Laporan Tugas Kecil 2
IF2211 Strategi Algoritma
Implementasi Convex Hull untuk Visualisasi Tes Linear Separability Dataset dengan Algoritma Divide and Conquer

Oleh:
Adzka Ahmadetya Zaidan 13520127

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
SEMESTER II 2021/2022

Daftar Isi

1. Algoritma *Divide and Conquer* untuk *Convex Hull*

Algoritma *divide and conquer* yang digunakan untuk membuat titik-titik *convex hull* 2 dimensi diawali dengan mengurutkan *array* titik berdasarkan *x-axis*-nya. Kemudian, jika titik memiliki *x-axis* yang sama, pengurutan dilakukan berdasarkan *y-axis*-nya. Setelah terurut, program mengambil 2 titik: titik awal dan titik akhir dan membagi 2 daerah titik berdasarkan garis yang dibuat dari kedua titik tersebut. Setelah itu, program akan melakukan *solving* untuk daerah kiri.

Setelah garis yang membagi titik-titik menjadi 2 daerah terbentuk, program akan mencari titik terjauh dari garis tersebut. Jika sudah ditemukan, program akan membuat 2 garis: dari titik awal ke titik terjauh tersebut dan dari titik terjauh tersebut ke titik akhir. Daerah di bagian kiri yang akan dilakukan *solving* terbagi lagi menjadi 2. Kemudian, program akan melakukan rekursif untuk kedua daerah tersebut dengan basis tidak ada titik, atau hanya terdapat satu titik pada daerah kiri. Jika hanya terdapat satu titik pada daerah kiri, titik tersebut yang akan dipilih.

Setelah bagian kiri selesai, program akan melanjutkan mencari titik-titik pada bagian kanan. Cara program mencarinya menggunakan fungsi yang sama. Namun, titik awal dan titik akhir garis awal tersebut dibalik. Jadi, program tetap mencari dengan cara rekursif mencari titik di daerah kiri.

2. Kode Program

2.1. File convexhull.ipynb

2.1.1. Blok Kode 1

```
# Implementasi fungsi ConvexHull buatan sendiri: myConvexHull
# 13520127 Adzka Ahmadetya Zaidan

import numpy as np

# Menghitung jarak dari 2 titik
def Distance(p1, p2):
    return float(np.sqrt((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2))

# Menghitung jarak suatu titik ke suatu garis yang dibuat dari 2 titik: start hingga end
def DistToLine(start, end, point):
    start = np.array(start)
    end = np.array(end)
    point = np.array(point)
    return np.cross(end-start,point-start)/np.linalg.norm(end-start)
```

```
# Menghitung nilai determinan dari 3 titik p1, p2, dan p3
#  p1 = (x1, y1); p2 = (x2, y2); p3 = (x3, y3)
# | x1 y1 1 |
# | x2 y2 1 |
# | x3 y3 1 |
def DetConvexHull(p1, p2, p3):
    return (p1[0]*p2[1]+p3[0]*p1[1]+p2[0]*p3[1]-p3[0]*p2[1]-p2[0]*p1[1]-p1[0]*p3[1])

# Fungsi untuk mencari titik terjauh dari garis yang dibentuk dari titik start dan end,
# sekaligus menambahkan titik tersebut ke array hull
def FindFurthest(hull, start, end, possibleCoords):
    if (len(possibleCoords) == 0): # Basis 0
        return
    elif (len(possibleCoords) == 1): # Basis 1
        hull.append(possibleCoords[0])
        return
    else:
        # Cari titik terjauh dari garis
        maxDist = 0
        maxIdx = 0
        # Mencari index terjauh dari garis
        for i in range(len(possibleCoords)):
            dist = DistToLine(start, end, possibleCoords[i])
            if dist > maxDist:
                maxDist = dist
                maxIdx = i

        # Merekursif bagian kiri dari garis
        # yang dibentuk dari titik start hingga titik terjauh
        coords1 = []
        for i in range(len(possibleCoords)):
            detTemp = DetConvexHull(start, possibleCoords[maxIdx], possibleCoords[i])
            if detTemp > 0:
                coords1.append(possibleCoords[i])
        if len(coords1) > 0:
            FindFurthest(hull, start, possibleCoords[maxIdx], coords1)

        hull.append(possibleCoords[maxIdx])

        # Merekursif bagian kanan dari garis
```

```
        # yang dibentuk dari titik start hingga titik terjauh
        coords2 = []
        for i in range(len(possibleCoords)):
            detTemp = DetConvexHull(possibleCoords[maxIdx], end, possibleCoords[i])
            if detTemp > 0:
                coords2.append(possibleCoords[i])
        if len(coords2) > 0:
            FindFurthest(hull, possibleCoords[maxIdx], end, coords2)
        return

# Fungsi yang menerima kumpulan data titik 2 dimensi
# Mengembalikan titik 2 dimensi dari data tersebut yang membentuk poligon convex
def myConvexHull(coords):
    hull = []

    # Melakukan sorting titik berdasarkan x-axisnya
    coords = sorted(coords , key=lambda k: [k[0], k[1]])
    coords = np.array(coords)

    # Membagi 2 array titik menjadi array di kanan dan di kiri berdasarkan garis
    # Pembelahan dilakukan di titik dengan x terkecil dan titik dengan x terbesar
    start = coords[0]
    end = coords[-1]

    coordsLeft = []
    coordsRight = []
    for i in range(len(coords)):
        detTemp = DetConvexHull(start, end, coords[i])
        if detTemp > 0:
            coordsLeft.append(coords[i])
        if detTemp < 0:
            coordsRight.append(coords[i])

    hull.append(start)

    # Cari titik terjauh dari garis di bagian kiri secara rekursif
    FindFurthest(hull, start, end, coordsLeft)

    hull.append(end)
```

```
    # Cari titik terjauh dari garis di bagian kanan secara rekursif
    FindFurthest(hull, end, start, coordsRight)

    #hull.append(start)

    # Jika hanya garis yang dapat terbentuk, maka bidang bukan convex hull
    # mengembalikan array kosong
    if len(coordsRight) == 0 and len(coordsLeft) == 0:
        return np.array([])

    return np.array(hull)
```

2.1.2. Blok Kode 2

```
# Importing libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
```

2.1.3. Blok Kode 3

```
data = datasets.load_iris() # Diganti sesuai dataset yang ingin digunakan

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

2.1.4. Blok Kode 4

```
# INPUT Attribute Index dan title graph yang sesuai
plt.figure(figsize = (10, 6))
plt.title('Petal Width vs Petal Length') # Diganti sesuai label atribut
attributeIdx1 = 2 # Diganti sesuai index atribut
attributeIdx2 = 3 # Diganti sesuai index atribut
```

2.1.5. Blok Kode 5

```
#visualisasi hasil ConvexHull
colors = ['b','r','g', 'c', 'm', 'y', 'k']
plt.xlabel(data.feature_names[attributeIdx1])
plt.ylabel(data.feature_names[attributeIdx2])
```

```
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[attributeIdx1,attributeIdx2]].values
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull
Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    x = []
    y = []
    for j in range(len(hull)):
        x.append(hull[j][0])
        y.append(hull[j][1])
    if len(hull) > 0:
        x.append(hull[0][0])
        y.append(hull[0][1])
    plt.plot(x, y, colors[i])

plt.legend()
```

3. *Screenshot Input* dan *Output*

3.1. *Dataset* Iris

```
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()
```

[18] ✓ 0.7s Python

```
(150, 5)
```

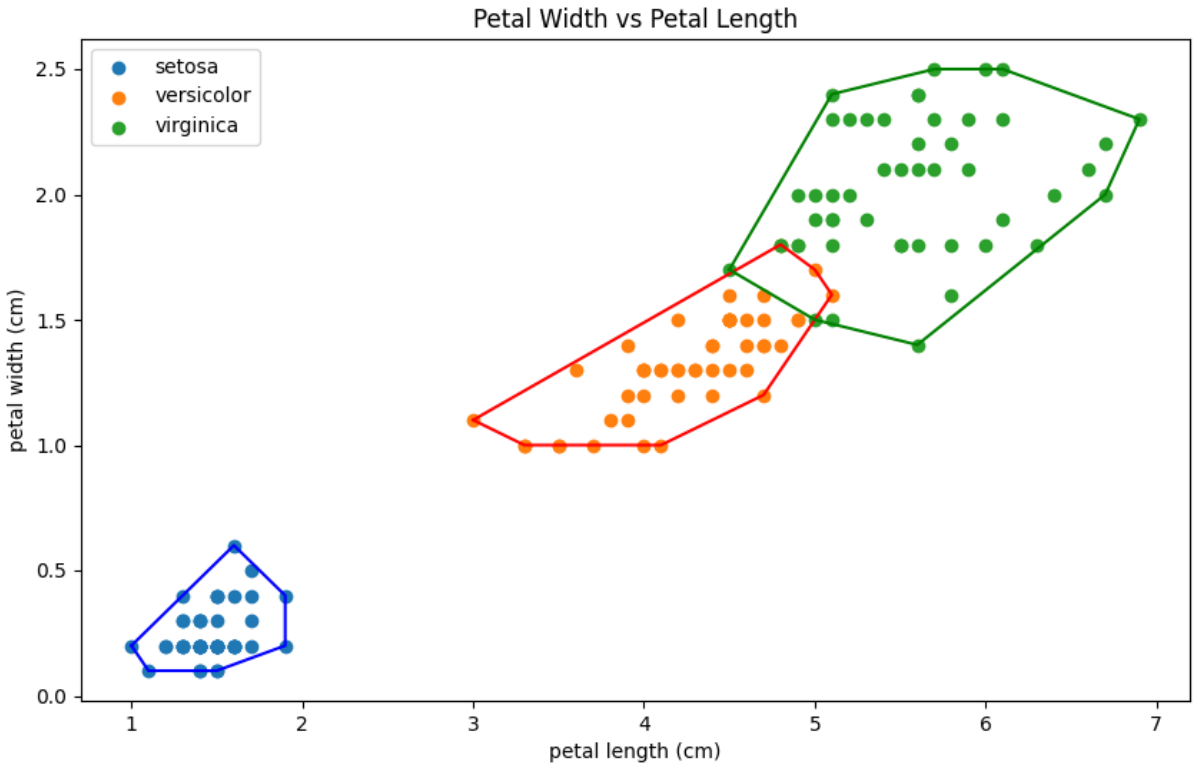
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

3.1.2. Petal Width vs Petal Length *Input*

```
# INPUT Attribute Index dan title graph yang sesuai
plt.figure(figsize = (10, 6))
plt.title('Petal Width vs Petal Length') # Diganti sesuai label atribut
attributeIdx1 = 2 # Diganti sesuai index atribut
attributeIdx2 = 3 # Diganti sesuai index atribut
```

[42] ✓ 0.2s Python

3.1.2. Petal Width vs Petal Length *Output*

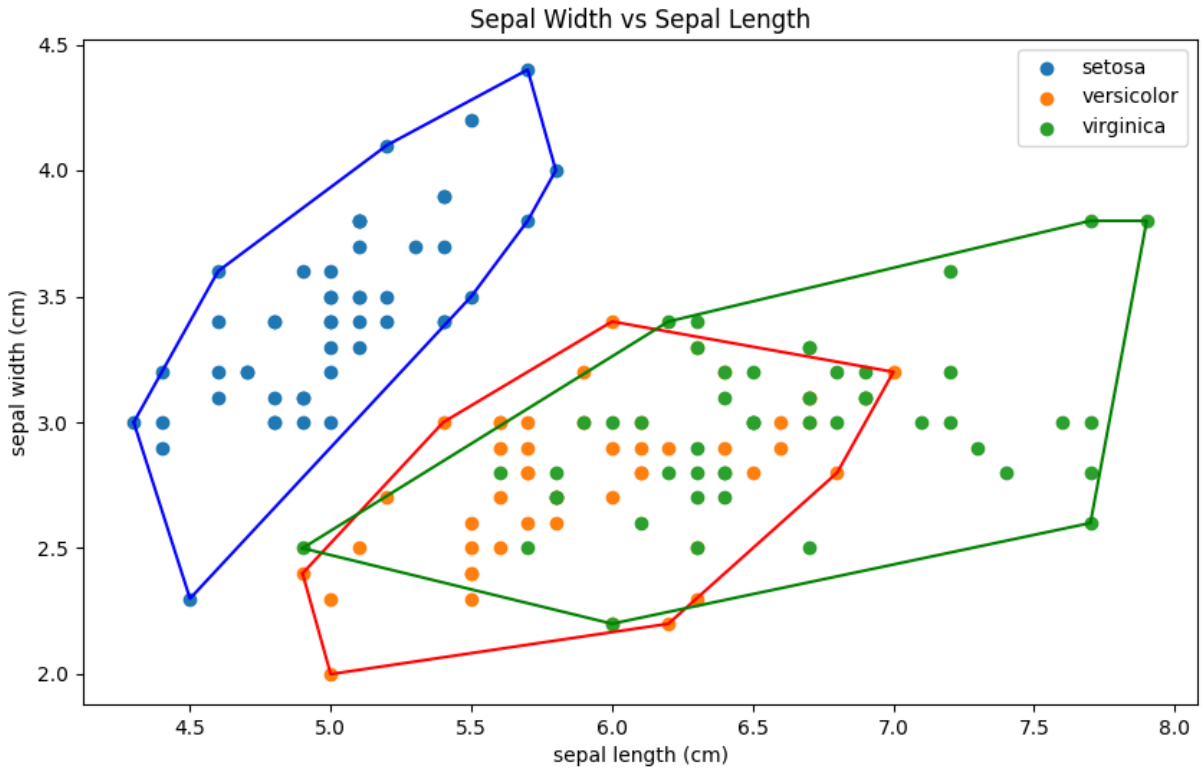


3.1.3. Sepal Width vs Sepal Length *Input*

```
# INPUT Attribute Index dan title graph yang sesuai
plt.figure(figsize = (10, 6))
plt.title('Sepal Width vs Sepal Length') # Diganti sesuai label atribut
attributeIdx1 = 0 # Diganti sesuai index atribut
attributeIdx2 = 1 # Diganti sesuai index atribut
```

[40] ✓ 0.3s Python

3.1.4. Sepal Width vs Sepal Length *Output*



3.2. *Dataset* Wine

```
data = datasets.load_wine()

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

[19] ✓ 0.1s Python

```
(178, 14)
```

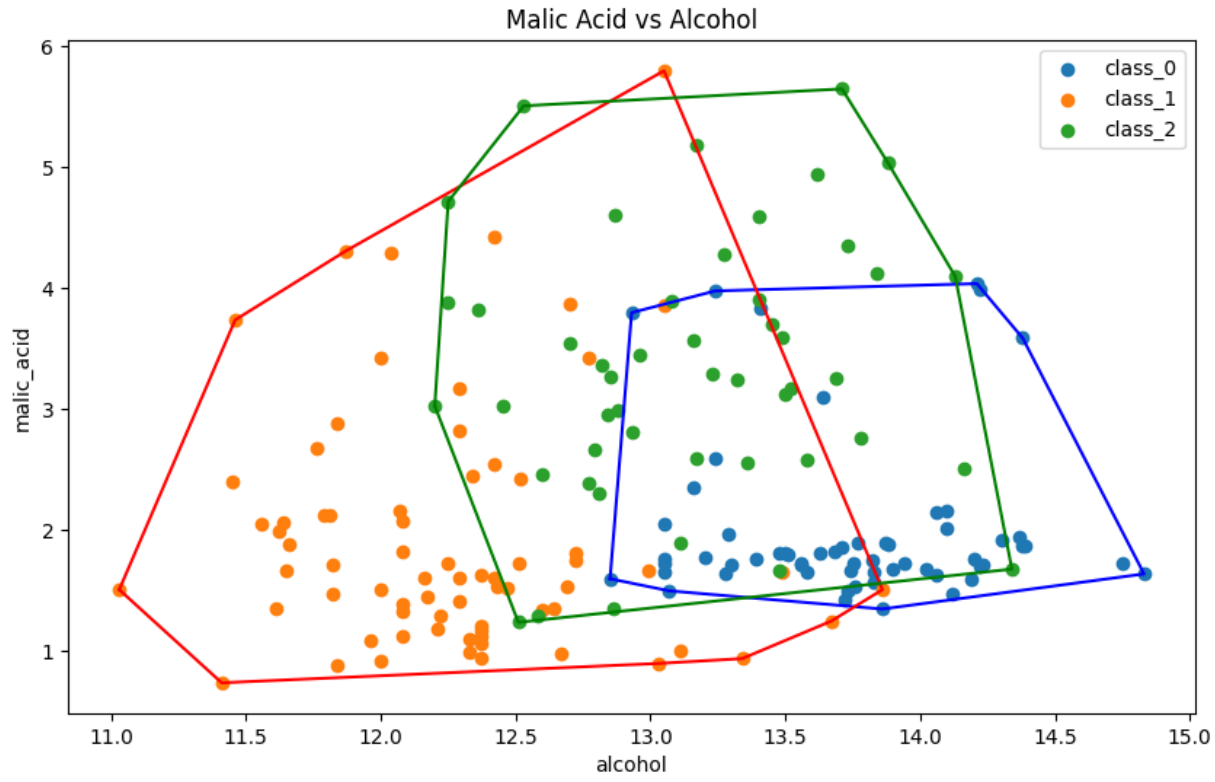
	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthc
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	

3.2.1. Malic Acid vs Alcohol *Input*

```
# INPUT Attribute Index dan title graph yang sesuai
plt.figure(figsize = (10, 6))
plt.title('Malic Acid vs Alcohol') # Diganti sesuai label atribut
attributeIdx1 = 0 # Diganti sesuai index atribut
attributeIdx2 = 1 # Diganti sesuai index atribut
```

[51] ✓ 0.2s Python

3.2.2. Malic Acid vs Alcohol *Output*

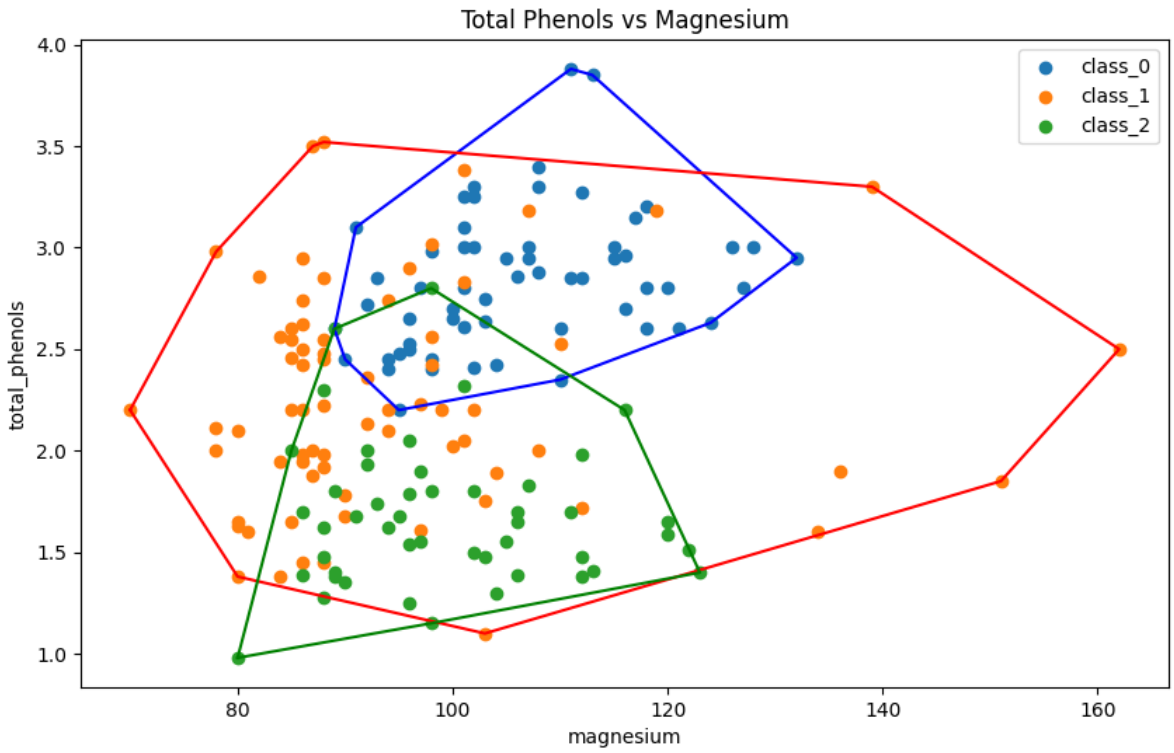


3.2.3. Total Phenols vs Magnesium *Input*

```
# INPUT Attribute Index dan title graph yang sesuai
plt.figure(figsize = (10, 6))
plt.title('Total Phenols vs Magnesium') # Diganti sesuai label atribut
attributeIdx1 = 4 # Diganti sesuai index atribut
attributeIdx2 = 5 # Diganti sesuai index atribut
```

[57] ✓ 0.2s Python

3.2.4. Total Phenols vs Magnesium *Output*



4. Alamat *Drive* Kode Program

https://github.com/Voguelish/convex-hull-stima

Checklist Program

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.	*√	

*Catatan: Input dilakukan dengan cara memodifikasi isi kode supaya sesuai dengan isi atribut (mengubah isi Blok Kode 4). Dataset yang terlalu besar mungkin mengeluarkan error.

Daftar Pustaka

"Algoritma Divide and Conquer Bagian 4". informatika.stei.itb.ac.id. 2022. Diakses pada tanggal 27 Februari 2022. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf