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

Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II Tahun 2021/2022

Disusun oleh:

Grace Claudia 13520078

DAFTAR ISI

BAB I

PENJELASAN ALGORITMA

A. Algoritma *Divide and Conquer*

Algoritma *divide and conquer* merupakan algoritma penyelesaian masalah dengan prinsip membagi masalah menjadi beberapa upa masalah yang lebih kecil. Tiap upa-masalah ini memiliki karakteristik yang sama dengan masalah asal sehingga dapat sering diselesaikan dengan skema rekursif. Kelebihan dari algoritma ini adalah dapat mengurangi kompleksitas pencarian solusi suatu masalah.

Algoritma ini terdiri dari 3 langkah utama:

1. *Divide*: membagi persoalan menjadi sub persoalan yang lebih kecil dengan tentunya mempertahankan karakteristik persoalan yang ada.
2. *Conquer*: disebut juga tahap solve yaitu tahap dimana sub-persoalan diselesaikan baik secara langsung maupun rekursif
3. *Combine*: tahap terakhir dimana menggabungkan solusi-solusi dari sub persoalan yang lebih kecil sehingga membentuk solusi persoalan semula

Algoritma ini dapat menyelesaikan beberapa persoalan antara lain:

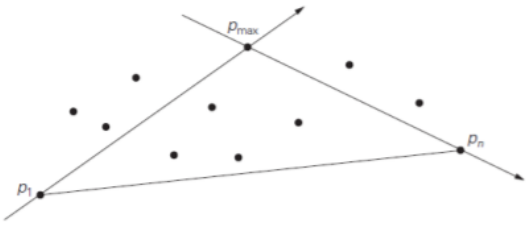
- Pencarian nilai maksimum dan minimum
- *Convex hull*
- Optimasi konversi bilangan desimal ke biner
- Dll.

B. Algoritma *divide and conquer* pada *Convex hull*

Convex hull merupakan persoalan pencarian poligon yang disusun dari subset titik hingga semua titik dari himpunan awal selain penyusun poligon berada di dalam area. *Convex hull* dimanfaatkan di kehidupan sehari0hari untuk collision detection, mendeteksi outliers pada data, dan lain sebagainya. Algoritma divide and conquer dapat digunakan pada penyelesaian persoalan *Convex Hull* ini.

Garis besar langkah-langkah penyelesaian :

1. Cari titik paling kiri(**P1**) dan paling kanan(**P2**) lalu tarik garis **S** yang membagi menjadi titik-titik menjadi dua bagian yaitu **S1** yaitu titik-titik yang berada di kiri//atas garis dan **S2** yaitu titik-titik yang berada di bawah/kanan garis.
2. Semua titik yang berada pada garis **S** tidak mungkin membentuk *convex hull* sehingga dapat diabaikan dalam pengecekan
3. Kumpulan titik pada **S1** akan membentuk *convex hull* bagian atas, begitu juga untuk titik pada **S2** dapat membentuk *convex hull* bagian bawah
4. Untuk sebuah *convex hull* terdapat 2 kemungkinan yaitu tidak ada titik selain bagian itu sendiri maka misalnya pada **S1**, **P1** dan **Pn** akan menjadi pembentuk *convex hull* pada bagian itu. Selain itu, ada juga kemungkinan **S1** tidak kosong sehingga akan dilakukan pemilihan titik dengan jarak terjauh dari garis. Titik yang di dalam segitiga yang dibentuk oleh titik terjauh dan garis diabaikan



Gambar 1. Ilustrasi *Convex Hull*
(sumber:https://informatika.stei.itb.ac.id/)

5. Tentukan sekumpulan titik yang berada di sebelah kiri garis **P1Pmax** menjadi bagian **S1.1** sedangkan sebelah kanan garis **PmaxPn** menjadi bagian **S1.2**, lanjutkan langkah 3 sampai tidak ada titik lagi di bagian tersebut lalu kembalikan hasil.
6. Lakukan langkah 4 dan 5 untuk bagian S2 hingga area ‘kanan’ dan ‘kiri’ kosong
7. Kembalikan gabungan pasangan titik yang dihasilkan.

C. Algoritma divide and conquer pada library sendiri (*myConvexHull*)

Pada tugas besar kali ini, ditugaskan untuk membuat *convex hull* dari *dataset-dataset* yang ada dengan menggunakan library sendiri yaitu *myConvexHull*. Algoritma secara garis besar program:

Program Utama

1. Menerima input *dataset*
2. *Dataset* yang berupa numpy array diubah ke array of poin
3. Menggunakan fungsi *myConvexHull*
4. Menerima return dari *myConvexHull* berupa titik-titik yang penyusun poligon dan memplotnya membentuk *convex hull* dari titik-titik yang ada

Fungsi *MyConvexHull*:

Parameter dari *myConvexhull* merupakan *points*(array of *points* dari *dataset*), a(titik paling kiri pada *dataset*), b(titik paling kanan pada *dataset*), Dan *direction*(arah dilakukannya pengecekan titik pembentuk *convexHull*). Langkah-langkah:

1. Pada pemanggilan pertama function (ditandakan dengan direction == None), akan disediakan 2 array kosong yaitu **up** dan **down**. **Up** akan menampung **titik yang berada di atas/kiri garis** sedangkan **down** akan menampung **titik yang berada di bawah/kanan garis**
2. Variabel **a** dan **b** dicari, **a** merupakan **titik terkiri** dari *dataset*(x terkecil) sedangkan **b** merupakan **titik terkanan** dari *dataset*(x terbesar).
3. Setelah ditemukan, maka kita akan membentuk sebuah **garis** yang menghubungkan **antara a dan b**, disebut **garis S**
4. Garis S akan digunakan untuk pengecekan posisi seluruh titik yang ada di *dataset* terkecuali elemen a dan b itu sendiri. Jika posisinya **diatas garis**, maka akan **dimasukkan ke array up**, jika posisinya **dibawah garis**, maka akan **dimasukkan ke array down**.
5. Lalu kita akan melanjutkan pengecekan dengan memanfaatkan rekursi dari fungsi *myConvexHull* dengan **membagi pengecekan** menjadi arah **atas** dan **bawah**. Untuk **arah atas** maka akan digunakan **array Up** dan **direction 1** sebagai penanda, sedangkan untuk **arah bawah** maka akan digunakan **array Down** dan

direction -1 sebagai penanda, untuk **arah atas** dan **bawah, a** dan **b** merupakan **titik terkiri dan terkanan** dari dataset. Akhir dari pemanggilan pertama fungsi *myConvexHull* akan mengembalikan titik a, b, dan hasil dari pengecekan arah atas dan bawah.

6. Untuk pengecekan selanjutnya, akan terbagi menjadi arah atas dan bawah.

   Pengecekan arah atas (direction == 1):

   1) Menyediakan array bernama **up** untuk menampung titik titik yang berada di atas garis
   2) Melakukan pengecekan titik-titik pada points dengan garis yang dilewati a dan b, apabila **berada di atas garis** maka akan ditambahkan ke array **up**
   3) Selanjutnya mencari titik terjauh dari garis dengan membandingkan semua array of points yang ada dengan meng-*assign* ke variabel bernama ***furthestPoint***
   4) Kembalian fungsi terakhir merupakan a dari parameter fungsi, b dari parameter fungsi, dan hasil pengecekan rekursi yang terbagi menjadi dua yaitu arah 'kiri' dan 'kanan'.
   5) Parameter arah kiri:
      - Points : array up
      - A : a dari parameter fungsi
      - B : furthestPoint
      - Direction : 1 (karena selalu ke atas)

      Parameter arah kanan:

      - Points : array up
      - A : furthestPoint
      - B : b dari parameter fungsi
      - Direction : 1 (karena selalu ke atas)

   Pengecekan arah bawah (direction == -1):

   1) Menyediakan array bernama **down** untuk menampung titik titik yang berada di bawah garis
   2) Melakukan pengecekan titik-titik pada points dengan garis yang dilewati a dan b, apabila **berada di bawah garis** maka akan ditambahkan ke array **down**

3) Selanjutnya mencari titik terjauh dari garis dengan membandingkan semua array of points yang ada dengan meng-*assign* ke variabel bernama ***furthestPoint***
4) Kembalian fungsi terakhir merupakan a dari parameter fungsi, b dari parameter fungsi, dan hasil pengecekan rekursi yang terbagi menjadi dua yaitu arah 'kiri' dan 'kanan'.
5) Parameter arah kiri:
   - Points : array down
   - A : a dari parameter fungsi
   - B : furthestPoint
   - Direction : -1 (karena selalu ke bawah)

   Parameter arah kanan:

   - Points : array down
   - A : furthestPoint
   - B : b dari parameter fungsi
   - Direction : -1 (karena selalu ke bawah)

7. Setelah pengecekan menyentuh **base case** yaitu **array of points** yang akan dicek **sudah kosong**(tidak ada kemungkinan lagi), maka akan dilakukan **return solusi** dari untuk **semua sub persoalan rekursi** dan akan dilakukan **penggabungan seluruh solusi** yang ada menjadi **solusi permasalahan utama**.

BAB II

KODE PROGRAM

Convex Hull Implementation

Grace Claudia - 13520078

Import *library* yang dibutuhkan

```
#import library yg dibutuhkan
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
```

[189] ✓ 0.9s

input *dataset* dan kolom yang akan digunakan untuk visualisasi

```
# title untuk plot datasset
title = ["Iris Classification Dataset", "Breast Cancer Wisconsin Classification Dataset", "Wine Classification Dataset"]
# memilih dataset yang akan digunakan
print("Choose Dataset:\n1. Iris dataset (classification)\n2. Breast Cancer Wisconsin (regression)\n3. Wine Dataset (classification)\n")
num = int(input("Masukkan dataset yang akan diuji: "))
# pengecekan apakah dataset yang dipilih ada, jika tidak ada user akan diminta menginput kembali
while num > 3 or num < 1:
    print("Masukan dataset salah")
    num = int(input("Masukkan dataset yang akan diuji: "))
if num == 1:
    data = datasets.load_iris()
elif num == 2:
    data = datasets.load_breast_cancer()
elif num == 3:
    data = datasets.load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
```

[190] ✓ 1.3s

```
... Choose Dataset:
1. Iris dataset (classification)
2. Breast Cancer Wisconsin (regression)
3. Wine Dataset (classification)
```

Print seluruh nama kolom yang ada dan memilih 2 kolom yang akan di plot

```
print("Kolom", title[num-1])
for i in range(len(df.columns)-1):
    print(str(i+1)+". "+str(df.columns[i]))
column1 = int(input("kolom 1: "))
column2 = int(input("kolom 2: "))
# apabila index kolom yang dipilih tidak ada, atau kolom1 dan kolom 2 sama, user akan diminta menginput kembali
while column1 > len(df.columns)-1 or column1 < 1 or column2 > len(df.columns)-1 or column2 < 1 or column1 == column2:
    print("Masukan kolom diluar index atau kolom 1 dan kolom 2 tidak boleh sama")
    column1 = int(input("kolom 1: "))
    column2 = int(input("kolom 2: "))
```

[191] ✓ 0.5s

```
... Kolom Wine Classification Dataset
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
```

Implementasi fungsi *Helper*

```python
import math
def leftMost(points):
    # mencari titik dengan x terkecil
    # input: array of points
    # output: left most point
    xMin = points[0][0]
    leftMostIndex = 0
    # mengiterasi points untuk mendapatkan x paling kecil
    for i in range (1,len(points)):
        if points[i][0] < xMin:
            xMin = points[i][0]
            leftMostIndex = i
    return points[leftMostIndex]

def rightMost(points):
    # mencari titik dengan x terbesar
    # input: array of points
    # output: right most point
    xMax = points[0][0]
    rightMostIndex = 0
    # mengiterasi points untuk mendapatkan x paling besar
    for i in range (1,len(points)):
        if points[i][0] > xMax:
            xMax = points[i][0]
            rightMostIndex = i
    return points[rightMostIndex]

def position(point, p1, p2):
    # menetukan apakah titik point berada diatas atau dibawah garis p1 dan p2
    # input: titik yang diuji (point), titik pembentuk garis(p1 dan p2)
    # output: jika hasil <= 0 maka letak titik diatas garis, jika hasil > 0 maka letak titik dibawah garis
    x1 = p1[0]
    y1 = p1[1]
    x2 = p2[0]
    y2 = p2[1]
    xA = point[0]
    yA = point[1]
    # Vector 1
    v1 = (x2-x1, y2-y1)
    # Vector 2
    v2 = (x2-xA, y2-yA)
    # Cross product
    xp = v1[0]*v2[1] - v1[1]*v2[0]
    return round(xp,10)

def mostDistancePoint(points, p1, p2):
    # mencari titik dengan jarak terjauh dari garis
    # input: titik-titik kandidat (points), titik pembentuk garis(p1 dan p2)
    # output: titik dengan jarak terjauh dari garis
    if len(points)==0:
        return None
    else:
        # mencari persamaan garis
        x1 = p1[0]
        y1 = p1[1]
        x2 = p2[0]
        y2 = p2[1]
        # mencari koefisien yaitu a, b, c dari ax + by + c = 0
        a = y2 - y1
        b = x1 - x2
        c = x2*y1 - x1*y2
        maxDistance = -999
        maxDistanceIndex = 0
        for i in range (len(points)):
            #cari dengan rumus d = |ax + by + c|/sqrt(a^2 + b^2)
            distanceNow = abs(a*points[i][0] + b*points[i][1] + c)/math.sqrt(a**2 + b**2)
            if  distanceNow > maxDistance:
                maxDistance = distanceNow
                maxDistanceIndex = i
        return points[maxDistanceIndex]
```

Implementasi algoritma convex hull dengan konsep *divide and conquer*

```python
def myConvexHull(points, a = None, b = None, direction = None):
    # mencari convex hull dari titik-titik yang ada
    # input: titik-titik kandidat (points), titik pembentuk garis(p1 dan p2), arah (direction)
    # output: convex hull dari titik-titik kandidat
    if len(points)==0:
        #apabila titik kandidat kosong, maka return kosong
        return []
    if direction == None:
        # apabila arah tidak ditentukan, maka arah rekursif akan terbagi menjadi arah atas dan bawah
        up = []
        down = []
        # a merupakan titik paling kiri, sedangkah b merupakan titik paling kanan
        a = leftMost(points)
        b = rightMost(points)
        for i in range (len(points)):
            # mengecek letak titik
            if (points[i] != a and points[i] != b):
                pos = position(points[i],a,b)
                # apabila hasil pengecekann <= 0 maka letak titik merupakan di kiri/atas garis, sedangkan jika > 0 maka di kanan garis
                if pos <= 0:
                    up += [points[i]]
                elif pos > 0:
                    down += [points[i]]
        # pengecekan masing-masin guntuk arah atas dan bawah secara rekursif
        return [a] + myConvexHull(up, a, b ,1) + [b] + myConvexHull(down, a, b, -1)[::-1]
    else:
        # jika arah merupakan 1 pengecekan dilakukan ke atas, jika -1 pengecekan dilakukan ke bawah
        if (direction == 1):
            up = []

            # mengecek posisi titik terhadap garis
            for i in range (len(points)):
                if (points[i] != a and points[i] != b):
                    if position(points[i],a, b) <= 0:
                        # menambahkan titik yang berada di atas garis ke array pengecekan
                        up += [points[i]]

            # mencari titik terjauh dari garis yang melewati a dan b
            furthestPoint = mostDistancePoint(up, a, b)

            # pengecekan dilanjutkan ke atas
            return [a] + myConvexHull(up, a, furthestPoint, 1) +  myConvexHull(up, furthestPoint, b, 1) + [b]

        elif (direction == -1):
            down = []

            # mengecek posisi titik terhadap garis

            for i in range (len(points)):
                # mengecek determinan
                if (points[i] != a and points[i] != b):
                    if position(points[i], a, b) > 0:
                        # menambahkan titik yang berada di bawah garis ke array pengecekan
                        down += [points[i]]

            # mencari titik terjauh dari garis yang melewati a dan b
            furthestPoint = mostDistancePoint(down,a, b)

            # pengecekan dilanjutkan ke bawah
            return [a] + myConvexHull(down, a, furthestPoint, -1) +  myConvexHull(down, furthestPoint, b, -1) + [b]
```

✓ 0.6s

Program visualisasi utama

```
    plt.figure(figsize = (10, 6))
    colors = ['blue','red','green','yellow','orange','black','brown','purple','pink','gray','cyan']
    plt.title(f"Convex Hull of {title[num-1]}")
    plt.xlabel(data.feature_names[column1-1])
    plt.ylabel(data.feature_names[column2-1])

    for j in range(len(data.target_names)):
        bucket = df[df['Target'] == j]
        buckets = bucket.iloc[:,[column1-1,column2-1]].values
        # changing from numpy array to list of points
        result = tuple(map(tuple, buckets))
        # menggunakan fungsi myConvexHull
        hull = myConvexHull(result)
        # plotting convex hull
        plt.scatter([p[0] for p in buckets], [p[1] for p in buckets], label=data.target_names[j], color=colors[j])
        plt.legend()
        for i in range(len(hull)-1):
            x = [hull[i][0], hull[i+1][0]]
            y = [hull[i][1], hull[i+1][1]]
            plt.plot(x, y, colors[j], linewidth=2, linestyle='-')
```

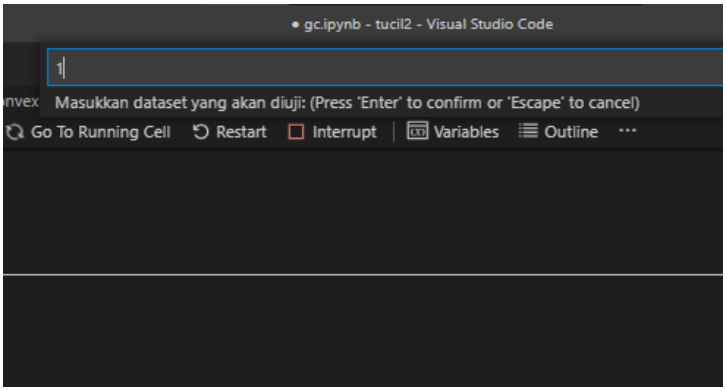
BAB III

SCREENSHOT INPUT DAN *OUTPUT* PROGRAM

Output Program saat pertama kali dijalankan berupa dataset yang dapat dipilih
Pilihan Dataset: 1. Iris dataset (classification) 2. Breast Cancer Wisconsin (regression) 3. Wine Dataset (classification)

1. Dataset Iris

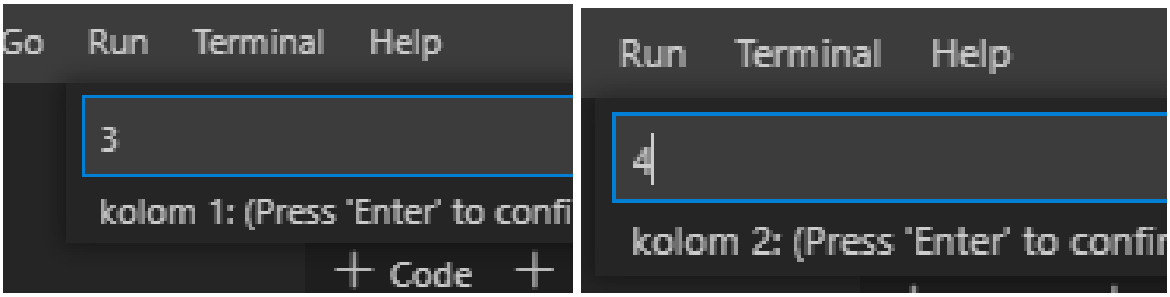
    1.1 Petal-length, petal-width

Input awal pemasukan program untuk memilih dataset


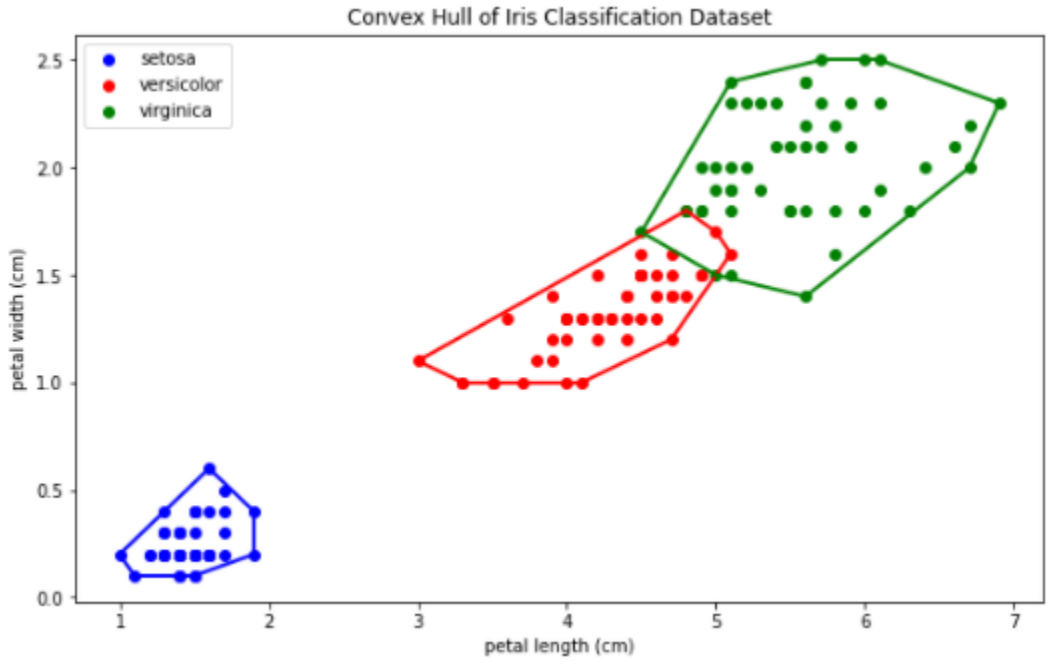
Output saat kita sudah memilih dataset, akan dikeluarkan kolom yang hendak kita pilih

```
[197]  25.4s
...  Kolom Iris Classification Dataset
     1. sepal length (cm)
     2. sepal width (cm)
     3. petal length (cm)
     4. petal width (cm)
```

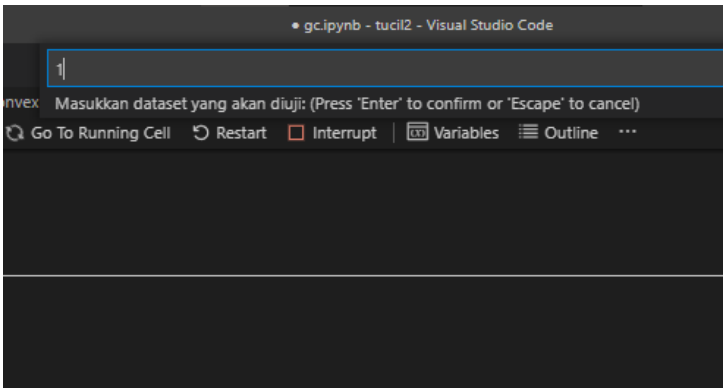
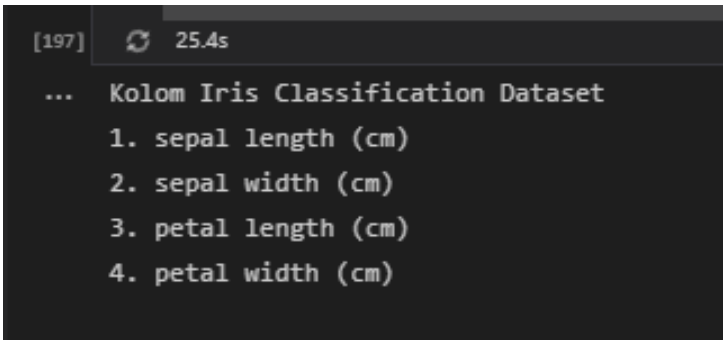
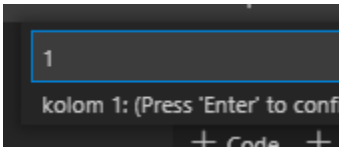
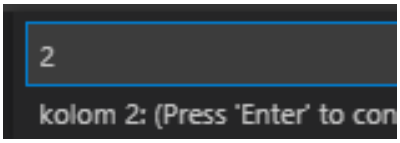
Input untuk memilih kolom mana yang akan di-*plot*

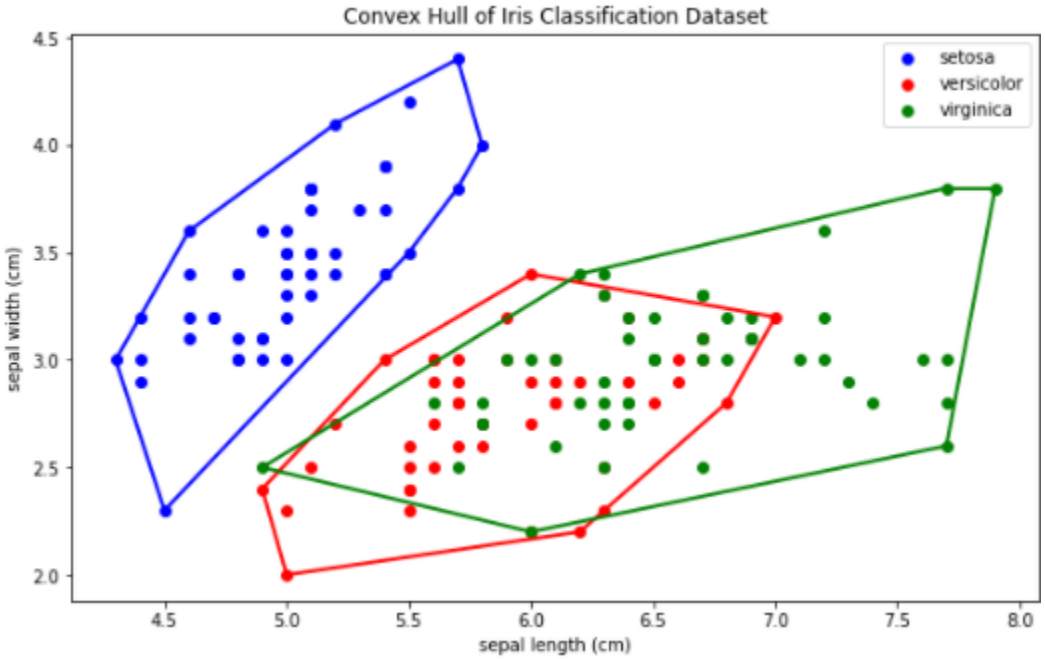


Output *convex hull* yang terbentuk



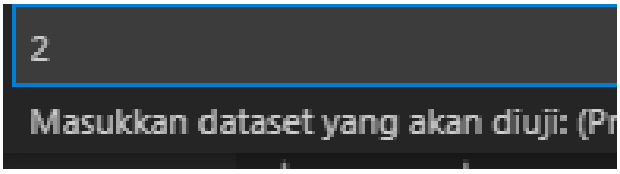
1.2 Sepal-length, sepal-width

Input awal pemasukan program untuk memilih dataset

Output saat kita sudah memilih dataset, akan dikeluarkan kolom yang hendak kita pilih

Input untuk memilih kolom mana yang akan di-*plot*
 

Output *convex hull* yang terbentuk


2. *Dataset* lainnya 1: *dataset breast cancer wisconsin*

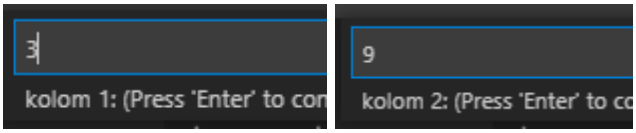
   2.1 mean perimeter, mean symmetry

Input awal pemasukan program untuk memilih dataset


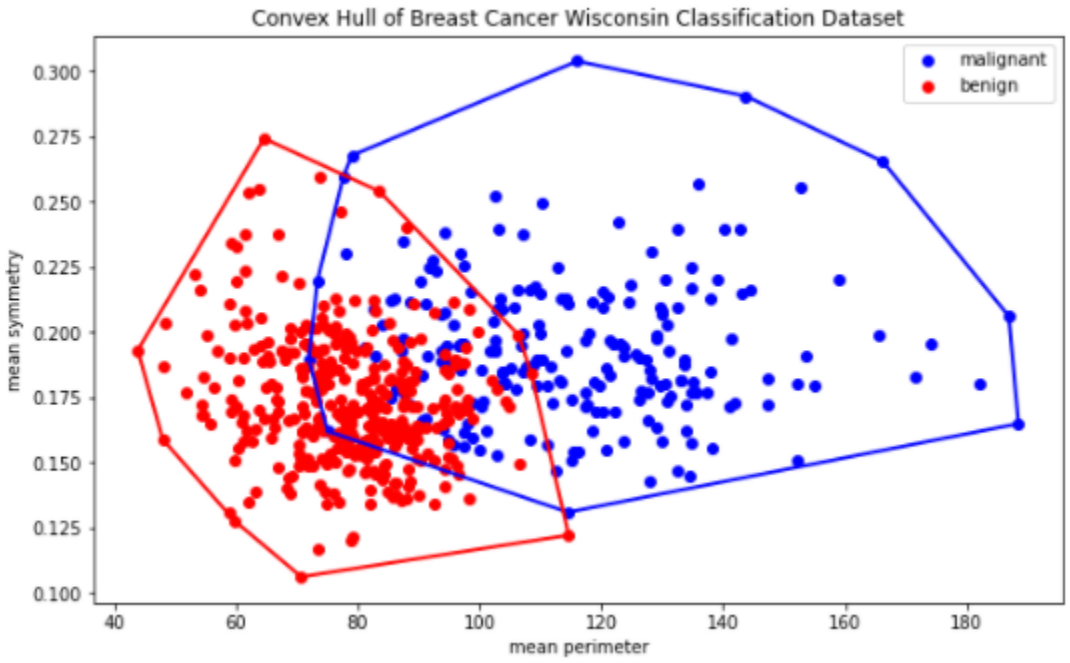
Output saat kita sudah memilih dataset, akan dikeluarkan kolom yang hendak kita pilih

```
Kolom Breast Cancer Wisconsin Classification Dataset
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
```

Input untuk memilih kolom mana yang akan di-*plot*

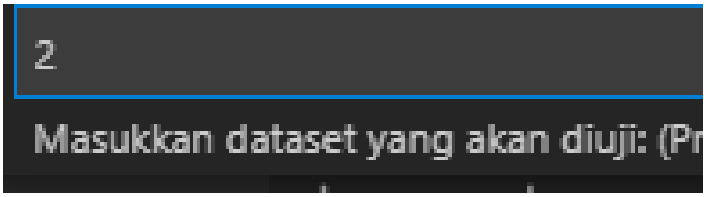


Output *convex hull* yang terbentuk



2.2 radius error, worst radius

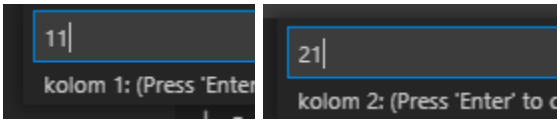
Input awal pemasukan program untuk memilih dataset

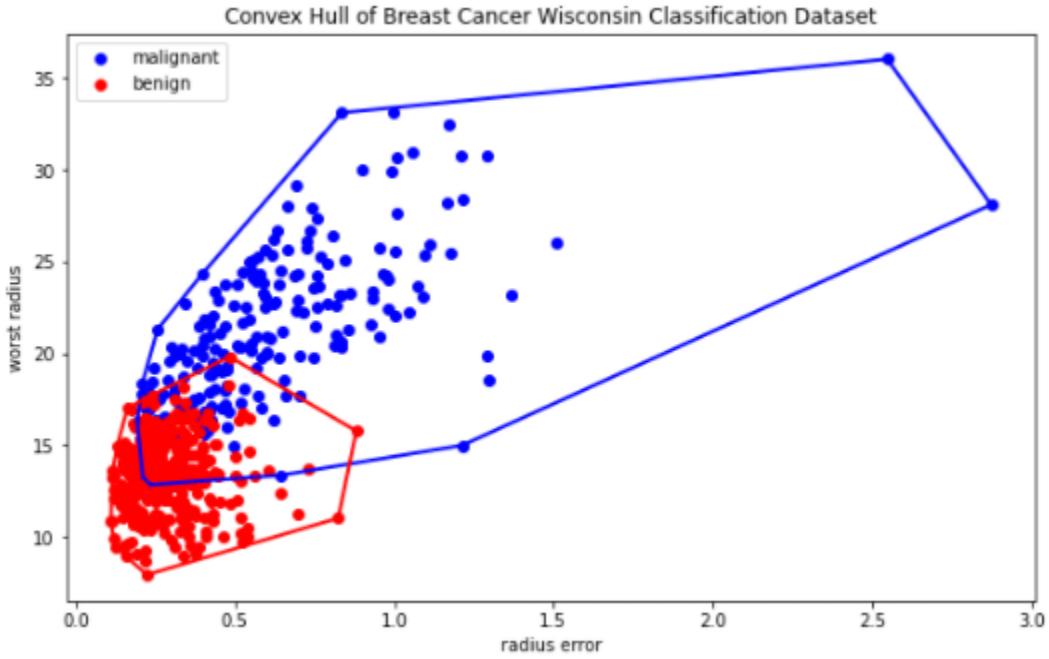


Output saat kita sudah memilih dataset, akan dikeluarkan kolom yang hendak kita pilih

```
Kolom Breast Cancer Wisconsin Classification Dataset
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
```

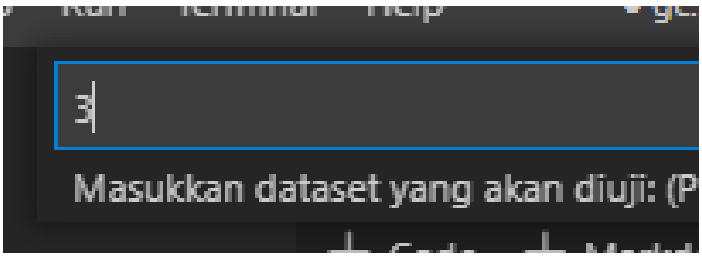
Input untuk memilih kolom mana yang akan di-*plot*



Output *convex hull* yang terbentuk


3. *Dataset* lainnya 2: *dataset wine*

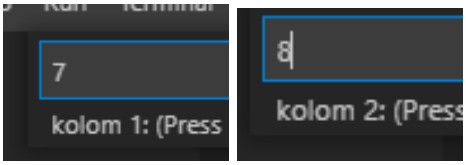
   3.1 flavanoids, nonflavanoid_phenols

Input awal pemasukan program untuk memilih dataset


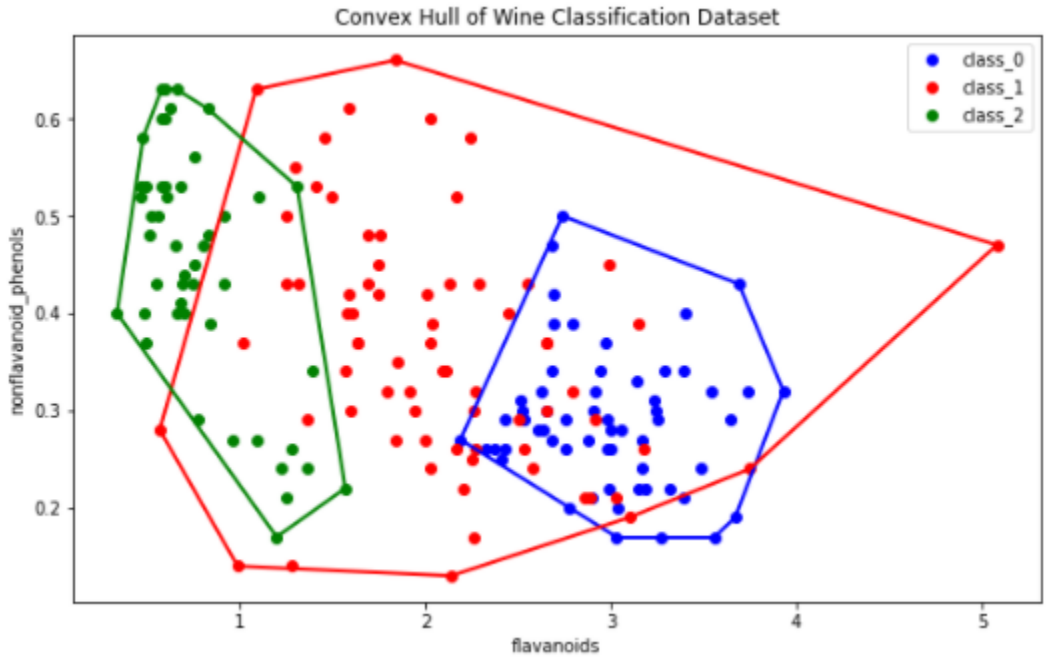
Output saat kita sudah memilih dataset, akan dikeluarkan kolom yang hendak kita pilih

```
Kolom Wine Classification Dataset
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
```

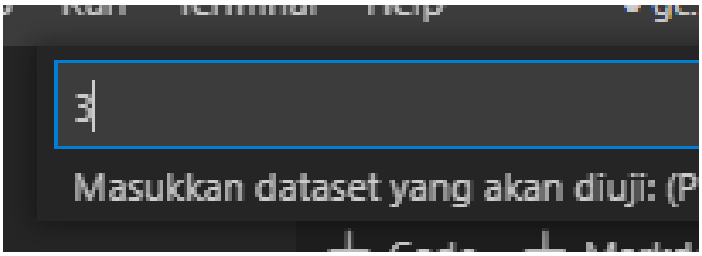
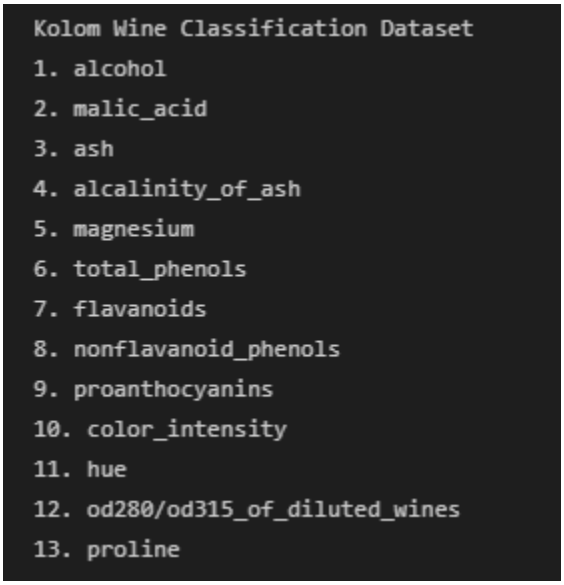
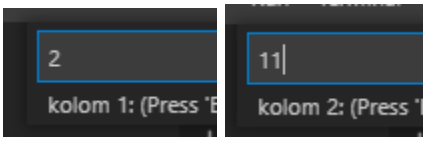
Input untuk memilih kolom mana yang akan di-*plot*



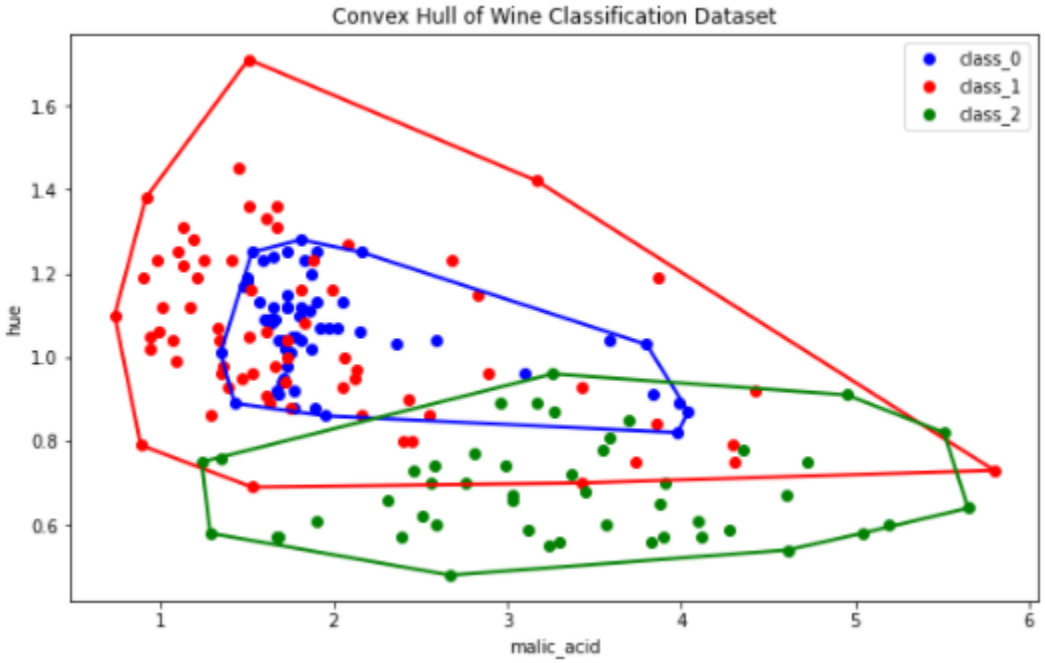
Output *convex hull* yang terbentuk



3.2 malic_acid, hue

Input awal pemasukan program untuk memilih dataset

Output saat kita sudah memilih dataset, akan dikeluarkan kolom yang hendak kita pilih

Input untuk memilih kolom mana yang akan di-*plot*


Output *convex hull* yang terbentuk



LAMPIRAN

I. ALAMAT GITHUB KODE PROGRAM DAN DRIVE

Github: https://github.com/graceclaudia19/Tucil2_13520078

Drive:
https://drive.google.com/drive/u/0/folders/1E2pjkyyj3ZPhAPgEOZfPUi3N4ZyeWmuL

II. 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.	✓	