Laporan Tugas Kecil 2

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

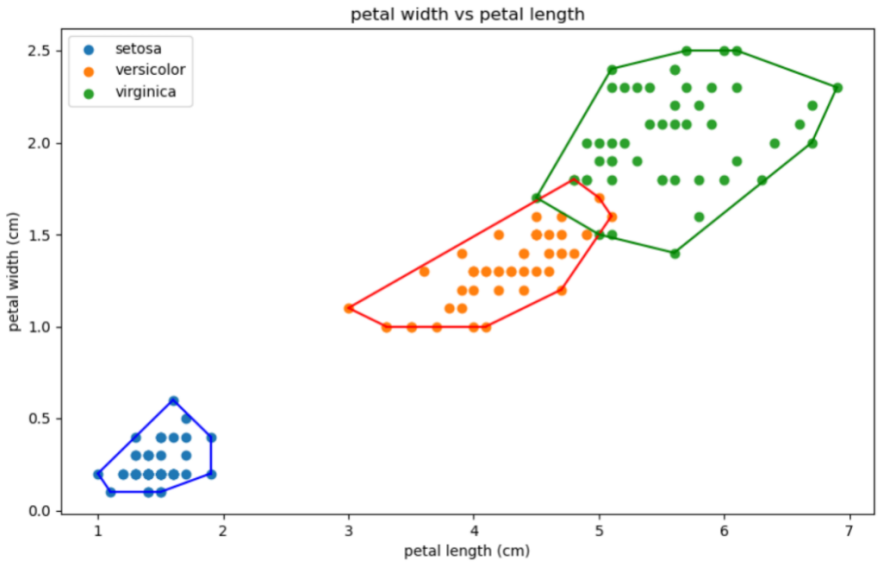
Yoseph Alexander Siregar
K03 / 13520141

Teknik Informatika

Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

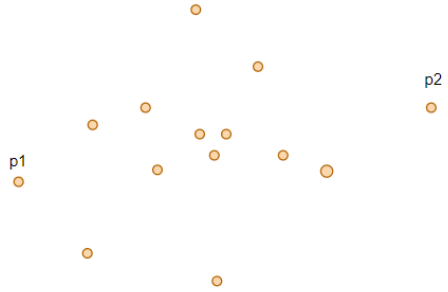
I. Algoritma Divide and Conquer - Convex Hull

Salah satu persoalan dalam komputasi geomerty adalah mencari linear separability dari sebuah dataset. Persoalan ini dapat diselesaikan dengan menggunakan convex hull. Himpunan titik akan dikatakan convex jika untuk 2 buah titik sembarang, seluruh segmen garis yang berakhir pada 2 titik sembarang itu ada pada himpunan titik tersebut. Dengan memanfaatkan hal tersebut, linear separability dari sebuah dataset dapat ditentukan. Gambar dibawah ini menunjukkan hasil visualisasi dari linear separability test dari dataset iris untuk petal width vs petal length dari program tugas kecil saya. Dapat dilihat bahwa setosa adalah kelas yang linearly separable dari kelas lainnya karena kelas versicolor dan virginica saling berpotongan pada convex hullnya yang terbentuk.



Gambar 1.1 - Sumber : Arsip Pribadi

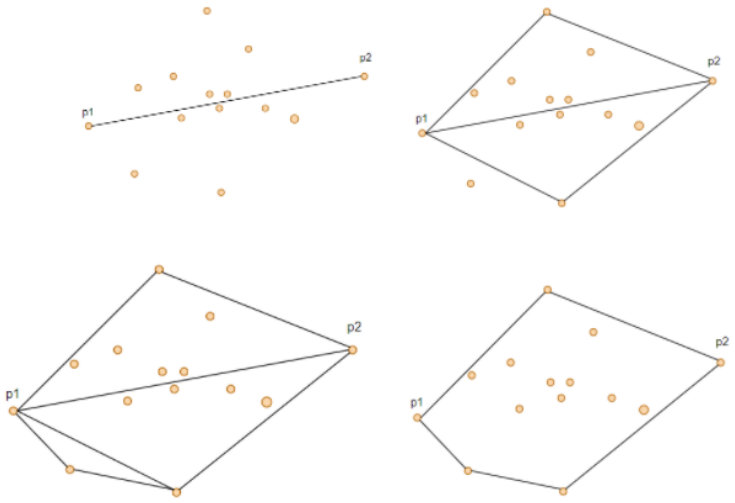
Metode convex hull ini dapat diimplementasikan menggunakan strategi divide and conquer. Misalkan terdapat sebuah himpunan titik seperti gambar dibawah, dua titik ekstrim kiri (absis terkecil, misal p1) dan ekstrim kanan (absis terbesar, misal p2) akan menjadi dasar pembentuk convex hull dari himpunan titik ini.



Gambar 1.2 - Sumber : Arsip Pribadi

Bentuk garis dari kedua titik ini dan bagi bidang menjadi 2 bagian, atas/kiri dan bawah/kanan, untuk tiap bagian dicari titik terjauh dari garis awal. Dari sinilah strategi divide and conquer dimulai, karena kumpulan titik pada bagian atas dan bawah lah yang nantinya akan membentuk convex hullnya. Misal tinjau bagian atas, setelah memilih titik terjauh buat garis kembali dari p1 ke titik terjauh dan titik terjauh ke p2. Untuk langkah selanjutnya akan digunakan proses rekursif, dimana basisnya adalah pengecekan diatas tiap garis yang dibentuk – apabila tidak ada lagi titik di atas garis tersebut, maka kedua titik yang membentuk garis itu akan termasuk dalam titik yang membentuk convexhull.

Berikut tampilan proses pembentukan convex hull dari kumpulan titik di atas :



Gambar 1.3 - Sumber : Arsip Pribadi

Apabila masih terdapat titik di atas garis tersebut ulangi proses ini lagi hingga seluruh bagian atas dari tiap garis kosong. Proses ini akan dilakukan pada kedua bagian yang terbentuk di awal dan titik-titik yang didapatkan akan membentuk convex hull dari kumpulan titik tersebut.

Pada tugas kecil kali ini akan dibuat implementasi dari metode convex hull di atas dan dilakukan visualisasi untuk dataset yang didapat dari library sklearn.

II. Kode Program

main.py

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from convexhull import myConvexHull

#load data
iris = datasets.load_iris()
wine = datasets.load_wine()
b_cancer = datasets.load_breast_cancer()

#create a DataFrame
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_df['Target'] = pd.DataFrame(iris.target)

wine_df = pd.DataFrame(iris.data, columns=iris.feature_names)
wine_df['Target'] = pd.DataFrame(iris.target)

bc_df = pd.DataFrame(iris.data, columns=iris.feature_names)
bc_df['Target'] = pd.DataFrame(iris.target)

# Visualization on convex hull implementasion using iris data set
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('petal width vs petal length')

# choose which atribute to be test
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
for i in range(len(iris.target_names)):
    bucket = iris_df[iris_df['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1],
label=iris.target_names[i])
    for j in range(len(hull)-1):
        plt.plot((hull[j][0], hull[j+1][0]), (hull[j][1],
```

```
hull[j+1][1]), colors[i])
plt.legend()
plt.show()
```

convexhull.py

```
import numpy as np

def myConvexHull(list):
    list = np.array(sort(list)) # save bucket as a numpy array
    # take the farmost left and right point from the data set
    extremeLeft = list[0]
    extremeRight = list[-1]
    # empty list to store point from the above and below of the line
made by the 2 point of extreme left and right
    labove =[]
    lbelow =[]

    # empty list to store convex hull point from above and below of
the line made by the 2 point of extreme left and right
    global final_up, final_down
    final_up= []
    final_down = []

    # check each point and assign it to labove / lbelow based on its
position
    for point in list:
        if(position(point,extremeLeft,extremeRight) == "above"):
            labove.append(point)
        elif(position(point,extremeLeft,extremeRight) =="below"):
            lbelow.append(point)

    #find the convexhullpoint on labove and lbelow
    convexhullPoint(labove,extremeLeft,extremeRight,"up")
    convexhullPoint(lbelow,extremeRight,extremeLeft,"down") # the
positon of extremeRight and left is switch because its position in
below
    sort(final_up) #sort the point based on the axis first then the
ordinate in ascending order
```

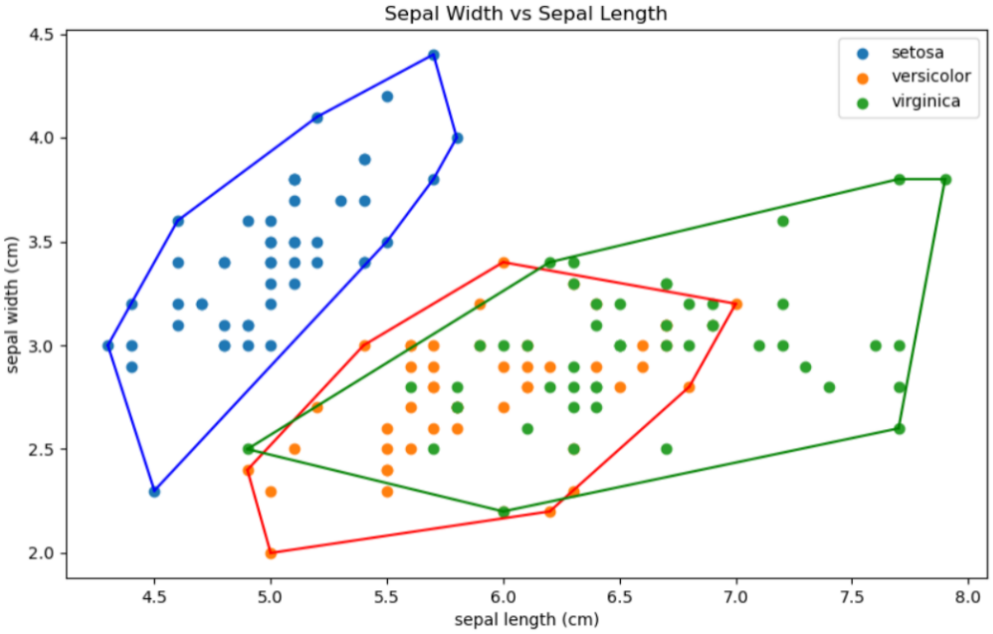
```
    sort_down(final_down) #sort but in descending order
    final = np.concatenate((final_up, final_down), axis=0) #merge the
two list as the final convexhull point
    return(final)

def convexhullPoint(list, min, max,pos):
    #basis
    if(len(list)==0): # list here is the list of point above the
line, if there is no point above the line
        # the two point that make up the line is included as
convexhullpoint
        if(pos =="up"):
            final_up.extend([min])
            final_up.extend([max])
        elif(pos == "down"):
            final_down.extend([min])
            final_down.extend([max])
    else: #recursive
        furthest = furthestpoint(list,min,max) #find the
furthestpoint from the line
        # empty list to store point above the line made by the
previous left point and furthest & furthest and previous right point
        checkabove_left=[]
        checkabove_right=[]
        # check each point and assign based on its position
        for point in list:
            if(position(point,min,furthest) == "above"):
                checkabove_left.extend([point])
        for point in list:
            if(position(point,furthest,max) =="above"):
                checkabove_right.extend([point])
        convexhullPoint(checkabove_left,min,furthest,pos) #recursive
        convexhullPoint(checkabove_right,furthest,max,pos) #recursive

"""
To find whether point(xp,yp) is above the line from (x1,y1) to
(x2,y2) or not, use the determinant concept
If its positive its above, if its negative its below
```

```
x1y2 + xpy1 + x2yp - xpy2 -x2y1 -x1yp
(x1,y1) = min[0][1]
(x2,y2) = max[0][1]
(xp,yp) = point[0][1]

"""
def position(point, min,max):
    xmin, ymin = min
    xmax, ymax = max
    xp,yp = point
    determinant = (xmin*ymax + xp*ymin + xmax*yp - xp*ymax -
xmax*ymin - xmin*yp)
    if(determinant > pow(10,-10)): # error range
        return "above"
    elif(determinant < 0):
        return "below"

def furthestpoint(list,p1,p2):
    furthest = []
    far = 0
    for point in list:
        if(distance(point,p1,p2) >= far):
            far = distance(point,p1,p2)
            furthest = point
    return furthest

def distance(point,p1,p2): # find the distance of point to line make
up by p1 and p2
    p1 = np.asfarray(p1) #convert type to float type
    p2 = np.asfarray(p2)
    point = np.asfarray(point)
    return(np.linalg.norm(np.cross(p2-p1,
p1-point))/np.linalg.norm(p2-p1)) # distance equation


def sort(list): #method to sort the original data set and final_up
    return sorted(list , key=lambda x:[x[0], x[1]])

def sort_down(list): #method to sort final_down, using reverse
```

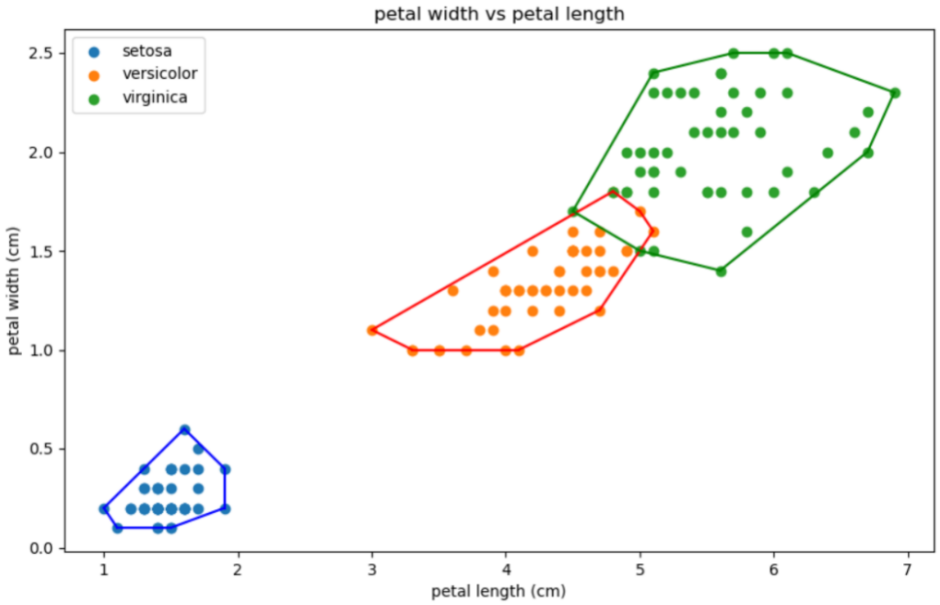
```
because the position of all the point is below
    return sorted(list , key=lambda x:[x[0], x[1]], reverse=True)
```

III. Hasil Eksekusi

1. Dataset iris

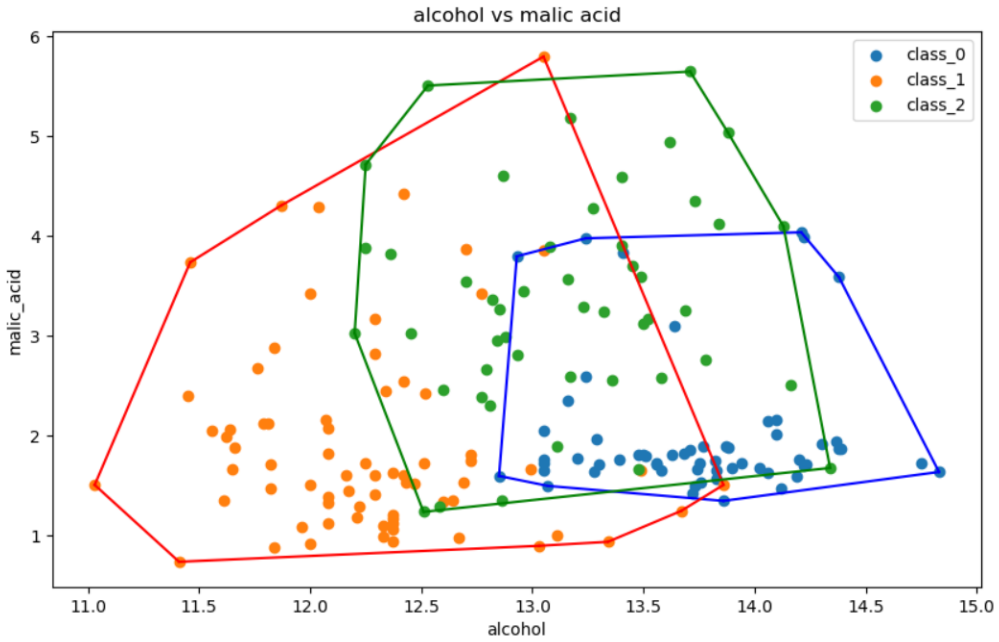


Gambar 3.1.1 - Convexhull pada atribut sepal width vs sepal length

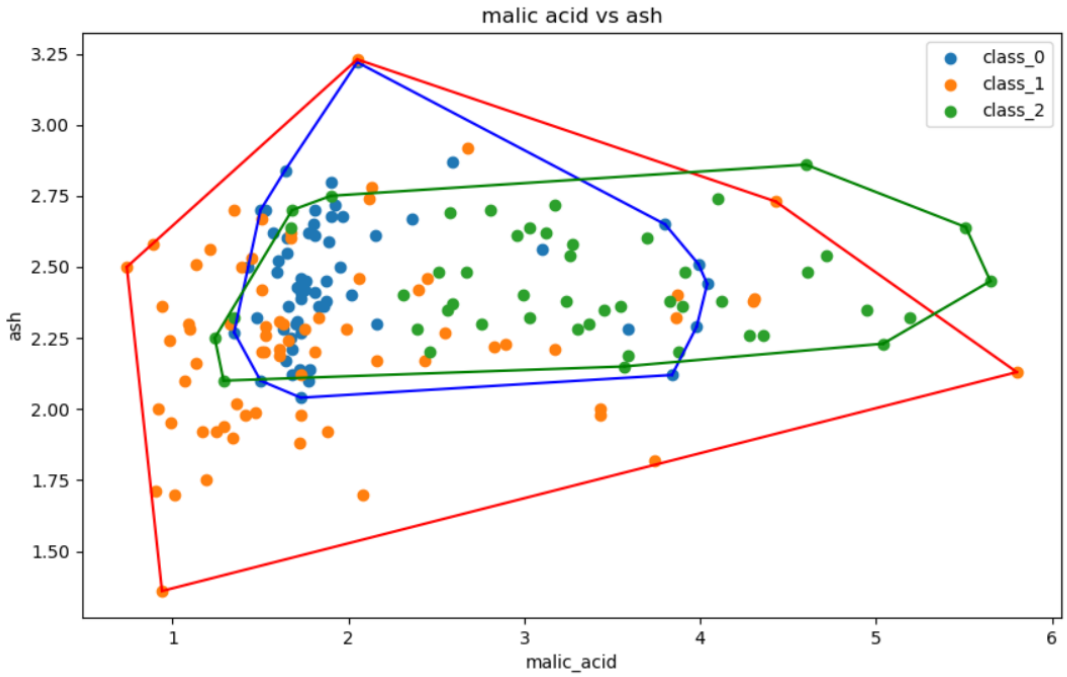


Gambar 3.1.2 - Convexhull pada atribut petal width vs petal length

2. Dataset wine

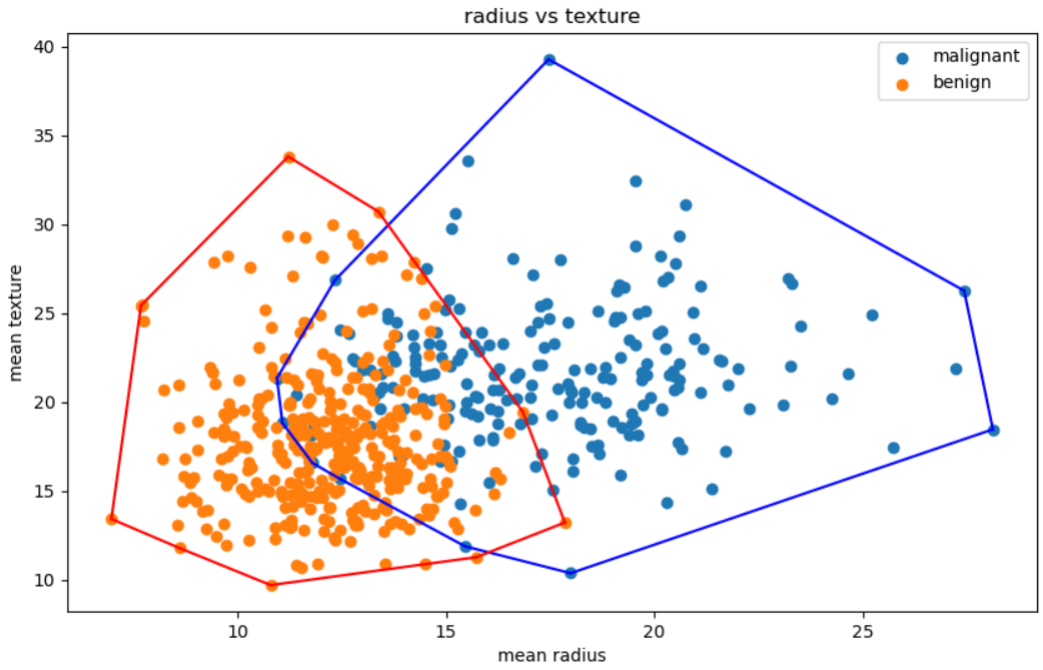


Gambar 3.2.1 - Convexhull pada atribut alcohol vs malic acid

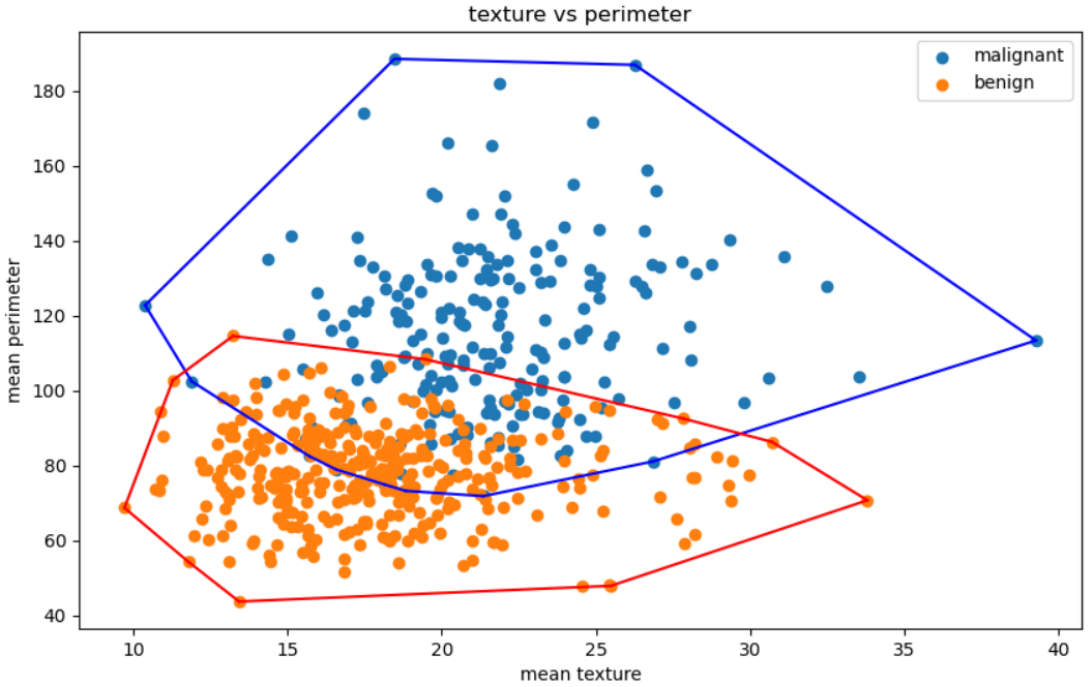


Gambar 3.2.2 - Convexhull pada atribut malic acid vs ash

3. Dataset breast cancer



Gambar 3.3.1 - Convexhull pada atribut mean radius vs mean texture



Gambar 3.3.2 - Convexhull pada atribut mean texture vs mean perimeter

4. 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**	

IV. Link

Repositoty Github : https://github.com/yosalx/LSD_TestvVisualisation-ConvexHull

V. Daftar Pustaka

1. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Tugas-Kecil-2-(2022).pdf
2. https://tatwan.github.io/blog/python/2017/12/31/linear-separability.html