Tugas Kecil 2 IF 2211 Strategi Algortima

Aplikasi Algoritma *Divide and Conquer* untuk Membuat Suatu Program untuk Mencari *Convex Hull* dari Suatu Dataset

Dibuat oleh :

Nama : Timothy Stanley Setiawan

NIM : 13520028

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

Daftar Isi

BAB 1: Algoritma *Divide and Conquer*

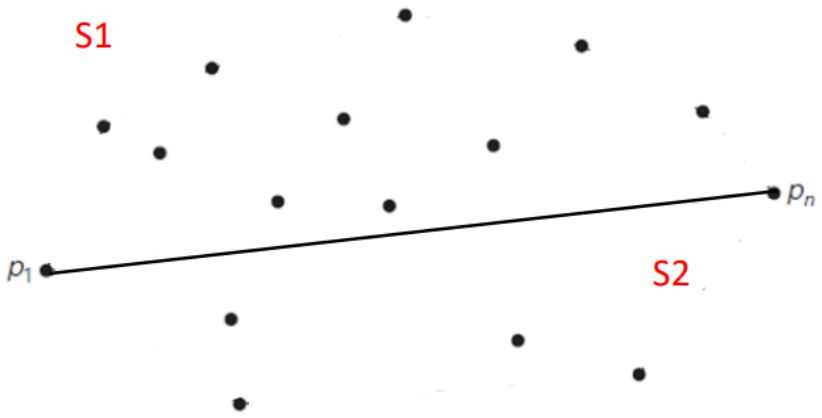
Pada tucil 2 mata kuliah strategi algoritma, penulis memanfaatkan algoritma *Divide and Conquer* dalam pembentukan *convex hull* dari persebaran data pada suatu dataset. Berikut ini penjelasan algoritam yang penulis gunakan.

1. *Prerequisite*, untuk mencari p1 dan p2 yang merupakan dua titik ekstrem yang akan membentuk *convex hull*. Data harus di urutkan terlebih dahulu berdasarkan nilai absis yang menaik, kemudian diikuti dengan nilai ordinat yang menaik (jika ada nilai absis yang sama) sehingga p1satu merupakan data pertama dan p2 adalah data terakhir dari hasil pengurutan.
2. Pertama, misalnya suatu persebaran data berada pada suatu bidang S, bagi bidang S tersebut dibagi menjadi dua bagian, yaitu atas (S1) dan bawah (S2) dari garis p1p2. Untuk memeriksa suatu titik berada pada bagian mana, penulis memanfaatkan nilai determinan,

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

Gambar 1. Detereminan untuk Menentukan Letak Titik

yaitu $\det = x1y2 + x3y1 + x2\,y3 - x3y2 - x2y1 - x1y3$. Titik p3 (x3,y3) berada di atas garis p1(x1,y1) p2(x2,y2) jika hasil determinannya positif

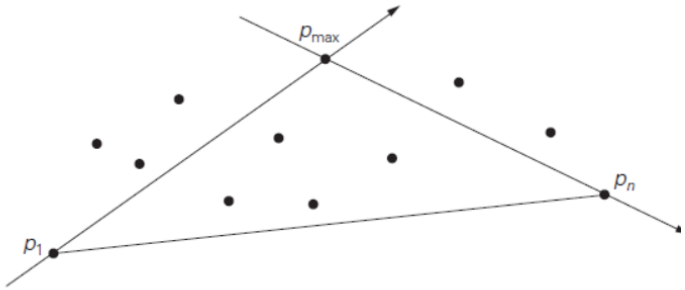


Gambar 2. Ilustrasi Langkah 1

3. Kedua, abaikan titik-titik yang berada pada garis p1p2 (selain titik p1 dan p2) karena titik-titik tersebut tidak mungkin dapat membentuk *convex hull*. Ciri titik yang berada pada garis p1p2 adalah nilai determinannya 0. Titik-titik yang terletak pada S1 atau S2 akan diproses lagi karena memiliki kemungkinan untuk membentuk *convexhull*
4. Ketiga, suatu bagian (misalnya S1) ada dua kemungkinan pemrosesan:
   a. Jika S1 kosong, titik p1 dan pn akan menjadi pasangan titik pembentuk untuk *convex hull* bagian S1.

b. Jika S1 tidak kosong, ambil suatu titik (pmax) yang memilik jarak terjauh dari garis p1pn. Jika ada dua titik yang memilik jarak yang sama, ambil titik yang menghasilkan sudut p1-pmax-pn yang paling besar.

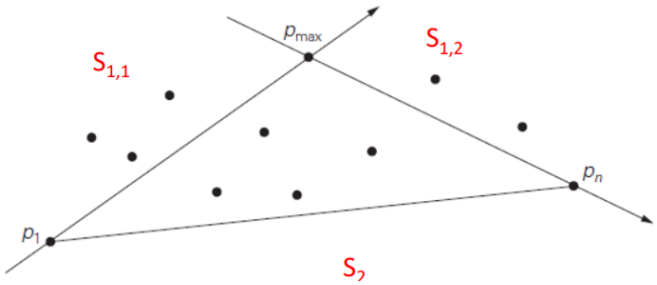
Titik-titik yang berada di dalam segitiga p1-pmax-pn bisa diabaikan karena tidak mungkin bisa membentuk *convex hull*.



Gambar 2. Ilustrasi Langkah 3

catatan : Untuk menentukan jarak titik ke garis, misalkan garis p1pn memiliki persamaan $ax + by + c = 0$, *jaraknya adalah* $S = \frac{|a*xn + b*yn + c|}{\sqrt{a^2+b^2}}$ *dengan xn adalah titik-titik pada S1. Untuk menentukan sudut p1-pm-pn, misalkan* $\overrightarrow{AB} = pm - p1$ *dan* $\overrightarrow{BC} = pn - pm$, *sudutnya adalah* $cos^{-1}\frac{AB.BC}{||AB||\,||BC|}$.

5. Keempat, kumpulkan titik-titik yang berada di atas garis p1pmax (S1,1) dan sebelah atas garis pmaxpn (S1,2)



Gambar 4. Ilustrasi Langkah 4

6. Ulangi, langkah kedua dan ketiga untuk bagian S2 hingga seluruh bagian menjadi kosong. (titik-titik yang dikumpulkan adalah kebalikan dari bagian S1, yaitu di bawah garis p1pmax2 dan sebelah bawah pmax2pn dengan pmax2 adalah titik terjauh dari garis p1pn dari kumpulan titik pada S2.
7. Gabungkan kembali semua pasangan titik yang membentuk *convex hull* pada bagiannya masing-masing

BAB 2: *Source Code Program* dalam bahasa Python

2.1. main.py

```
from pandas import DataFrame
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits, load_iris, load_wine, load_breast_cancer
from ConvexHull import myConvexHull

if __name__ == "__main__":
    while True:
        print("Pilihan Database : ")
        print("1. Dataset Iris")
        print("2. Dataset Wine")
        print("3. Dataset Breast Cancer")
        print("4. Dataset Digits")
        print("5. Keluar")
        print("==================================")
        # Memilih database yang ingin dibuat convexhullnya
        choices = int(input("Masukan dataset (1/2/3/4) : "))

        if (choices < 1 or choices > 4):
            break

        # Melakukan load database yang dipilih
        match choices:

            case 1:
                print("Pilihan Kolom : ")
                print("1. Sepal-length vs Sepal-width")
                print("2. Petal-length vs Petal-width")
                data = load_iris()
                coulom = int(input("Masukan kolom dataset (1/2) : "))
                if (coulom == 1):
                    n, m = 0, 1
                elif (coulom == 2):
                    n, m = 2,3

            case 2:
                data = load_wine()
                n = int(input("Masukan kolom1 dataset (0-12) : "))
                while True:
                    m = int(input("Masukan kolom2 dataset (0-12) excepet " + str(n) +
") : "))
                    if (n != m):
                        break
```

```
            case 3:
                data = load_breast_cancer()
                n, m = 0, 1
                n = int(input("Masukan kolom1 dataset (0-29) : "))
                while True:
                    m = int(input("Masukan kolom2 dataset (0-29) excepet " + str(n) +
") : "))
                    if (n != m):
                        break

            case 4:
                data = load_digits()
                n, m = 9, 10
                n = int(input("Masukan kolom1 dataset (0-64) : "))
                while True:
                    m = int(input("Masukan kolom2 dataset (0-64) excepet " + str(n) +
" : "))
                    if (n != m):
                        break

        # Membuat Data Frame
        dataframe = DataFrame(data.data, columns=data.feature_names)
        dataframe['Target'] = DataFrame(data.target)
        dataframe.head()

        # Visualisasi convexhull
        plt.figure(figsize = (10, 6))
        colors = ['b','r','g','y','c','m','y','k','bisque','gold','slategrey']
        plt.title(data.feature_names[n] + ' vs ' + data.feature_names[m])
        plt.xlabel(data.feature_names[n])
        plt.ylabel(data.feature_names[m])
        for i in range(len(data.target_names)):
            bucket = dataframe[dataframe['Target'] == i]
            bucket = bucket.iloc[:,[n,m]].values
            hull = myConvexHull(bucket)
            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%5])
        plt.legend()
        plt.show()
        print("====================================")
```

2.2. myConvexHull.py

```
from math import sqrt, acos, pi
import numpy as np
```

```
# Fungsi untuk mencari sudut
def angle(p1,pcheck,p2):
    x1, x2, x3 = p1[0], p2[0], pcheck[0]
    y1, y2, y3 = p1[1], p2[1], pcheck[1]
    # Mencari vektor AB dan BC
    AB = [x3-x1,y3-y1]
    BC = [x2-x3,y2-x2]
    # Mencari ||AB|| dan ||BC||
    nomAB = sqrt(AB[0]*AB[0]+AB[1]*AB[1])
    nomBC = sqrt(BC[0]*BC[0]+BC[1]*BC[1])
    # Mencegah perhitungan n/0
    if (nomAB != 0 and nomBC != 0):
        # Mencari sudut dari rumus sudut = arc_cos(AB.BC/(||AB||*||BC||))
        valuecos = (AB[0]*BC[0] + AB[1]*BC[1])/(nomAB*nomBC)
        return acos(valuecos)*180/pi
    return 0

# Fungsi untuk mencari distance
def distance(p1,pcheck,p2):
    x1, x2, x3 = p1[0], p2[0], pcheck[0]
    y1, y2, y3 = p1[1], p2[1], pcheck[1]
    #Mencari persamaan ax + by + c = 0
    a, b, c = -(y2-y1), (x2-x1), -(x2-x1)*y1 + (y2-y1)*x1
    # Rumus mencari distance dari persamaan garis ax + by + c = 0
    # Distance = abs(a*x + b*y + c)/sqrt(a*a + b*b)
    return abs(a*x3 + b*y3 + c)/sqrt(a*a + b*b)

# Fungsi untuk membagi bagian S menjadi S1 dan S2 berdasarkan garis p1p2
def divide(S,p1,p2):
    S1 = [] #sebelah atas
    S2 = [] #sebelah bawah
    x1, y1, x2, y2 = p1[0], p1[1], p2[0], p2[1]
    # p1 dan p2 menunjuk titik yang sama
    if (p1[0] == p2[0] and p1[1] == p2[1]):
        return S1, S2
    # Membagi bagian S dengan memanfaatkan determinan :
    # |x1 y1 1|
    # |x2 y2 1|
    # |x3 y3 1|
    for point in S:
        x3, y3 = point[0],point[1]
        det = x1*y2 + x3*y1 + x2*y3 - x3*y2 - x2*y1 - x1*y3
        # Determinan positif berarti titik berada di atas garis p1p2 (S1)
        if (det > 0):
            S1.append(point)
        # Determinan negatif berarti titik berada di bawah garis p1p2 (S2)
        elif (det < 0):
```

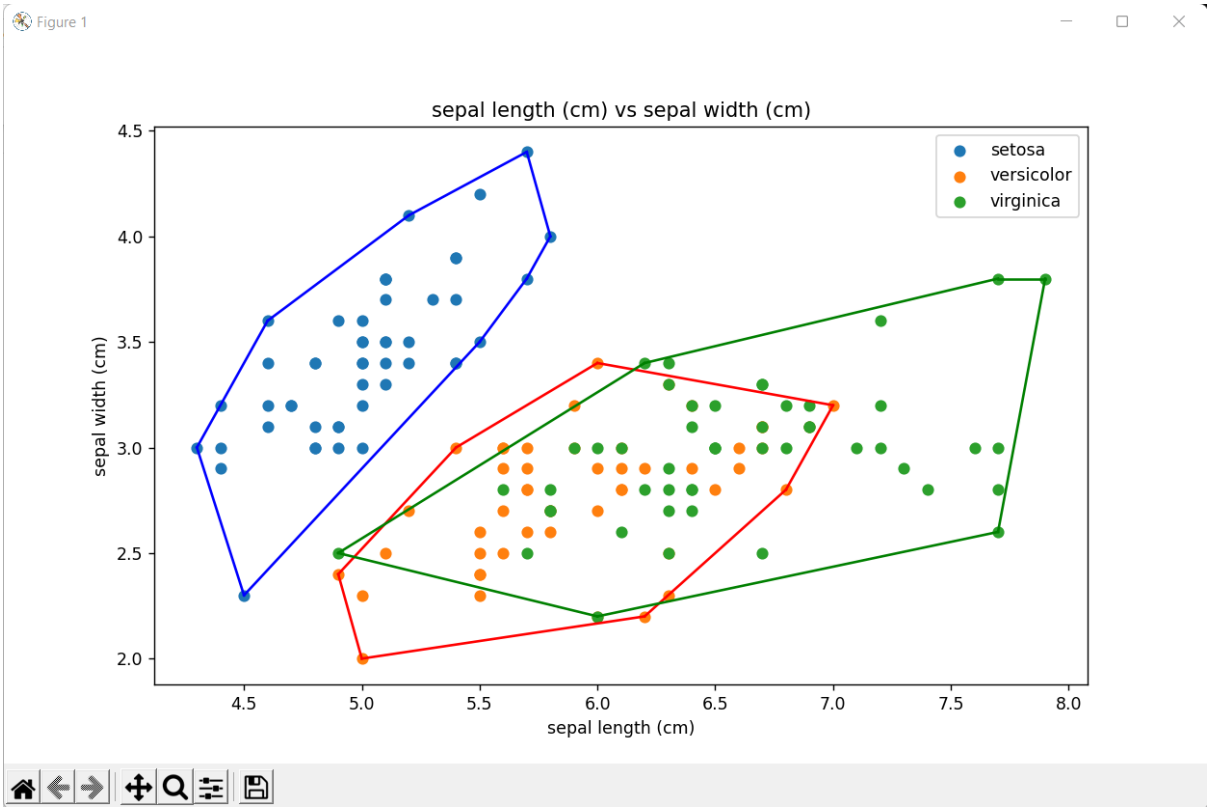
```
            S2.append(point)
        # Determinan nol berarti titik berada di garis p1p2 (tidak masuk S1 atau pun S2)
    return S1, S2

# Fungsi rekursif convexhull (Algoritma quickhull)
def quickhull(S,p1,p2,above):
    convex_hull = []
    # Apabila S kosong (tidak ada titik lain selain S)
    # p1 dan p2 menjadi bentuk convexhull
    if (S == []):
        convex_hull.append([p1.tolist(),p2.tolist()])
        return convex_hull
    # Mencari jarak terjauh dari garis p1p2
    far_point = S[0]
    far_distance = distance(p1,far_point,p2)
    far_angle = angle(p1,far_point,p2)
    n = 0
    for i, point in enumerate(S):
        temp_dist = distance(p1,point,p2)
        if (far_distance < temp_dist):
            far_point, far_distance, n = point, temp_dist, i
        # Apabila ada dua titik yang memiliki jarak yang sama
        # Mencari titik yang menghasilkan sudut terbesar
        elif (far_distance == temp_dist):
            temp_angle = angle(p1,point,p2)
            if (far_angle < temp_angle):
                far_point, far_distance, far_angle, n = point, temp_dist, temp_angle, i

    # Menghapus titik terjauh dari S
    S.pop(n)

    # Membagi S1 menjadi S11 (atas S1) dan S12 (bawah S1) berdasarkan garis p1far_point
    S11, S12 = divide(S,p1,far_point)
    # Membagi S2 menjadi S21 (atas S2) dan S22 (bawah S2) berdasarkan garis far_pointp2
    S21, S22 = divide(S,far_point,p2)

    # Lakukan rekursif untuk mengecek bagian yang sama
    if (above):
        # Bagian atas (S1) hanya melakukan pengecekan bagian atas saja (S11, S21)
        convex_hull += quickhull(S11,p1,far_point,True)
        convex_hull += quickhull(S21,far_point,p2,True)
    elif (not above) :
        # Bagian atas (S2) hanya melakukan pengecekan bagian atas saja (S12, S22)
        convex_hull += quickhull(S12,p1,far_point,False)
        convex_hull += quickhull(S22,far_point,p2,False)
    return convex_hull
```

```
# Fungsi mencari convexhull
def convexhull(bucket):
    convex_hull = []
    index_convex_hull = []
    # Melakukan sorting dengan key data di x kemudian di y untuk mencari p1 dan p2 (titik terjauh)
    # Perkiraan komplesitas O(log n)/ quicksort
    sort = sorted(bucket,key=lambda x:(x[0],x[1]))

    # Menghapus titik-titik terjauh
    p1 = sort.pop(0)
    p2 = sort.pop(-1)

    # Membagi S menjadi S1 (atas S) dan S2 (bawah S) berdasarkan garis p1p2
    S1, S2 = divide(sort,p1,p2)

    # Rekursif convexhull bagian atas (S1)
    convex_hull += quickhull(S1,p1,p2,True)
    # Rekursif convexhull bagian atas (S1)
    convex_hull += quickhull(S2,p1,p2,False)

    # Melakukan searching terhadap index dari titik convexhull yang terbentuk
    for hasil in convex_hull:
        for i, content in enumerate(bucket):
            if (content.tolist() == hasil[0]):
                break
        for j, content in enumerate(bucket):
            if (content.tolist() == hasil[1]):
                break
        index_convex_hull.append([i,j])

    # Mengubah list menjadi nparray
    if (index_convex_hull != []):
        index_convex_hull = np.vstack(index_convex_hull)

    return index_convex_hull
```

BAB 3: *Screen Shot* Input Output Program

3.1. *Dataset Iris* kolom *Sepal-length vs Sepal-width*

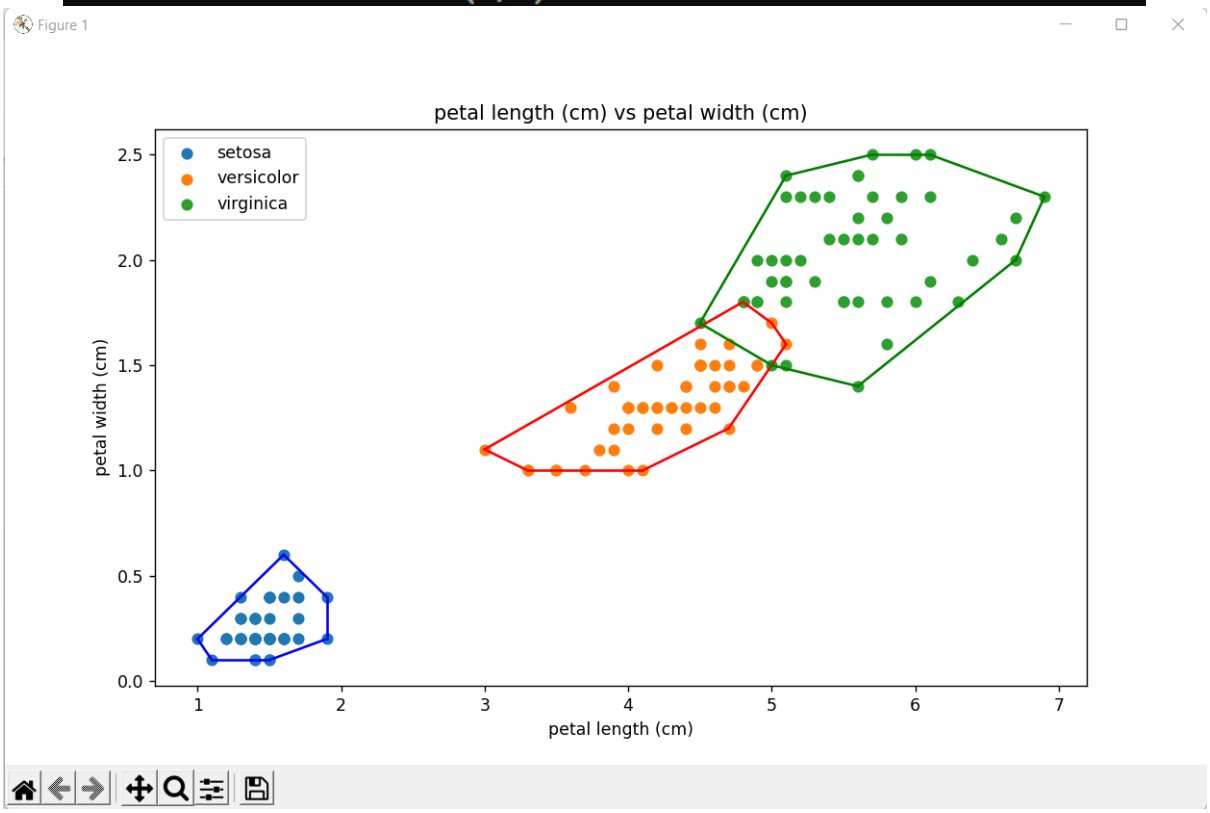
```
Pilihan Database :
1. Dataset Iris
2. Dataset Wine
3. Dataset Breast Cancer
4. Dataset Digits
5. Keluar
==================================
Masukan dataset (1/2/3/4) : 1
Pilihan Kolom :
1. Sepal-length vs Sepal-width
2. Petal-length vs Petal-width
Masukan kolom dataset (1/2) : 1
```



Gambar 5. Input dan Output Dataset Iris kolom Sepal-length vs Sepal-width

3.2. *Dataset Iris* kolom *Petal-length vs Petal-width*

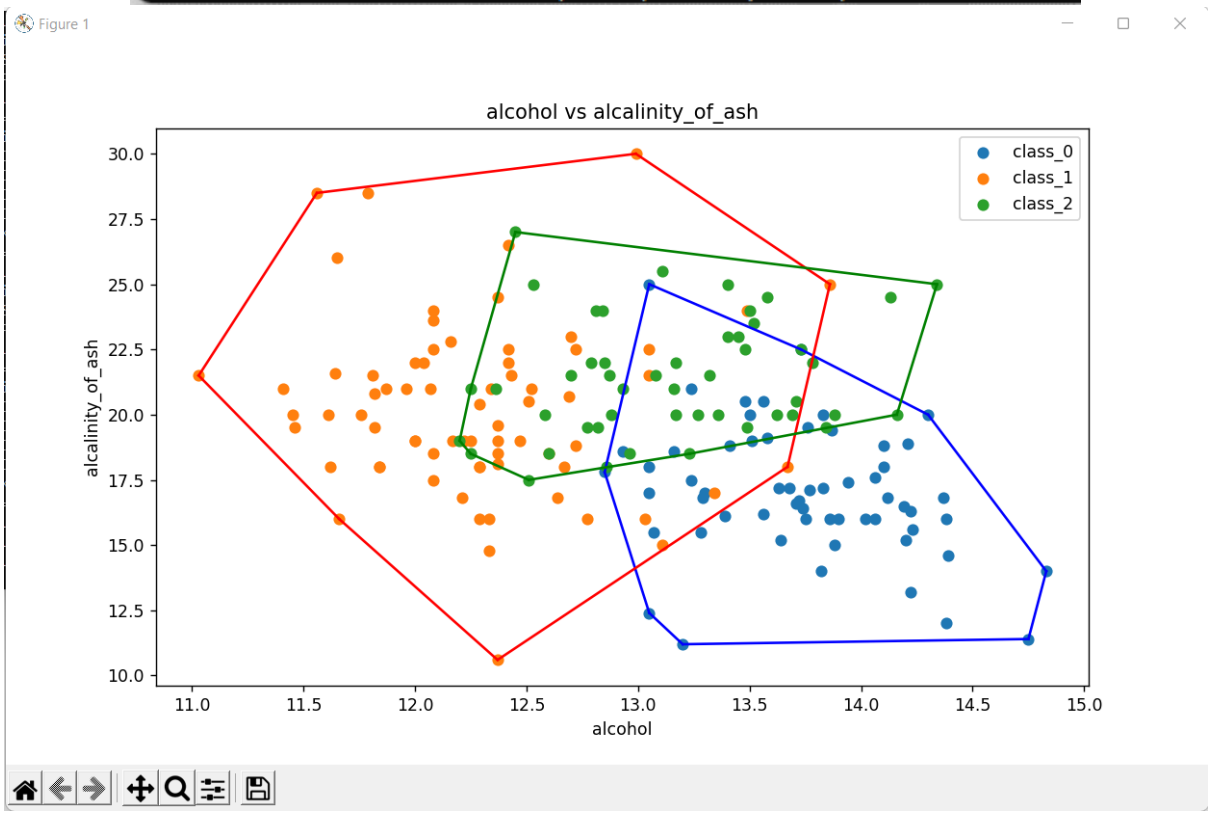
```
==================================
Pilihan Database :
1. Dataset Iris
2. Dataset Wine
3. Dataset Breast Cancer
4. Dataset Digits
5. Keluar
==================================
Masukan dataset (1/2/3/4) : 1
Pilihan Kolom :
1. Sepal-length vs Sepal-width
2. Petal-length vs Petal-width
Masukan kolom dataset (1/2) : 2
```



Gambar 6. Input dan Output Dataset Iris kolom Petal-length vs Petal-width

3.3. *Dataset Wine* kolom *Alcohol vs Alcalinity of Ash*

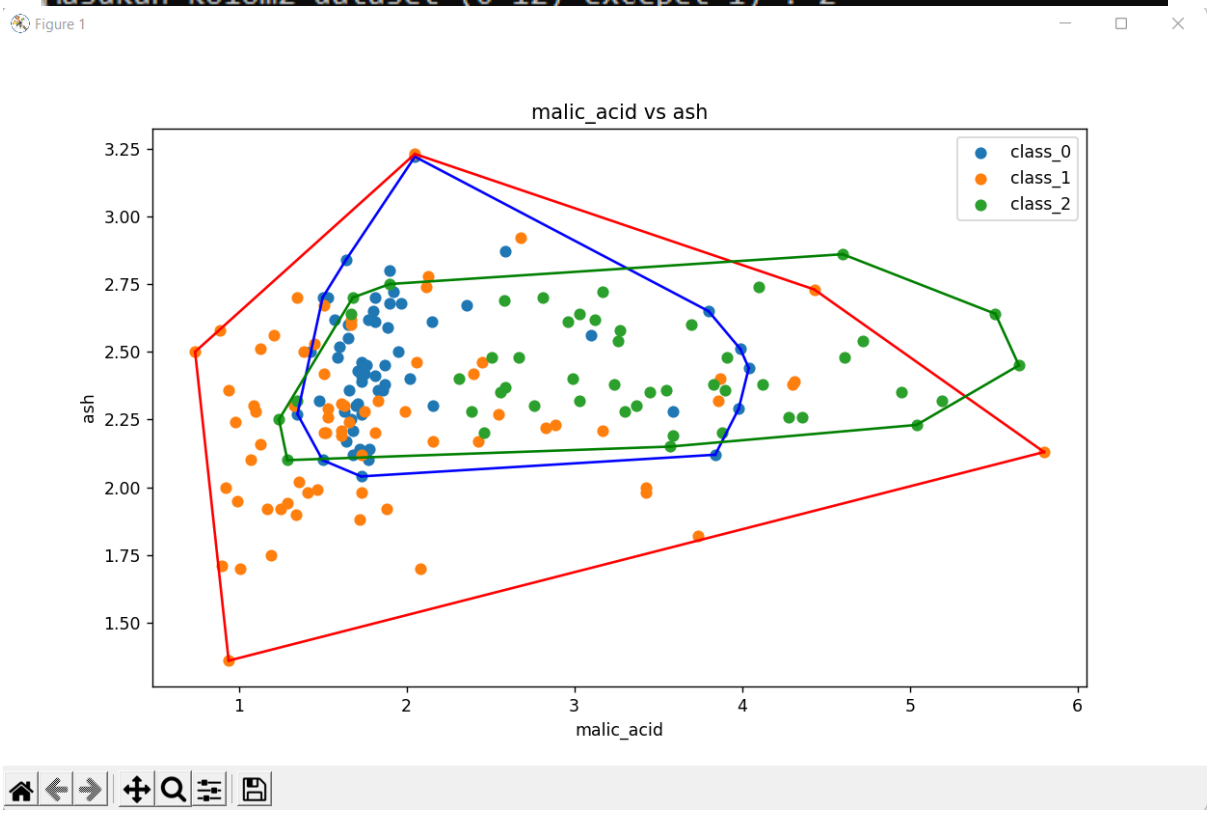
```
==================================
Pilihan Database :
1. Dataset Iris
2. Dataset Wine
3. Dataset Breast Cancer
4. Dataset Digits
5. Keluar
==================================
Masukan dataset (1/2/3/4) : 2
Masukan kolom1 dataset (0-12) : 0
Masukan kolom2 dataset (0-12) excepet 0) : 3
```



Gambar 7. Input dan Output Dataset Wine kolom *Alcohol vs Alcalinity of Ash*

3.4. *Dataset Wine* kolom *Malic Acid vs Ash*

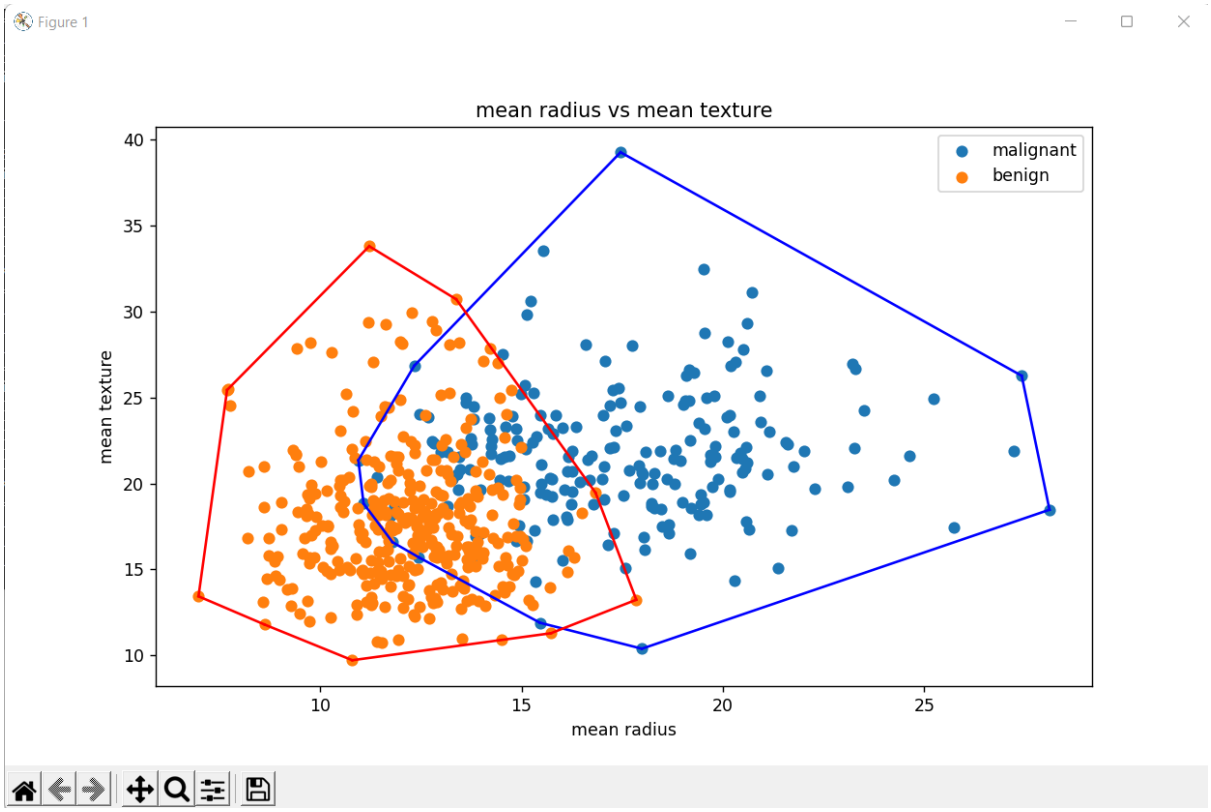
```
Pilihan Database :
1. Dataset Iris
2. Dataset Wine
3. Dataset Breast Cancer
4. Dataset Digits
5. Keluar
==================================
Masukan dataset (1/2/3/4) : 2
Masukan kolom1 dataset (0-12) : 1
Masukan kolom2 dataset (0-12) excepet 1) : 2
```



Gambar 8. Input dan Output Dataset Wine kolom *Malic Acid vs Ash*

3.5. *Dataset Breast Cancer* kolom *Mean Radius vs Mean Texture*

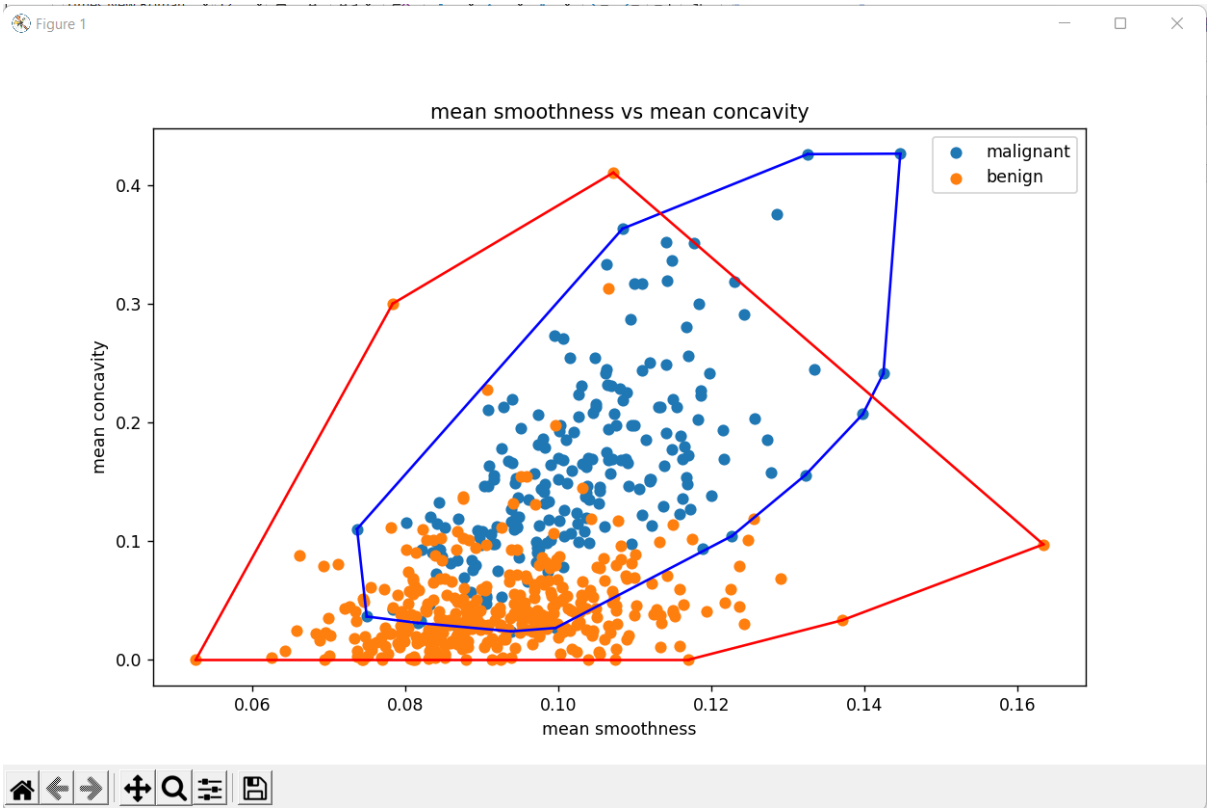
```
==================================
Pilihan Database :
1. Dataset Iris
2. Dataset Wine
3. Dataset Breast Cancer
4. Dataset Digits
5. Keluar
==================================
Masukan dataset (1/2/3/4) : 3
Masukan kolom1 dataset (0-29) : 0
Masukan kolom2 dataset (0-29) excepet 0) : 1
```



Gambar 9. Input dan Output Dataset Breast Cancer kolom *Mean Radius vs Mean Texture*

3.6. *Dataset Breast Cancer* kolom *Mean Smoothness vs Mean Concavity*

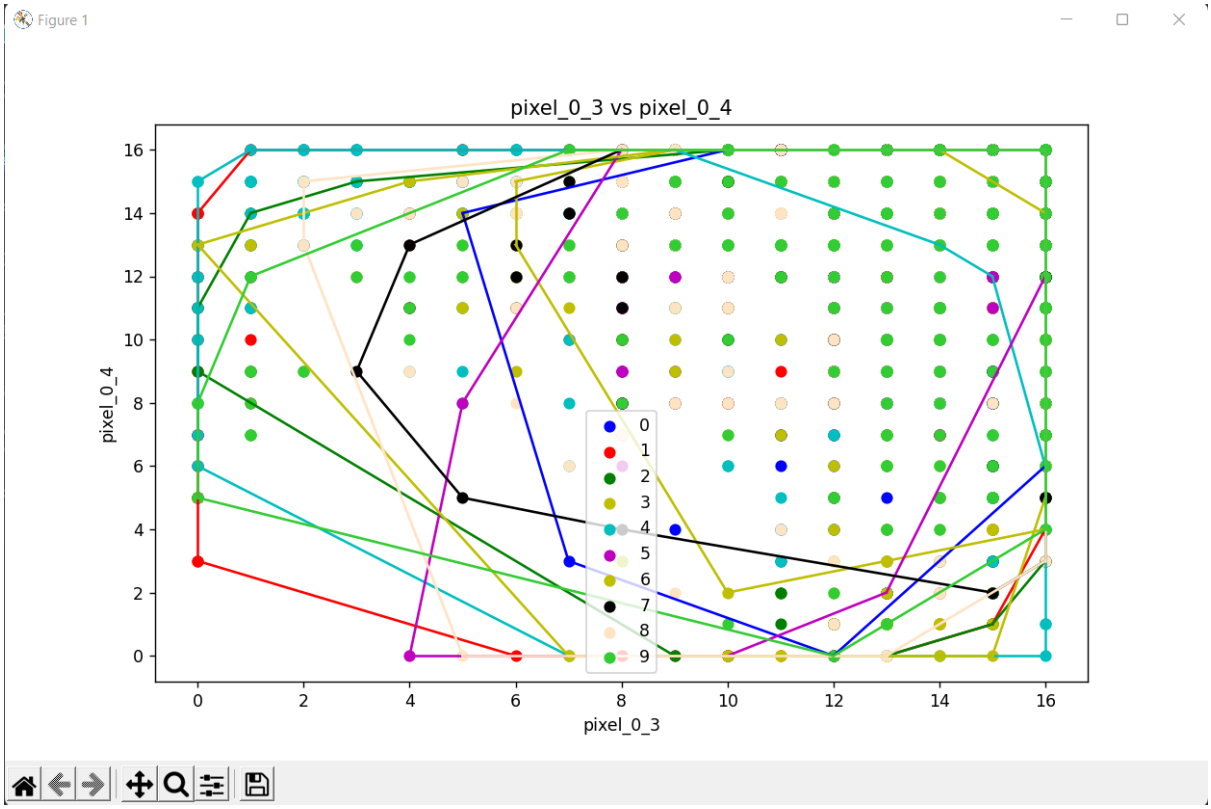
```
==================================
Pilihan Database :
1. Dataset Iris
2. Dataset Wine
3. Dataset Breast Cancer
4. Dataset Digits
5. Keluar
==================================
Masukan dataset (1/2/3/4) : 3
Masukan kolom1 dataset (0-29) : 4
Masukan kolom2 dataset (0-29) excepet 4) : 6
```



Gambar 10. Input dan Output Dataset Breast Cancer kolom *Mean Smoothness vs Mean Concavity*

3.7. *Dataset Digits* kolom *Pixel 03 vs Pixel 04*

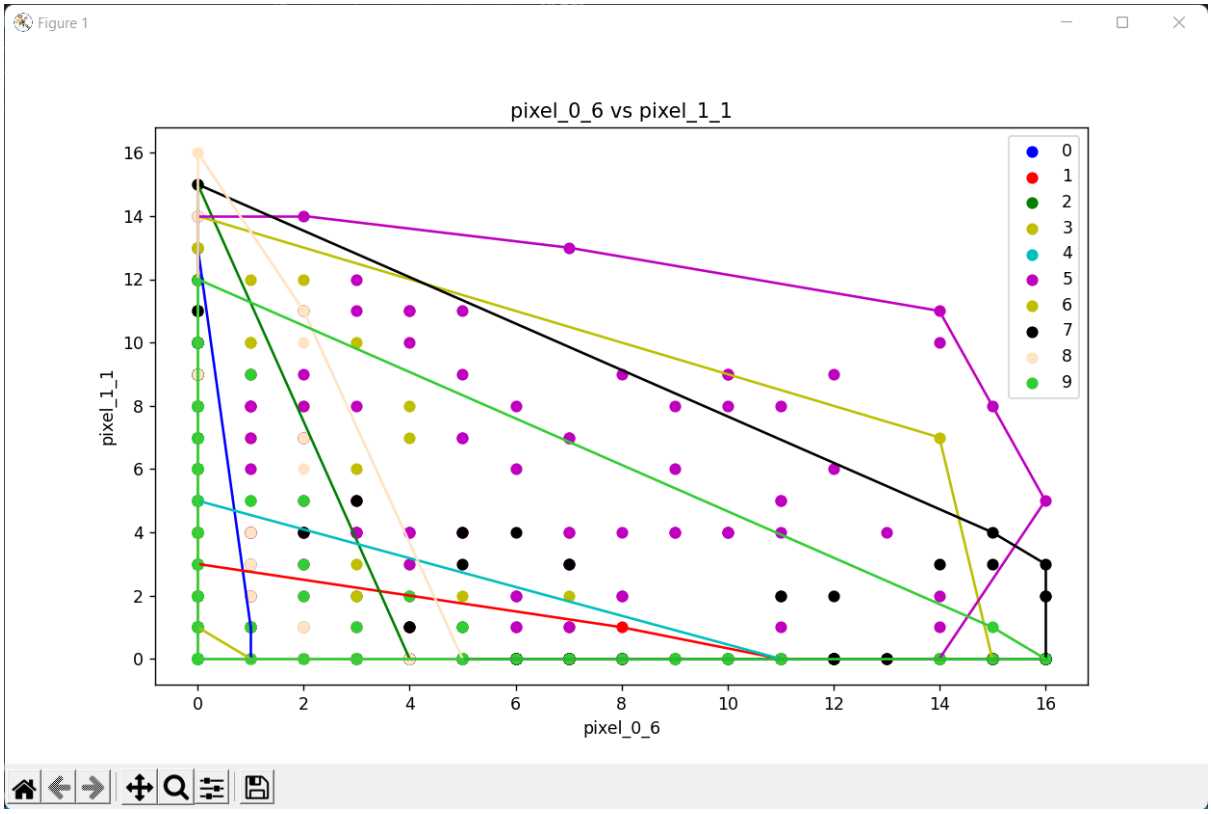
```
==================================
Pilihan Database :
1. Dataset Iris
2. Dataset Wine
3. Dataset Breast Cancer
4. Dataset Digits
5. Keluar
==================================
Masukan dataset (1/2/3/4) : 4
Masukan kolom1 dataset (0-64) : 3
Masukan kolom2 dataset (0-64) excepet 3 : 4
```



Gambar 11. Input dan Output Dataset Digits kolom *Pixel 03 vs Pixel 04*

3.8. *Dataset Digits* kolom *Pixel 06 vs Pixel 11*

```
==================================
Pilihan Database :
1. Dataset Iris
2. Dataset Wine
3. Dataset Breast Cancer
4. Dataset Digits
5. Keluar
==================================
Masukan dataset (1/2/3/4) : 4
Masukan kolom1 dataset (0-64) : 6
Masukan kolom2 dataset (0-64) excepet 6 : 9
```



Gambar 12. Input dan Output Dataset Digits kolom *Pixel 06 vs Pixel 11*

Lampiran

1. Alamat Kode Program (*Github*)

https://github.com/Stanley77-web/Tucil-2-Stima.git

2. Tabel Pengujian

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.	√	