Laporan Tugas Kecil 2
IF2211 Strategi Algoritma

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

Disusun Oleh:
13520120 Afrizal Sebastian

Sekolah Tinggi Elektro dan Informatika
Institut Teknologi Bandung
2021

DAFTAR ISI

BAB I
Algoritma *Divide and Conquer*

1.1 Deskripsi Langkah-Langkah Penggunaan Algoritma *Devide and Conquer* pada *Convex Hull*

Terlebih dahulu ambil 2 titik terluar sebagai garis pemotong awal pada kumpulan garis. Pada Kasus ini saya mengambil 2 titik terluar yang memiliki nilai x (absis) terkecil dan nilai x terbesar. Dua titik tersebut akan masuk himpunan titik Convex Hull. Setelah mendapatkan 2 titik tersebut, bagi menjadi 2 sisi yaitu kanan dan kiri sisi dengan menggunakan determinan

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

Gambar 1.1 Determinan, sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf

Jika determinan > 0, maka titik 3 akan berada di sebelah kiri begitu juga sebaliknya.

Setelah kumpulan titik sudah terbagi menjadi dua bagian. Tiap bagian akan diambil titik terjauh dari garis awal, lalu titik tersebut akan masuk himpunan titik Convex Hull. Setelah mendapatkan titik terjauh. hubungkan titik terjauh dengan 2 titik awal sehingga membentuk garis, lalu bagi kembali sisi tersebut menjadi 2 bagian dan ulangi lagi seperti langkah diatas. pengerjaan dilakukan dengan cara rekursif.

Setelah semua titik didapatkan, kumpulan titik tersebut akan di sort dengan cara memutar (clockwise) agar dapat dihungkan tiap titik. Setelah sudah disort, maka titik-titik akan dihubungkan agar nantinya dapat di lakukan plot. Namun, list untuk menampung titik-titik Convex Hull harus dikosongkan terlebih dahulu karena list yang digunakan varable global sehingga harus dikosongkan terlebih dahulu sebelum digunakan kemudian.

BAB II
Source Code Program dengan C++

2.1 myConvexHull

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


import math

#myConvexHull
#======================
listHull = []
'''Untuk Mengetahui titik berada di kiri atau kanan garis'''
def calculateDeterminant(point1, point2,point3):
    return (point1[0]*point2[1]) + (point3[0]*point1[1]) +
(point2[0]*point3[1]) - (point3[0]*point2[1]) - (point2[0]*point1[1]) -
(point1[0]*point3[1])


'''Mencari 2 titik awal sebagai pembagi awal
    2 titik tersebut diambil berdasarkan nilai x terkecil dan nilai x
terbesar'''
def find2InitialPoint(listPoint):
    minPoint = listPoint[0]
    maxPoint = listPoint[0]

    for point in listPoint :
        if(point[0] >= maxPoint[0]) :
            maxPoint = point

        if(point[0] <= minPoint[0]) :
            minPoint = point

    return minPoint, maxPoint

'''Menghitung jarak titik ke garis'''
def calculateDistance(point1, point2, point3) : #Point 3 yang akan dicari
jarak ke garis point1-point2
    return abs((point3[0]-point1[0]) * (point2[1]-point1[1]) - (point3[1]-
point1[1]) * (point2[0]-point1[0]))
    '''Hasil Penyederhanaan rumus jarak titik ke garis
abs(ax+by+c/sqrt(a^2+b^2)
        namun sqrt(a^2+b^2) tidak diperhitungkan'''


'''Mencari titik dengan jarak terjauh dari garis'''
def findMaxDistance(point1, point2, listPoint):
    max = 0
    idxMax = 0
```

```
    for i in range(len(listPoint)):
        tempMax = calculateDistance(point1, point2, listPoint[i])
        if(tempMax >= max) :
            idxMax = i
            max = tempMax

    return listPoint[idxMax]


'''Membagi Menjadi 2 sisi'''
def divideSide(point1, point2, listPoint):
    leftSide = []
    rightSide = []

    for point in listPoint:
        if(any(point1 != point)) and (any(point2 != point)) :
            if(calculateDeterminant(point1, point2, point) > 0) :
                leftSide.append(point)
            elif(calculateDeterminant(point1, point2, point) < 0):
                rightSide.append(point)

    return leftSide,rightSide

#Menyelesaikan sisi kiri dari 2 titik awal dan seterusnya
def solveLeftSide(point1, point2, leftListPoint, countPointInLeft):
    if( countPointInLeft > 0 ):
        pMax = findMaxDistance(point1, point2, leftListPoint)
        listHull.append(pMax)
        #Bagi Kembali sisi
        firstLeftSide,_ = divideSide(point1, pMax, leftListPoint)
        secondLeftSide,_ = divideSide(pMax, point2, leftListPoint)
        solveLeftSide(point1, pMax, firstLeftSide, len(firstLeftSide))
        solveLeftSide(pMax, point2, secondLeftSide, len(secondLeftSide))

#menyelesaikan sisi kanan dari 2 titik awal dan seterusnya
def solveRightSide(point1, point2, rightListPoint, countPointInRight):
    if( countPointInRight > 0 ):
        pMax = findMaxDistance(point1, point2, rightListPoint)
        listHull.append(pMax)
        #Bagi Kembali sisi
        _,firstRightSide= divideSide(point1, pMax, rightListPoint)
        _,secondRightSide = divideSide(pMax, point2, rightListPoint)

        solveRightSide(point1, pMax, firstRightSide, len(firstRightSide))
        solveRightSide(pMax, point2, secondRightSide, len(secondRightSide))


#Mengembalikan points yang menjadi point untuk convexHull
def myConvexHull(listAllPoint):
    minPoint, maxPoint = find2InitialPoint(listAllPoint)
    listHull.append(minPoint)
    listHull.append(maxPoint)

    leftSide, rightSide = divideSide(minPoint, maxPoint, listAllPoint)
```

```
    solveLeftSide(minPoint, maxPoint, leftSide, len(leftSide))
    solveRightSide(minPoint, maxPoint, rightSide, len(rightSide))

    #Clokwise Order point
    centerX = sum(hull[0] for hull in listHull)/len(listHull)
    centerY = sum(hull[1] for hull in listHull)/len(listHull)
    listHull.sort(key=lambda point:math.atan2(point[1]-centerY, point[0]-
centerX))
    orderedHull = []
    for i in range(len(listHull)):
        orderedHull.append(listHull[i])

    #kosongkan kembali listHull, karena variable global
    for i in range(len(orderedHull)):
        listHull.pop()

    return orderedHull

#menkoneksikan dari satu titik ke titik lain
def connectThePoint(hullPoint):
    lenList = len(hullPoint)
    connectPointX = []
    connectPointY = []

    for i in range(lenList):
        if(i == lenList - 1 ):
            connectPointX.append([hullPoint[i][0], hullPoint[0][0]])
            connectPointY.append([hullPoint[i][1], hullPoint[0][1]])
        else:
            connectPointX.append([hullPoint[i][0], hullPoint[i+1][0]])
            connectPointY.append([hullPoint[i][1], hullPoint[i+1][1]])

    return connectPointX, connectPointY
```

2.2 datasets.load_iris()

2.2.1 Petal Width vs Petal Length

```
data = datasets.load_iris()
# create df
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

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)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
```

```
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    connectPointX, connectPointY = connectThePoint(hull)
    for e in range(len(hull)):
        plt.plot(connectPointX[e], connectPointY[e], colors[i])
plt.legend()
```

2.2.1 Sepal Width vs Sepal Length

```
#load
data2 = datasets.load_iris()
# create df
df2 = pd.DataFrame(data2.data, columns=data2.feature_names)
df2['Target'] = pd.DataFrame(data2.target)
print(df2.shape)
df2.head()

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Sepal Width vs Sepal Length')
plt.xlabel(data2.feature_names[2])
plt.ylabel(data2.feature_names[3])
for i in range(len(data2.target_names)):
    bucket2 = df2[df2['Target'] == i]
    bucket2 = bucket2.iloc[:,[2,3]].values
    hull2 = myConvexHull(bucket2)
    plt.scatter(bucket2[:, 0], bucket2[:, 1], label=data2.target_names[i])
    connectPointX2, connectPointY2 = connectThePoint(hull2)
    for e in range(len(hull2)):
        plt.plot(connectPointX2[e], connectPointY2[e], colors[i])
plt.legend()
```

2.3 datasets.load_wine()

2.3.1 Alchohol vs color_intensity

```
#load
data3 = datasets.load_wine()
#create df
df3 = pd.DataFrame(data3.data, columns=data3.feature_names)
df3['Target'] = pd.DataFrame(data3.target)
print(df3.shape)
df3.head()
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Alchohol vs color_intensity')
plt.xlabel(data3.feature_names[0])
plt.ylabel(data3.feature_names[9])
for i in range(len(data3.target_names)):
    bucket3 = df3[df3['Target'] == i]
    bucket3 = bucket3.iloc[:,[0,9]].values
    hull3 = myConvexHull(bucket3)
    plt.scatter(bucket3[:, 0], bucket3[:, 1], label=data3.target_names[i])
    connectPointX3, connectPointY3 = connectThePoint(hull3)
    for e in range(len(hull3)):
```

```
        plt.plot(connectPointX3[e], connectPointY3[e], colors[i])
plt.legend()
```

2.3.2 Alchohol vs proanthocyanins

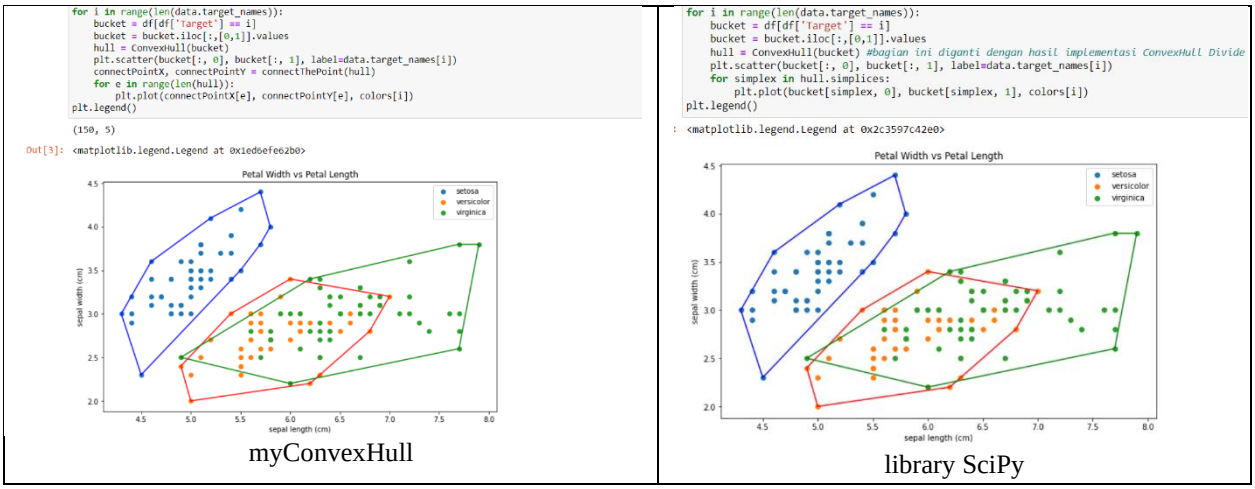
```
#load
data4 = datasets.load_wine()
#create df
df4 = pd.DataFrame(data4.data, columns=data4.feature_names)
df4['Target'] = pd.DataFrame(data4.target)
print(df4.shape)
df4.head()
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Alchohol vs proanthocyanins')
plt.xlabel(data4.feature_names[0])
plt.ylabel(data4.feature_names[8])
for i in range(len(data4.target_names)):
    bucket4 = df4[df4['Target'] == i]
    bucket4 = bucket4.iloc[:,[0,8]].values
    hull4 = myConvexHull(bucket4)
    plt.scatter(bucket4[:, 0], bucket4[:, 1], label=data4.target_names[i])
    connectPointX4, connectPointY4 = connectThePoint(hull4)
    for e in range(len(hull4)):
        plt.plot(connectPointX4[e], connectPointY4[e], colors[i])
plt.legend()
```

2.4 datasets.load_breast_cancer (worst concavity vs worst concave points)

```
data5 = datasets.load_breast_cancer()
#create df
df5 = pd.DataFrame(data5.data, columns=data5.feature_names)
df5['Target'] = pd.DataFrame(data5.target)
print(df5.shape)
df5.head()
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title("worst concavity vs worst concave points")
plt.xlabel(data5.feature_names[26])
plt.ylabel(data5.feature_names[27])
for i in range(len(data5.target_names)):
    bucket5 = df5[df5['Target'] == i]
    bucket5 = bucket5.iloc[:,[26,27]].values
    hull5 = myConvexHull(bucket5)
    plt.scatter(bucket5[:, 0], bucket5[:, 1], label=data5.target_names[i])
    connectPointX5, connectPointY5 = connectThePoint(hull5)
    for e in range(len(hull5)):
        plt.plot(connectPointX5[e], connectPointY5[e], colors[i])
plt.legend()
```

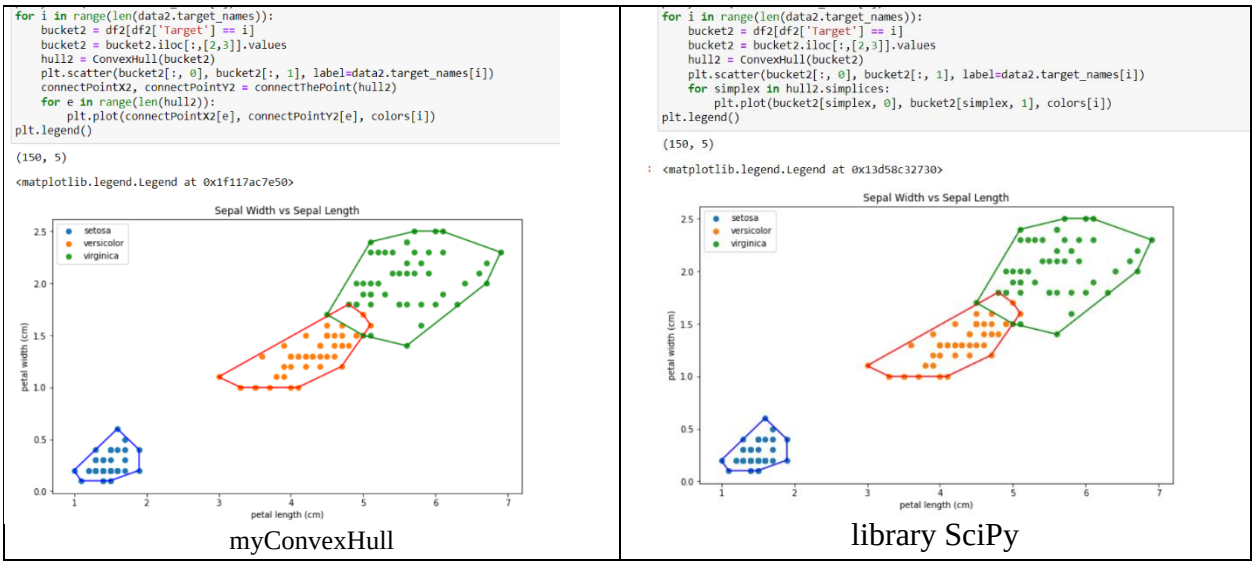
Bab III
Screeshot Hasil

3.1 Petal Width vs Petal Length



Gambar 3.1 : Petal Width vs Petal Length

3.2 Sepal Width vs Sepal Length



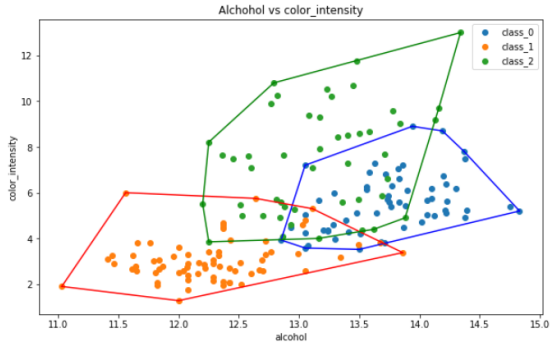
Gambar 3.2 : Sepal Width vs Sepal Length

3.3 Alchohol vs color_intensity

```
for i in range(len(data3.target_names)):
    bucket3 = df3[df3['Target'] == i]
    bucket3 = bucket3.iloc[:,[0,9]].values
    hull3 = ConvexHull(bucket3)
    plt.scatter(bucket3[:, 0], bucket3[:, 1], label=data3.target_names[i])
    connectPointX3, connectPointY3 = connectThePoint(hull3)
    for e in range(len(hull3)):
        plt.plot(connectPointX3[e], connectPointY3[e], colors[i])
plt.legend()
```

```
(178, 14)
```

```
<matplotlib.legend.Legend at 0x1ed6f0e8f70>
```

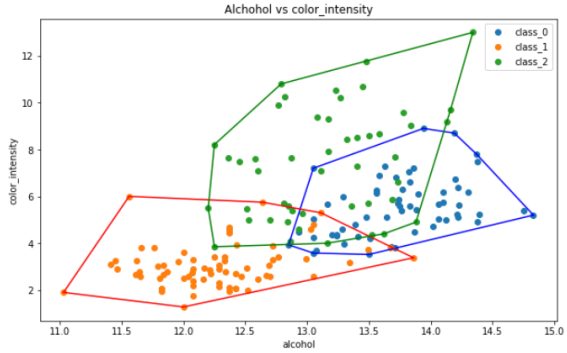


myConvexHull

```
for i in range(len(data3.target_names)):
    bucket3 = df3[df3['Target'] == i]
    bucket3 = bucket3.iloc[:,[0,9]].values
    hull3 = ConvexHull(bucket3)
    plt.scatter(bucket3[:, 0], bucket3[:, 1], label=data3.target_names[i])
    for simplex in hull3.simplices:
        plt.plot(bucket3[simplex, 0], bucket3[simplex, 1], colors[i])
plt.legend()
```

```
(178, 14)
```

```
<matplotlib.legend.Legend at 0x2c3598d1d00>
```



library SciPy

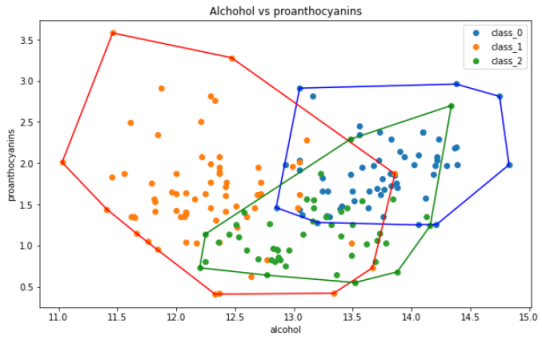
Gambar 3.3 : Alchohol vs color_intensity

3.4 Alchohol vs proanthocyanins

```
for i in range(len(data4.target_names)):
    bucket4 = df4[df4['Target'] == i]
    bucket4 = bucket4.iloc[:,[0,8]].values
    hull4 = ConvexHull(bucket4)
    plt.scatter(bucket4[:, 0], bucket4[:, 1], label=data4.target_names[i])
    connectPointX4, connectPointY4 = connectThePoint(hull4)
    for e in range(len(hull4)):
        plt.plot(connectPointX4[e], connectPointY4[e], colors[i])
plt.legend()
```

```
(178, 14)
```

```
<matplotlib.legend.Legend at 0x1ed6f19ba00>
```

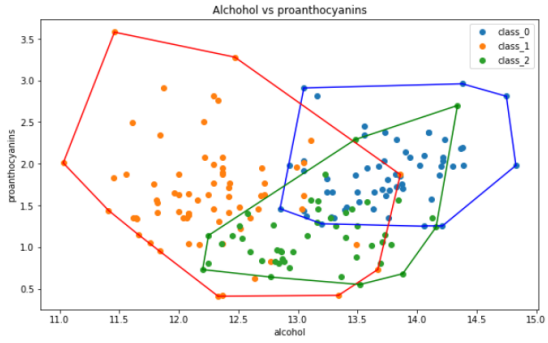


myConvexHull

```
for i in range(len(data4.target_names)):
    bucket4 = df4[df4['Target'] == i]
    bucket4 = bucket4.iloc[:,[0,8]].values
    hull4 = ConvexHull(bucket4)
    plt.scatter(bucket4[:, 0], bucket4[:, 1], label=data3.target_names[i])
    for simplex in hull4.simplices:
        plt.plot(bucket4[simplex, 0], bucket4[simplex, 1], colors[i])
plt.legend()
```

```
(178, 14)
```

```
<matplotlib.legend.Legend at 0x2c359d5c610>
```



library SciPy

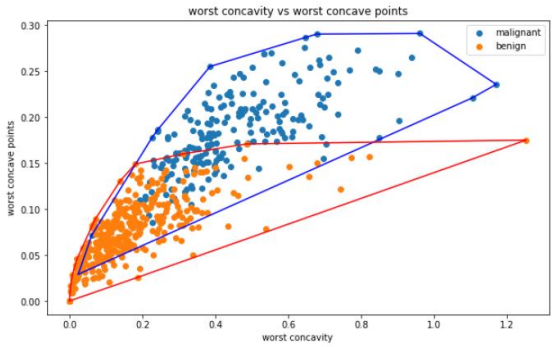
Gambar 3.4 : Alchohol vs proanthocyanins

3.2.2 worst concavity vs worst concave points

```
for i in range(len(data5.target_names)):
    bucket5 = df5[df5['Target'] == i]
    bucket5 = bucket5.iloc[:,[26,27]].values
    hull5 = ConvexHull(bucket5)
    plt.scatter(bucket5[:, 0], bucket5[:, 1], label=data5.target_names[i])
    connectPointX5, connectPointY5 = connectThePoint(hull5)
    for e in range(len(hull5)):
        plt.plot(connectPointX5[e], connectPointY5[e], colors[i])
plt.legend()
```

```
(569, 31)

<matplotlib.legend.Legend at 0x1ed69dcbcd0>
```

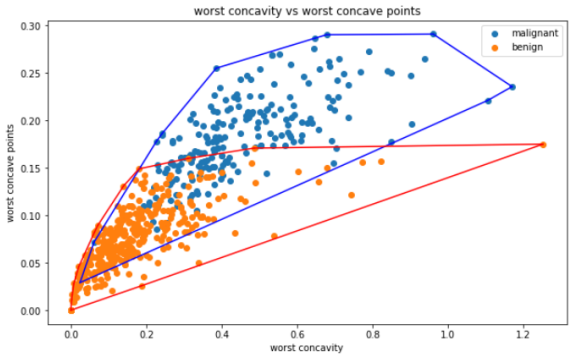


myConvexHull

```
for i in range(len(data5.target_names)):
    bucket5 = df5[df5['Target'] == i]
    bucket5 = bucket5.iloc[:,[26,27]].values
    hull5 = ConvexHull(bucket5)
    plt.scatter(bucket5[:, 0], bucket5[:, 1], label=data5.target_names[i])
    for simplex in hull5.simplices:
        plt.plot(bucket5[simplex, 0], bucket5[simplex, 1], colors[i])
plt.legend()
```

```
(569, 31)

<matplotlib.legend.Legend at 0x2c359d986d0>
```



library Scipy

Gambar 3.2.2 : Hasil Eksekusi Medium2.txt

Bab IV
Alamat GitHub

https://github.com/afrizalsebastian/Tucil2-STIMA

CheckList

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.	√	