IMPLEMENTASI *CONVEX HULL* UNTUK VISUALISASI TES *LINEAR SEPARABILITY DATASET* DENGAN ALGORITMA *DIVIDE AND CONQUER*

LAPORAN TUGAS KECIL 2

Diajukan untuk memenuhi Tugas Kecil 2
IF2211 Strategi Algoritma

Oleh

Damianus Clairvoyance Diva Putra 13520035

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022

BAB I
ALGORITMA

Algoritma Divide and Conquer adalah algoritma yang memecah persoalan menjadi sejumlah upapersoalan yang berukuran lebih kecil, tetapi memiliki kemiripan dengan persoalan semula, lalu menyelesaikan setiap upapersoalan tersebut secara langsung atau rekursif kembali, kemudian menggabungkan solusi setiap upapersoalan menjadi solusi persoalan semula. Apabila diberikan sejumlah titik pada suatu bidang planar, himpunan titik disebut *convex* bila untuk sembarang dua titik pada bidang tersebut (misal p dan q), seluruh segmen garis yang berakhir di p dan q berada pada himpunan tersebut.

Secara umum, algoritma Divide and Conquer dalam program ini mengimplementasikan algoritma dalam Slide Bahan Kuliah IF2211 Strategi Algoritma – Algoritma Divide and Conquer (Bagian 4), tetapi dengan sedikit penyesuaian. Berikut merupakan langkah memperoleh kumpulan titik yang membentuk Convex Hull tersebut.

1. Dari kumpulan n titik (misal bucket), temukan titik paling kiri (misal leftmost) dan kanan (rightmost)
2. Garis leftmost-rightmost membagi kumpulan titik menjadi dua bagian, yaitu atas (misal upperBucket) dan bawah (misal lowerBucket).
3. Untuk setiap titik di upperBucket, temukan titik terjauh dari garis leftmost-rightmost (misal furthest).
4. Kumpulan titik yang berada di atas leftmost-furthest (misal leftBucket) dan atas furthest-rightmost (misal rightBucket) akan diproses kembali secara rekursif.
    a. Apabila bucket masih berisi titik, temukan titik terjauh dan bagi lagi menjadi leftBucket dan rightBucket masing-masing.
    b. Apabila bucket tidak berisi titik (kosong), tambahkan titik leftmost dalam bucketCHull.
5. Tambahkan titik rightmost dalam bucketCHull.
6. Untuk setiap titik di lowerBucket, temukan titik terjauh dari garis leftmost-rightmost (misal furthest).
7. Kumpulan titik yang berada di bawah furthest-rightmost (misal leftBucket) dan bawah leftmost-furthest (misal rightBucket) akan diproses kembali secara rekursif.
    a. Apabila bucket masih berisi titik, temukan titik terjauh dan bagi lagi menjadi leftBucket dan rightBucket masing-masing.
    b. Apabila bucket tidak berisi titik (kosong), tambahkan titik leftmost dalam bucketCHull.

Konsep determinan tidak digunakan dan diganti dengan konsep penentuan apakah suatu titik berada di atas atau di bawah garis untuk mempermudah perhitungan.

BAB II
SOURCE CODE PROGRAM

Tugas kecil ini diselesaikan dengan Bahasa Python. Kakas (*library*) yang dimanfaatkan ialah math (bawaan Python), `numpy`, `pandas`, `matplotlib`, dan `sklearn`, yang penggunaannya dijabarkan sebagai berikut.

- `numpy`, sebagai kakas komputasi numerikal di Python, digunakan untuk membangun array, baik array *tuple* bilangan desimal (disebut point) maupun array point.
- `pandas`, sebagai kakas analisis dan manipulasi data di Python, digunakan membangun *dataframe* dari *dataset*.
- `matplotlib.pyplot`, sebagai kakas visualisasi di Python, digunakan untuk membangun dan mengatur hasil visualisasi Convex Hull.
- `sklearn` (`scikit-learn`), sebagai kakas analisis data prediktif di Python, digunakan untuk memperoleh *datasets* untuk *test case*, yakni iris, wine, dan breast cancer.

Penggunaan kakas tersebut dilakukan dengan kode berikut pada bagian atas program.

```
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
import matplotlib.pyplot as plt
from myConvexHull import ConvexHull # kakas hasil buatan sendiri
```

Perancangan program ini dilakukan dalam *environment* conda dengan Python 3.9.7. Apabila pengguna hendak menjalankan program, seluruh kakas di atas harus terinstalasi terlebih dahulu dalam *environment*, dengan menjalankan kode berikut.

```
conda install -c conda-forge numpy
conda install -c conda-forge pandas
conda install -c conda-forge matplotlib
conda install -c conda-forge scikit-learn
```

Berikut merupakan fungsi/prosedur pendukung yang dibuat dalam kakas myConvexHull. Penjelasan tiap fungsi/prosedur dituliskan dalam bentuk komentar dalam kode program.

A. Fungsi `findLeftMost`

```
def findLeftMost(bucket):
    # masukan: bucket berisi array of points, dengan point berupa tuple (x, y)
    # keluaran: point paling kiri berdasarkan absis dan ordinat
```

```
    leftmost = bucket[0]
    for point in bucket:
        if (point[0] < leftmost[0]):
            leftmost = point
        elif (point[0] == leftmost[0] and point[1] < leftmost[1]):
            leftmost = point
    return leftmost
```

B. Fungsi `findRightMost`

```
def findRightMost(bucket):
    # masukan: bucket berisi array of points, dengan point berupa tuple (x, y)
    # keluaran: point paling kanan berdasarkan absis dan ordinat
    rightmost = bucket[0]
    for point in bucket:
        if (point[0] > rightmost[0]):
            rightmost = point
        elif (point[0] == rightmost[0] and point[1] > rightmost[1]):
            rightmost = point
    return rightmost
```

C. Fungsi `findEquation`

```
def findEquation(x1, y1, x2, y2):
    # masukan: dua titik (x1, y1) dan (x2, y2) pada suatu garis l
    # luaran: a, b, dan c dalam persamaan garis 1
    # persamaan garis 1: ax + by + c = 0
    # persamaan garis 2: y - y1 = m(x - x1) = mx - mx1
    #                    mx - y + (y1 - mx1) = 0
    #              maka, a = m, b = -1, c = y1 - mx1
    m = (y2-y1)/(x2-x1)
    return m, -1, y1 - m*x1
```

D. Fungsi `findY`

```
def findY(x1, y1, x2, y2, x):
    # masukan: dua titik (x1, y1) dan (x2, y2) pada suatu garis l, serta nilai x
    # luaran: nilai y dari nilai x pada garis l
    # rumus gradien: m = (y2-y1)/(x2-x1)
    # rumus cari y: y = m(x - x1) + y1
```

```
    if (x1 != x2):
        m = (y2-y1)/(x2-x1)
        return m*(x - x1) + y1
    else:
        return 0
```

E. Fungsi `findDistance`

```
def findDistance(x1, y1, x2, y2, x3, y3):
    # masukan: dua titik P1(x1, y1) dan P2(x2, y2) pada suatu garis l,
    #          serta titik P3(x3, y3)
    # luaran: jarak titik P3 terhadap garis P1P2
    # rumus jarak: |ax + by + c|/akar(a^2 + b^2)
    a, b, c = findEquation(x1, y1, x2, y2)
    return abs(a*x3 + b*y3 + c)/math.sqrt(a*a + b*b)
```

F. Fungsi `findAngle`

```
def findAngle(x1, y1, x2, y2, x3, y3):
    # masukan: tiga titik P1(x1, y1), P2(x2, y2), dan P3(x3, y3),
    #          yang membentuk segitiga P1P3P2, dengan
    #          sisi a = sisi P3P1, sisi b = sisi P3P2, dan sisi c = sisi P1P2
    # luaran: sudut C, yang mengapit sisi a dan b
    # rumus dasar: c^2 = a^2 + b^2 - 2*a*b*cos(theta)
    a2 = (x3-x1)*(x3-x1) + (y3-y1)*(y3-y1)
    b2 = (x3-x2)*(x3-x2) + (y3-y2)*(y3-y2)
    c2 = (x2-x1)*(x2-x1) + (y2-y1)*(y2-y1)
    a = math.sqrt(a2)
    b = math.sqrt(b2)
    c = math.sqrt(c2)
    if (((a2 + b2 - c2)/(2*a*b)) <= 1 and ((a2 + b2 - c2)/(2*a*b)) >= -1):
        return math.acos((a2 + b2 - c2)/(2*a*b))
    else:
        return 0
```

G. Fungsi `findFurthest`

```
def findFurthest(bucket, left, right):
    # masukan: dua titik left(x1, y1) dan right(x2, y2) pada suatu garis l,
    #          serta bucket (array titik pada suatu section)
```

```
    # luaran: titik dengan jarak terjauh terhadap garis l
    furthest = bucket[0]
    for point in bucket:
        distCurrent = findDistance(left[0], left[1], right[0], right[1], point[0], point[1])
        distFurthest = findDistance(left[0], left[1], right[0], right[1], furthest[0], furthest[1])
        if (distCurrent > distFurthest):
            furthest = point
        elif (distCurrent == distFurthest):
            angleCurrent = findAngle(left[0], left[1], right[0], right[1], point[0], point[1])
            angleFurthest = findAngle(left[0], left[1], right[0], right[1], furthest[0], furthest[1])
            if (angleCurrent > angleFurthest):
                furthest = point
    return furthest
```

H. Prosedur `ConvexHullDnCUpper`

```
def ConvexHullDnCUpper(bucket, left, right, bucketDnC):
    # algoritma rekursif Divide and Conquer, algoritma dijelaskan pada laporan
    # pengolahan section lebih atas dari garis leftmost-rightmost
    # masukan: bucket (array titik pada suatu section),
    #          left (titik kiri), right (titik kanan)
    # masukan/keluaran: bucketDnC (array titik pembentuk ConvexHull)
    leftBucket = []
    rightBucket = []

    if (len(bucket) != 0):
        # pencarian titik terjauh dari garis left-right
        furthest = findFurthest(bucket, left, right)
        for point in bucket:
            # pembagian titik dalam bucket utuk pemrosesan lebih lanjut
            if (point != furthest):
                y = findY(left[0], left[1], furthest[0], furthest[1], point[0])
                # bila titik di atas garis left-furthest, masukkan dalam leftBucket
                if (y > point[1]):
                    leftBucket.append(point)
                y = findY(right[0], right[1], furthest[0], furthest[1], point[0])
                # bila titik di atas garis furthest-right, masukkan dalam rightBucket
                if (y > point[1]):
```

```
                rightBucket.append(point)
        # untuk tiap bucket baru, kembali lakukan divide and conquer
        ConvexHullDnCUpper(leftBucket, left, furthest, bucketDnC)
        ConvexHullDnCUpper(rightBucket, furthest, right, bucketDnC)
    else:
        # bila bucket kosong (tidak ada titik di atas garis), left pembentuk Convex Hull
        if (left not in bucketDnC):
            bucketDnC.append(left)
    return
```

I. Prosedur `ConvexHullDnCLower`

```
def ConvexHullDnCLower(bucket, left, right, bucketDnC):
    # algoritma rekursif Divide and Conquer, algoritma dijelaskan pada laporan
    # pengolahan section lebih atas dari garis leftmost-rightmost
    # masukan: bucket (array titik pada suatu section),
    #          left (titik kiri), right (titik kanan)
    # masukan/keluaran: bucketDnC (array titik pembentuk ConvexHull)
    leftBucket = []
    rightBucket = []

    if (len(bucket) != 0):
        # pencarian titik terjauh dari garis left-right
        furthest = findFurthest(bucket, left, right)
        for point in bucket:
            # pembagian titik dalam bucket utuk pemrosesan lebih lanjut
            if (point != furthest):
                y = findY(left[0], left[1], furthest[0], furthest[1], point[0])
                # bila titik di bawah garis left-furthest, masukkan dalam leftBucket
                if (y < point[1]):
                    leftBucket.append(point)
                y = findY(right[0], right[1], furthest[0], furthest[1], point[0])
                # bila titik di bawah garis furthest-right, masukkan dalam rightBucket
                if (y < point[1]):
                    rightBucket.append(point)
        # untuk tiap bucket baru, kembali lakukan divide and conquer
        ConvexHullDnCLower(rightBucket, furthest, right, bucketDnC)
        ConvexHullDnCLower(leftBucket, left, furthest, bucketDnC)
```

```
    else:
        # bila bucket kosong (tidak ada titik di atas garis), left pembentuk Convex Hull
        if (left not in bucketDnC):
            bucketDnC.append(left)
    return
```

J. Fungsi `ConvexHull`

```
def ConvexHull(bucket):
    cHullBucket = []
    upperBucket = []
    lowerBucket = []
    leftmost = findLeftMost(bucket)
    rightmost = findRightMost(bucket)

    # pembagian titik di atas dan bawah garis leftmost-rightmost
    for point in bucket:
        y = findY(leftmost[0], leftmost[1], rightmost[0], rightmost[1], point[0])
        if (y > point[1]):
            upperBucket.append(point)
        elif (y < point[1]):
            lowerBucket.append(point)
        # else
            # do nothing, karena artinya berada pada garis

    # lakukan divide and conquer section atas
    ConvexHullDnCUpper(upperBucket, leftmost, rightmost, cHullBucket)
    # rightmost pasti pembentuk ConvexHull
    cHullBucket.append(rightmost)
    # lakukan divide and conquer section bawah
    ConvexHullDnCLower(lowerBucket, leftmost, rightmost, cHullBucket)
    return cHullBucket
```

Berikut merupakan fungsi/prosedur pendukung yang dibuat dalam program utama. Sebagian besar kode program utama ini dikutip dari dokumen spesifikasi Tugas Kecil 2, dengan perbedaan sebagai berikut.

- Terdapat input pilihan tabel dataset dan atribut (kolom) yang ingin diproses.
- Terdapat transformasi struktur data *default numpy array* menjadi *list* biasa untuk mempermudah pemrosesan algoritma Convex Hull, lalu kembali ke *numpy array*.

- Terdapat pembangunan *list linked* nilai x dan nilai y, yang merupakan titik pembentuk Convex Hull.

Penjelasan tiap fungsi/prosedur dituliskan dalam bentuk komentar dalam kode program.

A. Fungsi `start`

```
def start():
    print("Selamat datang di implementasi Convex Hull dengan algoritma Divide and Conquer.")
    print()
    return
```

B. Fungsi `choose`

```
def choose():
    # cetak daftar dataset
    print("Pilihan dataset:")
    print("1. Iris")
    print("2. Wine")
    print("3. Breast Cancer")

    # pilih dataset
    pilihan = int(input("Pilihan Anda: "))
    while (pilihan > 3 or pilihan < 1):
        print("Pilihan tidak tersedia.")
        print()
        pilihan = int(input("Pilihan Anda: "))

    # load dataset
    if (pilihan == 1):
        data = datasets.load_iris()
    elif (pilihan == 2):
        data = datasets.load_wine()
    else: # pilihan == 3dataa
        data = datasets.load_breast_cancer()
    print()

    # buat dataFrame tabel yang telah dipilih
    df = pd.DataFrame(data.data, columns=data.feature_names)
    df['Target'] = pd.DataFrame(data.target)

    # cetak data atribut dari tabel yang telah dipilih
    nCol = len(data.feature_names)
    print("Pilihan atribut:")
    for i in range(nCol):
        print(i+1, data.feature_names[i])

    # pilih atribut dari tabel yang telah dipilih
    x = int(input("Pilihan atribut pertama Anda (contoh: 1): "))
    y = int(input("Pilihan atribut kedua Anda (contoh: 2): "))
    while (x == y or (x > nCol or x <= 0) or (y > nCol or y <= 0) ):
```

```
        print("Pilihan Anda tidak tersedia atau pilihan 1 dan 2 Anda sama.")
        print()
        x = int(input("Pilihan atribut pertama (contoh: 1): "))
        y = int(input("Pilihan atribut kedua (contoh: 2): "))
    return data, df, x, y
```

C. Fungsi `visualize`

```
def visualize(data, df, x, y):
    # atur kelengkapan visualisasi berupa ukuran, warna, dan legenda
    plt.figure(figsize = (10, 6))
    colors = ['b','r','g', 'c', 'm', 'y', 'k']
    xName = data.feature_names[x-1]
    yName = data.feature_names[y-1]
    plt.title(f"{xName} vs {yName}")
```

```
plt.xlabel(xName)
plt.ylabel(yName)

# tampilkan titik dan garis antartitik sesuai Convex Hull
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[x-1,y-1]].values
    bucket = bucket.tolist()
    hull = ConvexHull(bucket)
    bucket = np.asarray(bucket)
    hull.append(hull[0])
    xs, ys = zip(*hull)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    plt.plot(xs, ys, colors[i])
    plt.legend()
plt.show()
```

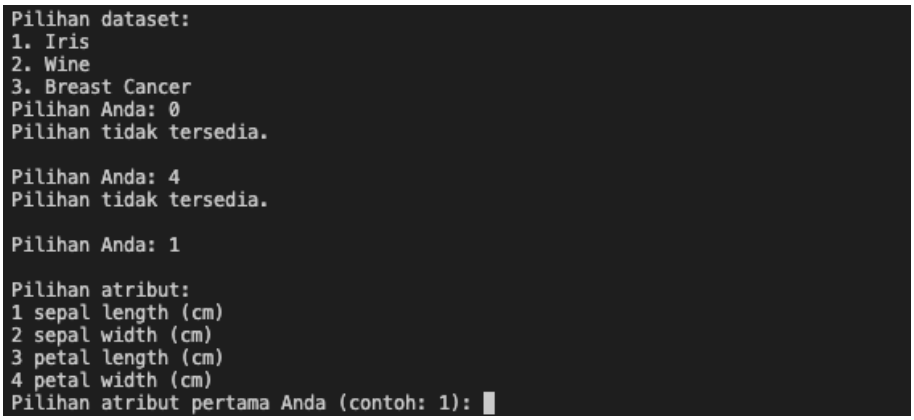
BAB III
HASIL PROGRAM DAN *TEST CASE*

Program dimulai dengan salam pembukaan berikut.

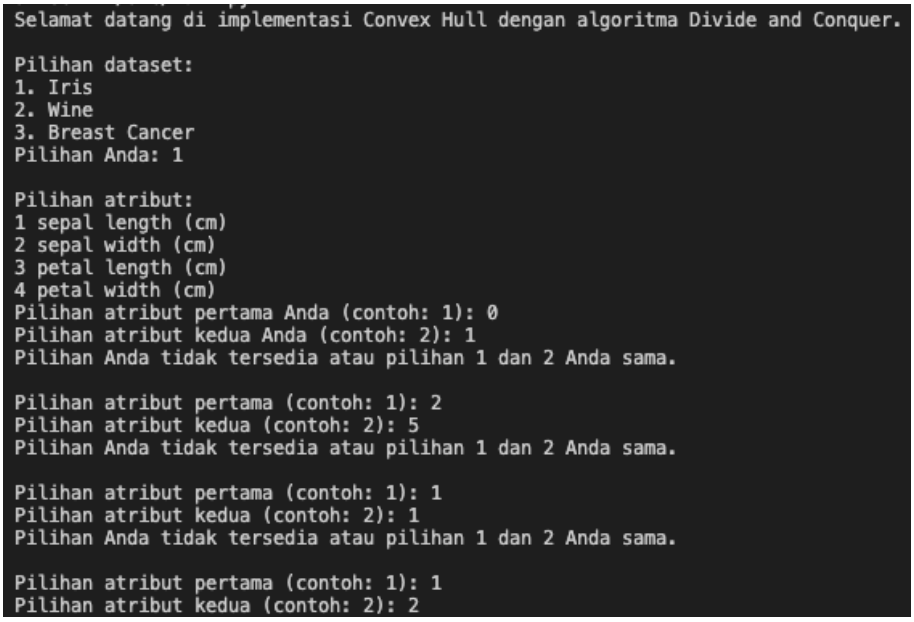
Gambar III.1. Salam Pembukaan

Sesuai spesifikasi dan jawaban di Questions-and-Answers (QnA), seluruh input data berasal dari test case yang diperoleh dari kakas `sklearn`. Oleh karena itu, pengguna hanya dapat memasukkan pilihan berupa nama tabel dari 3 pilihan dan atribut (kolom) yang akan diproses dari tabel masing-masing.

Gambar III.2. Pemilihan Dataset Berhasil



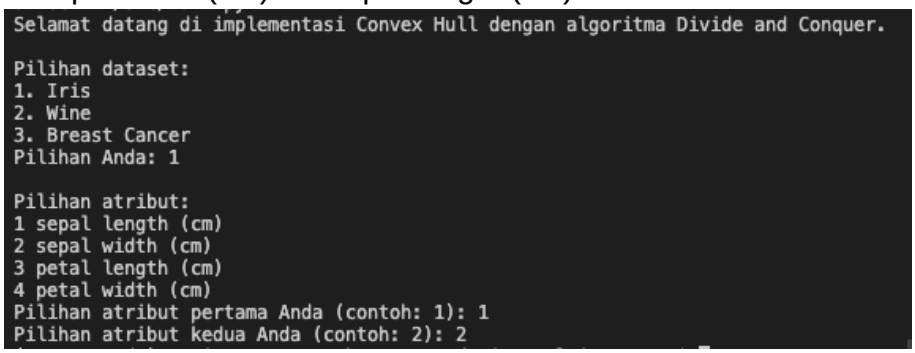
Gambar III.3. Pemilihan Dataset Gagal



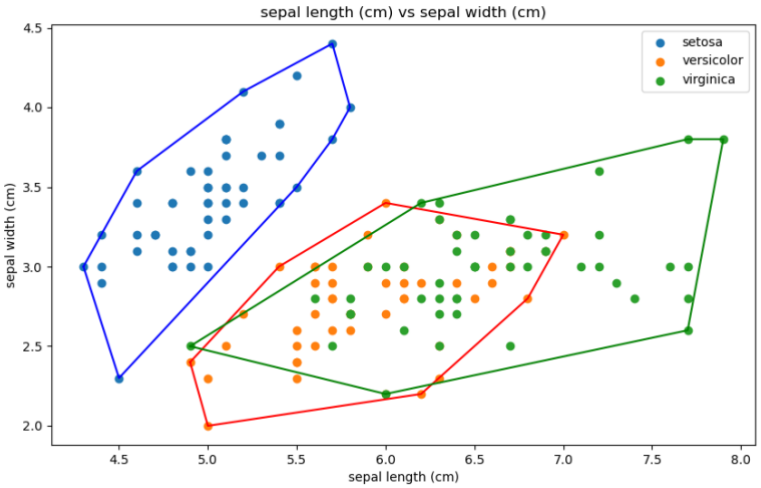
Gambar III.4. Pemilihan Atribut Gagal

Terdapat penanganan terhadap kesalahan masukan pengguna, yakni tidak tersedianya nomor *dataset* (tabel), tidak tersedianya nomor atribut, dan samanya pilihan atribut pertama dan kedua. Berikut merupakan tangkapan layar untuk 6 *test case*, dengan rincian 2 *test case* untuk masing-masing *dataset*.

A. Dataset iris: sepal width (cm) vs. sepal length (cm)



Gambar III.5. Proses *Test Case* 1



Gambar III.6. Hasil Visualisasi *Test Case* 1

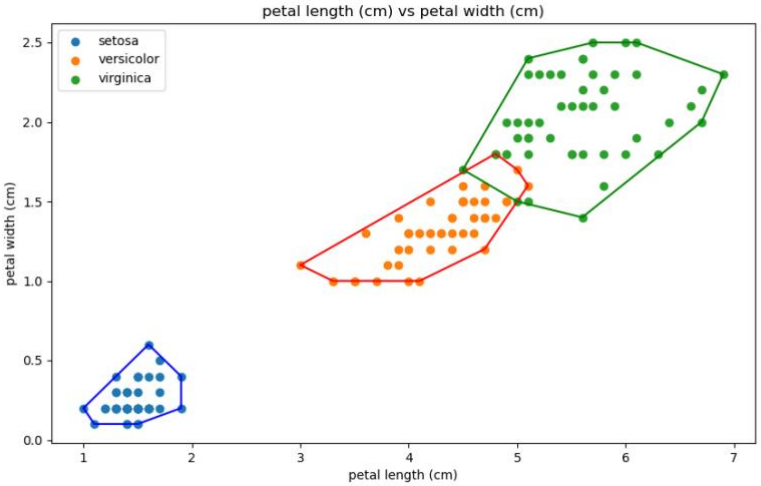
B. Dataset iris: petal width (cm) vs. petal length (cm)

```
Selamat datang di implementasi Convex Hull dengan algoritma Divide and Conquer.

Pilihan dataset:
1. Iris
2. Wine
3. Breast Cancer
Pilihan Anda: 1

Pilihan atribut:
1 sepal length (cm)
2 sepal width (cm)
3 petal length (cm)
4 petal width (cm)
Pilihan atribut pertama Anda (contoh: 1): 3
Pilihan atribut kedua Anda (contoh: 2): 4
```

Gambar III.7. Proses *Test Case* 2



Gambar III.8. Hasil Visualisasi *Test Case* 2

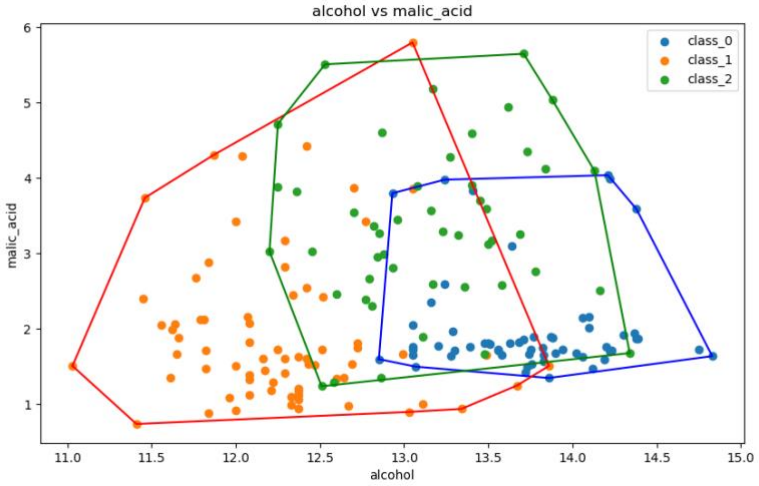
C. Dataset wine: alcohol vs. malic_acid

```
Selamat datang di implementasi Convex Hull dengan algoritma Divide and Conquer.

Pilihan dataset:
1. Iris
2. Wine
3. Breast Cancer
Pilihan Anda: 2

Pilihan atribut:
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
Pilihan atribut pertama Anda (contoh: 1): 1
Pilihan atribut kedua Anda (contoh: 2): 2
```

Gambar III.9. Proses *Test Case* 3



Gambar III.10. Hasil Visualisasi *Test Case* 3

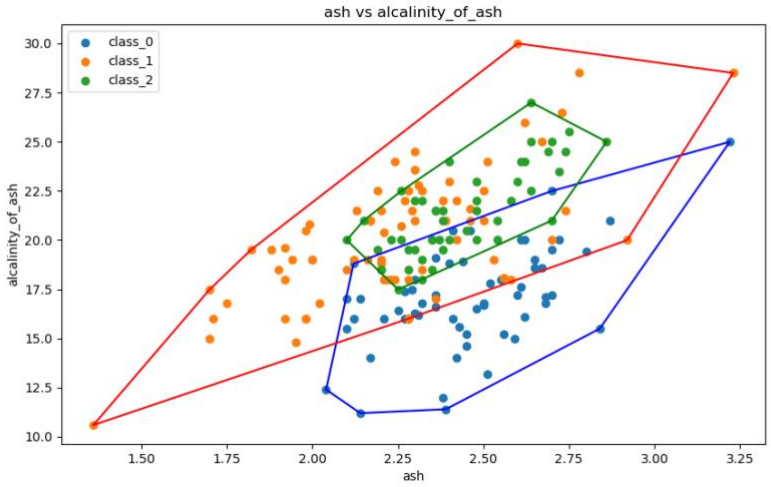
D. Dataset wine: ash vs. alcalinity_of_ash

```
Selamat datang di implementasi Convex Hull dengan algoritma Divide and Conquer.

Pilihan dataset:
1. Iris
2. Wine
3. Breast Cancer
Pilihan Anda: 2

Pilihan atribut:
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
Pilihan atribut pertama Anda (contoh: 1): 3
Pilihan atribut kedua Anda (contoh: 2): 4
```

Gambar III.11. Proses *Test Case* 4



Gambar III.12. Hasil Visualisasi *Test Case* 4

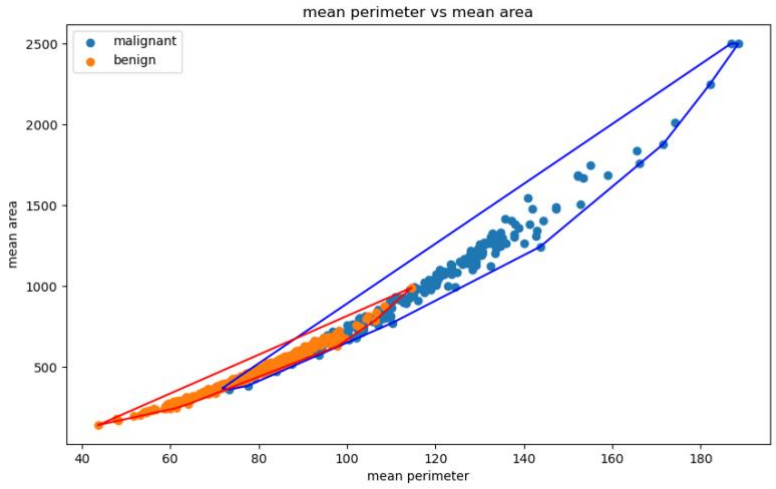
E. Dataset breast cancer: mean perimeter vs. mean area

```
Selamat datang di implementasi Convex Hull dengan algoritma Divide and Conquer.

Pilihan dataset:
1. Iris
2. Wine
3. Breast Cancer
Pilihan Anda: 3

Pilihan atribut:
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
25 worst smoothness
26 worst compactness
27 worst concavity
28 worst concave points
29 worst symmetry
30 worst fractal dimension
Pilihan atribut pertama Anda (contoh: 1): 3
Pilihan atribut kedua Anda (contoh: 2): 4
```

Gambar III.13. Proses *Test Case* 5



Gambar III.14. Hasil Visualisasi *Test Case* 5

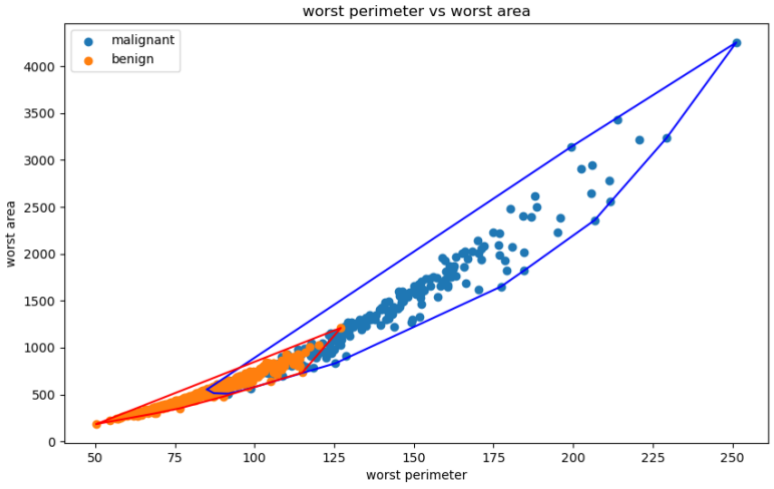
F. Dataset breast cancer: worst perimeter vs. worst area

```
Selamat datang di implementasi Convex Hull dengan algoritma Divide and Conquer

Pilihan dataset:
1. Iris
2. Wine
3. Breast Cancer
Pilihan Anda: 3

Pilihan atribut:
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
25 worst smoothness
26 worst compactness
27 worst concavity
28 worst concave points
29 worst symmetry
30 worst fractal dimension
Pilihan atribut pertama Anda (contoh: 1): 23
Pilihan atribut kedua Anda (contoh: 2): 24
```

Gambar III.15. Proses *Test Case* 6



Gambar III.15. Hasil Visualisasi *Test Case* 6

No	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.	√	

Tabel III.1. Tabel Penilaian Mandiri

BAB IV
AKSES *REPOSITORY*

Seluruh fail tugas kecil ini dapat diunduh melalui repository GitHub pada tautan berikut.
https://github.com/dclairvoyance/Tucil2Stima

Struktur repository tersebut ialah sebagai berikut.

- Folder src, berisi *source code* main.py dan myConvexHull.py (kakas ConvexHull).
- Folder doc, berisi laporan tugas kecil Tucil2_13520035.pdf.
- README.

Catatan : Sesuai arahan di QnA, folder bin tidak diperlukan karena tidak ada executable dan folder test tidak diperlukan karena tidak input dataset dari file eksternal.

BAB V
KESIMPULAN DAN SARAN

Tes *Linear Separability Dataset* dapat divisualisasikan dengan konsep *Convex Hull*, yang dapat diselesaikan dengan algoritma *Divide and Conquer*.

REFERENSI

Slide Bahan Kuliah IF2211 Strategi Algoritma – Algoritma Divide and Conquer (Bagian 1)
Slide Bahan Kuliah IF2211 Strategi Algoritma – Algoritma Divide and Conquer (Bagian 4)