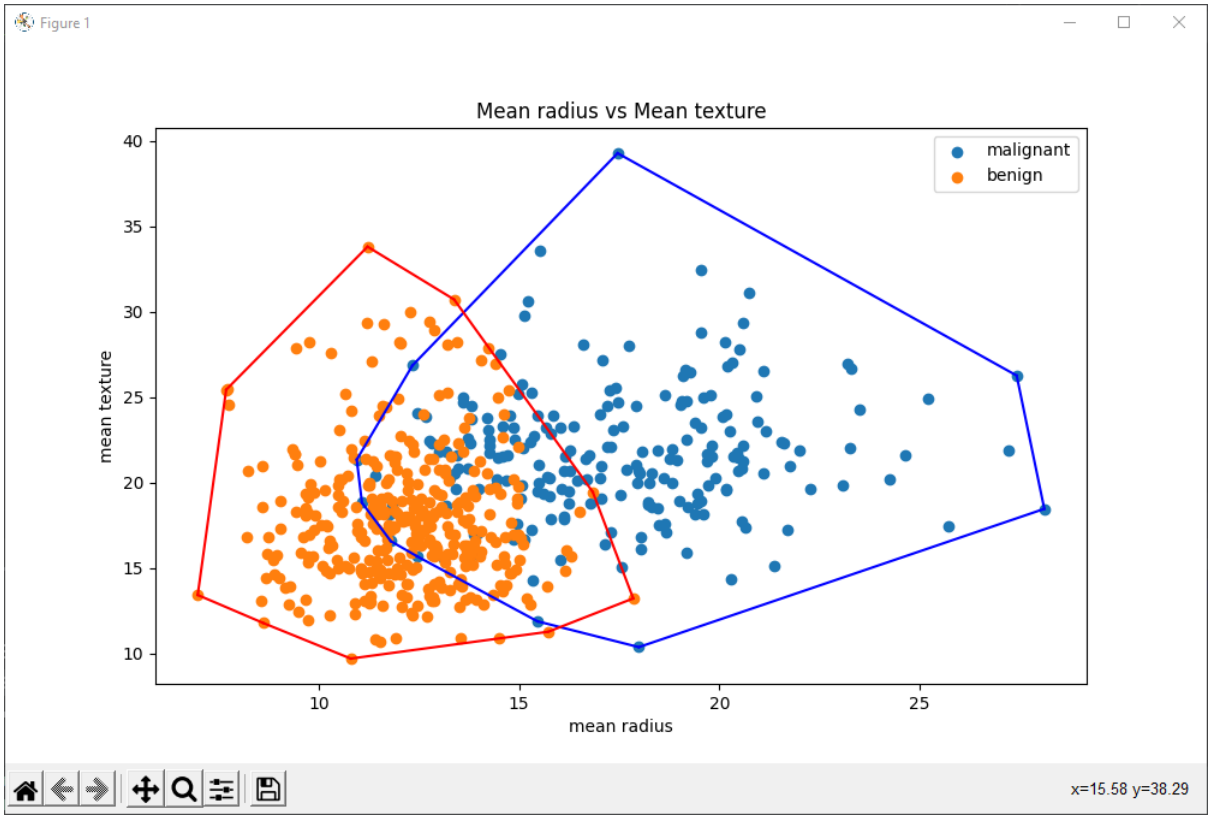
LAPORAN TUGAS KECIL IF2211 STRATEGI ALGORITMA SEMESTER II TAHUN 2021/2022

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



Disusun oleh:

13520156 Dimas Faidh Muzaki

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

Algoritma *Divide and Conquer*

Convex Hull yang diimplementasikan di pustaka saya merupakan sebuah objek dari kelas bernama CHull yang memiliki dua data member yaitu *points* dan *simplices*. Dalam pembentukan objek kelas (konstruktor), kelas menerima sebuah argumen berupa matriks berukuran n x 2. Matriks tersebut merupakan dataset yang ingin kita uji *linear separability* nya. Setiap elemen pada matriks memiliki 2 elemen yang dapat direpresentasikan sebagai titik 2 dimensi. Data member *points* merupakan Salinan dari matriks masukan yang setiap elemennya dinomori sebagai identitas. Sementara itu, *simplices* merupakan sebuah list yang isinya adalah pasangan-pasangan nomor elemen points yang membentuk *Convex Hull.* Dalam mencari *Convex Hull*, pustaka saya memiliki algoritma *Divide and Conquer* sebagai berikut:

1. Data yang dimasukkan sebagai argument konstruktor elemennya masing-masing akan diberikan nomor identitas. Elemen data selanjutnya akan disebut sebagai titik
2. Titik yang sudah dinomori akan diurutkan. Titik memiliki atribut absis dan ordinat. Pengurutan dilakukan dengan abis menaik titik terlebih dahulu, untuk nilai absis yang sama akan diurutkan berdasarkan ordinat.
3. Dari titik-titik terurut tersebut akan diambil 2 titik ekstrem, indeks pertama dan terakhir, untuk dijadikan dua titik awal proses *divide and conquer*.
4. Selanjutnya adalah membagi *points* menjadi 2 himpunan. Himpunan pertama, s1, adalah titik-titik yang terletak disebelah kiri garis yang menghubungkan dua titik ekstrem. Sementara himpunan kedua, s2, adalah titik-titik yang terletak di sebelah kanan garis yang menghubungkan dua titik ekstrem tadi. Hal ini dilakukan dengan dibantu fungsi determinan pada file utils.
5. S1 dan s2 masing masing akan dimasukkan ke fungsi rekursif untuk mencari *simplices.* Mereka akan dimasukkan ke fungsi bersama dua titik ekstrem sebagai p1 dan p2
6. Di dalam fungsi ada dua kemungkinan kondisi:
    a. Apabila himpunan titik yang masuk kosong. Maka p1 dan p2 adalah pembetuk *Convex Hull* dan nomor-nomor nya dimasukkan ke dalam *simplices* sebagai sebuah pasangan
    b. Apabila himpunan tidak kosong, akan dipilih sebuah titik pn yang memiliki jarak terjauh dari garis p1p2. Jika terdapat beberapa titik dengan jarak yang sama, akan dipilih titik yang memaksimal sudut p1pnp2. Hal ini dapat dicapai dengan menggunakan fungsi farthest_from_line dan point_distance_to_line
7. Selanjutnya akan dicari kumpulan titik yang berada di sebelah kiri garis p1pn dan di sebelah kanan garis pnp2
8. Untuk kedua kumpulan titik tadi akan dilakukan proses nomor 6 dan 7 sampai didapatkan kedua kumpulan titik kosong.
9. Tak lupa untuk s2 dilakukan proses nomor 6,7,8
10. Pada akhir proses rekursi akan didapatkan simplices yang berisi pasangan-pasangan titik pembentuk *Convex Hull.*

Source Pusataka dalam Bahasa Python

File Chull.py

```
import copy

from myConvexHull import utils


def ConvexHull(arr):
    ch = CHull(arr)
    return ch


class CHull:
    def __init__(self, points):
        self.simplices = []
        self.points = [[0, 0, 0] for _ in range(len(points))]
        for i in range(len(points)):
            self.points[i] = [points[i][0], points[i][1], i]
        self.__search_simplices()

    def __search_simplices(self):
        # Himpunan titik-titik diurutkan berdasarkan absis lalu ordinat
        sorted_points = utils.sort_coordinates(self.points)

        # Dua titik terekstrem merupakan bagian dari ConvexHull
        p_start = sorted_points.pop(0)
        p_end = sorted_points.pop()
        # self.simplices.append([p_start[2], p_end[2]])

        # Membagi ke 2 himpunan, ini merupakan 2 himpunan kanan dan kiri
pertama
        s1 = utils.left_set(copy.deepcopy(sorted_points), p_start, p_end)
        s2 = utils.right_set(copy.deepcopy(sorted_points), p_start, p_end)

        # Masuk ke dalam fungsi yang rekurens untuk mencari simplex-simplex
ConvexHull
        # Menggunakan algoritma divide and conquer
        self.__dnc_chull_left(s1, p_start, p_end)
        self.__dnc_chull_right(s2, p_start, p_end)

    def __dnc_chull_left(self, points, p_start, p_end):
        if len(points) == 0:
            self.simplices.append([p_start[2], p_end[2]])
        else:
            p_farthest = utils.farthest_point(points, p_start, p_end)
            s_left = utils.left_set(points, p_start, p_farthest)
            s_right = utils.left_set(points, p_farthest, p_end)
            self.__dnc_chull_left(s_left, p_start, p_farthest)
            self.__dnc_chull_left(s_right, p_farthest, p_end)

    def __dnc_chull_right(self, points, p_start, p_end):
        if len(points) == 0:
            self.simplices.append([p_start[2], p_end[2]])
        else:
            p_farthest = utils.farthest_point(points, p_start, p_end)
            s_left = utils.right_set(points, p_start, p_farthest)
            s_right = utils.right_set(points, p_farthest, p_end)
            self.__dnc_chull_right(s_left, p_start, p_farthest)
            self.__dnc_chull_right(s_right, p_farthest, p_end)
```

File utils.py

```
import copy
```

```
import numpy as np


def sort_coordinates(points):
    return sorted(points, key=lambda k: [k[0], k[1]])


def determinant(p_start, p_end, p):
    p_s = copy.deepcopy(p_start)
    p_e = copy.deepcopy(p_end)
    pp = copy.deepcopy(p)

    p_s[2] = 1
    p_e[2] = 1
    pp[2] = 1

    p_s = np.array(p_s)
    p_e = np.array(p_e)
    pp = np.array(pp)

    mat = [p_s, p_e, pp]
    return np.linalg.det(mat)


def point_distance_to_line(p_start, p_end, p):
    p_start, p_end, p = to_np_arr(p_start, p_end, p)
    return np.abs(np.cross(p_end - p_start, p_start - p)) /
np.linalg.norm(p_end - p_start)


def farthest_point(points, p_start, p_end):
    p = None
    for point in points:
        if p is None:
            p = point
        d_p = point_distance_to_line(p_start, p_end, p)
        d_point = point_distance_to_line(p_start, p_end, point)
        if d_point > d_p:
            p = point
        elif d_point == d_p and angle(p_start, point, p_end) >
angle(p_start, p, p_end):
            p = point
    return p


def angle(p1, p2, p3):
    a, b, c = to_np_arr(p1, p2, p3)

    ba = a - b
    bc = c - b

    cosine_angle = np.dot(ba, bc) / (np.linalg.norm(ba) *
np.linalg.norm(bc))
    return np.arccos(cosine_angle)


def to_np_arr(p1, p2=None, p3=None):
    p1 = np.array([p1[0], p1[1]])
    if p2 is None:
        return p1
    p2 = np.array([p2[0], p2[1]])
```

```
    if p3 is None:
        return p1, p2
    p3 = np.array([p3[0], p3[1]])
    return p1, p2, p3


def left_set(points, p_start, p_end):
    s = []
    for p in points:
        if determinant(p_start, p_end, p) > 0 and p[2] != p_start[2] and
p[2] != p_end[2]:
            s.append(p)
    return s


def right_set(points, p_start, p_end):
    s = []
    for p in points:
        if determinant(p_start, p_end, p) < 0 and p[2] != p_start[2] and
p[2] != p_end[2]:
            s.append(p)
    return s
```

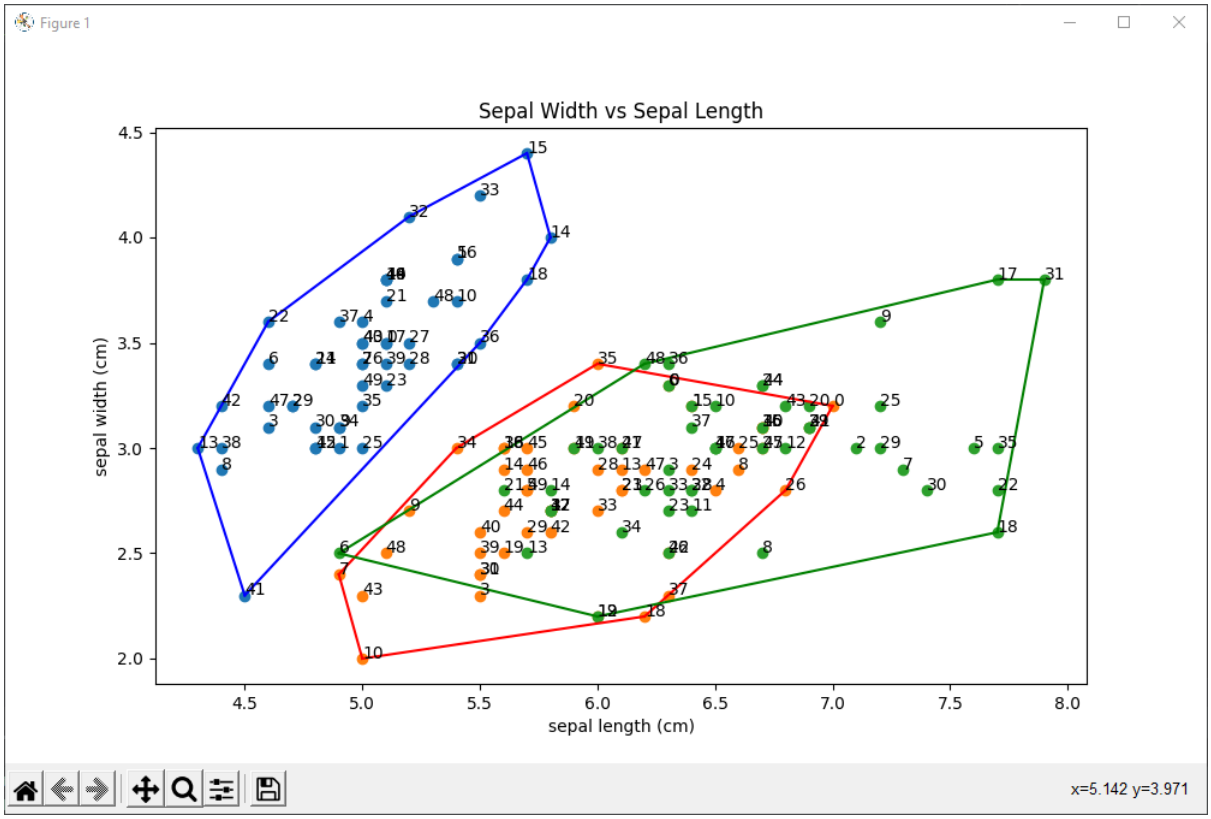
Screenshot Penggunaan Pustaka

Terdapat 3 kali uji pustaka menggunakan 2 dataset yaitu iris dan breast_cancer dari sklearn. Ketiga pengujian menggunakan header yang sama. Di dalam header terdapat proses *import* pustaka yang sudah dibuat sebelumnya. Header dapat dilihat pada gambar berikut:

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

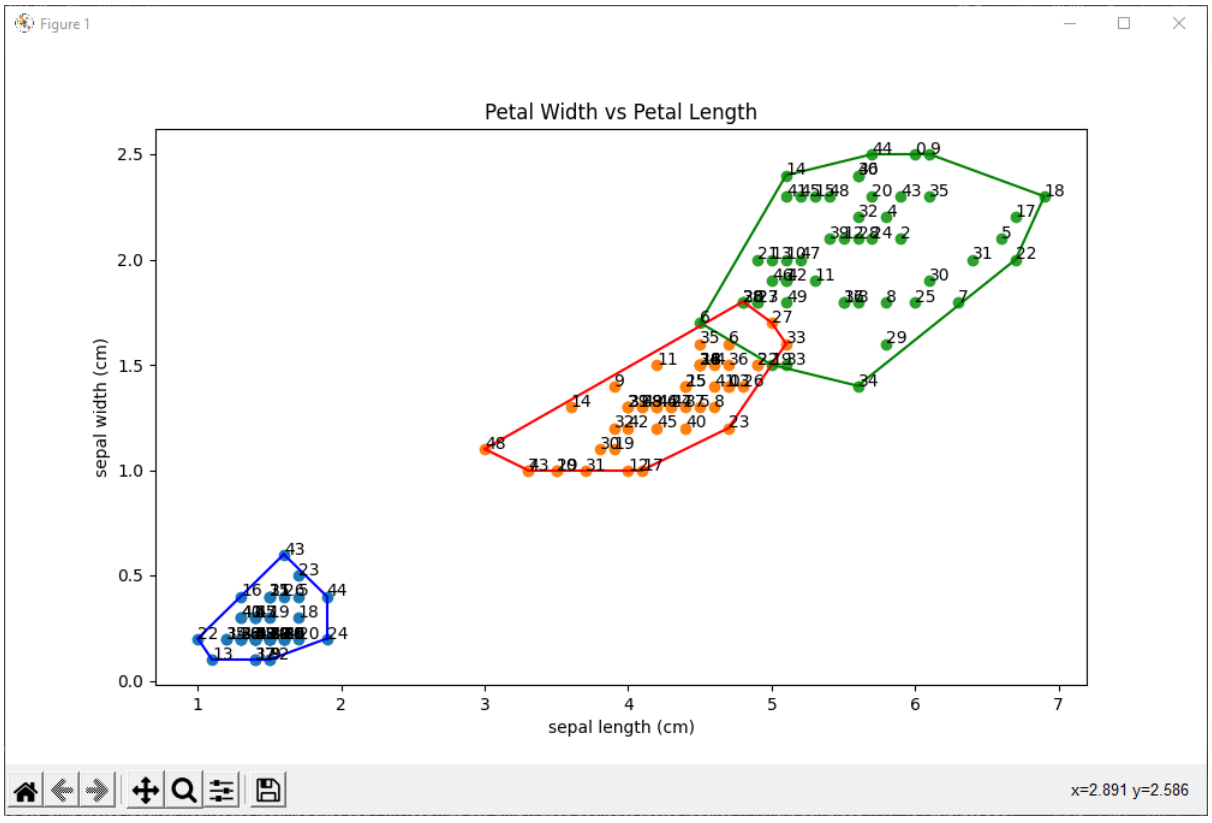
1. Iris (Sepal length-Sepal width)

```
data = datasets.load_iris()
# create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
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
    hull = ConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for j in range(len(bucket)):
        plt.annotate(j, (bucket[j][0], bucket[j][1]))
    for simplex in hull.simplices:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.show()
```



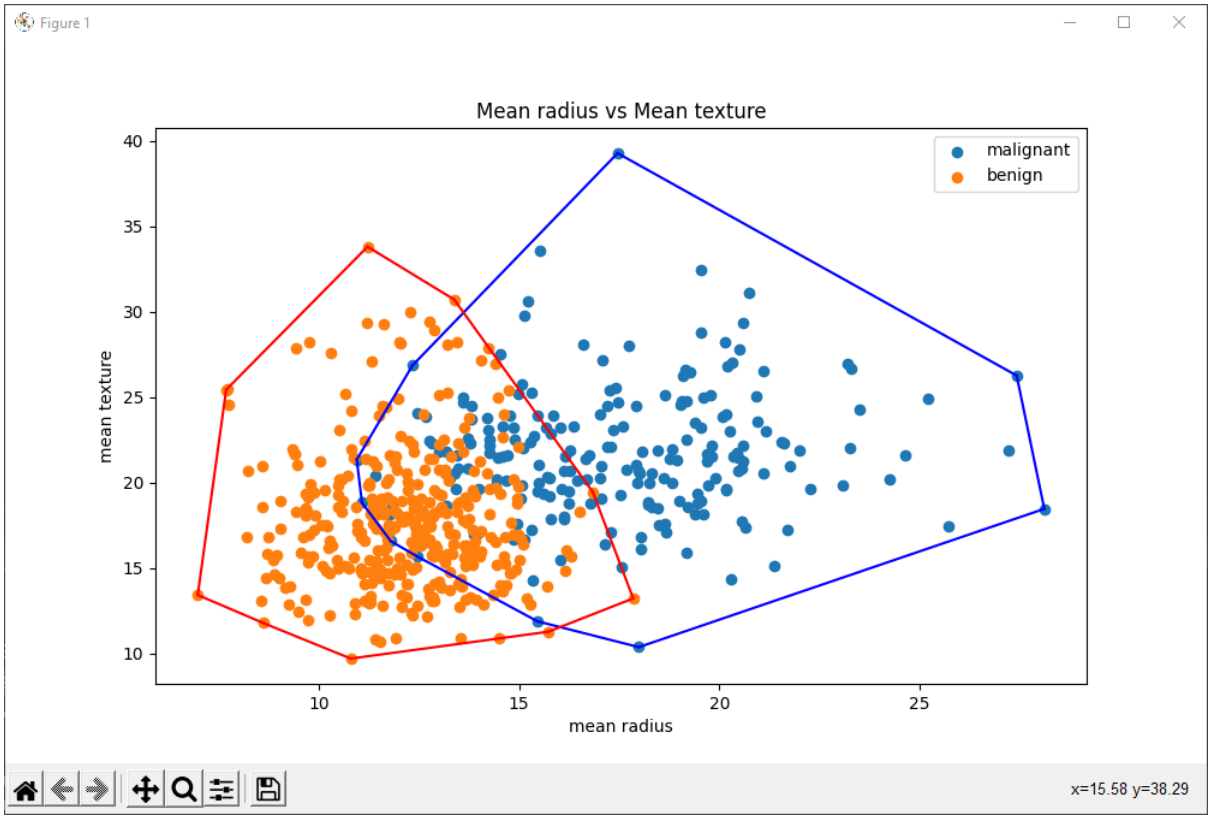
2. Iris (Petal length – Petal width)

```
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[:, [2, 3]].values
    hull = ConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for j in range(len(bucket)):
        plt.annotate(j, (bucket[j][0], bucket[j][1]))
    for simplex in hull.simplices:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.show()
```



3. Breast Cancer (Mean radius – Mean texture)

```
data = datasets.load_breast_cancer()
# create a DataFrame
pd.set_option('display.max_columns', None)
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
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
    hull = ConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for j in range(len(bucket)):
        plt.annotate(j, (bucket[j][0], bucket[j][1]))
    for simplex in hull.simplices:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
plt.show()
```



Check Point

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	✓	

Link Repositori

https://github.com/maspaitujaki/ConvexHullPython