LAPORAN TUGAS KECIL 2

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

IF2211 Strategi Algoritma

Oleh:

Farhan Hafiz 13520027

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

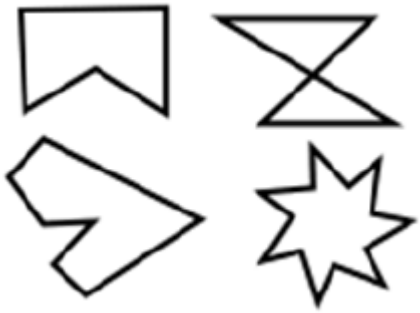
2022

Algoritma *Divide and Conquer* untuk Implementasi Convex Hull

Algoritma Divide and Conquer merupakan salah satu algoritma yang cukup efektif untuk digunakan. Divide berarti membagi persoalan menjadi beberapa upa persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil idealnya berukuran hampir sama. Conquer (solve) berarti menyelesaikan masing masing upa persoalan (secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar). Jadi, algoritma divide and conquer adalah algoritma yang mengabungkan solusi masing masing upa persoalan sehingga membentuk solusi persoalan semula. Salah satu contoh aplikasi penggunaan algoritma divide and conquer adalah implementasi convex hull.

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. Contoh gambar 1 adalah poligon yang convex , sedangkan gambar 2 menunjukkan contoh yang non convex.

Gambar 1 Convex



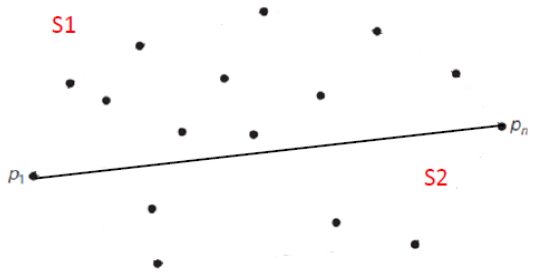
Gambar 2 Convex

Untuk dua titik, maka convex hull berupa garis yang menghubungkan 2 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 3 Garis Convex Hull

Langkah-langkah Algoritma Divide and Conquer dalam implementasi convex hull :

1. Garis yang menghubungkan p_1 dan p_n p_1p_n) membagi S menjadi dua bagian yaitu S1 (kumpulan titik di sebelah kiri atau atas garis p_1p_n) dan S2 (kumpulan titik di sebelah kanan atau bawah garis p_1p_n).



Gambar 4 Ilustrasi Garis p_1p_n

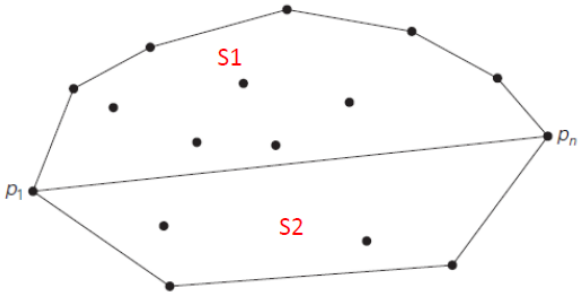
Untuk memeriksa apakah sebuah titik berada di sebelah kiri atau atas) suatu garis yang dibentuk dua titik, gunakan penentuan determinan :

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = x_1y_2 + x_3y_1 + x_2y_3 - x_3y_2 - x_2y_1 - x_1y_3$$

Titik (x3,y3) berada di sebelah kiri dari garis ((x1,y1),(x2,y2)) jika hasil determinan positif.

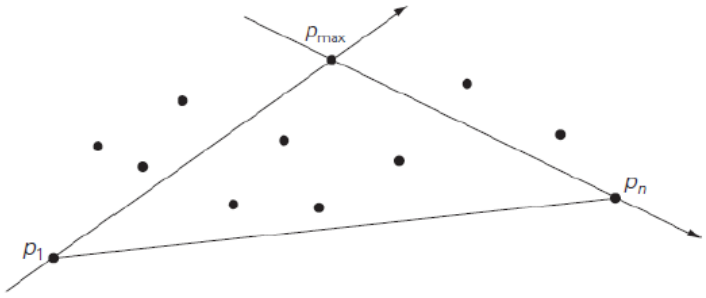
2. Semua titik pada S yang berada pada garis p_1p_n (selain titik p_1 dan p_n) tidak mungkin membentuk convex hull , sehingga bisa diabaikan dari pemeriksaan

3. Kumpulan titik pada S1 bisa membentuk convex hull bagian atas, dan kumpulan titik pada S2 bisa membentuk convex hull bagian bawah dengan menggunakan divide and conquer Kembali.



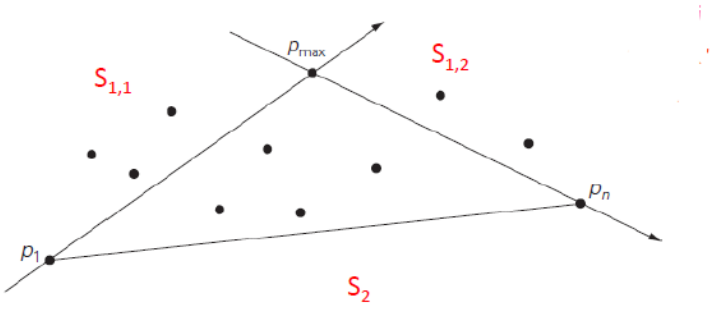
Gambar 5 Ilustrasi convex hull atas dan bawah

4. Untuk sebuah bagian (misal S1), terdapat dua kemungkinan:
   • Jika tidak ada titik lain selain S1, maka titik $p_1$dan p_n menjadi pembentuk convex hull bagian S1
   • Jika S1 tidak kosong, pilih sebuah titik yang memiliki jarak terjauh dari garis p_1p_n(misal p_{max}).Jikaterdapatbeberapa titik dengan jarak yang sama, pilih sebuah titik yang memaksimalkan sudut $pm_{ax}p_1p_n$
   Semua titik yang berada di dalam daerah segitiga $p_{max}p_1p_n$ diabaikan untuk pemeriksaan lebih lanjut



Gambar 6 Ilustrasi mencari titik terjauh

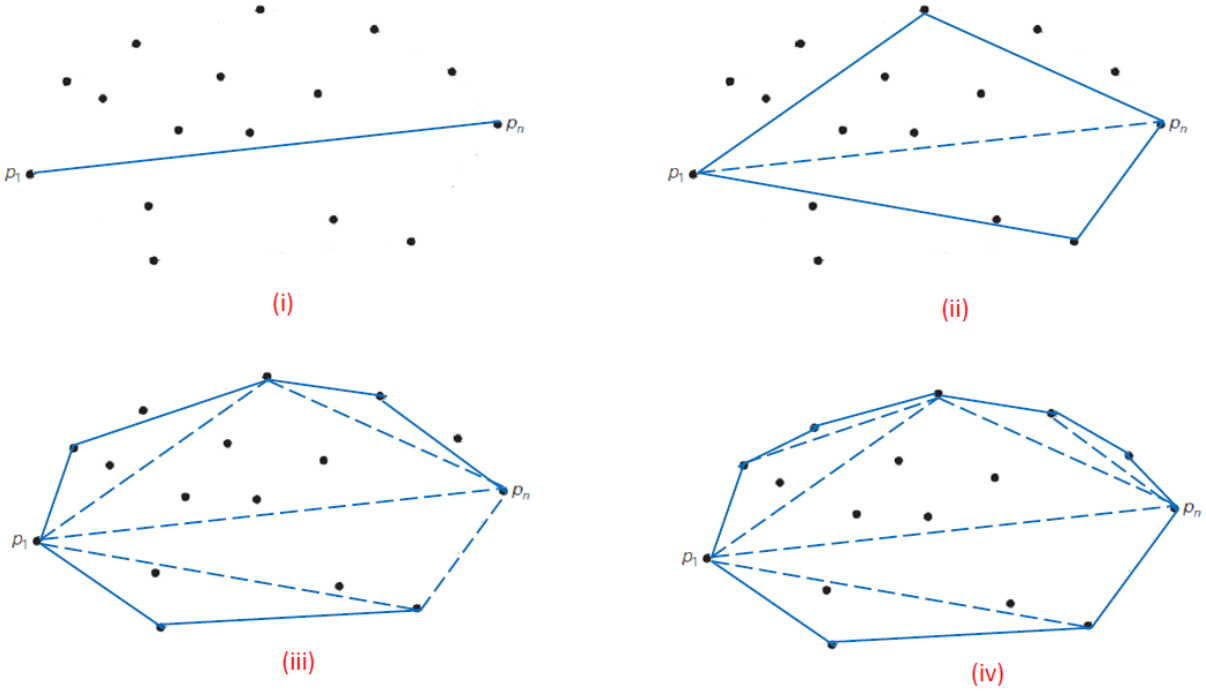
5. Tentukan kumpulan titik yang berada di sebelah kiri garis $p_1p_{ma}x$ (menjadi bagian $S_{1,1}$) dan di sebelah kanan garis p_1p_{max} (menjadi bagian $S_{1,2}$)



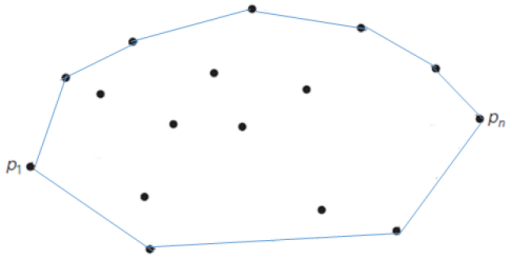
Gambar 7 Ilustrasi kumpulan titik sebelah titik terjauh

6. Lakukan hal yang sama (butir 4 dan 5) untuk bagian S2, hingga bagian ‘kiri’ dan ‘kanan’ kosong

7. Kembalikan pasangan titik yang dihasilkan



Hasil akhir:



Gambar 8 Ilustrasi tahapan implementasi convex hull dengan Divide and Conquer

Source Program

Source Program dibuat dalam Bahasa python dengan nama ‘Tucil2_ConvexHull.ipynb’ pada file ini terdapat beberapa bagian source code antara lain

Bagian Import Dataset

Bagian Mengimport Dataset yaitu Iris dan Wine data-datanya digunakan untuk mencari Convex Hull-nya.

Import Dataset Iris yang disimpan dalam df :

```
# Import Data Set Iris

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
data = datasets.load_iris()

#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
```

Import Dataset Wine yang disimpan dalam df2 :

```
# Import Data Set Wine

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
data = datasets.load_wine()

#create a DataFrame
df2 = pd.DataFrame(data.data, columns=data.feature_names)
df2['Target'] = pd.DataFrame(data.target)
print(df2.shape)
df2.head()
```

Bagian Pustaka myConvexHull

Inisialisasi beberapa list yang akan digunakan :

```
# List untuk menyimpan titik-titik koordinat data set yang ada
coordinates = []
# List untuk menyimpan titik-titik terluar/solusi convex hull
solution = []
# List untuk menyimpan titik-titik partisi/sub persoalan convex hull
partition = []
```

Fungsi get_side :

```
# Fungsi yang mengembalikan sisi dari sebuah titik terhadap garis AB (kiri/kanan/pada garis)
def get_side (A,B,coordinate):

    # Set titik-titik pada garis AB ke x1,y1 dan x2,y2, serta coordinate ke x3,y3
    x1, y1 = A
    x2, y2 = B
    x3, y3 = coordinate

    # Mencari determinan dari titik (x3,y3) ke garis A(x1,y1),B(x2,y2)
    Det = (y2-y1)*x3 + (x1-x2)*y3 + (x2*y1 - x1*y2)

    # Mencari apakah titik berada di kiri atau kanan berdasarkan determinan
    if Det > 0:
        return 'right'
    elif Det < 0:
        return 'left'
    else:
        return 'in-line'
```

Fungsi get_side digunakan untuk mengetahui apakah sebuah titik berada di kiri atau kanan dari sebuah garis. Pencarian dilakukan dengan mengambil nilai determinan dari ketiga titik menggunakan rumus yang telah dijelaskan pada bagian sebelumnya. Penjelasan lebih lanjut dapat dilihat pada comment source code diatas.

Fungsi convex_hull_initial :

```python
def convex_hull_initial(data):

    # Lakukan pengurutan titik-titik pada data terlebih dahulu berdasarkan nilai X-membesar
    # jika nilai X sama, urutkan berdasarkan Y-membesar
    sorted_coordinates = sorted(data , key=lambda k: [k[0], k[1]])

    # Cari titik terjauh sebelah kanan dan kiri dan simpan menjadi garis AB
    A = sorted_coordinates[0]           #titik terjauh kiri
    B = sorted_coordinates[-1]          #titik terjauh kanan

    # Lakukan iterasi titik-titik (coordinate) pada sorted_coordinates
    # Jika titik coordinates berada di sebelah kanan garis AB, masukkan ke list Right_AB
    # Jika titik coordinates berada di sebelah kiri garis AB, masukkan ke list Left_AB
    Right_AB = []
    Left_AB = []
    for coordinate in sorted_coordinates:
        side = get_side(A,B,coordinate)
        if side == 'right':
            Right_AB.append(coordinate)
        elif side == 'left':
            Left_AB.append(coordinate)
        else:
            pass

    # Masukkan garis AB ke dalam list solution sebagai initial solution
    solution.append((A,B))

    # Panggil next_hull() untuk mencari titik-titik solusi convex hull dengan divide dan conquer
    next_hull(Right_AB,A,B)
    next_hull(Left_AB,B,A)
```

Fungsi convex_hull_initial merupakan fungsi yang digunakan saat pertama kali membaca data set. Fungsi ini akan mengurutkan terlebih dahulu titik-titik koordinat yang ada pada dataset. Kemudian, titik-titik tersebut akan dicari titik terjauhnya yang nantinya dibuat garis dan setiap titik lainnya akan dicari letaknya apakah berada di kiri atau kanan garis yang dibuat. Lalu, akan dilanjutkan dengan fungsi next_hull yang menjadi algoritma divide and conquer selanjutnya Penjelasan lebih lanjut pada tiap Langkah dapat dilihat pada comment source code diatas.

Fungsi next_hull :

```python
# Fungsi yang digunakan untuk mencari titik terluar selanjutnya guna mendapatkan convex hull
def next_hull (partition,A,B):
    #Fungsi melakukan return apabila tidak terdapat titik lagi pada partition
    if len(partition) == 0:
        return
    else:
        # Set titik-titik pada garis AB ke x1,y1 dan x2,y2
        x1, y1 = A
        x2, y2 = B

        # Simpan nilai a,b,c yang nantinya digunaakan untuk mencari jarak
        a = y2 - y1
        b = x1 - x2
        c = x2*y1 - x1*y2

        # Set jarak terjauh = -1, titik C = None
        max_distance = -1
        C = None

        # Lakukan iterasi titik-titik (coordinate) pada partition
        # Untuk mencari nilai titik C, dimana C merupakan titik dengan
        # jarak terjauh dari garis AB
        for coordinate in partition:
            x, y = coordinate
            distance = abs(a*x + b*y + c)
            if distance >= max_distance:
                max_distance = distance
                C = coordinate

        # Simpan garis AB pada list newAB
        newAB = []
        newAB.append((A,B))
```

```python
        # Lakukan update pada list solution
        # Jika ditemukan titik C, maka hapus garis AB pada list solution
        # Kemudian masukan garis AC dan CB ke list solution

        # Iterasi untuk menghapus garis AB jika ditemukan titik C
        found = False
        for j in range (len(solution)):
            Test = solution[j]
            Test2 = newAB[0]
            if (np.array_equal(Test,Test2)):
                found = True
                break
        if (found):
            solution.pop(j)

        # Masukkan garis AC dan CB ke list solution
        solution.append((A,C))
        solution.append((C,B))

        # Hapus titik C pada partition setelah update list solution
        for i in range (len(partition)):
            Test3 = partition[i]
            if (np.array_equal(C,Test3)):
                break
        partition.pop(i)
```

```
    # Cari titik-titik yang berada di sebelah kanan garis AC
    Right_AC = []
    for coordinate in partition:
        side = get_side(A,C,coordinate)
        if side == 'right':
            Right_AC.append(coordinate)
    # Cari titik-titik yang berada di sebelah kanan garis CB
    Right_CB = []
    for coordinate in partition:
        side = get_side(C,B,coordinate)
        if side == 'right':
            Right_CB.append(coordinate)


    # Lakukan hal yang sama pada garis AC dan CB menggunakan divide and conquer
    # dengan memanggil fungsi next_hull() kembali
    next_hull(Right_AC,A,C)
    next_hull(Right_CB,C,B)
```

Fungsi next_hull merupakan implementasi algoritma divide and conquer secara berulang untuk melakukan update titik-titik convex hull secara terus menerus hingga ke upa-persoalan paling kecil. Source code di atas merupakan implementasi dari Langkah-langkah algoritma divide and conquer untuk convex hull yang telah dijelaskan sebelumnya. Penjelasan lebih lanjut pada tiap Langkah dapat dilihat pada comment source code diatas.

Fungsi myConvexHull :

```
def myConvexHull(data):
    # Panggil fungsi convex_hull_initial() dan simpan titik-titik solusi ke dalam list 'line'
    convex_hull_initial(data)
    line = []
    line.clear()
    for i in range (len(solution)):
        (x1,y1),(x2,y2) = solution[i]
        line_new = (x1,x2)
        line_new2 = (y1,y2)
        line.append(line_new)
        line.append(line_new2)

    return line
```

Fungsi myConvexHull merupakan fungsi yang nantinya menjadi Pustaka dan dipanggil untuk menyelesaikan convexhull dataset yang diinginkan. Fungsi ini akan memanggil fungsi sebelumnya yaitu convex_hull_initial dan menyimpan titik-titik solusi yang dihasilkan pada list ‘line’. Isi dari list ‘line’ ini yang nantinya akan di-plot menghasilkan convex hull yang sesuai

Bagian Visualisasi Dataset

```
#visualisasi hasil ConvexHull 'Petal Width vs Petal Length'
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal Length')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])

solution.clear()
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    hull = myConvexHull(bucket)    #bagian ini diganti dengan hasil implementas ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    if (i==0) :
        j = 0
    while (j <len(hull)):
        plt.plot(hull[j],hull[j+1], colors[i])
        j += 2
    hull.clear()
```

Source code di atas merupakan salah satu source code yang akan memvisualisasikan diagram convex hull yang diinginkan. Semua input source code visualisasi dan output visualisasi akan dijelaskan pada bagian Input-Output Program

Input-Output Program

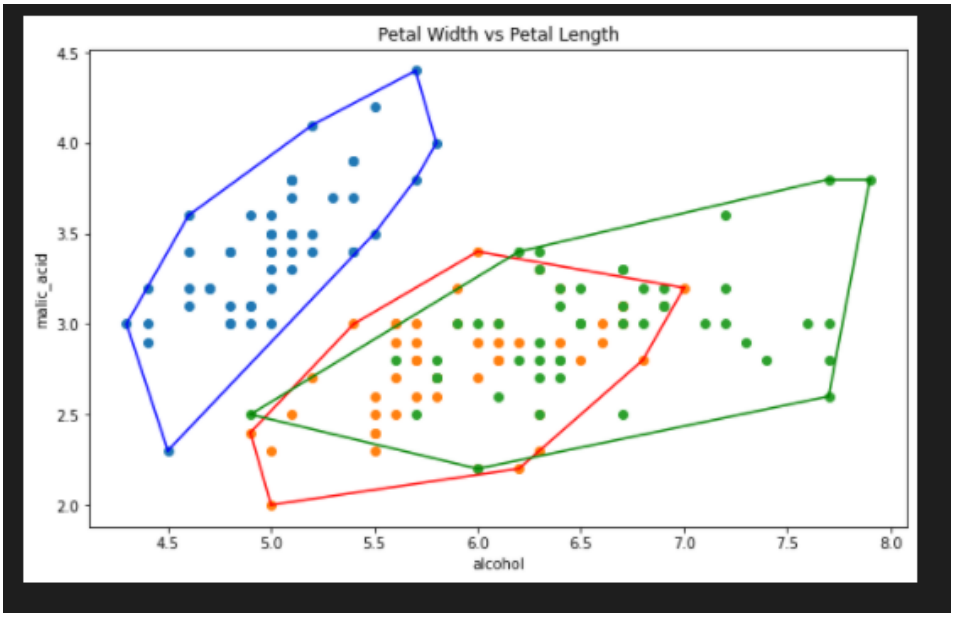
- **Visualisasi ConvexHull 'Petal Width vs Petal Length' pada data Iris**

Input :

```
#visualisasi hasil ConvexHull 'Petal Width vs Petal Length'
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal Length')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])

solution.clear()
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    hull = myConvexHull(bucket)    #bagian ini diganti dengan hasil implementas ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    if (i==0) :
        j = 0
    while (j <len(hull)):
        plt.plot(hull[j],hull[j+1], colors[i])
        j += 2
    hull.clear()
```

Output:



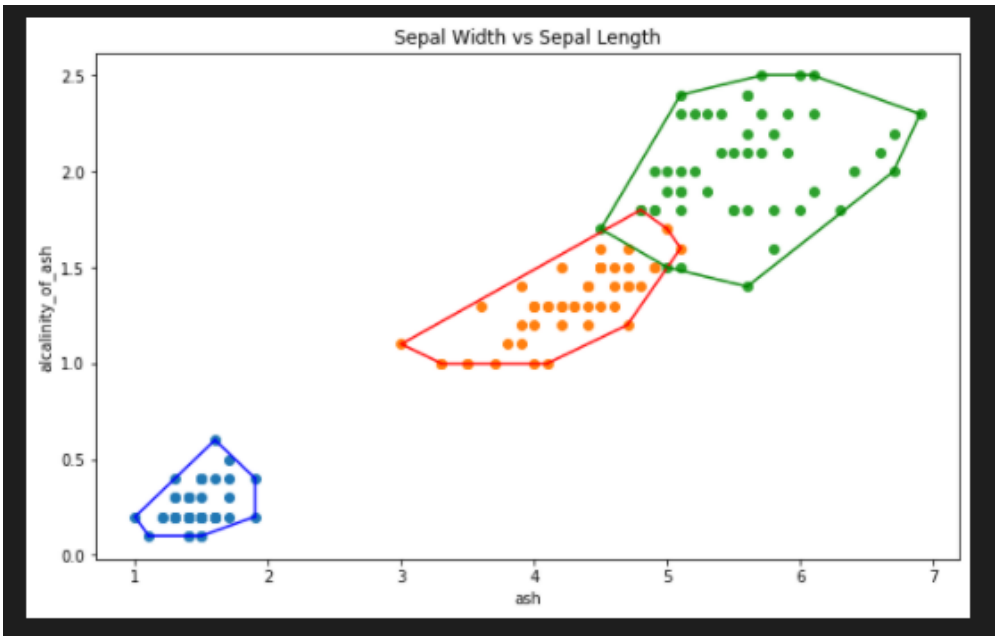
- **Visualisasi ConvexHull ‘Sepal Width vs Sepal Length' pada data Iris**

Input :

```
#visualisasi hasil ConvexHull 'Sepal Width vs Sepal Length' pada Data Iris
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Sepal Width vs Sepal Length')
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])

solution.clear()
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[2,3]].values
    hull2 = myConvexHull(bucket)    #bagian ini diganti dengan hasil implementas ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    if (i==0) :
        j = 0
    while (j <len(hull2)):
        plt.plot(hull2[j],hull2[j+1], colors[i])
        j += 2
    hull2.clear()
```

Output:



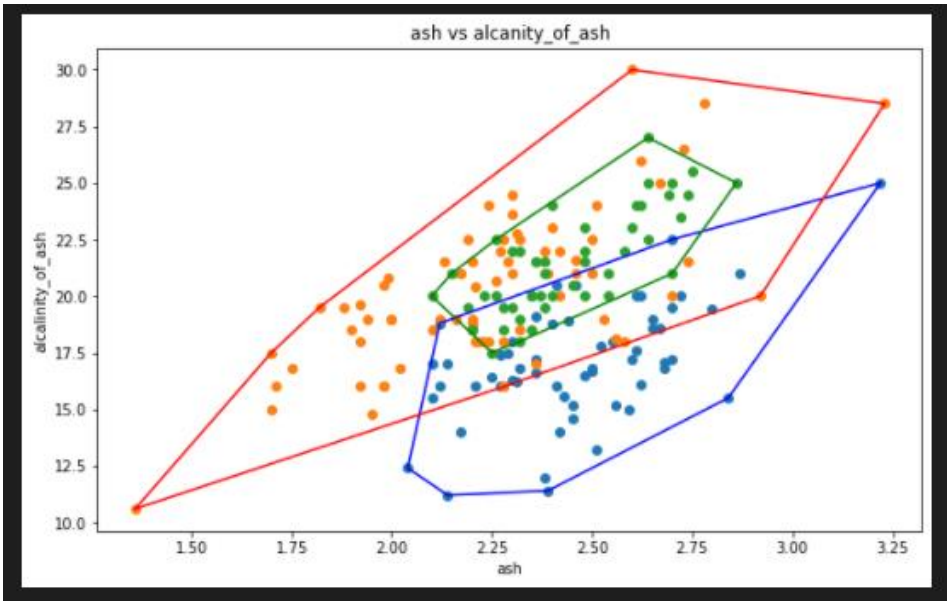
- **Visualisasi ConvexHull 'ash vs alcanity_of_ash' pada Data Wine (Bonus)**

Input :

```
#visualisasi hasil ConvexHull 'ash vs alcanity_of_ash' pada Data Wine
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('ash vs alcanity_of_ash')
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])

solution.clear()
for i in range(len(data.target_names)):
    bucket = df2[df2['Target'] == i]
    bucket = bucket.iloc[:,[2,3]].values
    hull2 = myConvexHull(bucket)    #bagian ini diganti dengan hasil implementas ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    if (i==0) :
        j = 0
    while (j <len(hull2)):
        plt.plot(hull2[j],hull2[j+1], colors[i])
        j += 2
    hull2.clear()
```

Output:



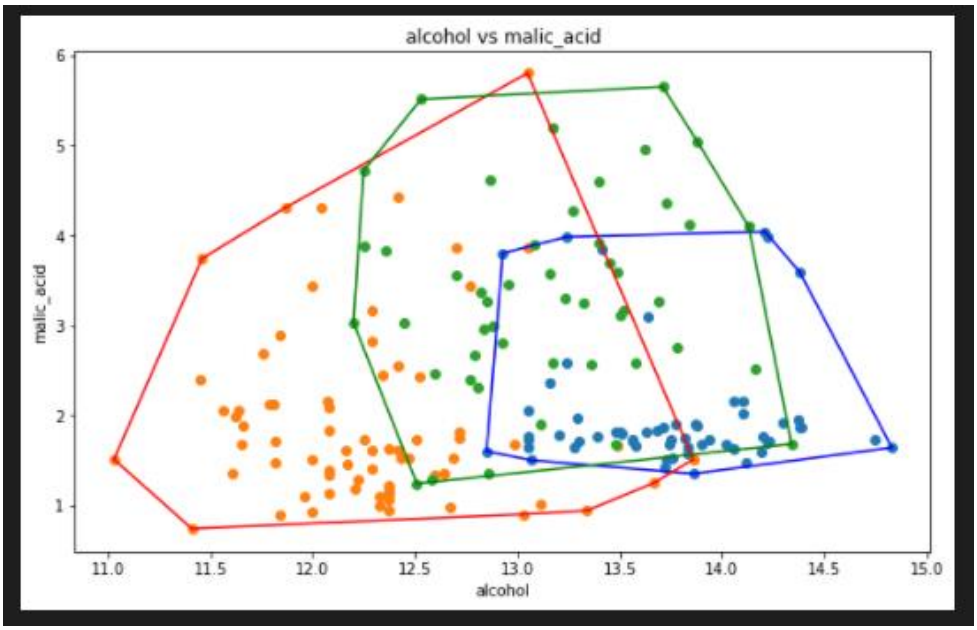
- **Visualisasi ConvexHull 'alcohol vs malic_acid' pada Data Wine (Bonus)**

Input :

```
#visualisasi hasil ConvexHull 'alcohol vs malic_acid' pada Data Wine
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('alcohol vs malic_acid')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])

solution.clear()
for i in range(len(data.target_names)):
    bucket = df2[df2['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    hull2 = myConvexHull(bucket)    #bagian ini diganti dengan hasil implementas ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    if (i==0) :
        j = 0
    while (j <len(hull2)):
        plt.plot(hull2[j],hull2[j+1], colors[i])
        j += 2
    hull2.clear()
```

Output:



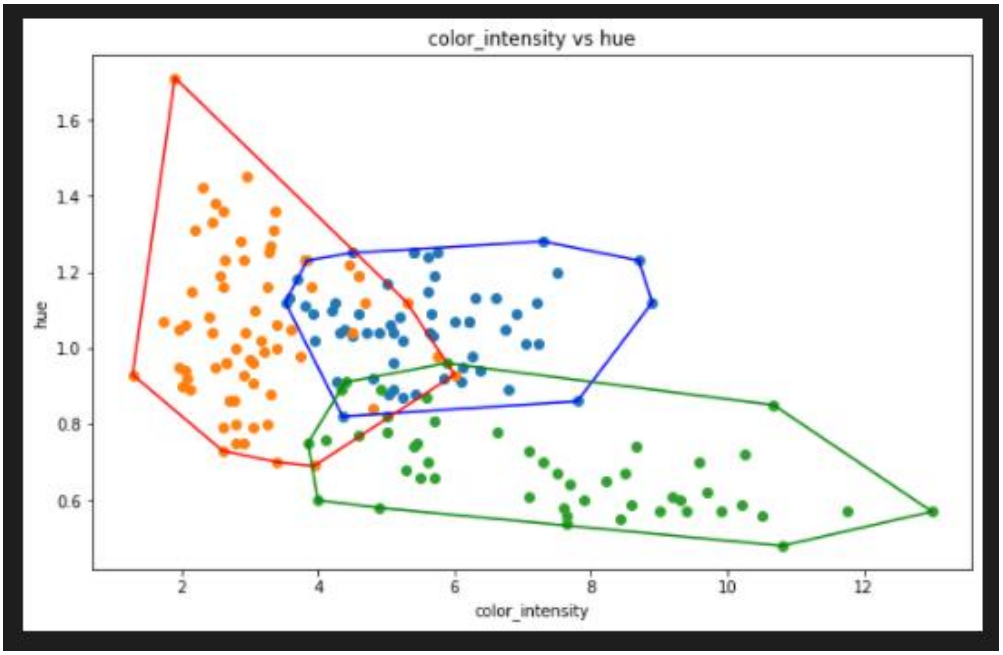
- **Visualisasi ConvexHull 'color_intensity vs hue' pada Data Wine (Bonus)**

Input :

```
#visualisasi hasil ConvexHull 'color_intensity vs hue' pada Data Wine
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('color_intensity vs hue')
plt.xlabel(data.feature_names[9])
plt.ylabel(data.feature_names[10])

solution.clear()
for i in range(len(data.target_names)):
    bucket = df2[df2['Target'] == i]
    bucket = bucket.iloc[:,[9,10]].values
    hull2 = myConvexHull(bucket)    #bagian ini diganti dengan hasil implementas ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    if (i==0) :
        j = 0
    while (j <len(hull2)):
        plt.plot(hull2[j],hull2[j+1], colors[i])
        j += 2
    hull2.clear()
```

Output:



Link Github Program

https://github.com/Fnhafiz/Tucil2_13520027.git

Drive Program (Alternative Link)

https://drive.google.com/drive/folders/1G72EtEkSj_kkI59dl0uMNbODCVSEc8aN?usp=sharing

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