LAPORAN TUGAS KECIL 2

IF2211 STRATEGI ALGORITMA

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

Disusun oleh:

Eiffel Aqila Amarendra 13520074

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2022

1. PENJELASAN ALGORITMA *DIVIDE AND CONQUER*

Algoritma *divide and conquer* merupakan strategi algoritma pemecahan permasalahan dengan membagi permasalahan menjadi beberapa upa-permasalahan yang berukuran lebih kecil dengan mempertahaknkan kemiripannya dengan permasalahan yang semula. Kemudian, masing-masing upa-permasalahan diselesaikan secara langsung atau melalui proses rekursif jika masih berukuran besar. Terakhir, seluruh solusi masing-masing upa-permasalahan digabung sehingga membentuk solusi persoalan semula.

Algoritma *divide and conquer* dapat digunakan untuk menyelesaikan berbagai macam persoalan, mulai dari pencarian nilai ekstrem, persoalan pengurutan (*sorting*), hingga *Convex Hull*. Pada tugas kecil ini, algoritma *divide and conquer* digunakan untuk mengimplementasikan *Convex Hull* untuk Visualisasi Tes *Linear Separability Dataset*.

Secara umum, penjelasan algoritma *divide and conquer* di dalam *file* 'myConvexHull.py' pada program ini adalah sebagai berikut,

1. Program menerima *ndarray* yang berisi kumpulan titik kemudian dikonversi menjadi list.
2. Program mengurutkan list titik tersebut berdasarkan nilai absis secara menaik. Jika terdapat nilai absis yang sama, list diurutkan dengan nilai ordinat yang menaik.
3. Program memilih dua titik ekstrem dari list terurut tersebut dengan titik P_1 adalah titik dengan indeks pertama dan titik P_n adalah titik dengan indeks terakhir.
4. Titik P_1 dan P_n membentuk sebuah garis yang menghubungkan kedua titik tersebut yang membagi kumpulan titik tersebut menjadi dua bagian, yakni bagian atas (kumpulan titik di sebelah kiri garis P_1P_n) dan bagian bawah atas (kumpulan titik di sebelah kiri garis $P_n P_1$). Pemeriksaan apakah suatu titik P_3 (x_3, y_3) berada di sebelah kiri garis yang dibentuk P_1 (x_1, y_1) *dan* P_2 (x_1, y_1) menggunakan persamaan determinan dari matriks sebagai berikut

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

5. Pada setiap bagian, terdapat dua kemungkinan:
    a. Jika tidak ada titik di suatu bagian, maka pembentuk *convex hull* pada bagian tersebut adalah P_1 dan P_2.

b. Jika ada, maka dipilih titik terjauh dari garis P_1P_2 misalnya P_3. Apabila terdapat beberapa titik dengan jarak yang sama dipilih titik dengan sudut $P_3P_1P_2$ terbesar.

6. Tentukan kumpulan titik solusi yang berada di kiri garis P_1P_3 dan kumpulan titik solusi yang berada di kiri garis P_3P_2.
7. Ulangi langkah 5 dan 6 hingga bagian 'kiri' kosong.
8. Kembalikan pasangan titik solusi yang dihasilkan.

2. *SOURCE CODE* PROGRAM

Program implementasi *Convex Hull* dalam tugas ini menggunakan bahasa pemrograman Python. Berikut hasil tangkapan layar program,

2.1. myConvexHull.py

```python
import math
import numpy as np

def isLeft(p1, p2, p3):
    """
    Mengembalikan true jika p3 terletak di kiri garis p1-p2 dan false jika tidak
    berdasarkan persamaan determinan
    """
    return (np.linalg.det(np.array([[p1[0], p1[1], 1], [p2[0], p2[1], 1], [p3[0], p3[1], 1]])) > 0)

def distance(p1, p2, p3):
    """
    Menghitung jarak p3 dengan garis p1-p2
    """
    return math.dist(p1, p3) + math.dist(p2, p3)

def leftSide(p1, p2, list_of_points):
    """
    Mengembalikan list yang berisi point-point yang
    terletak di kiri garis p1-p2
    """
    leftSide = []
    for i in range(len(list_of_points)):
        if (isLeft(p1, p2, list_of_points[i])):
            leftSide.append(list_of_points[i])

    return leftSide

def findFarthest(p1, p2, list_of_points):
    """
    Mengembalikan point yang letaknya paling jauh dari garis p1-p2
    jika terdapat point dengan jarak yang sama, mengembalikan point dengan
    besar sudut paling besar
    """
    max_distance = 0
    max_index = 0
    max_angle = 0
    for i in range (len(list_of_points)):
        currDistance = distance(p1,p2,list_of_points[i])
        currAngle = angle(p1, p2, list_of_points[i])
        if (currDistance > max_distance):
            max_distance = currDistance
            max_index = i
            max_angle = currAngle
        elif ((currDistance == max_distance) and (currAngle > max_angle)):
            max_distance = currDistance
            max_index = i
            max_angle = currAngle
    return list_of_points[max_index]

```

```python
def angle(p1, p2, p3):
    """
    Mengembalikan besar sudut p3-p1-p2
    """
    ang = math.degrees(math.atan2(p3[1]-p1[1], p3[0]-p1[0]) - math.atan2(p2[1]-p1[1], p2[0]-p1[0]))
    if ang < 0:
        return ang+360
    else:
        return ang

def FindHull(p1,p2,list_of_points,list_of_hull):
    """
    Implementasi Algoritma Divide and Conquer untuk mencari solusi Convex Hull
    """

    if (len(list_of_points) == 0):  # Basis
        # List kosong
        return list_of_hull
    else:   # Rekurens
        # Menentukan titik terjauh dari garis p1-p2
        max_point = findFarthest(p1, p2, list_of_points)
        list_of_hull.append(max_point)

        list_of_points.remove(max_point)

        # Memanggil kembali fungsi FindHull
        # Mencari solusi di kiri p1-max_point
        list_of_hull = FindHull(p1, max_point, leftSide(p1, max_point, list_of_points), list_of_hull)
        # Mencari solusi di kiri max_point-p2
        list_of_hull = FindHull(max_point, p2, leftSide(max_point, p2, list_of_points), list_of_hull)
        return list_of_hull

def ConvexHull(list_of_points):
    """
    Fungsi utama untuk menentukan ConvexHull

    Mengembalikan pasangan titik pembentuk convex hull
    """
    # Inisialisasi
    list_of_hull = []   # Kumpulan titik-titik pembentuk convex hull
    tuple_of_hull = []  # Kumpulan pasangan titik pembentuk convex hull

    # Mengurutkan list of points berdasarkan absis
    list_of_points = sorted(list_of_points.tolist())

    # Menentukan titik minimum (p1) dan maksimum (pn)
    p1 = list_of_points[0]
    pn = list_of_points[len(list_of_points)-1]
    list_of_hull.append(p1)
    list_of_hull.append(pn)

    # Memanggil fungsi FindHull untuk mencari solusi di atas dan bawah garis
    list_of_hull = FindHull(p1, pn, leftSide(p1,pn,list_of_points), list_of_hull)
    list_of_hull = FindHull(pn, p1, leftSide(pn,p1,list_of_points), list_of_hull)

    # Mengurutkan list of hull
    mid_x = sum(p[0] for p in list_of_hull)/len(list_of_hull)
    mid_y = sum(p[1] for p in list_of_hull)/len(list_of_hull)
    list_of_hull.sort(key = lambda p: math.atan2(p[0] - mid_x, p[1] - mid_y))

    # Membuat tuple of hull
    for j in range(0,len(list_of_hull)):
        if(j == len(list_of_hull)-1):
            tuple_of_hull.append([list_of_hull[j],list_of_hull[0]])
        else:
            tuple_of_hull.append([list_of_hull[j],list_of_hull[j+1]])

    return tuple_of_hull
```

2.2. main.py

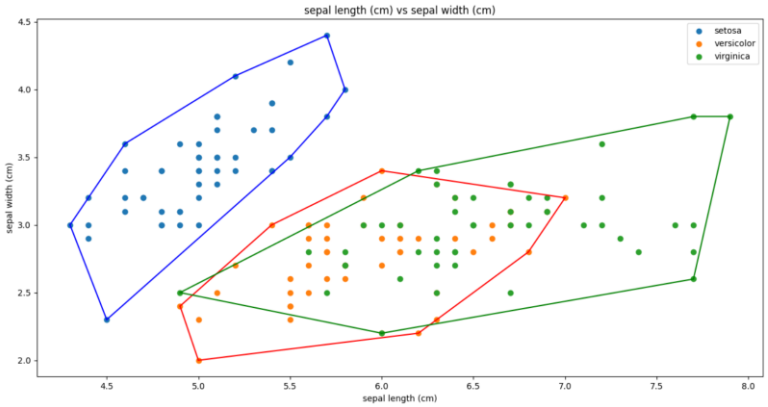
```python
from myConvexHull import ConvexHull
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

if __name__ == '__main__':
    # Mencetak datasets yang tersedia
    print("Datasets yang tersedia:")
    print("1. Iris")
    print("2. Wine")
    print("3. Breast Cancer")

    # Memilih dataset
    pilihanDataset = int(input("Masukkan pilihan datasets (dalam nomor): "))

    if (pilihanDataset == 1):
        data = datasets.load_iris()
    elif (pilihanDataset == 2):
        data = datasets.load_wine()
    else:
        data = datasets.load_breast_cancer()

    df = pd.DataFrame(data.data, columns=data.feature_names)
    df['Target'] = pd.DataFrame(data.target)

    # Memilih kolom
    print("Daftar kolom yang tersedia:")
    for i in range(len(data.feature_names)):
        print(str(i+1) + ". " + str(data.feature_names[i]))

    X = int(input("Masukan atribut-x: "))
    Y = int(input("Masukan atribut-y: "))

    #visualisasi hasil ConvexHull
    plt.figure(figsize = (10, 6))
    colors = ['b','r','g']
    plt.title(data.feature_names[X-1] + ' vs ' + data.feature_names[Y-1])
    plt.xlabel(data.feature_names[X-1])
    plt.ylabel(data.feature_names[Y-1])

    for i in range(len(data.target_names)):
        bucket = df[df['Target'] == i]
        bucket = bucket.iloc[:,[X-1,Y-1]].values
        hull = ConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
        plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])

        for j in range(len(hull)):
            plt.plot([hull[j][0][0],hull[j][1][0]], [hull[j][0][1],hull[j][1][1]], colors[i])

    plt.legend()
    plt.show()
```

3. *SCREENSHOT INPUT* DAN *OUTPUT*

3.1. Iris

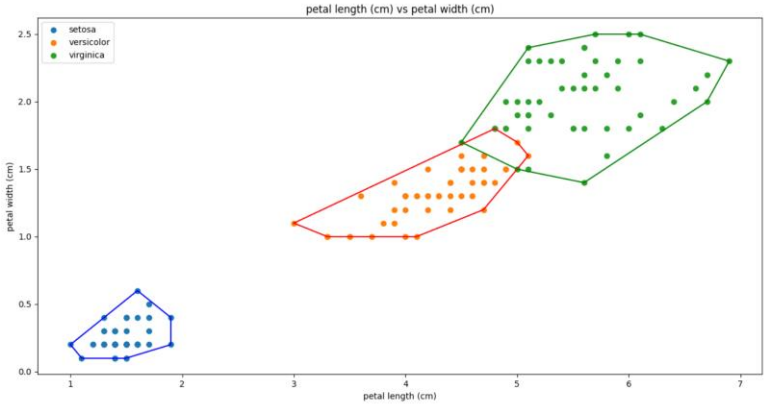
3.1.1. Sepal Length vs Sepal Width

```
Datasets yang tersedia:
1. Iris
2. Wine
3. Breast Cancer
Masukkan pilihan datasets (dalam nomor): 1
Daftar kolom yang tersedia:
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)
Masukan atribut-x: 1
Masukan atribut-y: 2
```



3.1.2. Petal Length vs Petal Width

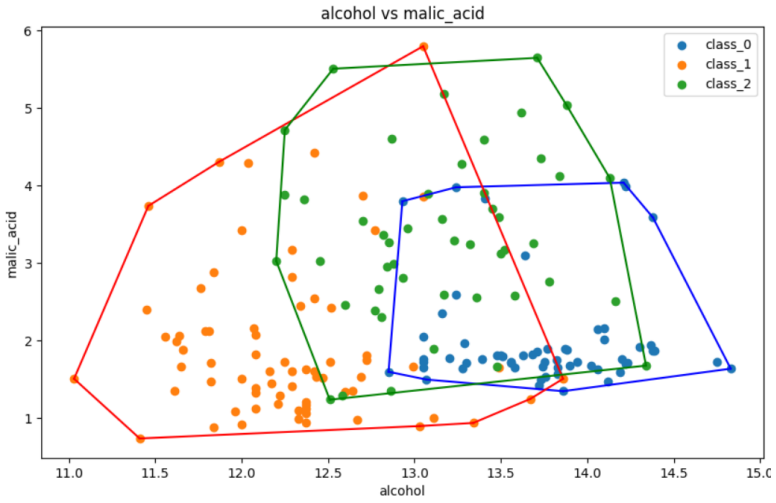
```
Datasets yang tersedia:
1. Iris
2. Wine
3. Breast Cancer
Masukkan pilihan datasets (dalam nomor): 1
Daftar kolom yang tersedia:
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)
Masukan atribut-x: 3
Masukan atribut-y: 4
```



3.2. Wine

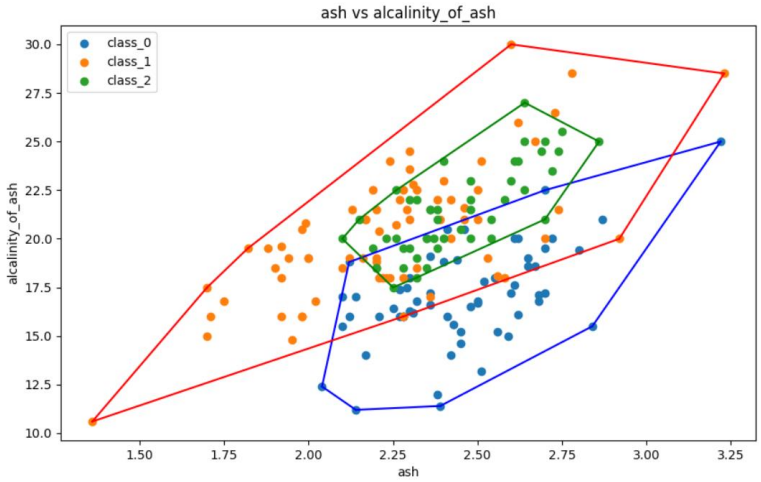
3.2.1. alcohol vs malic_acid

```
Datasets yang tersedia:
1. Iris
2. Wine
3. Breast Cancer
Masukkan pilihan datasets (dalam nomor): 2
Daftar kolom yang tersedia:
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
Masukan atribut-x: 1
Masukan atribut-y: 2
```



3.2.2. ash vs alcalinity_of_ash

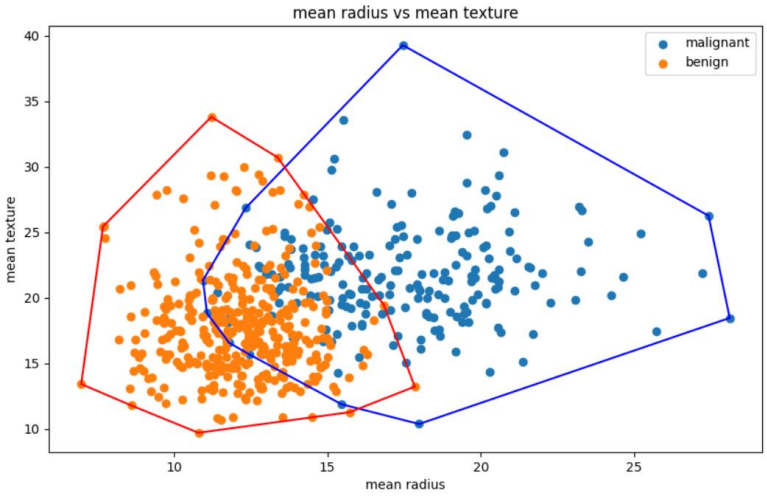
```
Datasets yang tersedia:
1. Iris
2. Wine
3. Breast Cancer
Masukkan pilihan datasets (dalam nomor): 2
Daftar kolom yang tersedia:
1. alcohol
2. malic_acid
3. ash
4. alcalinity_of_ash
5. magnesium
6. total_phenols
7. flavanoids
8. nonflavanoid_phenols
9. proanthocyanins
10. color_intensity
11. hue
12. od280/od315_of_diluted_wines
13. proline
Masukan atribut-x: 3
Masukan atribut-y: 4
```



3.3. Breast Cancer

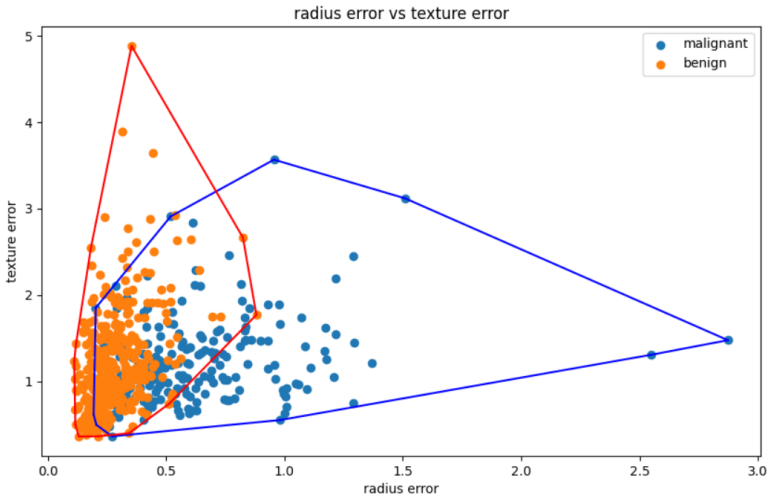
3.3.1. mean radius vs mean texture

```
Datasets yang tersedia:
1. Iris
2. Wine
3. Breast Cancer
Masukkan pilihan datasets (dalam nomor): 3
Daftar kolom yang tersedia:
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
Masukan atribut-x: 1
Masukan atribut-y: 2
```



3.3.2. radius error vs texture error

```
1. Iris
2. Wine
3. Breast Cancer
Masukkan pilihan datasets (dalam nomor): 3
Daftar kolom yang tersedia:
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
Masukan atribut-x: 11
Masukan atribut-y: 12
```


radius error vs texture error
malignant
benign
texture error
radius error
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1
2
3
4
5

4. TAUTAN *REPOSITORY* GITHUB

Tautan *repository* Github kode program ini adalah sebagai berikut

https://github.com/eiffelaqila/Tucil2_13520074

5. CEKLIST

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.	√	