**LAPORAN TUGAS KECIL 2 IF2211 Strategi Algoritma**

**Semester II tahun 2021/2022**

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

**Tahun Akademik 2021-2022**



**Oleh**

|  |  |
| --- | --- |
| **Ken Kalang Al Qalyubi** | **13520010** |

**PROGRAM STUDI TEKNIK INFORMATIKA INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**

**2022**

**BAB I**

***Algoritma Divide and Conquer* Secara Umum**

Algoritma Divide and Conquer secara umum adalah strategi pemecahan masalah yang besar dengan cara melakukan pembagian masalah yang besar tersebut menjadi beberapa bagian yang lebih kecil secara rekursif hingga masalah tersebut dapat dipecahkan secara langsung. Solusi yang didapat dari setiap bagian kemudian digabungkan untuk membentuk sebuah solusi yang utuh. Divide pada algoritma berarti membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya berukuran hampir sama). Conquer (solve) berarti menyelesaikan masing-masing upa-persoalan ( secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar). Combine berarti menggabungkan solusi masing-masing upa-persoalan sehingga membentuk solusi persoalan semula.

Obyek permasalahan yang dibagi adalah masukan (input) atau *instances* yang berukuran n: tabel (larik), matriks, eksponen, dan sebagainya, bergantung pada masalahnya. Tiap-tiap upa-masalah mempunyai karakteristik yang sama (the same type) dengan karakteristik masalah asal, sehingga metode Divide and Conquer lebih natural diungkapkan dalam skema rekursif.

**BAB II**

**Source Code Program dalam bahasa Python**

import math

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn import datasets

import matplotlib.pyplot as plt

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()

#hitung determinan

def determinan(titik1,titik2,titik3):

det = titik1[0]\*titik2[1] + titik3[0]\*titik1[1] + titik2[0]\*titik3[1] - titik3[0]\*titik2[1] - titik2[0]\*titik1[1] - titik1[0]\*titik3[1]

return det

#menentukan dua titik terpanjang berdasarkan x ataupun y terkecil dan terbesar

def titikutama(titikx,titiky):

panjangx = (titikx[-1,0] - titikx[0,0])

panjangy = (titiky[-1,1] - titiky[0,1])

if (panjangy > panjangx):

terpanjang = titiky[0],titiky[-1]

elif (panjangx > panjangy):

terpanjang = titikx[0],titikx[-1]

return terpanjang

#untuk memisahkan antara titik di atas atau di bawah garis utama

def UpperLower(titik1,titik2,titik3):

sortedx = titik3[titik3[:,0].argsort()]

isi = sortedx.tolist()

Upper = []

Lower = []

for i in range (len(isi)):

if isi[i] != titik1 and isi[i] != titik2:

if determinan(titik1,titik2,isi[i]) > 0:

Upper.append(isi[i])

elif determinan(titik1,titik2,isi[i]) < 0:

Lower.append(isi[i])

return Upper,Lower

# cari gradien

def gradien(s1,s2) :

if(s2[0]!=s1[0]):

slope = (s2[1]-s1[1])/(s2[0]-s1[0])

return slope

#perhitungan persamaan garis

def titiktogaris(point,terpanjangx,terpanjangy) :

slope = gradien(terpanjangx,terpanjangy)

yInt = terpanjangx[1] - (slope \* terpanjangx[0])

panjang = abs(-slope\*point[0]+1\*point[1]-yInt)/(math.sqrt(1\*\*2+(slope\*\*2)))

return panjang

#titik terpanjang menggunakan perhitungan persamaan garis

def terpanjang (terpanjangx,terpanjangy,Upper):

Upper = sorted(Upper,key=lambda x:x[0])

temp = 0

terjauh = Upper[0]

for i in range (1,len(Upper)):

if titiktogaris(Upper[i],terpanjangx,terpanjangy) > temp:

temp = titiktogaris(Upper[i],terpanjangx,terpanjangy)

terjauh = Upper[i]

return terjauh

#titik terpanjang menggunakan perhitungan determinan

def titikterpanjang(terpanjangx,terpanjangy,Upper):

Upper = sorted(Upper,key=lambda x:x[0])

temp = 0

terjauh = Upper[0]

for i in range (len(Upper)):

if determinan(terpanjangx,terpanjangy,Upper[i]) > temp:

temp = determinan(terpanjangx,terpanjangy,Upper[i])

terjauh = Upper[i]

return terjauh

#cek titik dalam segitiga atau tidak pakai determinan

def isInsideSegitiga(A,B,C,point):

if(determinan(A,B,point )>=0 and determinan(A, C, point)<=0 and determinan(C, B, point)<=0):

return True

else :

return False

#convexhull utama

def ConvexUpper(titikA,titikB,isi):

isi = sorted(isi,key=lambda x:x[0])

tempUpper = []

tempLower = []

if (len(isi) == 0):

return

else:

temp = titikterpanjang(titikA,titikB,isi)

isi.remove(temp)

convex.append(temp)

for i in range (len(isi)):

if (not isInsideSegitiga(titikA,titikB,temp,isi[i])):

if (determinan(titikA,temp,isi[i]) > 0):

tempUpper.append(isi[i])

elif(determinan(titikB,temp,isi[i]) < 0):

tempLower.append(isi[i])

ConvexUpper(titikA,temp,tempUpper)

ConvexUpper(temp,titikB,tempLower)

#program utama

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)):

convex = []

bucket = df[df['Target'] == i]

bucket = bucket.iloc[:,[0,1]].values

isi = bucket.tolist()

sortedx = bucket[bucket[:,0].argsort()]

sortedy = bucket[bucket[:,1].argsort()]

titik1,titik2 = titikutama(sortedx,sortedy)

#ubah ke list biar mudah

terpanjangx = titik1.tolist()

terpanjangy = titik2.tolist()

#tambahkan titik utama pada array convex

convex.append(terpanjangx)

convex.append(terpanjangy)

#pisahin dari garis utama, atas atau bawah

Upper,Lower = UpperLower(terpanjangx,terpanjangy,bucket)

ConvexUpper(terpanjangx,terpanjangy,Upper)

ConvexUpper(terpanjangy,terpanjangx,Lower)

plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target\_names[i])

#buat list baru buat misahin titik hull atas atau bawah biar plot gampang

atas=[]

bawah=[]

convex = sorted(convex,key=lambda x:x[0])

for j in range(len(convex)):

if(determinan(convex[0],convex[-1],convex[j]) > 0):

atas.append(convex[j])

if(determinan(convex[0],convex[-1],convex[j]) < 0):

bawah.append(convex[j])

bawah.append(convex[0])

atas.append(convex[0])

atas = sorted(atas,key=lambda x:x[0])

bawah = sorted(bawah,reverse=True,key=lambda x:x[0])

convex = atas + bawah

convex\_x=[]

convex\_y=[]

for e in convex :

convex\_x.append(e[0])

convex\_y.append(e[1])

plt.plot(convex\_x,convex\_y,colors[i])

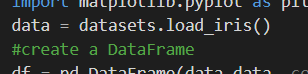
plt.legend()

plt.show()

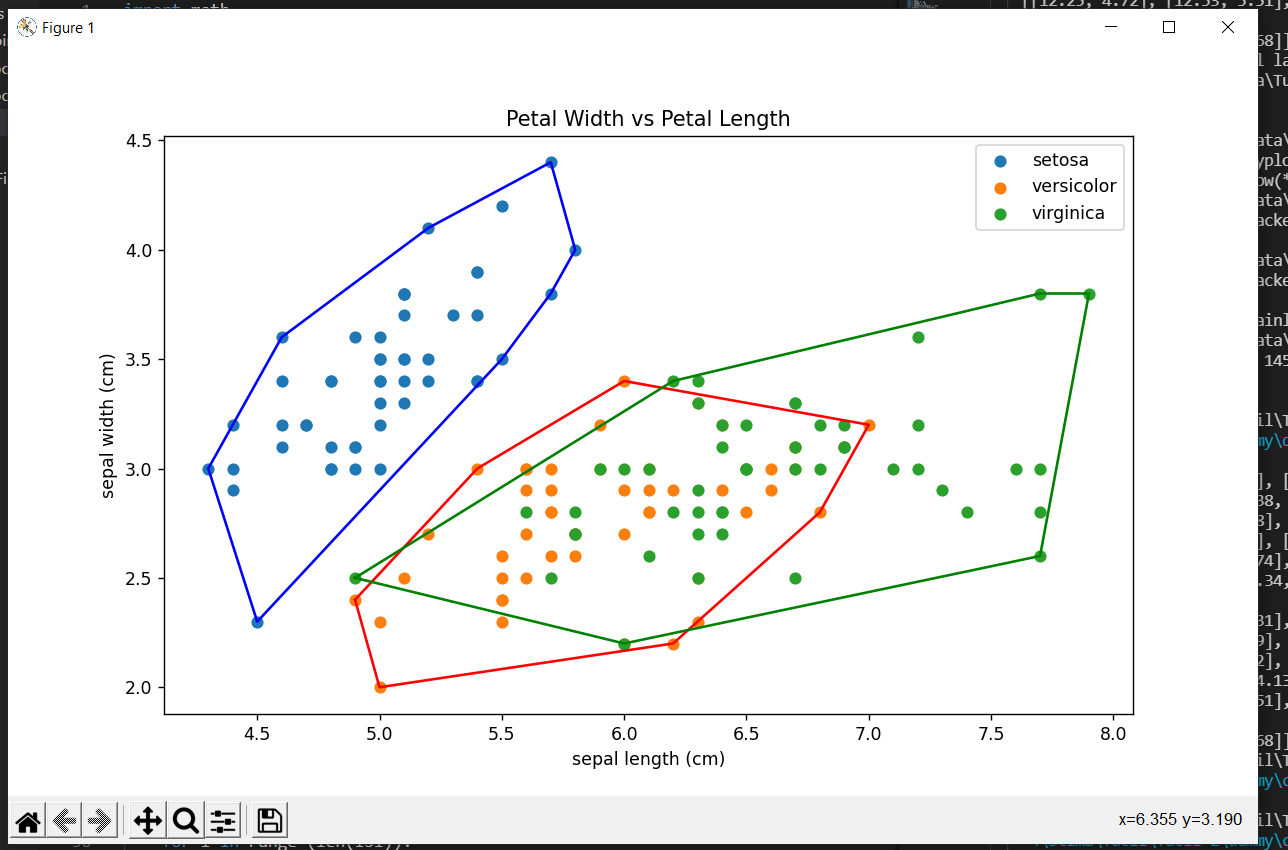
**BAB III**

**Input dan Output Test Case dengan menggunakan program**

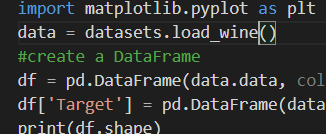
1. Input dataset iris



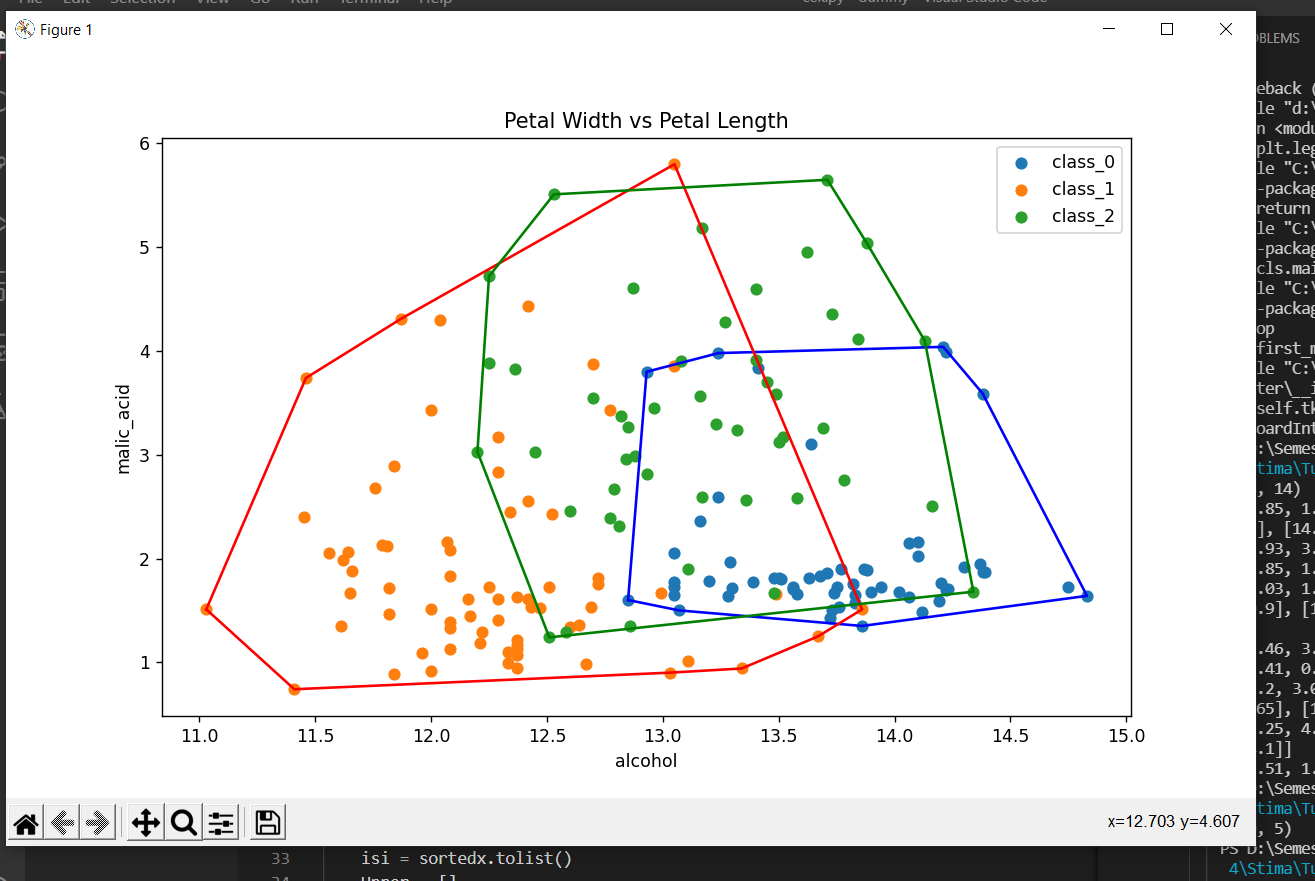
Output :



1. Input datasets wine



Output :



|  |  |  |
| --- | --- | --- |
| 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 | √ |  |

Link github : <https://github.com/kenkalang/Tucil-2-STIMA-Convex-Hull>