LAPORAN TUGAS KECIL 2

IF2211 Strategi Algoritma

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

Disusun oleh:

Nama : Gede Sumerta Yoga

NIM : 13520021

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

Algoritma *Divide and Conquer*

Divide and Conquer merupakan salah satu algoritma fundamental dalam ilmu komputer sekarang. Sesuai namanya, algoritma ini terdiri dari bagian *Divide, Conquer*, serta *Combine*. Tahap *Divide* berarti membagi persoalan menjadi bagian-bagian atau upa-persoalan yang memiliki karakteristik yang mirip dengan persoalaan utamanya. Kemudian, bagian *Conquer* berarti menyelesaikan permasalahan dari upa-persoalan. Yang terakhir adalah tahap *Combine* yaitu menyatikan solusi dari setiap upa-persoalan dan menjadi solusi dari persoalan utama yang ingin diselesaikan. Sebenarnya, algoritma ini sangat berkaitan dengan skema rekursif.

Dalam tugas kecil kali ini, saya diberi persoalan untuk membuat sebuah pustaka myConvexHull yang dapat mengembalikan Convex Hull dari data yang dimasukkan. Yang diperlukan adalah data yang berukuran dua dimensi. Kemudian, saya akan menjelaskan fungsi – fungsi yang digunakan untuk tugas kecil ini, yaitu:

- findIndex(bucket, array) : Fungsi ini akan menerima masukan sebuah data bucket dan array yang merupakan elemen bucket. Keluaran yang dihasilkan adalah sebuah integer yang menandakan indeks dari array di data bucket.
- farthestPoint(point1, pointn, setOfPoint) : Fungsi ini menerima masukan dua titik dan sebuat himpunan titik. Keluaran fungsi ini adalah sebuah titik di himpunan titik yang memiliki jarak terjauh dari garis yang dibentuk oleh titik point1 dan pointn
- outerPoint(p1, p2, s) : Fungsi ini menerima masukan dua buah titik dan sebuah himpunan titik. Keluaran dari fungsi ini adalah himpunan titik yang berada dibagian luar garis p1p2.
- DandC(bucket, p1, p2, s) : Fungsi ini menerima masukan data bucket, dua buah titik, dan sebuah himpunan titik. Fungsi ini merupakan implementasi dari bagian divide dan conquer dari algoritma Divide and Conquer yang digunakan.
- convexHull(bucket) : Fungsi ini merupakan fungsi utama dari modul ini. Alur kerja fungsi ini adalah sebagai berikut. Pertama fungsi akan menerima masukan data bucket dan akan mengurutkannya menaik sesuai nilai pertama dari array sebagai elemennya. Kemudian diambil titik pertama dan terakhir dari data terurut tersebut dan membuat garis khayal dari dua titik itu untuk membagi dua data tersebut. Ini merupakan proses Divide. Selanjutnya diperoleh dua himpunan data dan akan menjadi upa-persoalan dari persoalan utama yang akan diselesaikan dengan fungsi DandC. Ini merupakan proses

conquer. Terakhir, solusi yang telah diperoleh dari setiap upa-persoalan akan digabungkan menjadi sebuah himpunan hull yang akan menjadi keluaran dari fungsi ini. Ini adalah proses combine.

Kode Program

Kode Program yang berisi fungsi utama myConvexHull

```
from cmath import sqrt
import numpy as np


def findIndex(bucket, array):
    #Fungsi ini akan menerima dataset awal dan mencari indeks dari elemennya
    idx = 0
    for i in bucket:
        if(i[0] == array[0] and i[1] == array[1]):
            return idx
        idx += 1

def farthestPoint(point1, pointn, setOfPoint):
    # Mencari titik terjauh dari kumpulan titik ke garis
    # Persamaan ax+by+c=0
    m = (point1[1] - pointn[1])/(point1[0] - pointn[0])
    a = m
    b = -1
    c = point1[1] - m*point1[0]
    maks = 0
    pmaks = []

    #Mengiterasi jarak untuk setiap titik
    for i in setOfPoint:
        temp = abs((a*i[0] + b*i[1] + c)/sqrt(a*a + b*b))
        if temp >= maks:
            maks = temp
            pmaks = i
    return pmaks


def convexHull(bucket):
    sorted_bucket = sorted(bucket, key=lambda x: [x[0], x[1]])
    sorted_bucket = np.asarray(sorted_bucket)
    arrayLength = len(sorted_bucket)
    p1 = sorted_bucket[0]
    pn = sorted_bucket[arrayLength-1]

    # Membuat dua area baru
    # s1 untuk det positif atau dikiri
    # s2 untuk det negatif atau dikanan
    s1 = np.array([p1])
    s1 = np.append(s1, [pn], axis=0)
```

```
    s2 = s1
    for i in range(1, arrayLength-1):
        det = sorted_bucket[0, 0] * sorted_bucket[arrayLength-1, 1] +
sorted_bucket[i, 0] * sorted_bucket[0, 1] + sorted_bucket[arrayLength-1, 0] *
sorted_bucket[i,


      1] - sorted_bucket[i, 0] * sorted_bucket[arrayLength-1, 1] -
sorted_bucket[arrayLength-1, 0] * sorted_bucket[0, 1] - sorted_bucket[0, 0] *
sorted_bucket[i, 1]
        if det > 0:
            s1 = np.append(s1, [sorted_bucket[i]], axis=0)
        elif det < 0:
            s2 = np.append(s2, [sorted_bucket[i]], axis=0)
    s2 = np.unique(s2, axis=0)
    s1 = np.unique(s1, axis=0)

    #Divide menjadi upa-persoalan
    hull1 = DandC(bucket, p1, pn, s1)
    hull2 = DandC(bucket, pn, p1, s2)

    #Combine solusi dari upa-persoalan
    hull = np.append(hull1, hull2, axis=0)

    return hull


def outerPoint(p1, p2, s):
    #Mencari titik yang berada diluar segitiga atau garis p1p2
    part = np.array([p1, p2])
    for i in range(len(s)):
        a = p1[0] * p2[1] + s[i, 0] * p1[1] + p2[0] * s[i, 1]
        b = s[i, 0] * p2[1] + p2[0] * p1[1] + p1[0] * s[i, 1]
        det = a-b
        if det > 0:
            part = np.append(part, [s[i]], axis=0)
    part = np.unique(part, axis=0)
    return part


def DandC(bucket, p1, p2, s):
    #Proses Divide dan Conquer yang mirip rekursif
    #Akan mengembalikan hull atau solusi dari upa-persoalan
    if(len(s) == 2):
        pos1 = findIndex(bucket, p1)
        pos2 = findIndex(bucket, p2)
        hull = np.array([[pos1, pos2]])
    else:
        pmaks = farthestPoint(p1, p2, s)
```

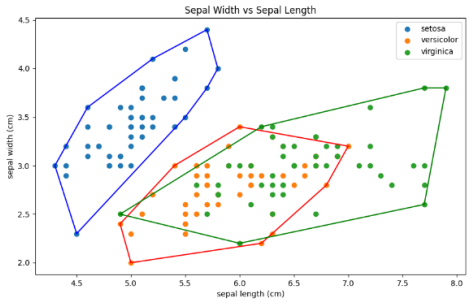
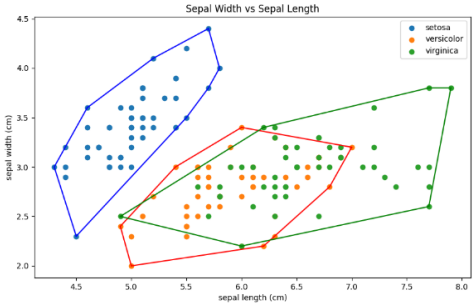
```
        s1 = outerPoint(p1, pmaks, s)
        s2 = outerPoint(pmaks, p2, s)
        hull1 = DandC(bucket, p1, pmaks, s1)
        hull2 = DandC(bucket, pmaks, p2, s2)
        hull = np.append(hull1, hull2, axis=0)
    return hull
```

Kode yang berisi contoh penggunaan

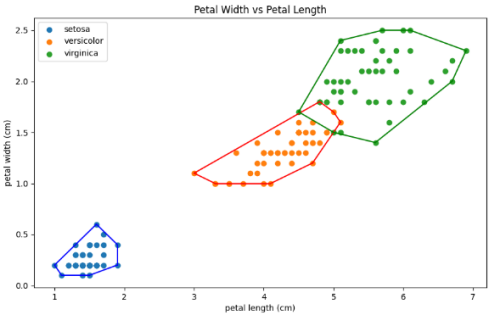
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn import datasets


data = datasets.load_iris()
# create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
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 = convexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        # Simple disini isinya pasangan urutan titik di bucket
        # pasangan tersebut akan dihubungkan garis/diplotkan
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
plt.show()
```

Screenshot Program

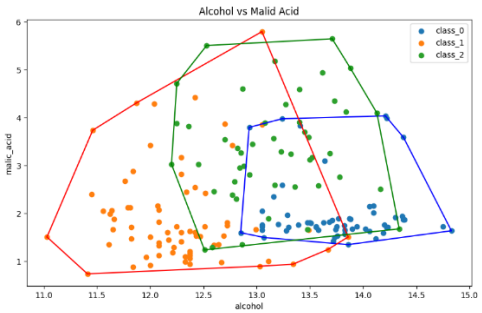
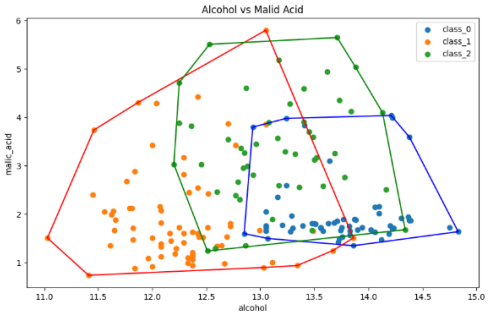
Pengujian sepal-length vs sepal width (Iris Dataset)

 Menggunakan ConvexHull Scipy	 Menggunakan convexHull myConvexHull

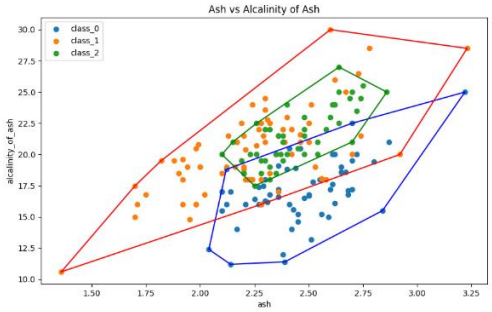
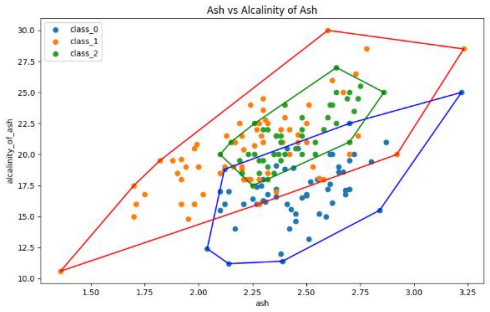
Pengujian Petal Length vs Petal Width (Iris Dataset)

 Menggunakan ConvexHull Scipy	 Menggunakan convexHull myConvexHull

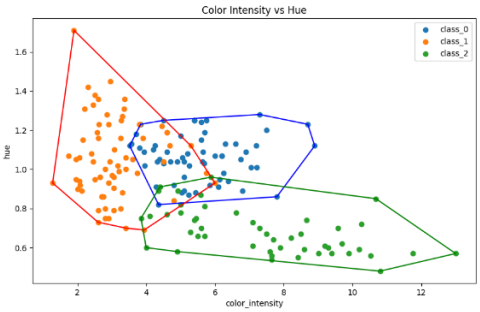
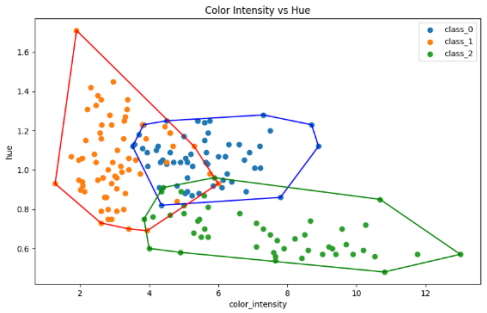
Pengujian Alcohol vs Malic Acid (Wine Dataset)

 Menggunakan ConvexHull Scipy	 Menggunakan convexHull myConvexHull

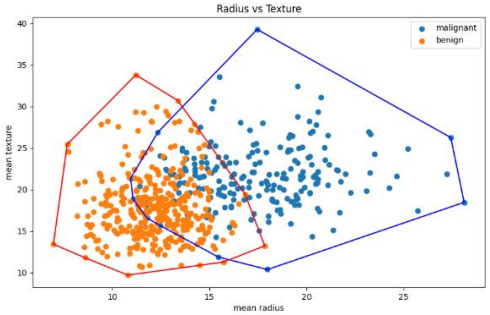
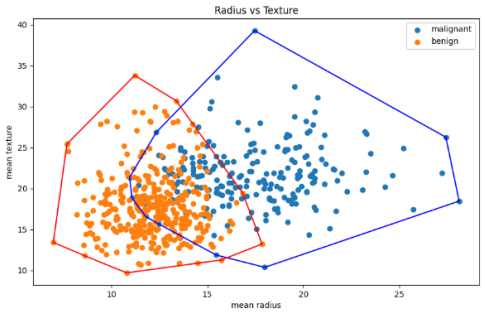
Pengujian Ash vs Alcalinity of Ash (Wine Dataset)

 Menggunakan ConvexHull Scipy	 Menggunakan convexHull myConvexHull

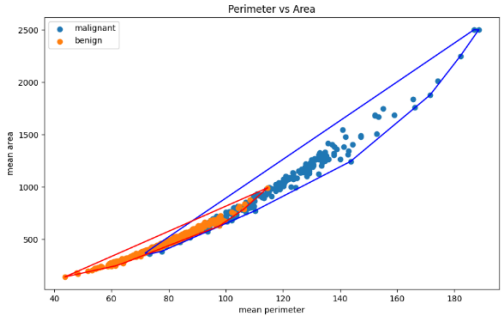
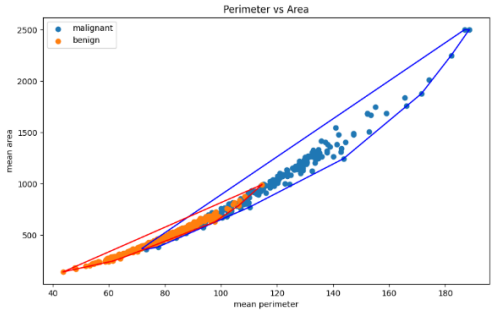
Pengujian Color Intensity vs Hue (Wine Dataset)

 Menggunakan ConvexHull Scipy	 Menggunakan convexHull myConvexHull

Pengujian Radius vs Texture (Breast Cancer Dataset)

 Menggunakan ConvexHull Scipy	 Menggunakan convexHull myConvexHull

Pengujian Perimeter vs Area (Breast Cancer Dataset)

	
Menggunakan ConvexHull Scipy	Menggunakan convexHull myConvexHull

No.	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.	Program dapat menerima input dan menuliskan output untuk dataset lainnya	✓	

Link repository GitHub : https://github.com/sumertayoga/Tucil-2-Stima.git