IMPLEMENTASI *CONVEX HULL* UNTUK VISUALISASI TES *LINEAR SEPARABILITY DATASET* DENGAN ALGORITMA *DIVIDE AND CONQUER*

LAPORAN TUGAS KECIL

diajukan untuk memenuhi persyaratan
IF2211 Strategi Algoritma

oleh
Jeremy S.O.N. Simbolon
13520042

TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG

2022

A. Deskripsi Algoritma

Untuk memperoleh himpunan titik pembentuk *convex hull* dari *dataset* masukan pengguna, program akan menggunakan pendekatan *divide-and-conquer*. Pertama, program akan mengurutkan titik *dataset* yang diterima berdasarkan nilai absis secara menaik. Apabila terdapat dua titik dengan nilai absis yang sama, titik akan diurutkan berdasarkan nilai ordinat secara menaik. Setelah pengurutan, dua titik dengan nilai absis terkecil dan terbesar akan dimasukkan ke himpunan solusi karena kedua titik ini selalu menjadi bagian *convex hull*. Selanjutnya, sisa titik pada himpunan akan dibagi menjadi dua subset berdasarkan orientasinya terhadap garis imajiner yang dibentuk kedua titik ekstrem pada himpunan solusi. Tiap subset titik akan kita proses secara rekursif.

Proses rekursif pada subset titik *dataset* dimulai dengan menentukan titik dengan jarak terjauh dari garis imajiner yang telah dibahas sebelumnya. Titik ini pasti menjadi bagian *convex hull*, sehingga akan kita masukkan ke himpunan solusi. Ketiga titik yang telah ditentukan sebelumnya akan membentuk sebuah segitiga imajiner dan membagi subset titik *dataset* menjadi tiga subset baru. Titik-titik yang terletak di dalam segitiga jelas tidak akan menjadi bagian *convex hull*, sehingga dapat kita abaikan untuk tahapan selanjutnya. Proses rekursif pun diulang kembali untuk kedua subset titik baru, dengan kedua sisi imajiner baru segitiga dan pasangan titik pembentuknya menjadi garis imajiner dan titik sudut alas segitiga yang baru. Proses ini akan terus dilakukan hingga tidak ada lagi titik *dataset* yang dapat diproses. Titik-titik pada himpunan solusi setelah proses ini usai adalah pembentuk *convex hull* dari *dataset* masukan program.

B. Kode Sumber

1. myConvexHull.py

```
"""
Provide convex hull vertices for numpy 2d-array visualisation. The library is built
using a divide-and-conquer approach called the QuickHull, first described by
William F. Eddy in 1977.

Reference: http://www.cse.yorku.ca/~aaw/legacy/Hang/quick_hull/Algorithm.html
"""

import numpy as np

hull_points = []
hull_points_copy = []
hull_point_count = 0
center_weight = [0, 0]
center_weight_calculated = False


def convex_hull(points):
    """
    Returns a list of ndarray denoting the dataset points that correspond to
    the convex hull vertices.

    :param points: List of 2d dataset values
    :return: List of ndarray dataset points that corresponds to the convex
    hull vertices to be plotted
    """
```

```
    global hull_points, hull_points_copy, hull_point_count, center_weight,
    center_weight_calculated

    hull_points = []
    center_weight = [0, 0]
    center_weight_calculated = False

    points = points[np.argsort(points[:, 0])]
    leftmost_point = points[0]
    rightmost_point = points[-1]
    np.delete(points, 0)
    np.delete(points, -1)

    hull_points.append(leftmost_point)
    hull_points.append(rightmost_point)

    left_point = []
    right_point = []

    for point in points:
        point_position = point_position_from_line(leftmost_point, rightmost_point,
                                                  point)
        if point_position == "left":
            left_point.append(point)
        elif point_position == "right":
            right_point.append(point)
        else:
            continue

    find_hull(leftmost_point, rightmost_point, left_point)
    find_hull(leftmost_point, rightmost_point, right_point, inverted=True)

    hull_point_count = len(hull_points)
    hull_points_copy = [_ for _ in hull_points]
    hull_points.sort(key=cyclic_order_hull_points)

    return hull_points


def point_position_from_line(left_point, right_point, point_checked):
    """
    Returns the position of a point according to a line defined by two points.

    :param left_point: The leftmost point of the two that defines the line
    :param right_point: The rightmost point of the two that defines the line
    :param point_checked: The point being checked
    :return: The point's position, either left, right, or collinear with the
    line
    """
    triangle_matrix = [[left_point[0], left_point[1], 1],
                       [right_point[0], right_point[1], 1],
                       [point_checked[0], point_checked[1], 1]]

    triangle_area = np.linalg.det(triangle_matrix)
    if triangle_area > 0:
        return "left"
    elif triangle_area < 0:
```

```
        return "right"
    else:
        return "collinear"


def find_hull(left_point, right_point, points, inverted=False):
    """
    Appends the furthest point in `points` from the line defined by `left_point`
    and `right_point` to `hull_points`, then recursively calls this function for
    the point subsets created.

    :param left_point: The leftmost point of the two that defines the line
    :param right_point: The rightmost point of the two that defines the line
    :param points: List of ndarray dataset points being checked
    :param inverted: `True` if checking the lower part of the hull for the
    first time.
    :return: `None` when `points` have been exhausted
    """
    if inverted:
        left_point, right_point = right_point, left_point

    if not points:
        return
    else:
        left_point = np.asarray(left_point)
        right_point = np.asarray(right_point)
        points = np.asarray(points)

        furthest_point = points[0]
        max_distance = (np.abs(np.cross(right_point - left_point, furthest_point -
                                        left_point)) / (np.linalg.norm(right_point -
                                                                       left_point)))

        for i in range(1, len(points)):
            point_distance = (np.abs(np.cross(right_point - left_point, points[i] -
                                              left_point)) / np.linalg.norm
                                              (right_point - left_point))
            if point_distance > max_distance:
                furthest_point = points[i]
                max_distance = point_distance

        hull_points.append(furthest_point)

        left_triangle_points = []
        right_triangle_points = []

        for point in points:
            if (point_position_from_line(left_point, furthest_point, point) == "left"
                    and point_position_from_line(furthest_point, right_point, point)
                        == "right"):
                left_triangle_points.append(point)
            elif (point_position_from_line(left_point, furthest_point, point) ==
                  "right" and point_position_from_line(furthest_point, right_point,
                  point) == "left"):
                right_triangle_points.append(point)
            else:
                continue
```

```
        find_hull(left_point, furthest_point, left_triangle_points)
        find_hull(furthest_point, right_point, right_triangle_points)


def cyclic_order_hull_points(point):
    """
    Returns the angle of a given point relative to the x-axis and the hull's
    center weight

    :param point: The point whose angle is being calculated
    :return: Angle of the given point relative to the x-axis in radian
    """
    global center_weight_calculated

    if not center_weight_calculated:
        for _point in hull_points_copy:
            center_weight[0] += _point[0]
            center_weight[1] += _point[1]
        center_weight[0] /= hull_point_count
        center_weight[1] /= hull_point_count
        center_weight_calculated = True

    angle = np.arctan2([point[1] - center_weight[1]], [point[0] - center_weight[0]])

    return angle[0]
```

2. convex_hull.ipynb

```
import matplotlib.pyplot as plt
import myConvexHull as cH
import pandas as pd
from scipy.spatial import ConvexHull
from sklearn import datasets
```

```
first_data = datasets.load_iris()
df = pd.DataFrame(first_data.data, columns=first_data.feature_names)
df["Target"] = pd.DataFrame(first_data.target)

plt.figure(figsize=(10, 6))
plt.title("Sepal Length vs Sepal Width")
plt.xlabel(first_data.feature_names[0])
plt.ylabel(first_data.feature_names[1])
for i in range(len(first_data.target_names)):
    bucket = df[df["Target"] == i]
    bucket = bucket.iloc[:, [0, 1]].values
    plt.scatter(bucket[:, 0], bucket[:, 1], label=first_data.target_names[i])
    hull = cH.convex_hull(bucket)

    x_coordinates = []
    y_coordinates = []
    for j in range(len(hull)):
        x_coordinates.append(hull[j][0])
        y_coordinates.append(hull[j][1])
    x_coordinates.append(hull[0][0])
    y_coordinates.append(hull[0][1])
```

```
    plt.plot(x_coordinates, y_coordinates)

plt.legend()
```

```
plt.figure(figsize=(10, 6))
plt.title("Sepal Length vs Sepal Width")
plt.xlabel(first_data.feature_names[0])
plt.ylabel(first_data.feature_names[1])
for i in range(len(first_data.target_names)):
    bucket = df[df["Target"] == i]
    bucket = bucket.iloc[:, [0, 1]].values
    plt.scatter(bucket[:, 0], bucket[:, 1], label=first_data.target_names[i])
    hull = ConvexHull(bucket)

    for simplex in hull.simplices:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1])

plt.legend()
```

```
plt.figure(figsize=(10, 6))
plt.title("Petal Length vs Petal Width")
plt.xlabel(first_data.feature_names[2])
plt.ylabel(first_data.feature_names[3])
for i in range(len(first_data.target_names)):
    bucket = df[df["Target"] == i]
    bucket = bucket.iloc[:, [2, 3]].values
    plt.scatter(bucket[:, 0], bucket[:, 1], label=first_data.target_names[i])
    hull = cH.convex_hull(bucket)

    x_coordinates = []
    y_coordinates = []
    for j in range(len(hull)):
        x_coordinates.append(hull[j][0])
        y_coordinates.append(hull[j][1])
    x_coordinates.append(hull[0][0])
    y_coordinates.append(hull[0][1])
    plt.plot(x_coordinates, y_coordinates)

plt.legend()
```

```
plt.figure(figsize=(10, 6))
plt.title("Petal Length vs Petal Width")
plt.xlabel(first_data.feature_names[2])
plt.ylabel(first_data.feature_names[3])
for i in range(len(first_data.target_names)):
    bucket = df[df["Target"] == i]
    bucket = bucket.iloc[:, [2, 3]].values
    plt.scatter(bucket[:, 0], bucket[:, 1], label=first_data.target_names[i])
    hull = ConvexHull(bucket)

    for simplex in hull.simplices:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1])

plt.legend()
```

```
second_data = datasets.load_breast_cancer()
df = pd.DataFrame(second_data.data, columns=second_data.feature_names)
df["Target"] = pd.DataFrame(second_data.target)

plt.figure(figsize=(10, 6))
plt.title("Diagnostic Radius vs Diagnostic Texture")
plt.xlabel(second_data.feature_names[0])
plt.ylabel(second_data.feature_names[1])
for i in range(len(second_data.target_names)):
    bucket = df[df["Target"] == i]
    bucket = bucket.iloc[:, [0, 1]].values
    plt.scatter(bucket[:, 0], bucket[:, 1], label=second_data.target_names[i])
    hull = cH.convex_hull(bucket)

    x_coordinates = []
    y_coordinates = []
    for j in range(len(hull)):
        x_coordinates.append(hull[j][0])
        y_coordinates.append(hull[j][1])
    x_coordinates.append(hull[0][0])
    y_coordinates.append(hull[0][1])

    plt.plot(x_coordinates, y_coordinates)

plt.legend()
```

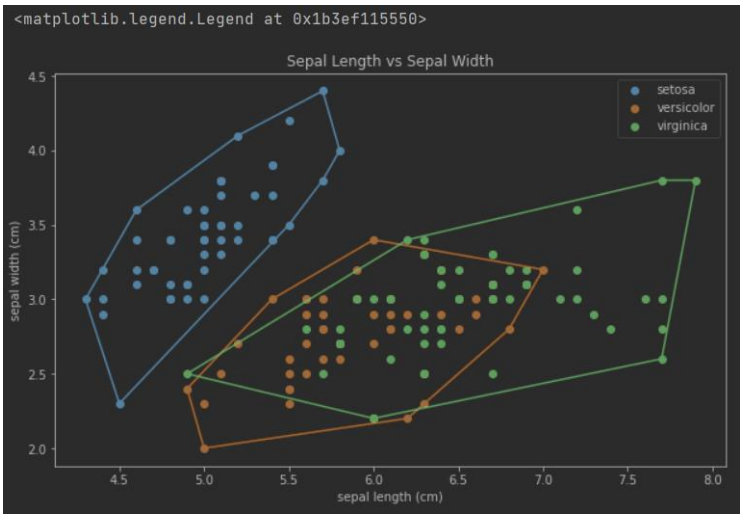
```
plt.figure(figsize=(10, 6))
plt.title("Diagnostic Radius vs Diagnostic Texture")
plt.xlabel(second_data.feature_names[0])
plt.ylabel(second_data.feature_names[1])
for i in range(len(second_data.target_names)):
    bucket = df[df["Target"] == i]
    bucket = bucket.iloc[:, [0, 1]].values
    plt.scatter(bucket[:, 0], bucket[:, 1], label=second_data.target_names[i])
    hull = ConvexHull(bucket)

    for simplex in hull.simplices:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1])

plt.legend()
```

```
third_data = datasets.load_wine()
df = pd.DataFrame(third_data.data, columns=third_data.feature_names)
df["Target"] = pd.DataFrame(third_data.target)

plt.figure(figsize=(10, 6))
plt.title("Alcohol vs Malic Acid")
plt.xlabel(third_data.feature_names[0])
plt.ylabel(third_data.feature_names[1])
for i in range(len(third_data.target_names)):
    bucket = df[df["Target"] == i]
    bucket = bucket.iloc[:, [0, 1]].values
    plt.scatter(bucket[:, 0], bucket[:, 1], label=third_data.target_names[i])
    hull = cH.convex_hull(bucket)

    x_coordinates = []
    y_coordinates = []
    for j in range(len(hull)):
        x_coordinates.append(hull[j][0])
        y_coordinates.append(hull[j][1])
    x_coordinates.append(hull[0][0])
    y_coordinates.append(hull[0][1])

    plt.plot(x_coordinates, y_coordinates)

plt.legend()
```

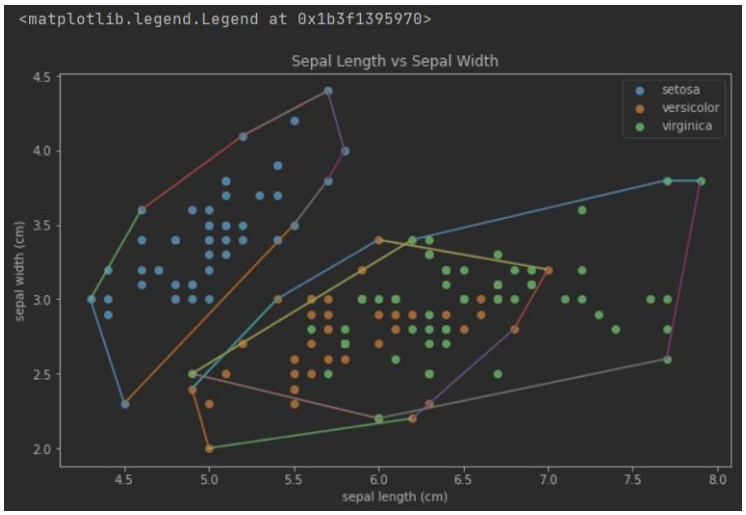
```
plt.figure(figsize=(10, 6))
plt.title("Alcohol vs Malic Acid")
plt.xlabel(third_data.feature_names[0])
plt.ylabel(third_data.feature_names[1])
for i in range(len(third_data.target_names)):
    bucket = df[df["Target"] == i]
    bucket = bucket.iloc[:, [0, 1]].values
    plt.scatter(bucket[:, 0], bucket[:, 1], label=third_data.target_names[i])
    hull = ConvexHull(bucket)

    for simplex in hull.simplices:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1])

plt.legend()
```

C. Uji Coba Program

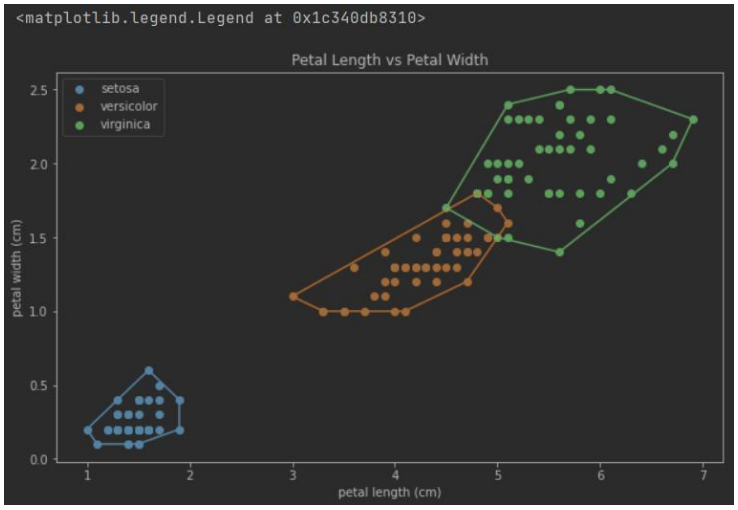
1. *Dataset* Iris Plants



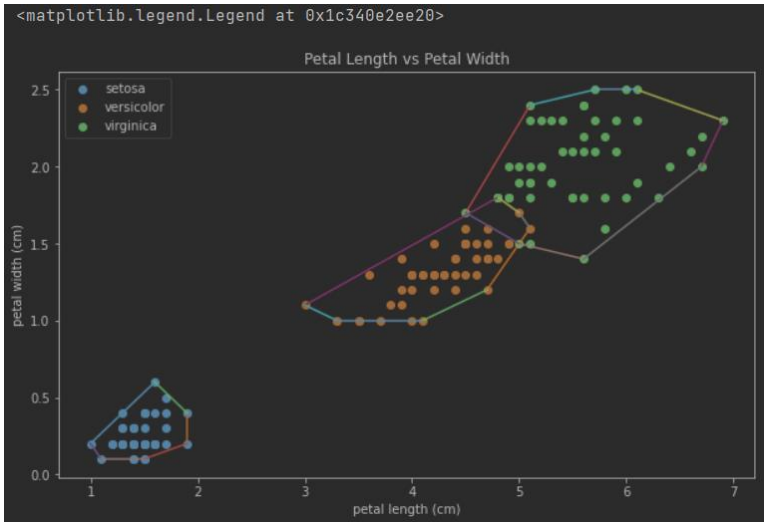
Gambar 1 Dataset Iris Plants dengan myConvexHull (sepal)



Gambar 2 Dataset Iris Plants dengan Scipy's ConvexHull (sepal)

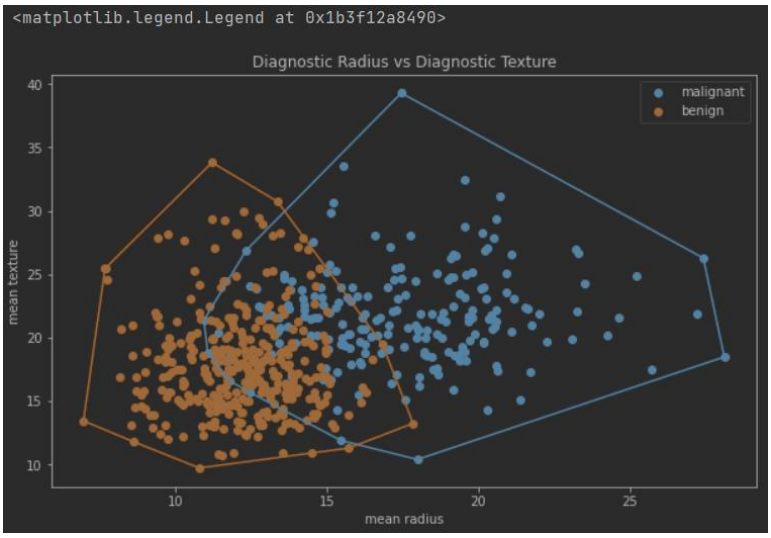


Gambar 3 Dataset Iris Plants dengan myConvexHull (petal)

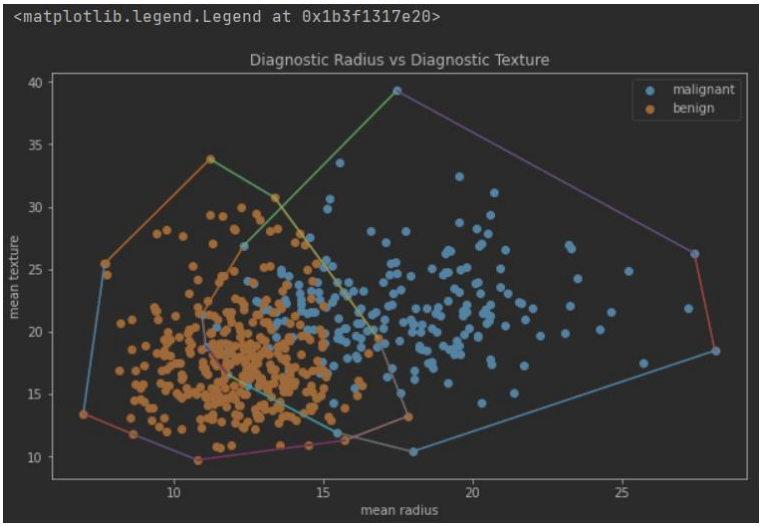


Gambar 4 Dataset Iris Plants dengan Scipy's ConvexHull (petal)

2. *Dataset* Breast Cancer Diagnostic

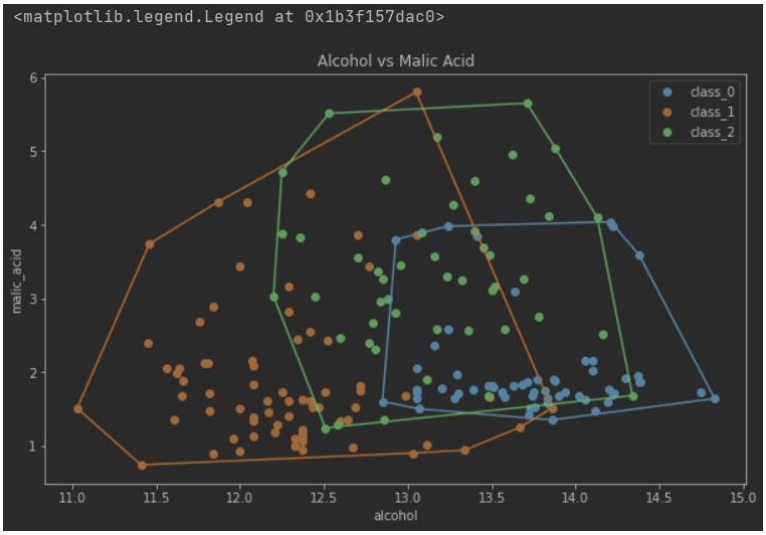


Gambar 5 Dataset Breast Cancer Diagnostic dengan myConvexHull

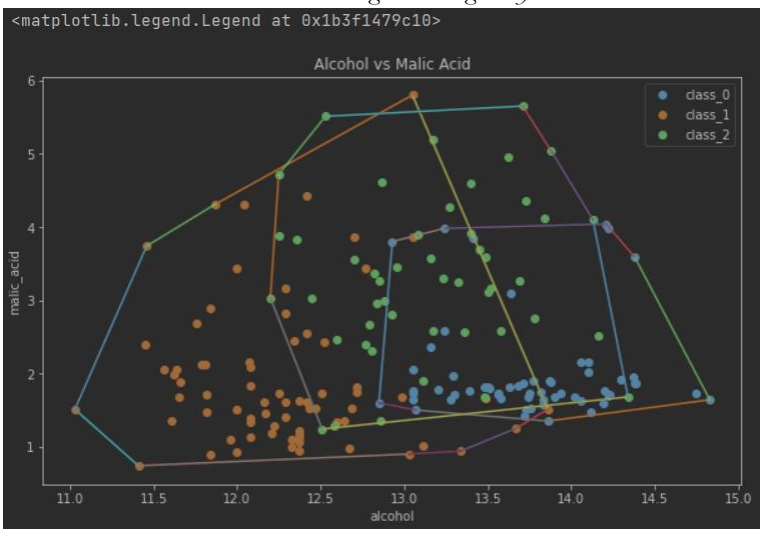


Gambar 6 Dataset Breast Cancer Diagnostic dengan Scipy's ConvexHull

3. *Dataset* Wine Recognition



Gambar 7 Dataset Wine Recognition dengan myConvexHull



Gambar 8 Dataset Wine Recognition dengan Scipy's ConvexHull

4. Repositori Program

Kode sumber dari program ini dapat diakses pada repositori GitHub yang terdapat pada laman https://github.com/tastytypist/convex-hull-visualiser.

5. Lampiran

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