

Laporan Tugas Kecil 2

IF2211 Strategi Algoritma

Semester II Tahun 2021/2022



disusun oleh:

Muhammad Fikri Ranjabi 13520002

K-02

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG 2022**

Daftar Isi

Daftar Isi	2
A. Algoritma Divide and Conquer.....	3
B. Kode Program dalam Bahasa Python	4
C. Screenshot Input dan Output.....	12
D. Alamat Kode Program	14

A. Algoritma Divide and Conquer

Program pencarian *convex hull* dibuat menggunakan algoritma *divide and conquer*. Langkah-langkah algoritma pencarian *convex hull* adalah sebagai berikut:

1. Cari titik ekstrem p_1 dan p_n dari kumpulan titik yang tersedia.
2. Bagi himpunan titik (S) menjadi dua bagian dengan garis yang menghubungkan p_1 dan p_n . S_1 adalah kumpulan titik di sebelah kiri atau atas garis p_1p_n dan S_2 adalah kumpulan titik di sebelah kanan atau bawah garis p_1p_n .
3. S_1 dan S_2 menjadi kandidat untuk membentuk *convex hull* bagian atas dan bawah.
4. Untuk sebuah bagian (misal S_1), terdapat dua kemungkinan:
 - Jika tidak ada titik selain S_1 , maka titik p_1 dan p_n menjadi pembentuk *convex hull* bagian S_1 .
 - Jika S_1 tidak kosong, pilih sebuah titik yang memiliki jarak terjauh dari garis p_1p_n (misal p_{max}).
5. Tentukan kumpulan titik yang berada di sebelah kiri garis p_1p_{max} menjadi bagian $S_{1,1}$ dan di sebelah kanan garis p_1p_{max} menjadi bagian $S_{1,2}$. Abaikan titik yang berada di dalam daerah segitiga (p_{max}, p_1, p_n) untuk pemeriksaan lebih lanjut.
6. Lakukan langkah 4 dan 5 untuk bagian S_2 , hingga bagian kiri dan kanan kosong.
7. Kembalikan pasangan titik yang dihasilkan.

B. Kode Program dalam Bahasa Python

File helper.py:

```
import numpy as np
from numpy import linalg as LA

def findDet(p1, pn, arrayDet):
    # Mengembalikan determinan dari tiga titik (p1, pn, arrayDet)
    x1 = p1[0]
    y1 = p1[1]
    x2 = pn[0]
    y2 = pn[1]
    x3 = arrayDet[0]
    y3 = arrayDet[1]
    a = np.array([[x1,y1,1], [x2,y2,1], [x3,y3,1]])
    return (np.linalg.det(a))

def area(x1, y1, x2, y2, x3, y3):
    # Mengembalikan area segitiga dari (x1,y1), (x2,y2), (x3,y3)
    return abs((x1 * (y2 - y3) + x2 * (y3 - y1)
               + x3 * (y1 - y2)) / 2.0)

def isInside(x1, y1, x2, y2, x3, y3, x, y):
    # Mengecek apakah p(x,y) terdapat di dalam segitiga A(x1,y1), B(x2,y2), dan C(x3,y3)

    # Menghitung area segitiga ABC
    A = area (x1, y1, x2, y2, x3, y3)
    # Menghitung area segitiga PBC
    A1 = area (x, y, x2, y2, x3, y3)
    # Menghitung area segitiga PAC
    A2 = area (x1, y1, x, y, x3, y3)
    # Menghitung area segitiga PAB
    A3 = area (x1, y1, x2, y2, x, y)

    # Mengecek apakah A = hasil penjumlahan A1 + A2 + A3
    if(A == A1 + A2 + A3):
        return True
    else:
        return False

def printList(list):
    # Mencetak setiap elemen list ke layar
    for x in list:
        print(x)

def findNorm(p1,p2,p3):
    # Mengembalikan nilai norma pada vektor
```

```

        return (LA.norm(np.cross(p2-p1, p1-p3))/LA.norm(p2-p1))

def ifAlready(elmt, list):
    # Mengembalikan nilai True jika elmt sudah ada di list
    found = False
    for i in range(len(list)):
        v = np.array(elmt) == np.array(list[i])
        if (v.all()):
            found = True
    return found

def remove_duplicate(list):
    # Menghapus elemen duplikat di list
    new = []
    for elmt in list:
        if not(ifAlready(elmt,new)):
            new.append(elmt)
            # print("append")
    list = new

```

File convexHull.py:

```

import math
from helper import *

def findP1(array):
    # Mengembalikan point dengan x terletak di koordinat paling kiri
    min = array[0]
    for i in range(len(array)):
        if (array[i][0] <= min[0]):
            min = array[i]
    return min

def findPn(array):
    # Mengembalikan point dengan x terletak di koordinat paling kanan
    max = array[0]
    for i in range(len(array)):
        if (array[i][0] >= max[0]):
            max = array[i]
    return max

def cekPosNeg(bucketone, s1, s2, p1, pn):
    # Menambahkan point list ke s1 jika determinan > 0, menambahkan ke s2
    # jika determinan < 0
    for i in range(len(bucketone)):
        det = findDet(p1, pn, bucketone[i])
        if (det > 0):
            s1.append(bucketone[i])
        elif (det < 0):
            s2.append(bucketone[i])

```

```

def findPmax(s1p1,s1p2,s1):
# Mengembalikan nilai Pmax dari p1 dan p2 pada s1
# Berlaku juga untuk s2
    max = findNorm(s1p1,s1p2,s1[0])
    maxIndex = 0
    for x in range(len(s1)):
        temp = findNorm(s1p1,s1p2,s1[x])
        if (temp>max):
            max = temp
            maxIndex = x
    # print("temp: ", temp, maxIndex, max)
    return s1[maxIndex]

def findLeftRightTriS1(p1,pn,pmax,s11,s12,s1):
# Menyimpan elemen s1 bagian kiri dan kanan dari segitiga (p1,pn,pmax) ke
dalam s11 dan s12
    temp=[]
    cekPosNeg(s1,s11,temp,p1,pmax)
    cekPosNeg(s1,temp,s12,pn,pmax)

def findLeftRightTriS2(p1,pn,pmax,s21,s22,s2):
# Menyimpan elemen s2 bagian kiri dan kanan dari segitiga (p1,pn,pmax) ke
dalam s21 dan s22
    temp=[]
    cekPosNeg(s2,s21,temp,pmax,p1)
    cekPosNeg(s2,temp,s22,pmax,pn)

def deletePointTriangle(s11,s12,p1,pn,pmax):
# Menghapus titik pada s11 yang terdapat di dalam segitiga (p1,pn,pmax)
# Berlaku juga untuk s21 dan s22 pada s2
    s11delpos = []
    s12delpos = []

    if len(s11) != 0:
        for i in range(len(s11)):
            if
isInside(p1[0],p1[1],pn[0],pn[1],pmax[0],pmax[1],s11[i][0],s11[i][1]):
                s11delpos.append(i)
        if (len(s11delpos)!=0):
            for i in range(len(s11delpos)):
                s11.pop(s11delpos[len(s11delpos)-i-1])

    if len(s12) != 0:
        for i in range(len(s12)):
            if
isInside(p1[0],p1[1],pn[0],pn[1],pmax[0],pmax[1],s12[i][0],s12[i][1]):
                s12delpos.append(i)
        if (len(s12delpos)!=0):
            for i in range(len(s12delpos)):
                s12.pop(s12delpos[len(s12delpos)-i-1])

```

```

def addConvexLeft(sn,chlist,p1,pn):
    # Menambahkan convex list bagian kiri ke s11 dan bagian kanan ke s12
    s11 = []
    s12 = []
    if (len(sn)==0):
        chlist.append(p1)
        chlist.append(pn)
    else:
        pmax = findPmax(p1,pn,sn)

        findLeftRightTriS1(p1,pn,pmax,s11,s12,sn)
        deletePointTriangle(s11,s12,p1,pn,pmax)

        addConvexLeft(s11,chlist,p1,pmax)
        addConvexLeft(s12,chlist,pmax,pn)

def addConvexRight(sn,chlist,p1,pn):
    # Menambahkan convex list bagian kiri ke s21 dan bagian kanan ke s22
    s21 = []
    s22 = []
    if (len(sn)==0):
        chlist.append(p1)
        chlist.append(pn)
    else:
        pmax = findPmax(p1,pn,sn)

        findLeftRightTriS2(p1,pn,pmax,s21,s22,sn)
        deletePointTriangle(s21,s22,p1,pn,pmax)

        addConvexRight(s21,chlist,p1,pmax)
        addConvexRight(s22,chlist,pmax,pn)

def findConvex(listInput):
    # Mengembalikan convex hull dari listInput
    s = listInput

    # bagi s menjadi dua bagian
    s1 = []
    s2 = []

    # cari p1 dan pn
    p1 = findP1(s)
    pn = findPn(s)

    # isi elemen s1 dan s2
    cekPosNeg(s,s1,s2,p1,pn)

    convexList = []

    # cari convex hull pada s1 dan s2

```

```

addConvexLeft(s1,convexList,p1,pn)
addConvexRight(s2,convexList,p1,pn)
# printList(convexList)

# hapus elemen duplikat dan sort berlawanan arah jarum jam
remove_duplicate(convexList)
convexList = sort_counterclockwise(convexList)
convexList.append(convexList[0])

return convexList

def sort_counterclockwise(points):
# Mengurutkan elemen ada points berlawanan arah jarum jam dengan
menghitung theta dari koordinat polar

# Mencari titik tengah dari points dengan menghitung mean
center_x, center_y = sum([x for x,_ in points])/len(points), sum([y for
_,y in points])/len(points)

# Menghitung theta
angles = [math.atan2(y - center_y, x - center_x) for x,y in points]

# Urutkan berdasarkan theta
counterclockwise_temp = sorted(range(len(points)), key=lambda i:
angles[i])
counterclockwise_points = [points[i] for i in counterclockwise_temp]

return counterclockwise_points

```

File main.py:

```

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

from convexHull import *
from sklearn import datasets

def run(data_name):
    if (data_name == "iris sepal"):
        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)
        print(df.head())

        bucketInput = []

        #visualisasi hasil ConvexHull
        plt.figure(figsize = (10, 6))

```



```

        colors = ['b','r','g']

        plt.title('Sepal Width vs Sepal Length')
        plt.xlabel(data.feature_names[0])
        plt.ylabel(data.feature_names[1])
        for i in range(len(data.target_names)):
            bucket = df[df['Target'] == i]
            bucket = bucket.iloc[:,[0,1]].values
            bucketInput.append(bucket)
            # print("bucket: ", i, "\n", bucket)

            # ConvexHull Divide & Conquer
            hull = findConvex(bucketInput[i]) # Implementasi Convex Hull

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

                for elmt in hull:
                    plt.plot([x for x,_ in hull], [y for _,y in hull],
colors[i])
        elif (data_name == "iris petal"):
            data = datasets.load_iris()
            df = pd.DataFrame(data.data, columns=data.feature_names)
            df['Target'] = pd.DataFrame(data.target)
            print(df.shape)
            print(df.head())

            bucketInput = []

            plt.figure(figsize = (10, 6))
            colors = ['b','r','g']

            plt.title('Petal Width vs Petal Length')
            plt.xlabel(data.feature_names[2])
            plt.ylabel(data.feature_names[3])
            for i in range(len(data.target_names)):
                bucket = df[df['Target'] == i]
                bucket = bucket.iloc[:,[2,3]].values
                bucketInput.append(bucket)
                # print("bucket: ", i, "\n", bucket)

                hull = findConvex(bucketInput[i])

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

                    for elmt in hull:
                        plt.plot([x for x,_ in hull], [y for _,y in hull],
colors[i])
            elif (data_name == "breast cancer"):
                data = datasets.load_breast_cancer()

```

```

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

bucketInput = []

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Mean Radius vs Mean Symmetry')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[8])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[0,8]].values
    bucketInput.append(bucket)
    # print("bucket: ", i, "\n", bucket)

    hull = findConvex(bucketInput[i])

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

        for elmt in hull:
            plt.plot([x for x,_ in hull], [y for _,y in hull],
colors[i])

elif (data_name == "wine"):
    data = datasets.load_wine()
    df = pd.DataFrame(data.data, columns=data.feature_names)
    df['Target'] = pd.DataFrame(data.target)
    print(df.shape)
    print(df.head())

    bucketInput = []

    plt.figure(figsize = (10, 6))
    colors = ['b','r','g']
    plt.title('Alcohol vs Malic Acid')
    plt.xlabel(data.feature_names[0])
    plt.ylabel(data.feature_names[1])
    for i in range(len(data.target_names)):
        bucket = df[df['Target'] == i]
        bucket = bucket.iloc[:,[0,1]].values
        bucketInput.append(bucket)
        # print("bucket: ", i, "\n", bucket)

        hull = findConvex(bucketInput[i])

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

```

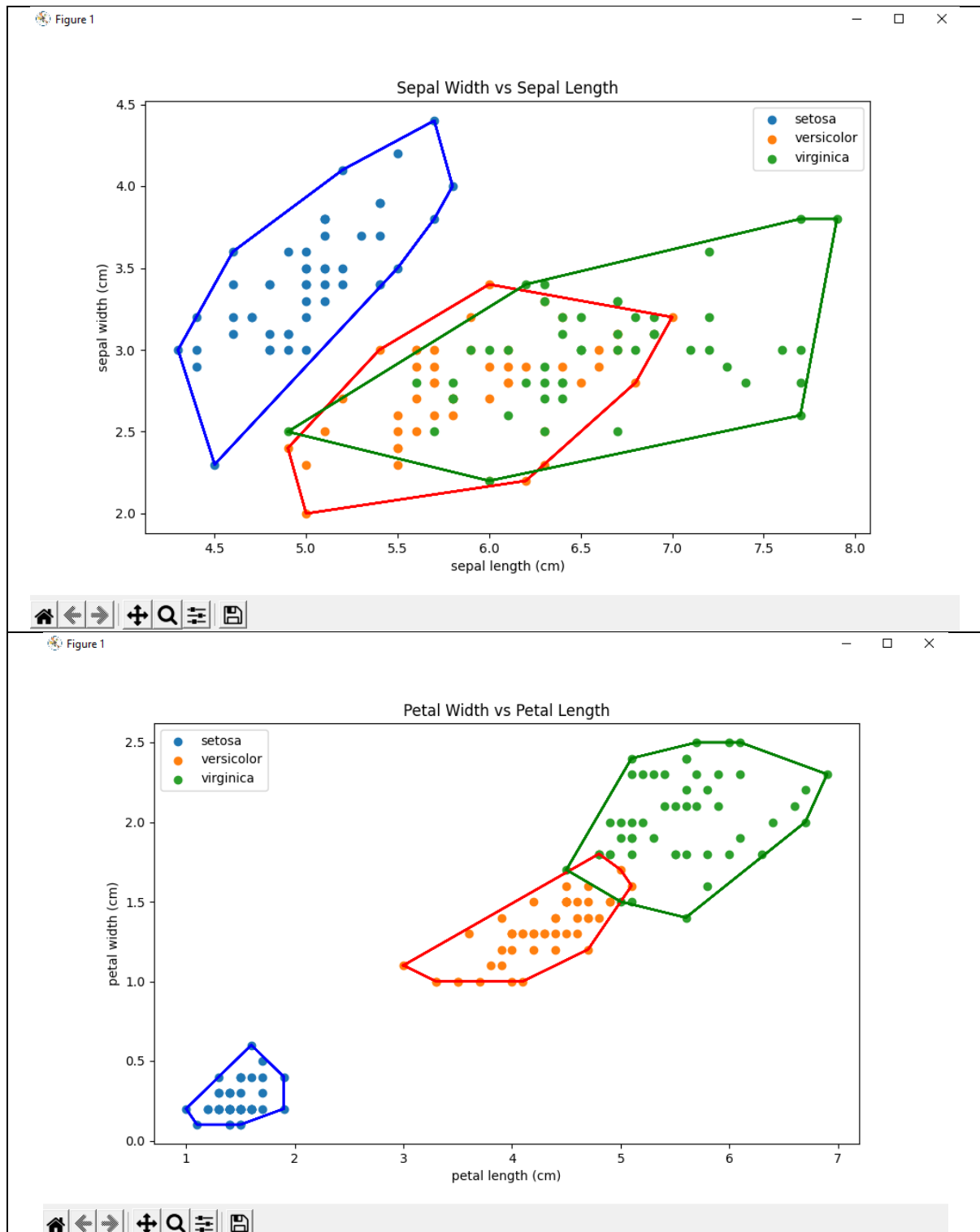
```
        for elmt in hull:
            plt.plot([x for x,_ in hull], [y for _,y in hull],
colors[i])
        plt.legend()
        plt.show()

# Pilih sesuai data yang diinginkan
run("iris sepal")
# run("iris petal")
# run("breast cancer")
# run("wine")
```

C. Screenshot Input dan Output

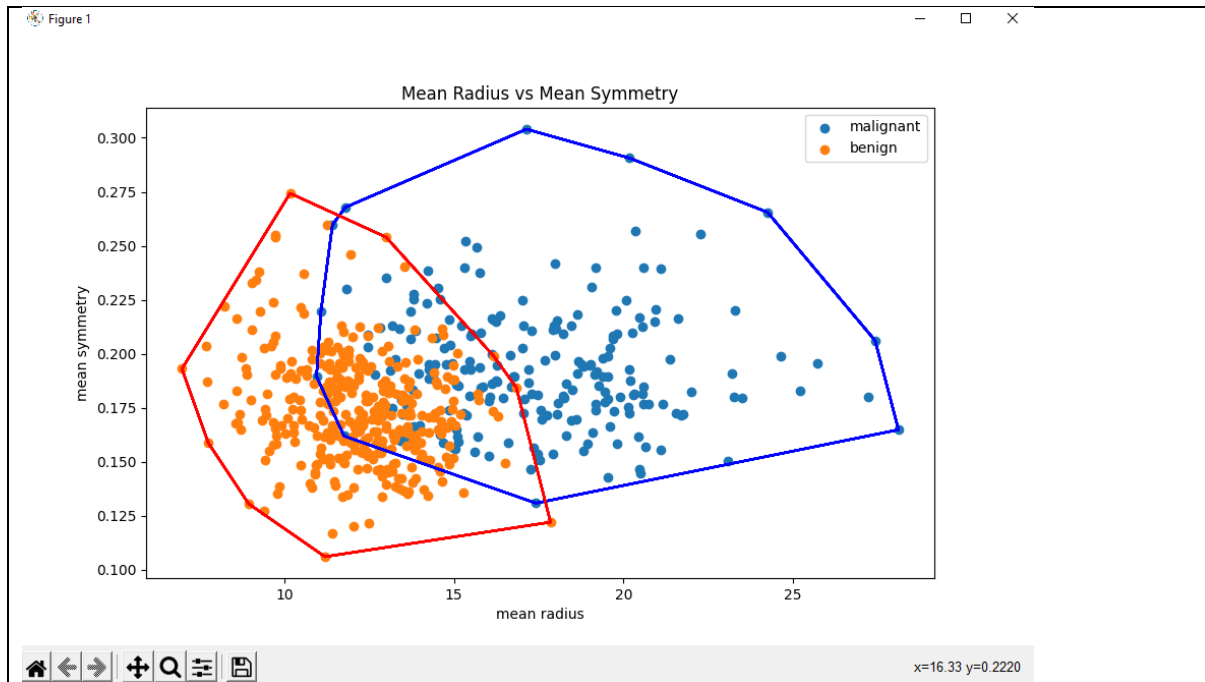
Input pertama diambil dari dataset iris (modul sklearn). Dataset memiliki 150 baris, 5 kolom, dan target sebagai label. Target memiliki 3 value yang mewakili label dataset “setosa, versicolor, dan virginica”.

Berikut adalah output program yang dihasilkan:



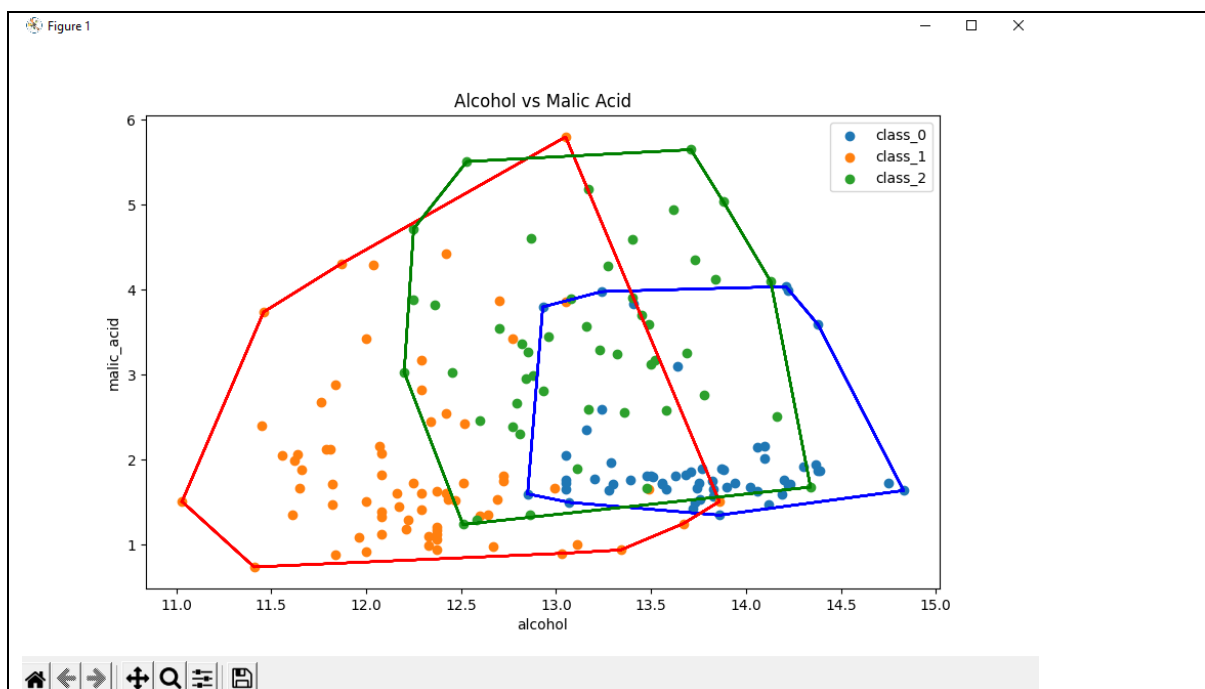
Input kedua diambil dari dataset breast cancer (modul sklearn). Dataset memiliki 569 baris, 31 kolom, dan target sebagai label. Target memiliki 2 value yang mewakili label dataset “malignant, benign”.

Berikut adalah output program yang dihasilkan:



Input ketiga diambil dari dataset wine (modul sklearn). Dataset memiliki 178 baris, 14 kolom, dan target sebagai label. Target memiliki 3 value yang mewakili label dataset “class_1, class_2, class_3”.

Berikut adalah output program yang dihasilkan:



D. Alamat Kode Program

Repository Github:

<https://github.com/ranjabi/Convex-Hull-Divide-and-Conquer>

Google Colab:

<https://colab.research.google.com/drive/1FAlhWu6uDvqAtcFugKK1Lqusuy-5jjmn?usp=sharing>

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.	✓	