LAPORAN TUGAS KECIL II
IF2211 STRATEGI ALGORITMA

Laporan ini dibuat untuk memenuhi tugas
Mata Kuliah IF 2211 Strategi Algoritma

Disusun Oleh :

Vincent Ho (13520093)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
SEMESTER I TAHUN 2021/2022

BAB I

ALGORITMA DIVIDE AND CONQUER

Algoritma Divide and Conquer yang saya terapkan dalam menyelesaikan permasalahan Convex Hull memiliki langkah – langkah sebagai berikut.

Tahap Divide:
Langkah pertama yang harus dilakukan adalah membagi atau mengelompokkan kumpulan titik menjadi 2 bagian yaitu atas/kiri dan bawah/kanan dengan cara menarik garis antara 2 titik ekstrem Pmin dan Pmax yang diurutkan berdasarkan nilai absis yang menaik (jika ada nilai absis yang sama, maka akan diurutkan dengan nilai ordinat yang menaik). Setelah membagi kumpulan titik menjadi 2 bagian, akan ditentukan apakah titik tersebut berada di sebelah atas/kiri suatu garis atau di sebelah bawah/kanan. Penentuan apakah sebuah titik berada di sebelah atas/kiri atau bawah/kanan suatu garis yang dibentuk oleh dua titik dapat dilakukan dengan penentuan determinan sebagai berikut.

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

Titik (x3, y3) berada di sebelah atas/kiri dari garis yang dibentuk titik (x1, y1) dan (x2, y2) jika hasil dari determinannya positif dan sebaliknya akan berada di sebelah bawah/kanan dari garis apabila hasil determinannya negatif.

Selanjutnya untuk sebuah bagian jika masih terdapat kumpulan titik, maka akan dipilih sebuah titik Pmax yang memiliki jarak terjauh dari garis. Kemudian akan dilakukan lagi pembagian atau pengelompokan kumpulan titik menjadi 2 bagian (tahap divide) yaitu atas(kiri) dan bawah(kanan). Hal ini akan dilakukan terus secara rekursif sampai bagian tersebut tidak tersisa titik lagi.

Tahap Solve:
Apabila tidak ada titik lagi pada sebuah bagian, maka titik Pmin dan Pn akan menjadi pembentuk Convex Hull dari bagian tersebut.

Tahap Combine:
Setelah ditemukan semua titik pembentuk Convex Hull (vertices) maka kumpulan titik atau solusi dari masing-masing upa-persoalan akan digabungkan membentuk pasangan titik (simplices) yang bila dihubungkan akan membentuk Convex Hull.

Beberapa fungsi yang diterapkan dalam library yang saya buat antara lain:

Fungsi	Deskripsi
vertices(arrayOfPoint)	Fungsi yang menerima parameter berupa kumpulan titik yang ada dalam bentuk array dan mengembalikan array dari titik-titik terluar yang membentuk convex hull.
simplices(vertices)	Fungsi yang menerima parameter berupa kumpulan titik (vertices) terluar yang membentuk convex hull dan mengembalikan pasangan dari titik-titik tersebut untuk dihubungkan pada proses visualisasi tes linear separability dataset.
extremePoint(arrayOfPoint)	Fungsi yang menerima parameter berupa array dari titik kumpulan titik yang ada dan mengembalikan index dari titik absis yang minimum (paling kiri/kecil) dan titik absis yang maksimum (paling kanan/kanan).
groupPoints(idxLeft, idxRight, arrayOfPoints)	Fungsi yang menerima parameter index dari titik kiri, index dari titik kanan, dan array of points dan membagi array titik tersebut menjadi 2 bagian berdasarkan garis yang dibentuk oleh titik kiri dan titik kanan. Fungsi ini mengembalikan dua array of index dari titik yang dibagi menjadi 2 bagian.
distanceBetweenPointAndLine(left, right, point)	Fungsi yang menerima parameter 3 titik, yaitu titik kiri, titik kanan, dan sebuah titik sembarang yang kemudian mengembalikan jarak antara garis yang dibentuk oleh titik kiri dan kanan dengan titik sembarang.
findHull(arrayOfIndexes, idxMin, idxMax, arrayOfPoints)	Fungsi yang menerima parameter berupa array of index, index dari titik minimum, index dari titik maksimum, dan array of points yang kemudian akan melakukan pencarian kumpulan titik terluar yang membentuk convex hull.

BAB II

SOURCE CODE PROGRAM

Library myConvexHull dibuat dalam bahasa Python.

myConvexHull.py

```
import numpy as np
import math

class ConvexHull:
    def __init__(self, arrayOfPoints):
        def vertices(arrayOfPoints):
            idxMin, idxMax = extremePoint(arrayOfPoints)
            above = np.array([])
            below = np.array([])
            above, below = groupPoints(idxMin, idxMax, arrayOfPoints)

            aboveHull = findHull(above, idxMin, idxMax, arrayOfPoints)
            belowHull = findHull(below, idxMax, idxMin, arrayOfPoints)

            vertices = np.array([idxMin])
            for i in aboveHull:
                vertices = np.append(vertices, i)

            vertices = np.append(vertices, [idxMax])
            for i in belowHull:
                vertices = np.append(vertices, i)

            return vertices

        def simplices(vertices):
            simplices = np.array([[]])
            for i in range(len(vertices)):

                last = len(vertices) - 1
                if i != last:
                    simplices = np.append(simplices, [vertices[i], vertices[i
+ 1]])
                else:
                    simplices = np.append(simplices, [vertices[i],
vertices[0]])

            simplices = np.reshape(simplices, (-1, 2))
            simplices = simplices.astype(int)

            return simplices
```

```
def extremePoint(arrayOfPoints):
    idxMin = 0
    idxMax = 0

    for i in range(len(arrayOfPoints)):
        if arrayOfPoints[i][0] < arrayOfPoints[idxMin][0]:
            idxMin = i
        elif arrayOfPoints[i][0] == arrayOfPoints[idxMin][0]:
            if arrayOfPoints[i][1] < arrayOfPoints[idxMin][1]:
                idxMin = i

        if arrayOfPoints[i][0] > arrayOfPoints[idxMax][0]:
            idxMax = i
        elif arrayOfPoints[i][0] == arrayOfPoints[idxMax][0]:
            if arrayOfPoints[i][1] > arrayOfPoints[idxMax][1]:
                idxMax = i

    return idxMin, idxMax

def groupPoints(idxLeft, idxRight, arrayOfPoints):
    left = arrayOfPoints[idxLeft]
    right = arrayOfPoints[idxRight]

    left = np.insert(left, 2, 1)
    right = np.insert(right, 2, 1)

    above = np.array([])
    below = np.array([])

    for i in range(len(arrayOfPoints)):
        if i != idxLeft and i != idxRight:
            point = arrayOfPoints[i]
            point = np.insert(point, 2, 1)

            det = np.linalg.det([left, right, point])
            if det > 0:
                above = np.append(above, i)
            else:
                below = np.append(below, i)

    above = above.astype(int)
    below = below.astype(int)

    return above, below

def distanceBetweenPointAndLine(left, right, point):
    x1 = left[0]
```

```
            y1 = left[1]
            x2 = right[0]
            y2 = right[1]
            x0 = point[0]
            y0 = point[1]

            a = abs((x2 - x1) * (y1 - y0) - (x1 - x0) * (y2 - y1))
            b = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)

            return a / b

        def findHull(arrayOfIndexes, idxMin, idxMax, arrayOfPoints):
            if len(arrayOfIndexes) == 0 or len(arrayOfIndexes) == 1:
                return arrayOfIndexes
            else:
                left = arrayOfPoints[idxMin]
                right = arrayOfPoints[idxMax]

                maxDistIdx = int(arrayOfIndexes[0])
                maxDistPoint = arrayOfPoints[maxDistIdx]
                maxDist = distanceBetweenPointAndLine(left, right,
maxDistPoint)

                for i in arrayOfIndexes:
                    index = i
                    point = arrayOfPoints[index]
                    dist = distanceBetweenPointAndLine(left, right, point)

                    if dist > maxDist:
                        maxDistIdx = index
                        maxDist = dist

                above1, below1 = groupPoints(idxMin, maxDistIdx,
arrayOfPoints)
                hull1 = findHull(above1, idxMin, maxDistIdx, arrayOfPoints)
                above2, below2 = groupPoints(maxDistIdx, idxMax,
arrayOfPoints)
                hull2 = findHull(above2, maxDistIdx, idxMax, arrayOfPoints)

                hull = np.concatenate((hull1, [maxDistIdx], hull2))

                return hull

        self.vertices = vertices(arrayOfPoints)
        self.simplices = simplices(self.vertices)
```

main.py

```
#%%
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import ConvexHull

#%%
# visualisasi hasil ConvexHull dari iris dataset
# petal width vs petal height
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()

plt.figure(figsize=(10, 6))
colors = ["b", "r", "g"]
x = 0
y = 1
plt.title(data.feature_names[x] + " vs " + data.feature_names[y])
plt.xlabel(data.feature_names[x])
plt.ylabel(data.feature_names[y])
for i in range(len(data.target_names)):
    bucket = df[df["Target"] == i]
    bucket = bucket.iloc[:, [x, y]].values
    hull = ConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull.simplices:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()

# %%
# visualisasi hasil ConvexHull dari iris dataset
# sepal length vs sepal width
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()

plt.figure(figsize=(10, 6))
colors = ["b", "r", "g"]
x = 2
```

```
y = 3
plt.title(data.feature_names[x] + " vs " + data.feature_names[y])
plt.xlabel(data.feature_names[x])
plt.ylabel(data.feature_names[y])
for i in range(len(data.target_names)):
    bucket = df[df["Target"] == i]
    bucket = bucket.iloc[:, [x, y]].values
    hull = ConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull.simplices:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()

# %%
# visualisasi hasil ConvexHull dari wine dataset
# flavanoids vs nonflavanoid_phenols
data = datasets.load_wine()

# create a DataFrame
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"]
x = 6
y = 7
plt.title(data.feature_names[x] + " vs " + data.feature_names[y])
plt.xlabel(data.feature_names[x])
plt.ylabel(data.feature_names[y])
for i in range(len(data.target_names)):
    bucket = df[df["Target"] == i]
    bucket = bucket.iloc[:, [x, y]].values
    hull = ConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull.simplices:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()

# %%
# visualisasi hasil ConvexHull dari breast cancer dataset
# mean radius vs mean texture
data = datasets.load_breast_cancer()

# create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df["Target"] = pd.DataFrame(data.target)
```

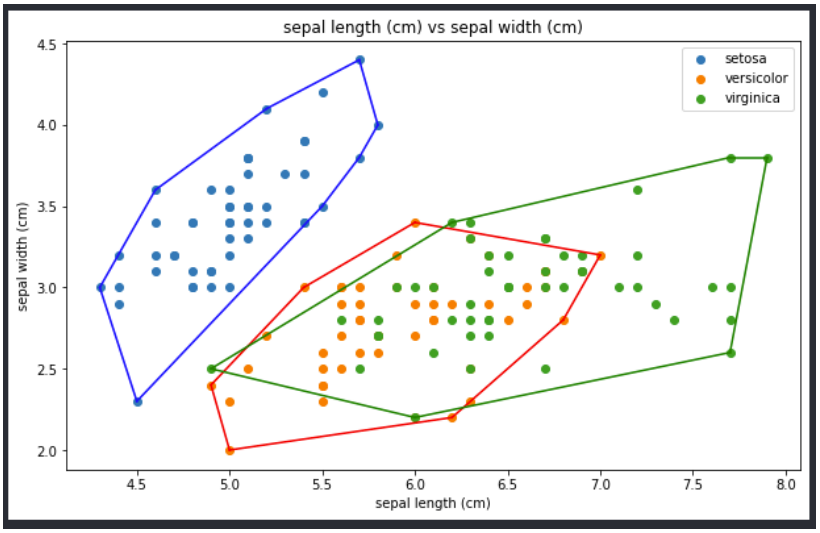
```
print(df.shape)
df.head()


plt.figure(figsize=(10, 6))
colors = ["b", "r", "g"]
x = 0
y = 1
plt.title(data.feature_names[x] + " vs " + data.feature_names[y])
plt.xlabel(data.feature_names[x])
plt.ylabel(data.feature_names[y])
for i in range(len(data.target_names)):
    bucket = df[df["Target"] == i]
    bucket = bucket.iloc[:, [x, y]].values
    hull = ConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull.simplices:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
# %%
```

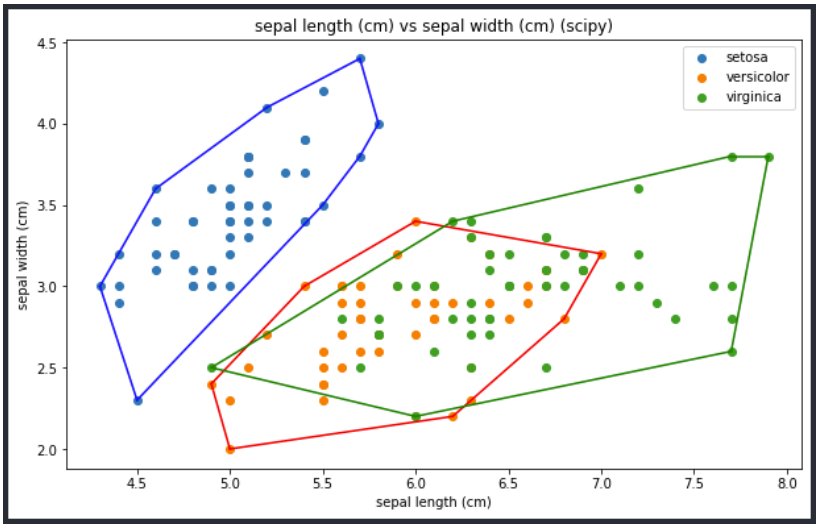
BAB III

EKSEKUSI PROGRAM

1. Visualisasi hasil Convex Hull dari iris dataset (sepal length vs sepal width)

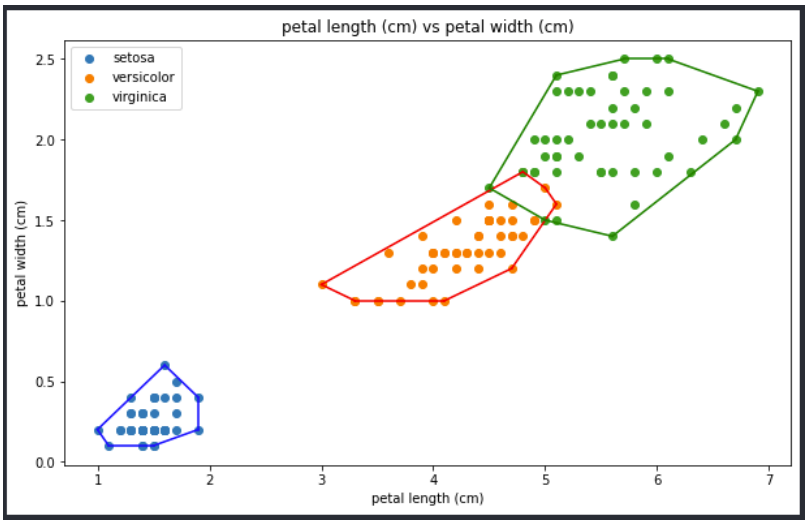


Gambar 1 sepal length v sepal width (myConvexHull)

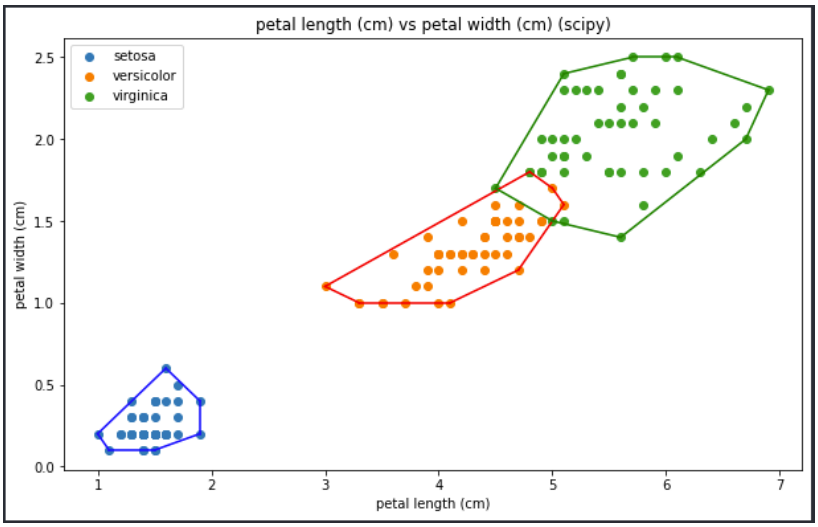


Gambar 2 sepal length v sepal width (scipy)

2. Visualisasi hasil Convex Hull dari iris dataset (petal length vs petal width)

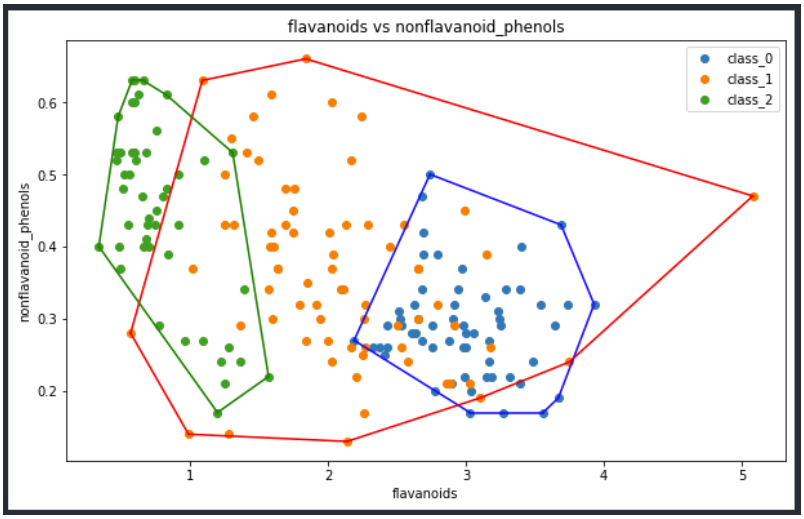


Gambar 3 petal length vs petal width (myConvexHull)

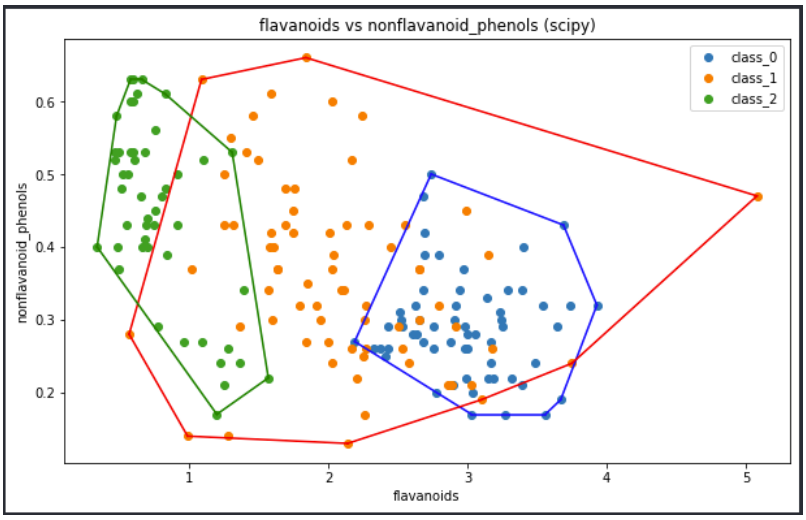


Gambar 4 petal length vs petal width (scipy)

3. Visualisasi hasil Convex Hull dari wine dataset (flavanoids vs nonflavanoid_phenols)

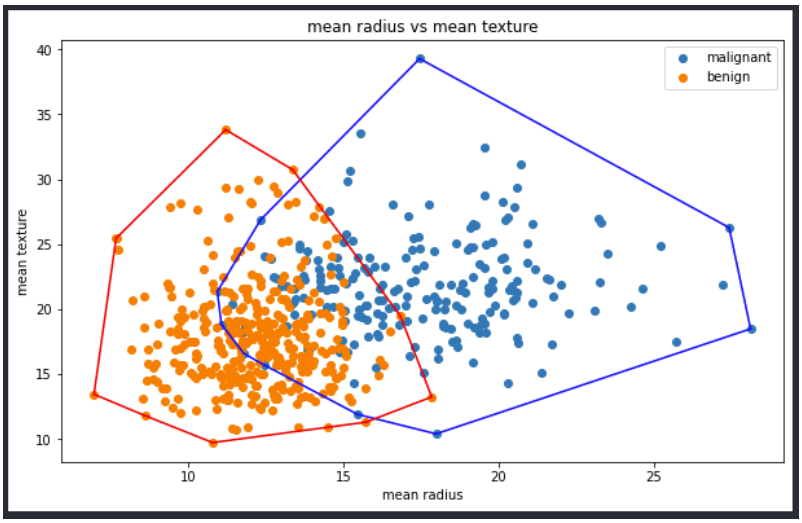


Gambar 5 flavanoids vs nonflavanoid_phenols (myConvexHull)

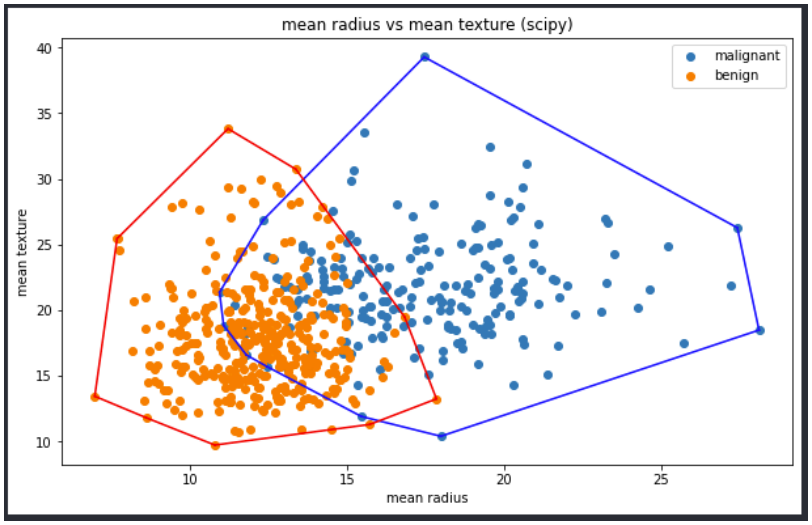


Gambar 6 flavanoids vs nonflavanoid_phenols (scipy)

4. Visualisasi hasil Covex Hull dari breast cancer dataset (mean radius vs mean texture)



Gambar 7 mean radius vs mean texture (myConvexHull)



Gambar 8 mean radius vs mean texture (scipy)

BAB IV

PENUTUP

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.	✓	

Repo Github: https://github.com/vincen-tho/Linear-Separability-Test-using-Convex-Hull