LAPORAN TUGAS KECIL 2

IF2211 Strategi Algoritma

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

Disusun oleh:

Nama : Johannes Winson Sukiatmodjo

NIM : 13520123

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

Algoritma *Divide and Conquer*

Algoritma divide and conquer adalah algoritma yang membagi persoalan menjadi beberapa upa persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya berukuran hampir sama), menyelesaikan masing-masing upa persoalan (secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar), dan menggabungkan solusi masing-masing upa persoalan sehingga membentuk solusi persoalan semula.

Salah satu persoalan yang dapat diselesaikan dengan algoritma divide and conquer adalah mencari convex hull dari kumpulan data 2 dimensi (dapat dianggap kumpulan titik 2 dimensi). Berikut merupakan ide algoritma divide and conquer untuk menyelesaikan persoalan convex hull.

1. Mengurutkan kumpulan titik berdasarkan nilai absis yang menaik, dan jika ada nilai absis yang sama, maka diurutkan dengan nilai ordinat yang menaik.
2. Menentukan initial_point dan final_point sebagai dua titik ekstrim yang akan membentuk convex hull untuk kumpulan titik tersebut.
3. Menarik garis yang menghubungkan initial_point dan final_point untuk membagi persoalan menjadi dua bagian, yaitu part1 (kumpulan titik di sebelah kiri atau atas garis tersebut) dan part2 (kumpulan titik di sebelah kanan atau bawah garis tersebut).
4. Mengabaikan semua titik yang berada pada garis tersebut (selain titik initial_point dan final_point) karena tidak mungkin membentuk convex hull.
5. Jika tidak ada titik lain dalam part1, maka titik initial_point dan final_point menjadi pembentuk convex hull bagian part1.
6. Jika ada 1 titik lain dalam part1, maka titik initial_point dan titik tersebut serta titik tersebut dan final_point menjadi pembentuk convex hull bagian part1.
7. Jika ada lebih dari 1 titik lain dalam part1, pilih sebuah titik yang memiliki jarak terjauh dari garis tersebut (misal titiksampel).
8. Mengabaikan semua titik yang berada di dalam daerah segitiga initial_point titiksampel final_point untuk pemeriksaan lebih lanjut.
9. Menentukan kumpulan titik yang berada di sebelah luar garis initial_point titiksampel, dan di sebelah luar garis titiksampel final_point.
10. Mengulangi langkah 7, 8, dan 9 sampai tidak ada bagian yang bisa dibagi lagi.
11. Melakukan hal yang sama untuk bagian part2, hingga semua bagian tidak dapat dibagi lagi.
12. Mengumpulkan semua pasangan titik yang dihasilkan dan dimasukkan ke dalam variabel hull.
13. Menghubungkan semua pasangan titik tersebut agar menghasilkan convex hull.

Kode Program

1. myConvexHull.py

```
import numpy as np
from scipy import linalg

def kiri(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] > 0

def kanan(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] < 0

def jaraktitikkegaris(p1, p2, p3):
    return linalg.norm(np.cross(p2-p1, p1-p3)) / linalg.norm(p2-p1)

def area(p1, p2, p3):
    return abs((p1[0] * (p2[1] - p3[1]) + p2[0] * (p3[1] - p1[1]) + p3[0] *
(p1[1] - p2[1])) / 2.0)

def isInside(p1, p2, p3, p):
    A = area(p1, p2, p3)
    A1 = area(p, p2, p3)
    A2 = area(p1, p, p3)
    A3 = area(p1, p2, p)
    if (A == A1 + A2 + A3):
        return True
    else:
        return False

def divideandconquer(semua_bagian, bagian, titik1, titik2, solusi):
    global hull
    hull = solusi
    if (len(bagian) == 0):
        hull.append((titik1, titik2))
    elif (len(bagian) == 1):
        hull.append((titik1, bagian[0]))
        hull.append((bagian[0], titik2))
    else:
        global all_part
        global total_part
        global titiksampel

        jarakmaxtitikkegaris = 0
        titiksampel = bagian[0]
```

```
        for i in range(len(bagian)):
            jarak = jaraktitikkegaris(titik1, titik2, bagian[i])
            if (jarak >= jarakmaxtitikkegaris):
                jarakmaxtitikkegaris = jarak
                titiksampel = bagian[i]

        all_part = semua_bagian
        total_part = len(all_part)

        a = total_part
        all_part.append([])
        total_part += 1

        left = 0
        right = 0
        for i in range(len(bagian)):
            if (kiri(titik1, titiksampel, bagian[i])):
                left += 1
            elif (kanan(titik1, titiksampel, bagian[i])):
                right += 1
        if (left < right):
            for i in range(len(bagian)):
                if (kiri(titik1, titiksampel, bagian[i]) and not
isInside(titik1, titiksampel, titik2, bagian[i])):
                    all_part[a].append(bagian[i])
        elif (left > right):
            for i in range(len(bagian)):
                if (kanan(titik1, titiksampel, bagian[i]) and not
isInside(titik1, titiksampel, titik2, bagian[i])):
                    all_part[a].append(bagian[i])

        b = total_part
        all_part.append([])
        total_part += 1

        left = 0
        right = 0
        for i in range(len(bagian)):
            if (kiri(titiksampel, titik2, bagian[i])):
                left += 1
            elif (kanan(titiksampel, titik2, bagian[i])):
                right += 1
        if (left < right):
            for i in range(len(bagian)):
                if (kiri(titiksampel, titik2, bagian[i]) and not
isInside(titik1, titiksampel, titik2, bagian[i])):
                    all_part[b].append(bagian[i])
        elif (left > right):
```

```
            for i in range(len(bagian)):
                if (kanan(titiksampel, titik2, bagian[i]) and not
isInside(titik1, titiksampel, titik2, bagian[i])):
                    all_part[b].append(bagian[i])

        divideandconquer(all_part, all_part[a], titik1, titiksampel, hull)
        divideandconquer(all_part, all_part[b], titiksampel, titik2, hull)
```

2. main.ipynb

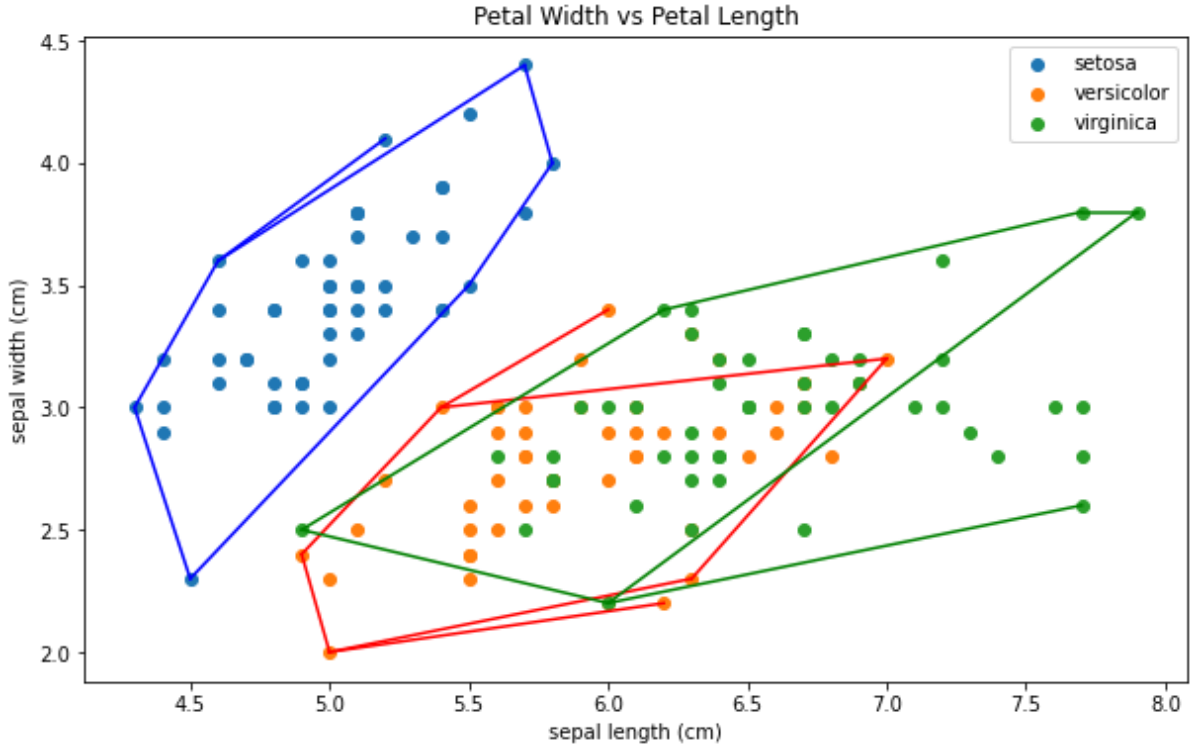
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

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()
```

```
#visualisasi hasil ConvexHull
import myConvexHull

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])
for i in range(len(data.target_names)):
    warna = i
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    bucket = sorted(bucket, key=lambda x: (x[0], x[1]))

    initial_point = bucket[0]
    final_point = bucket[len(bucket)-1]

    part1 = []
    part2 = []
    for i in range(len(bucket)):
        if (myConvexHull.kiri(initial_point, final_point, bucket[i])):
            part1.append(bucket[i])
        elif (myConvexHull.kanan(initial_point, final_point, bucket[i])):
```

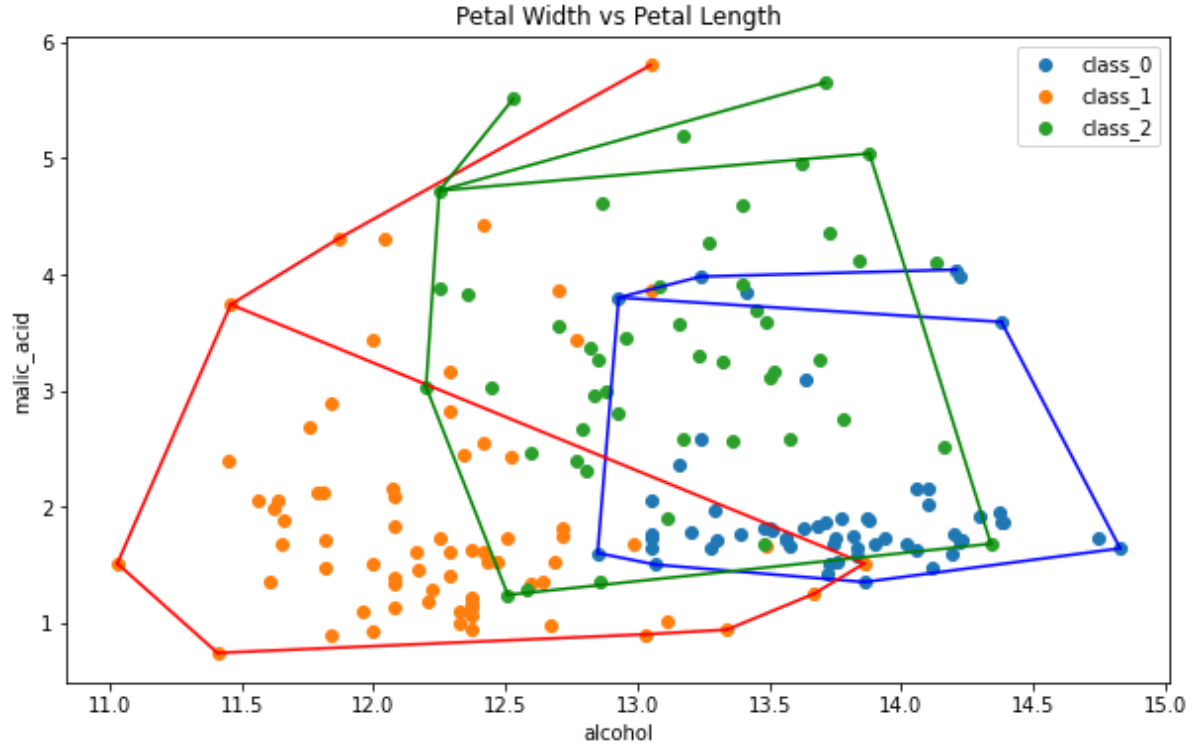
```
            part2.append(bucket[i])

    all_part = []
    total_part = 0
    hull = []
    myConvexHull.divideandconquer(all_part, part1, initial_point, final_point,
hull)
    myConvexHull.divideandconquer(all_part, part2, initial_point, final_point,
hull)

    for i in range(len(hull)):
        x_values = [hull[i][0][0], hull[i][1][0]]
        y_values = [hull[i][0][1], hull[i][1][1]]
        plt.plot(x_values, y_values, colors[warna])

plt.legend()
```

Screenshot Program

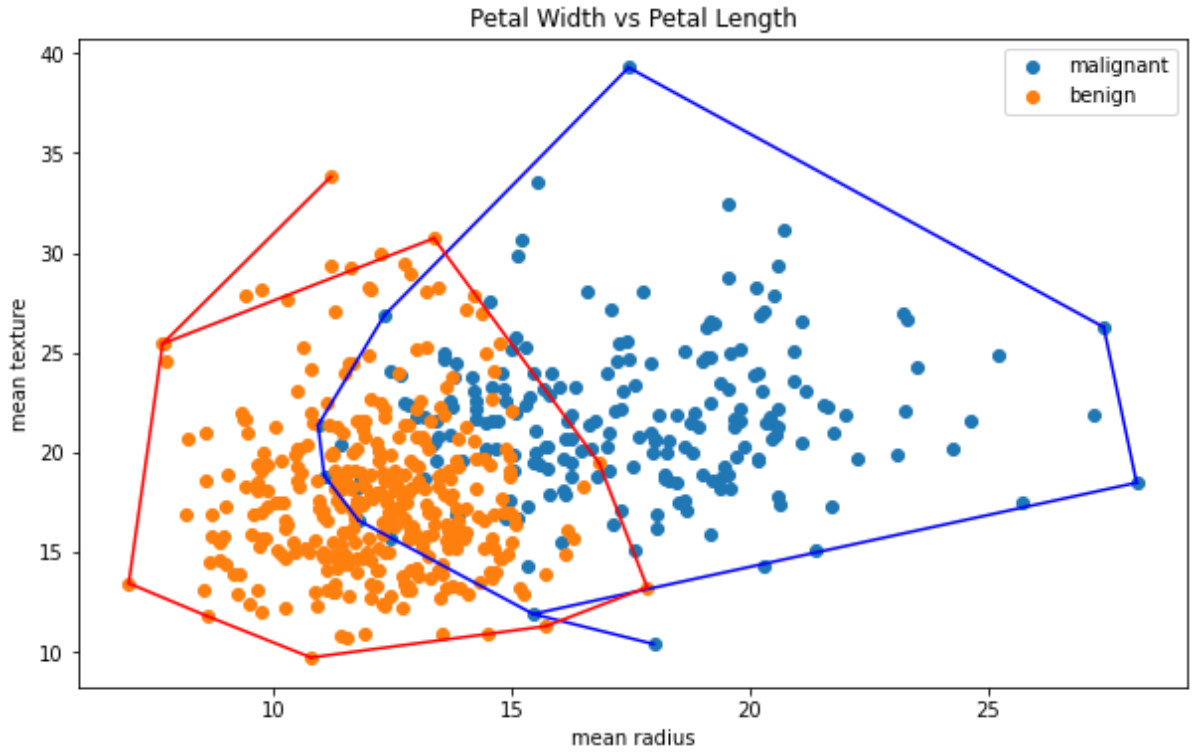
1. load_iris()



2. load_wine()



3. load_breast_cancer()



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.	Program dapat menerima input dan menuliskan output untuk dataset lainnya	✓	

Link repository GitHub : https://github.com/johannes-ws/Tucil-2-Stima