Laporan Tugas Kecil 2

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

Rizky Ramadhana P. K.

13520151

Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

I. Algoritma *Divide and Conquer*

Salah satu strategi dalam pemecahan masalah adalah *divide and conquer*. Pada strategi ini, permasalahan dipecah menjadi sub-permasalahan yang ukurannya lebih kecil. Tiap sub-permasalahan akan diselesaikan dan solusi dari tiap sub-permasalahan tersebut akan digabungkan menjadi satu. Strategi ini seringkali lebih baik daripada strategi *brute force* yang mencoba semua kemungkinan yang ada. Hal ini dimungkinkan terjadi karena strategi *divide and conquer* membagi persoalan menjadi persoalan yang lebih kecil yang cenderung lebih mudah untuk dipecahkan.

Salah satu permasalahan yang bisa diselesaikan dengan strategi *divide and conquer* adalah permasalahan pembentukan *convex hull* dari kumpulan titik pada sebuah bidang. Mula-mula dibentuk garis dari titik dengan absis terkecil dan titik dengan absis terbesar. Untuk tiap titik di atas garis dihitung jaraknya ke garis tersebut. Titik yang memiliki jarak terjauh dari garis akan menjadi salah satu pembentuk *convex hull* bersama-sama dengan titik dengan absis terkecil dan titik dengan absis terbesar yang sudah dipilih di awal. Saat ini terbentuk sebuah segitiga, titik-titik di dalam segitiga diabaikan. Titik di sebelah kiri atas segitiga akan dilakukan prosedur yang sama persis seperti langkah sebelumnya, cari titik terjauh untuk ditambahkan ke titik pembentuk *convex hull* sampai tidak ada titik yang tersisa. Kelompok titik di kanan atas juga dilakukan hal yang sama. Lakukan hal yang sama pada titik-titik di bawah garis yang terbentuk dari titik dengan absis terkecil dan titik dengan absis terbesar sebagaimana hal yang dilakukan pada titik-titik di atas garis tersebut.

Pada tugas ini akan dibuat implementasi dari algoritma yang dijelaskan pada paragraf sebelumnya dan algoritma tersebut akan dibandingkan dengan metode pencarian *convex hull* yang dimiliki *library* scipy. Poligon *convex hull* tersebut akan diterapkan pada kumpulan dataset milik *library* sklearn. Akan diambil dua atribut dan dibentuk *convex hull* untuk masing-masing kelompok data dengan nilai target yang sama. Kumpulan data tersebut beserta *convex hull* nya akan disimpan dalam bentuk gambar. Gambar inilah yang akan dibandingkan antara algoritma yang dibuat sendiri dengan algoritma milik *library scipy.*

II. Kode Program

Untuk catatan, beberapa baris kode pada bagian ini terlalu panjang untuk dituliskan dalam satu baris. Sehingga baris kode tersebut dipisahkan dalam dua baris yang mungkin saja meyebabkan error saat menjalankannya. Untuk referensi kode yang

benar dan berhasil dijalankan dapat dilihat pada *source code* yang ikut dikumpulkan.

main.py

```
import numpy as np
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from utils import *

#dataset iris
data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
drawConvexHull(df,data.target_names,"scipy-iris",1)
drawConvexHull(df,data.target_names,"myConvexHull-iris",0)

#bonus : dataset lain
data = datasets.load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
drawConvexHull(df,data.target_names,"scipy-wine",1)
drawConvexHull(df,data.target_names,"myConvexHull-wine",0)
```

utils.py

```
#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
from scipy.spatial import ConvexHull
from myConvexHull import *

def drawConvexHull(df,target_names,filename,mode):
    count=0
    for i in range(0,min(3,len(df.columns)-1)): #take 3 pairs for every dataset if possible
        for j in range(i+1,min(3,len(df.columns)-1)):
            plt.figure(figsize = (10, 6))
            colors = ['b','r','g','y','p','o']
            plt.xlabel(df.columns[i])
            plt.ylabel(df.columns[j])
            for k in range(len(target_names)):
                bucket = df[df['Target'] == k]
                bucket = bucket.iloc[:,[i,j]].values
                if(mode==0): #using myConvexHull
                    plt.title(f'{df.columns[j]} vs {df.columns[i]} (self-made)')
```

```
                    hull = myConvexHull(bucket)
                    plt.scatter(bucket[:, 0], bucket[:, 1], label=target_names[k])
                    for simplex in hull :
                        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[k])
                else : #using scipy's ConvexHull
                    plt.title(f'{df.columns[j]} vs {df.columns[i]} (scipy)')
                    hull = ConvexHull(bucket)
                    plt.scatter(bucket[:, 0], bucket[:, 1], label=target_names[k])
                    for simplex in hull.simplices:
                        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[k])


            plt.legend()
            plt.savefig('result/'+filename+'-'+str(count)+'.png')
            count+=1
```

myConvexHull.py

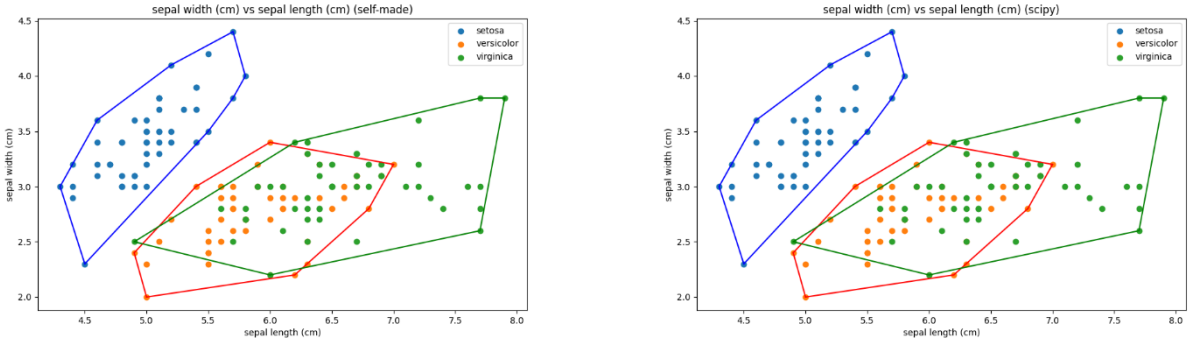
```
from numpy import argmin,argmax,cross,arccos,clip,dot
from numpy.linalg import norm

def myConvexHull(points):
    #return list of lines that shape the convex hull
    #line is a pair of integer [a,b]. a and b are
    #index of the points that shape the line
    rows = len(points)
    minAbsis=argmin(points,0)[0]
    maxAbsis=argmax(points,0)[0]
    p1 = points[minAbsis]
    pn = points[maxAbsis]

    above = [i for i in range(0,rows) if
                aboveLine(p1[0],p1[1],pn[0],pn[1],points[i][0],points[i][1])==1]
    below = [i for i in range(0,rows) if
                aboveLine(p1[0],p1[1],pn[0],pn[1],points[i][0],points[i][1])==-1]

    hullAbove=convexHullAbove(points,above,[minAbsis,maxAbsis])
    hullBelow=convexHullAbove(points,below,[maxAbsis,minAbsis])
    result=hullAbove + hullBelow
    return result

def convexHullAbove(points,indexes,line):
    #return list of lines that shapes the convex hull at the left of the line
    rows = len(indexes)
    if rows==0 : #basis
        return [[line[0],line[1]]]
    elif rows==1: #basis
        return [[line[0],indexes[0]],[indexes[0],line[1]]]
```

```
    else : #reccurence
        next = getNextPoint(points,indexes,[points[line[0]],points[line[1]]])
        left=[]
        right=[]
        for index in indexes :
            isAbove=aboveLine(points[line[0]][0],points[line[0]][1],points[next][0]
                              ,points[next][1],points[index][0],points[index][1])
            if isAbove==1 :
                left+=[index]

        for index in indexes :
            isAbove=aboveLine(points[next][0],points[next][1],points[line[1]][0],
                              points[line[1]][1],points[index][0],points[index][1])
            if isAbove==1 :
                right+=[index]

        #divide to two regions and solve each of them
        leftConvexHull = convexHullAbove(points,left,[line[0],next])
        rightConvexHull = convexHullAbove(points,right,[next,line[1]])

        #combine
        result=leftConvexHull+rightConvexHull
        return result

def getNextPoint(points,indexes,line): #indexes have two elements at least
    #return the furthest point from a line. If several points have same distance,
    #will return the point that maximize the angle formed
    maxDistance=-1
    maxIndex=0
    for index in indexes:
        distance = calculateDistance(points[index],line)
        if  distance > maxDistance :
            maxDistance=distance
            maxIndex=index
        elif distance==maxDistance :
            maxIndex=getMaxAngle(points,index,maxIndex,line)
    return maxIndex

def calculateDistance(point,line) :
    return norm(cross(line[1]-line[0], line[0]-point)) / norm(line[1]-line[0])

def getMaxAngle(points,index,maxIndex,line):
    p=line[1]-line[0]
    q=points[index]-line[0]
    r=points[maxIndex]-line[0]
    angle1=arccos(clip(dot(p,q)/norm(p)/norm(q),-1,1))
    angle2=arccos(clip(dot(p,r)/norm(p)/norm(r),-1,1))
    if angle1>angle2 :
```

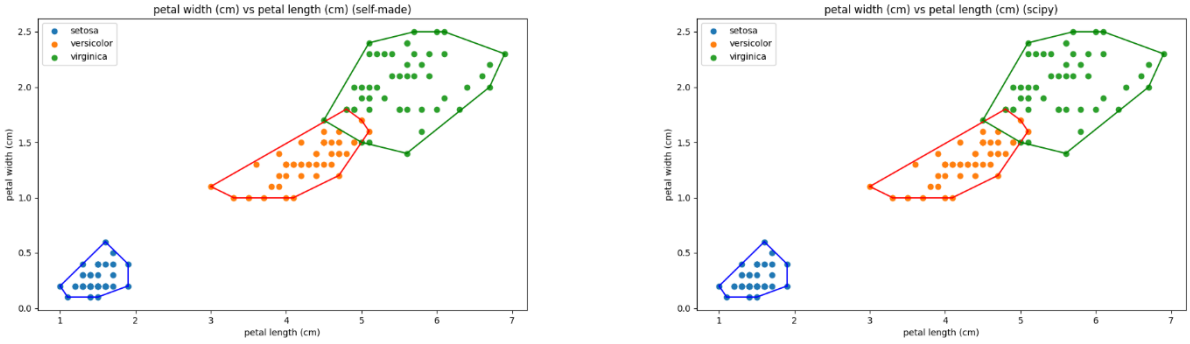
```
        return index
    else :
        return maxIndex

def aboveLine(x1,y1,x2,y2,x3,y3):
    param = (x1*y2+x3*y1+x2*y3) - (x3*y2+x2*y1+x1*y3)
    if param>0:
        return 1 #x3,y3 located above line
    elif param<0:
        return -1 #x3,y3 located below line
    else :
        return 0 #x3,y3 located in line
```

III. Hasil Eksekusi Program

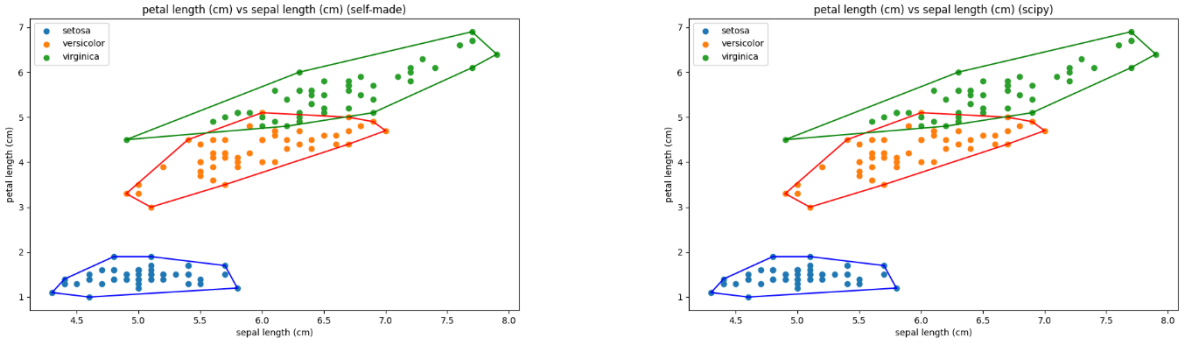
Sebagai catatan, spesifikasi input telah dituliskan pada kode bagian main.py. Berikut hanya ditampilkan hasil output program.



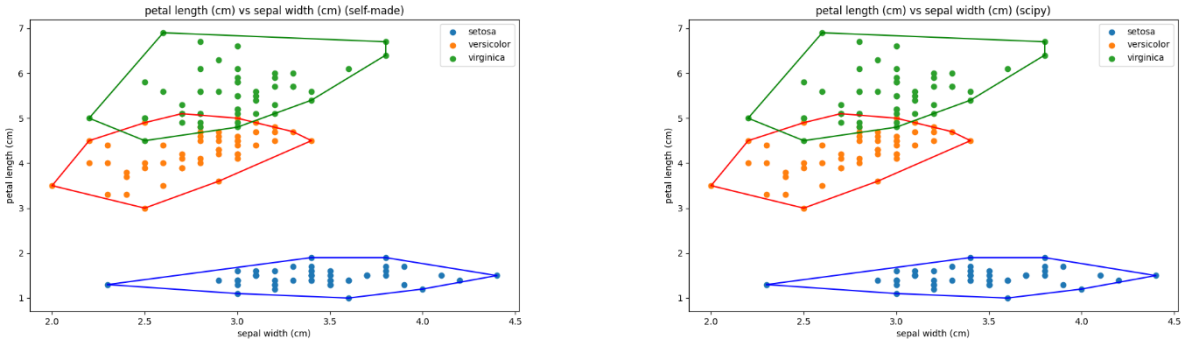
Gambar 1 Perbandingan convex hull pada atribut sepal width dan sepal length



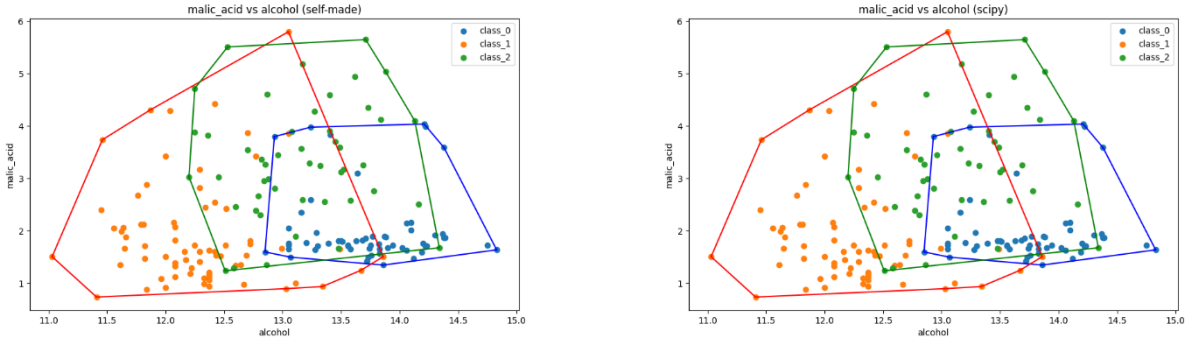
Gambar 2 Perbandingan convex hull pada atribut petal width dan petal length



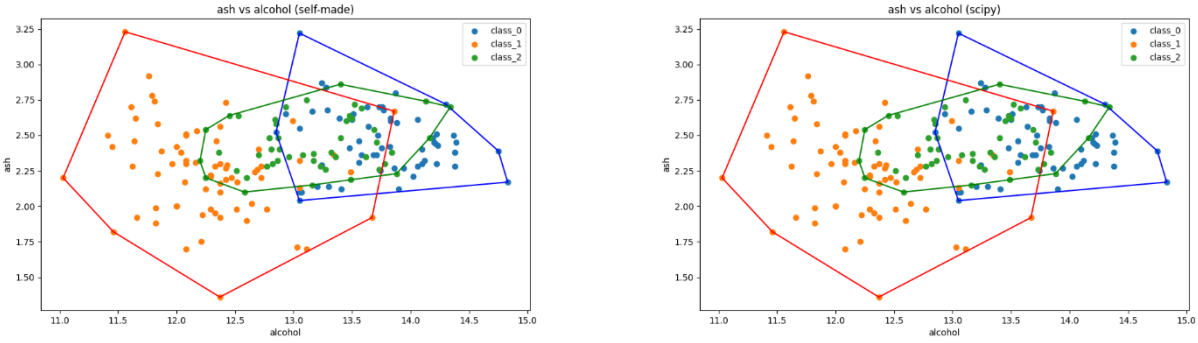
Gambar 3 Perbandingan convex hull pada atribut petal length dan sepal length



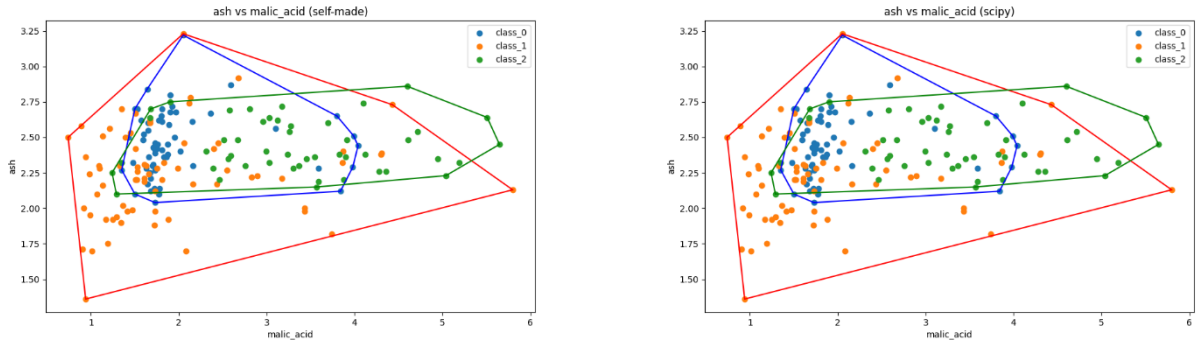
Gambar 4 Perbandingan convex hull pada atribut petal length dan sepal width



Gambar 5 Perbandingan convex hull pada atribut malic_acid dan alcohol



Gambar 6 Perbandingan convex hull pada atribut ash dan alcohol



Gambar 7 Perbandingan convex hull pada atribut ash dan malic_acid

IV. Lampiran

Checklist

Poin	Ya	Tidak
1. Pustaka myConvexHull berhasil dibuat dan tidak ada kesalahan	**V**	
2. Convex hull yang dihasilkan sudah benar	**V**	
3. Pustaka myConvexHull dapat digunakan untuk menampilkan convex hull setiap label dengan warna yang berbeda.	**V**	
4. Bonus: program dapat menerima input dan menuliskan output untuk dataset lainnya.	**V**	

Link Repository

https://github.com/rizkyramadhana26/Tucil-Stima-2