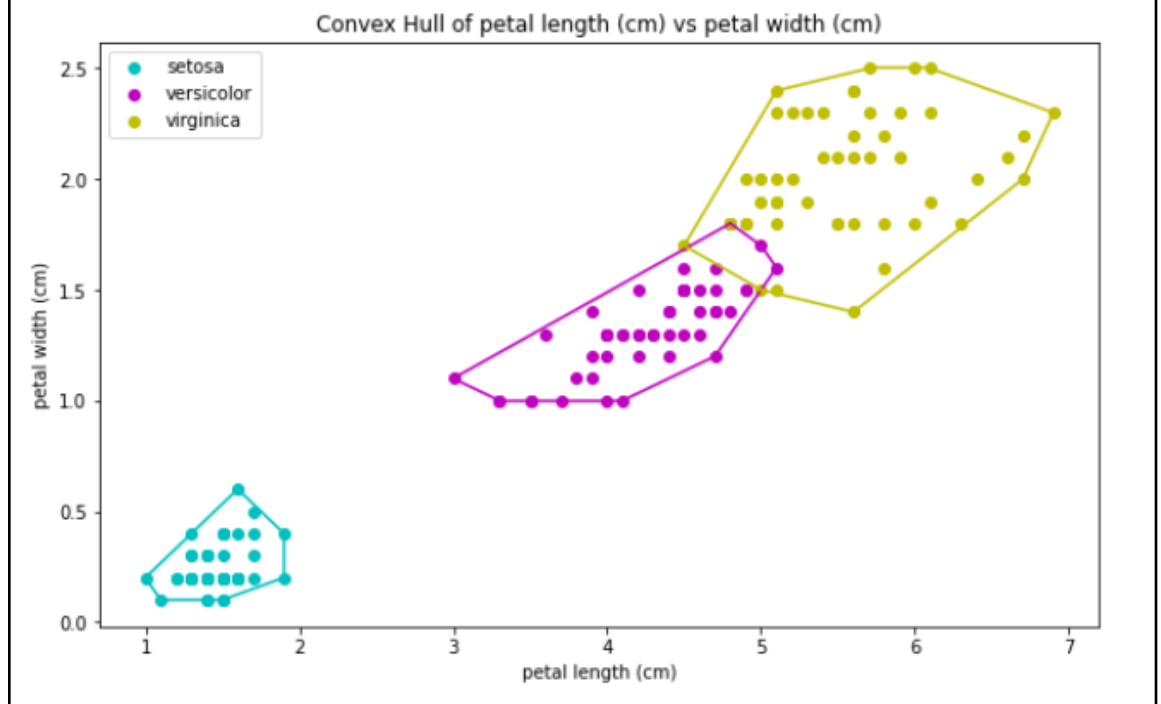
1. Dataset Iris

Input/Output	Screenshot	Keterangan
input		Memasukkan dataset
output		Output nama dataset yang telah dipilih serta pengeluaran kolom yang ada pada dataset terpilih

- petal-length(3), petal-width(4)

Input/Output	Screenshot	Keterangan
input		Pemilihan 2 kolom
output	<p>Choose two columns to be classified: 1 : sepal length (cm) 2 : sepal width (cm) 3 : petal length (cm) 4 : petal width (cm) You have chosen first column: petal length (cm) and second column: petal width (cm)</p> 	Output pengeluaran kolom 1 dan 2 terpilih beserta <i>convex hull</i>

Gambar lebih besar:



- Sepal-length(1), sepal-width(2)

Input/Output	Screenshot	Keterangan
input		Pemilihan 2 kolom

output

Choose two columns to be classified:

1 : sepal length (cm)

2 : sepal width (cm)

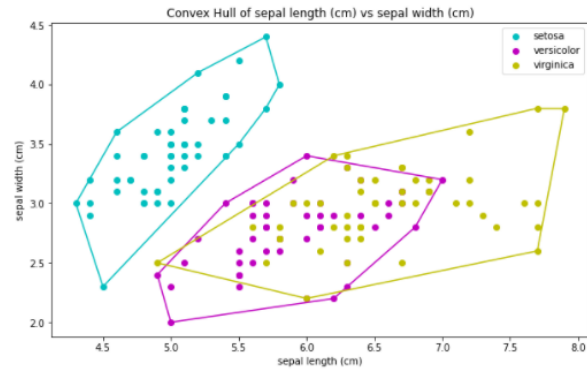
3 : petal length (cm)

4 : petal width (cm)

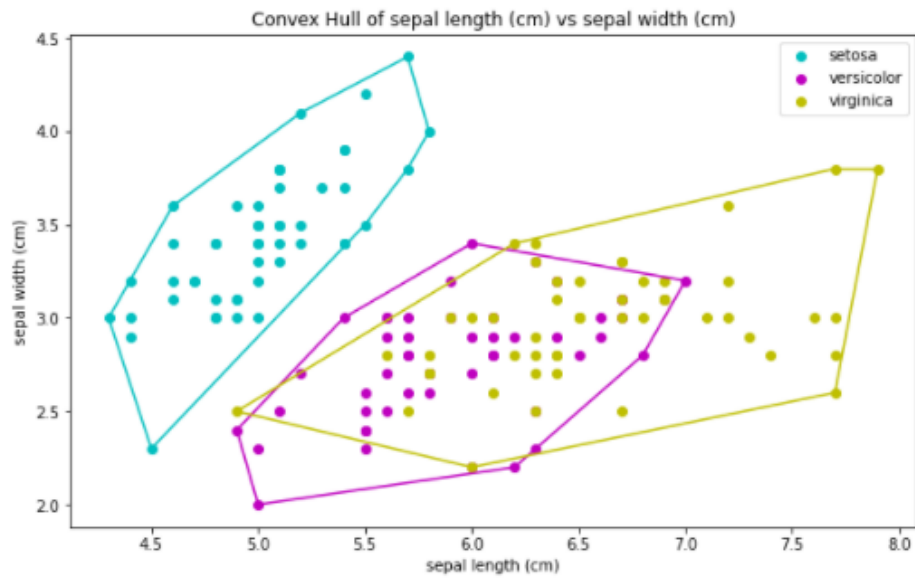
You have chosen first column: sepal length (cm) and second

column: sepal width (cm)

Output pengeluaran
kolom 1 dan 2 terpilih
berserta *convex hull*



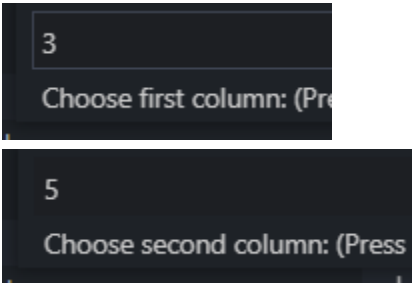
Gambar lebih besar:



2. Dataset Wine

Input/Output	Screenshot	Keterangan
input		Memasukkan dataset
Output		Output nama dataset yang telah dipilih serta pengeluaran kolom yang ada pada dataset terpilih

- Ash(3). magnesium(5)

Input/Output	Screenshot	Keterangan
input		Pemilihan 2 kolom

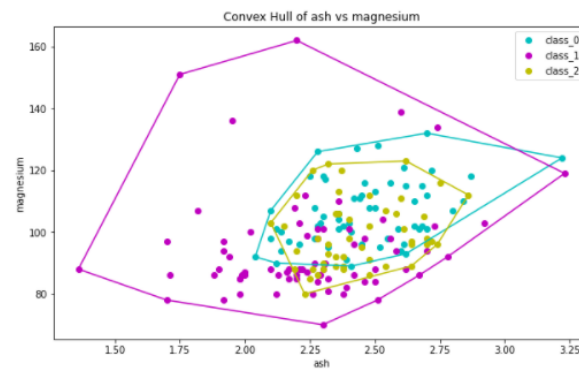
output

Choose two columns to be classified:

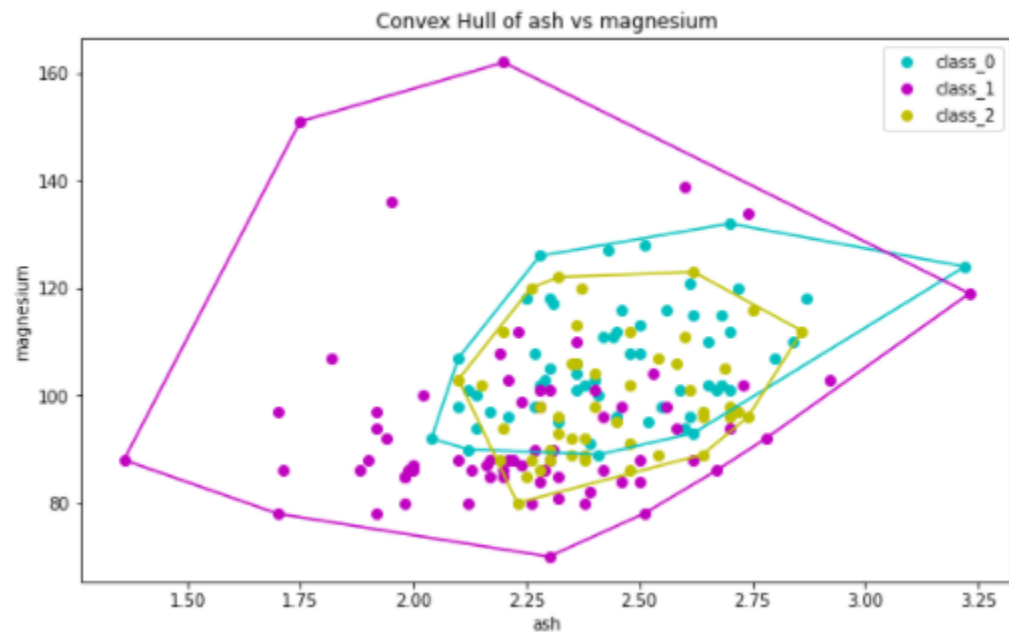
```
1 : alcohol
2 : malic_acid
3 : ash
4 : alcalinity_of_ash
5 : magnesium
6 : total_phenols
7 : flavanoids
8 : nonflavanoid_phenols
9 : proanthocyanins
10 : color_intensity
11 : hue
12 : od280/od315_of_diluted_wines
13 : proline
```

You have chosen first column: ash and second column: magnesium

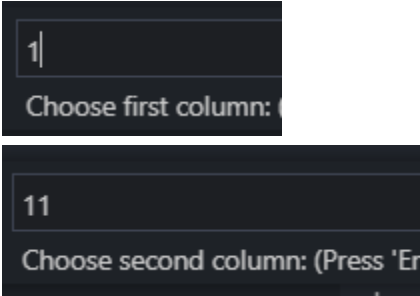
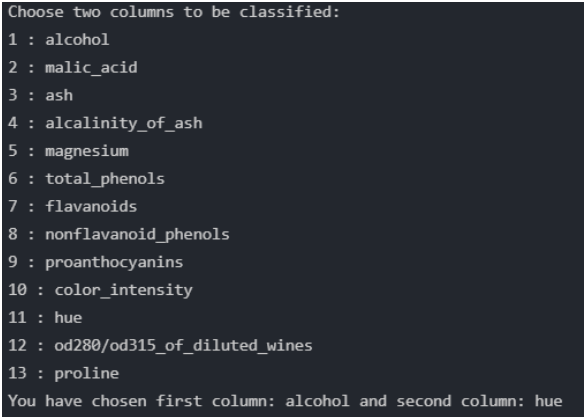
Output pengeluaran
kolom 1 dan 2 terpilih
beserta *convex hull*



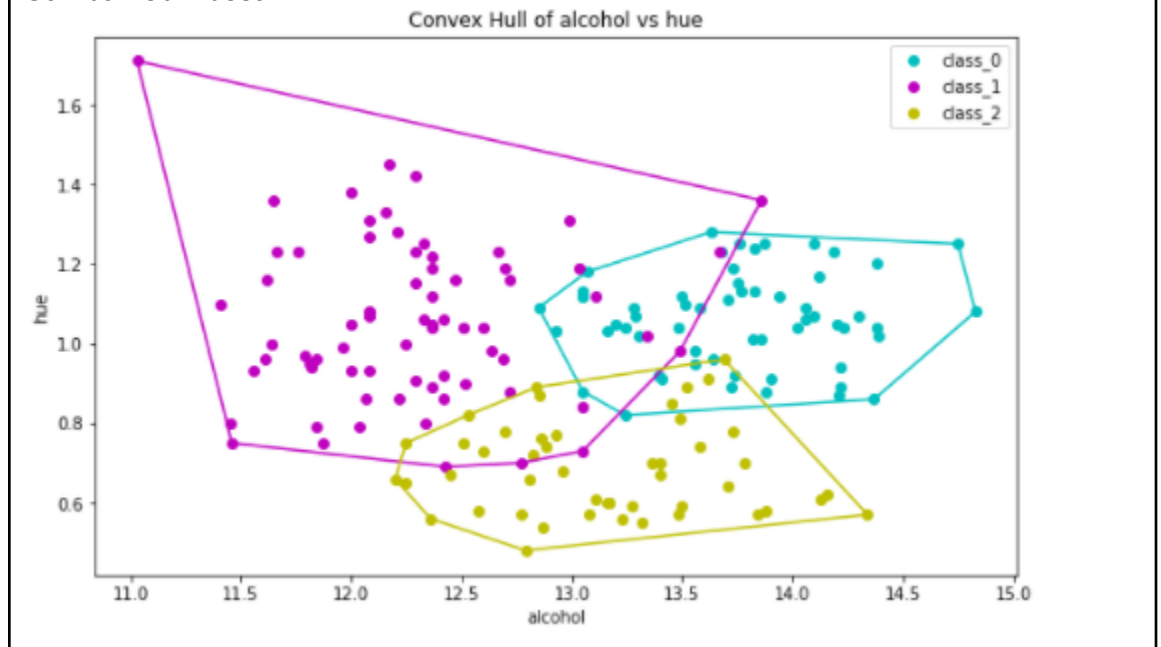
Gambar lebih besar:



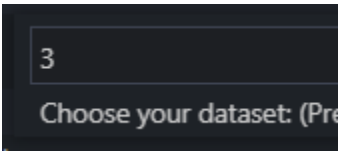
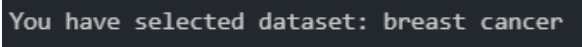
- Alcohol(1), hue(11)

Input/Output	Screenshot	Keterangan
input		Pemilihan 2 kolom
output		Output pengeluaran kolom 1 dan 2 terpilih beserta <i>convex hull</i>

Gambar lebih besar:



3. Dataset Breast Cancer

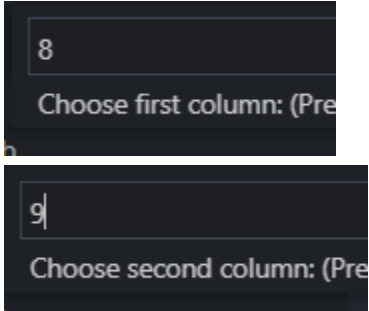
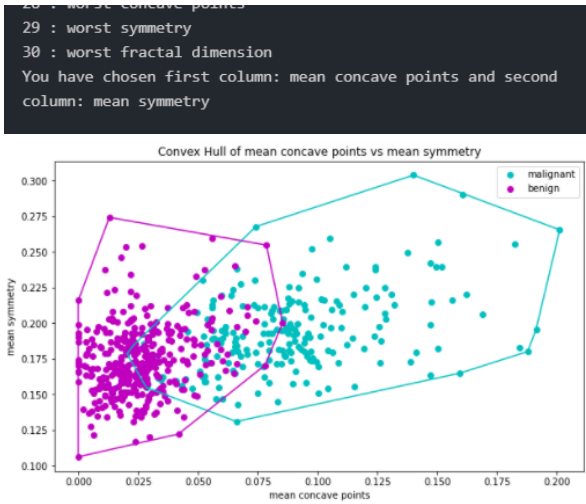
Input/Output	Screenshot	Keterangan
input		Memasukkan dataset
output		Output nama dataset yang telah dipilih serta pengeluaran kolom yang ada pada dataset terpilih

text editor

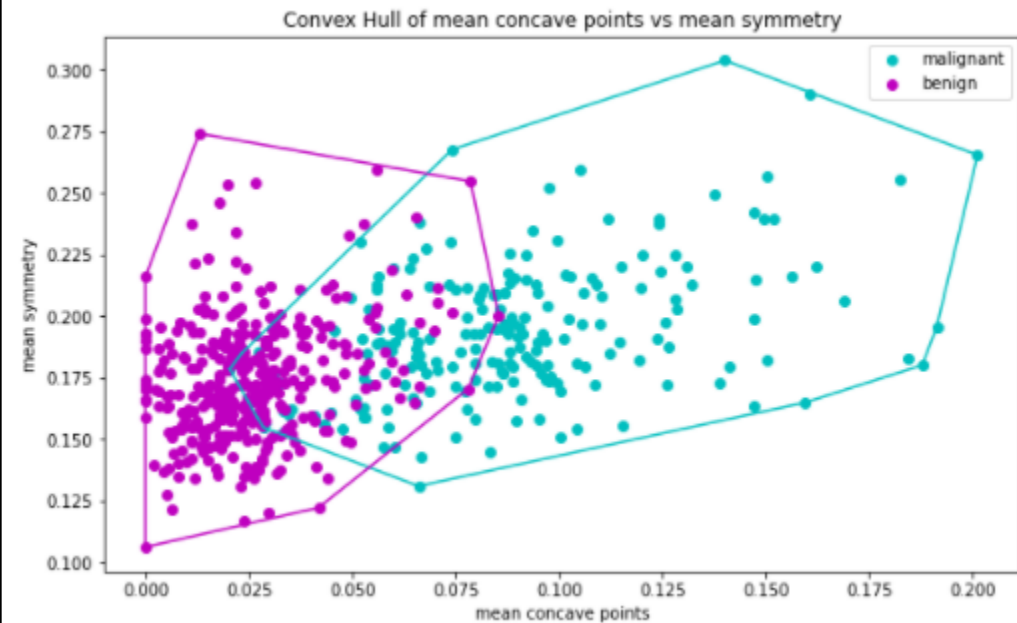
Choose two columns to be classified:

- 1 : mean radius
- 2 : mean texture
- 3 : mean perimeter
- 4 : mean area
- 5 : mean smoothness
- 6 : mean compactness
- 7 : mean concavity
- 8 : mean concave points
- 9 : mean symmetry
- 10 : mean fractal dimension
- 11 : radius error
- 12 : texture error
- 13 : perimeter error
- 14 : area error
- 15 : smoothness error
- 16 : compactness error
- 17 : concavity error
- 18 : concave points error
- 19 : symmetry error
- 20 : fractal dimension error
- 21 : worst radius
- 22 : worst texture
- 23 : worst perimeter
- 24 : worst area
- ...
- 26 : worst compactness
- 27 : worst concavity
- 28 : worst concave points
- 29 : worst symmetry
- 30 : worst fractal dimension

- Mean concave points(8), mean symmetry(9)

Input/Output	Screenshot	Keterangan
input		Pemilihan 2 kolom
output		Output pengeluaran kolom 1 dan 2 terpilih beserta <i>convex hull</i>

Gambar lebih besar:



- Area error(14), worst radius(21)

Input/Output	Screenshot	Keterangan
input		Pemilihan 2 kolom

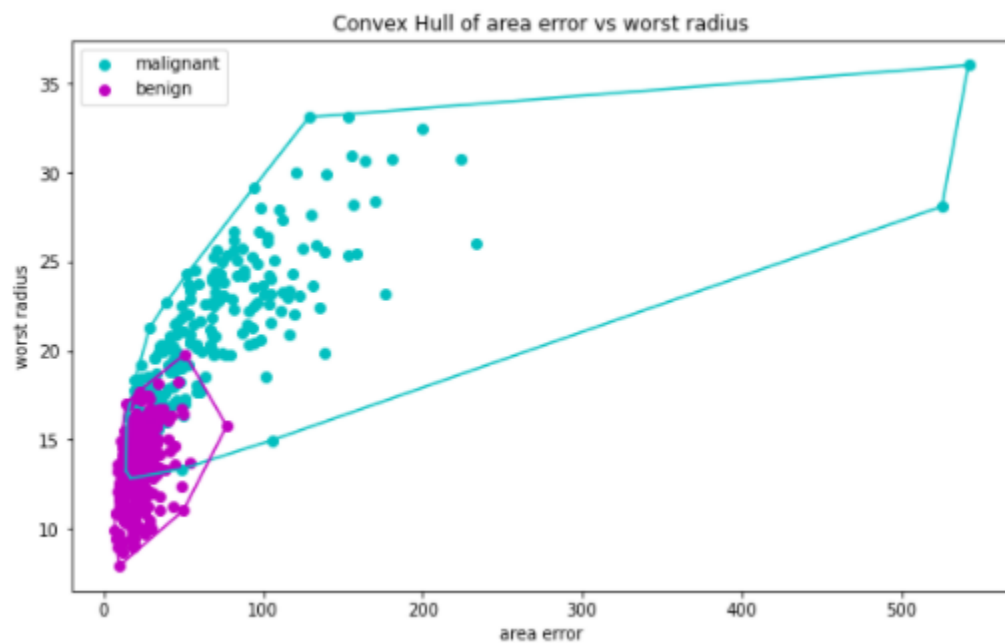
output

```
28 : worst concave points
29 : worst symmetry
30 : worst fractal dimension
You have chosen first column: area error and second column: worst
radius
```



Output pengeluaran
kolom 1 dan 2 terpilih
beserta *convex hull*

Gambar lebih besar:



IV. Link Github

- **ALAMAT GITHUB KODE PROGRAM**

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

- **CHECK LIST**

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.	✓	