Laporan Tugas Kecil 2

IF2211 Strategi Algoritma

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

Oleh :

Lyora Felicya

13520073

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

BAB 1 – Algoritma *Divide and Conquer*

Divide and Conquer merupakan sebuah metode pemecahan masalah dengan membagi masalah ke dalam beberapa upa-masalah yang lebih kecil. Secara garis besar, algoritma *Divide and Conquer* terdiri atas tiga langkah, yaitu :

1. ***Divide***
   Membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya berukuran hampir sama),
2. ***Conquer (solve)***
   Menyelesaikan masing-masing upa-persoalan (secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar).
3. ***Combine***
   Mengabungkan solusi masing-masing upa-persoalan sehingga membentuk solusi persoalan semula.

Algoritma *Divide and Conquer* ini dapat diaplikasikan dalam penentuan *Convex Hull* dari sekumpulan titik. Himpunan titik pada bidang planar disebut convex jika untuk sembarang dua titik pada bidang tersebut (misal p dan q), seluruh segmen garis yang berakhir di p dan q berada pada himpunan tersebut. Salah satu contoh pemanfaatan Convex Hull adalah untuk tes *Linearly Separable Dataset.*

Adapun tahap-tahap menemukan convex hull yang digunakan dalam program myConvexHull dibawah adalaht adalah sebagai berikut :

1. Urutkan kumpulan titik berdasarkan nilai absis yang menaik, jika ada nilai absis yang sama, maka diurutkan dengan nilai ordinat yang menaik.
2. Tentukan titik awal yang akan membentuk *convex hull*, yaitu p1 dan pn yang merupakan titik dengan absis terkecil dan terbesar (titik ekstrim).
3. Garis yang menghubungkan p1 dan pn akan membagi S menjadi dua bagian yaitu S1 (kumpulan titik di sebelah kiri atau atas garis $p1pn$) dan S2 (kumpulan titik di sebelah kanan atau bawah garis $p1pn$), digunakan determinan untuk menentukannya.
4. Semua titik pada S yang berada pada garis $p1pn$ (selain titik $p1$ dan pn) tidak mungkin membentuk *convex hull*, sehingga bisa diabaikan dari pemeriksaan
5. Kumpulan titik pada S1 bisa membentuk *convex hull* bagian atas/kiri, dan kumpulan titik pada S2 bisa membentuk *convex hull* bagian bawah/kanan
6. Untuk setiap bagian, pilih sebuah titik yang memiliki jarak maksimum dari garis $p1pn$
7. Tentukan kumpulan titik yang berada di sebelah kiri garis $p1pmax$ (menjadi bagian S1,1), dan di sebelah kanan garis $p1pmax$(menjadi bagian S1,2)
8. Ulangi langkah 5 - 7 secara rekursif hingga tidak terdapat titik lagi di sebelah kiri dan kanan gairs.
9. Gabungkan titik ekstrim awal, *convex hull* kiri, dan convex hull kanan, dan kembalikan pasangan titik yang dihasilkan.

Kompleksitas algoritma *Convex Hull* secara *Divide and Conquer* ini adalah **O(n log n)**

BAB 2 – Kode Program

Kode Pustaka ConvexHull

```
# Tugas Kecil 2 IF2211 Strategi Algoritma
# Lyora Felicya
# 13520073
# Convex Hull dengan Divide and Conquer
# Tujuan    : Menemukan kumpulan titik 'terluar' yang membentuk
convex hull

# Array untuk menyimpan solusi
arrConvexHull = []

# Mencari determinan
# Digunakan untuk menentukan posisi titik p3 terhadap garis yang
dibentuk p1 dan p2
def determinant(p1, p2, p3):
    return (p1[0]*p2[1] + p3[0]*p1[1] + p2[0]*p3[1]) - (p3[0]*p2[1] +
p2[0]*p1[1] + p1[0]*p3[1])

# Mencari titik ekstrim dari sekumpulan titik
def findMinMax(arrTitik):
    pMin = arrTitik[0]
    pMax = arrTitik[0]

    for point in arrTitik:
        if(point[0] <= pMin[0]):
            pMin = point
        if(point[0] >= pMax[0]):
            pMax = point

    return pMin, pMax

# Mengembalikan jarak dari sebuah titik p3 ke garis yang dibentuk p1
dan pn
def pointToLineDistance(p1, pn, p3):
    return abs((p3[1] - p1[1]) * (pn[0] - p1[0]) - (pn[1] - p1[1]) *
(p3[0] - p1[0]))

# Membagi sekumpulan titik ke dalam 2 bagian
def divideSection(points,min_X,max_X):
    leftHull = []
    rightHull = []

    for point in points:
        if(point != min_X) and (point != max_X):
            if (determinant(min_X,max_X,point) > 0):
                leftHull.append(point)
```

```
            if (determinant(min_X,max_X,point) < 0):
                rightHull.append(point)

    return leftHull,rightHull

# Memproses convex hull kiri
def leftConvexHull(p1, pn, arrKiri):
    if (len(arrKiri)!= 0):
        # Mengambil titik dengan jarak terjauh
        arrKiri.sort(key=lambda x:pointToLineDistance(p1, pn, x),
reverse=True)
        maxPoint = arrKiri[0]
        arrConvexHull.append(maxPoint)
        arrKiri.remove(maxPoint)

        # Mengulangi proses pembentukan convex hull kiri hingga tidak
terdapat titik lagi di sebelah kiri
        titikKiri1,_ = divideSection(arrKiri,p1,maxPoint)
        titikKiri2,_ = divideSection(arrKiri,maxPoint,pn)
        leftConvexHull(p1,maxPoint,titikKiri1)
        leftConvexHull(maxPoint,pn,titikKiri2)

# Memproses convex hull kanan
def rightConvexHull(p1, pn, arrKanan):
    if (len(arrKanan)!= 0):
        # Mengambil titik dengan jarak terjauh
        arrKanan.sort(key=lambda x:pointToLineDistance(p1, pn, x),
reverse=True)
        maxPoint = arrKanan[0]
        arrConvexHull.append(maxPoint)
        arrKanan.remove(maxPoint)

        # Mengulangi proses pembentukan convex hull kanan hingga
tidak terdapat titik lagi di sebelah kanan
        _,titikKanan1 = divideSection(arrKanan, p1, maxPoint)
        _,titikKanan2 = divideSection(arrKanan, maxPoint, pn)
        rightConvexHull(p1,maxPoint,titikKanan1)
        rightConvexHull(maxPoint,pn,titikKanan2)

# Mengurutkan titik agar dapat digambar berbentuk poligon
def makePolygon(arr):
    leftmost, rightmost = findMinMax(arr)
    arrLeft, arrRight =  divideSection(arr, leftmost, rightmost)

    arrLeft.sort(key=lambda k: [k[0], k[1]])
    arrRight.sort(key=lambda k: [k[0], k[1]], reverse=True)

    return [leftmost]+arrLeft+[rightmost]+arrRight+[leftmost]
```

```
# Inisiasi convex hull pertama kali
def convexHull(arrTitik):
    # Mengosongkan arrConvexHull yang merupakan variabel global
    arrConvexHull.clear()

    # Mengambil titik ekstrim sebagai titik awal yang akan membentuk
convex hull
    P1, Pn = findMinMax(arrTitik)
    arrConvexHull.append(P1)
    arrConvexHull.append(Pn)

    # Tahap pertama dari divide and conquer, membagi sekumpulan titik
S menjadi 2 bagian
    arrKiri, arrKanan = divideSection(arrTitik, P1, Pn)

     # Divide and Conquer
    leftConvexHull(P1,Pn,arrKiri)
    rightConvexHull(P1,Pn,arrKanan)

    # Mengembalikan titik yang dihasilkan setelah digabung dengan
convex hull kiri dan kanan
    arrPolygon = arrConvexHull
    arrPolygon = makePolygon(arrPolygon)
    return(arrPolygon)
```

Kode Visualisasi Hasil Convex Hull

catatan : kode ini perlu diubah – ubah sesuai dengan keperluan data yang ingin diperiksa

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

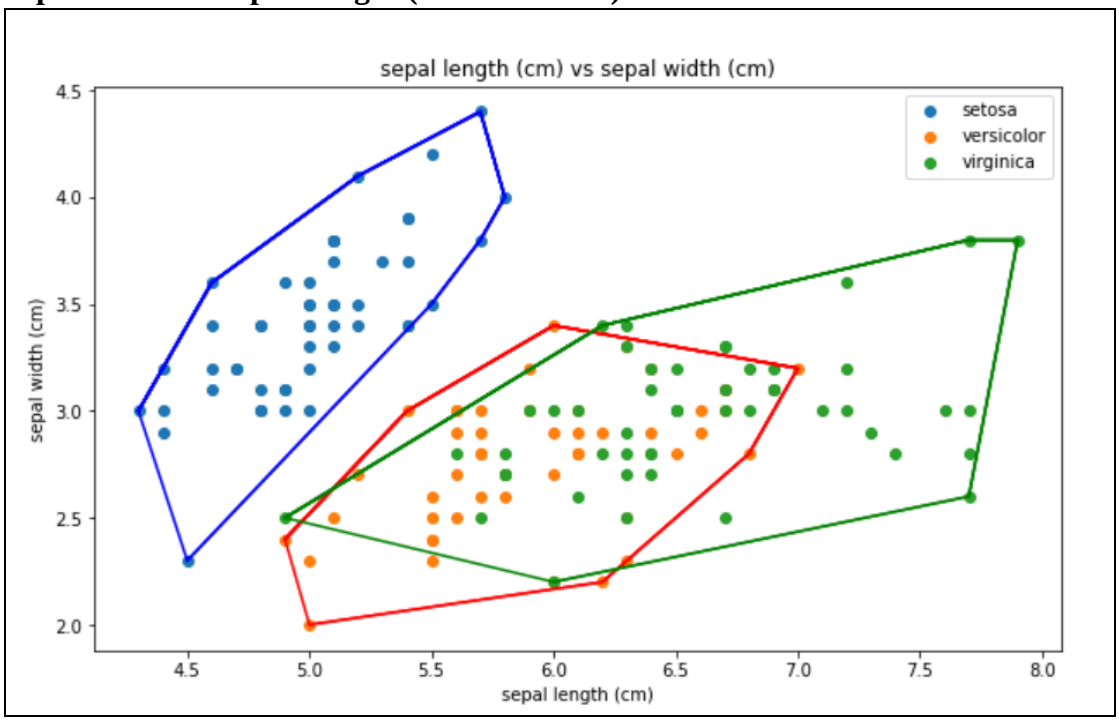
```
#visualisasi hasil ConvexHull (implementasi sendiri)
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
```

```
#kolom yang ingin digunakan
a = 0
b = 1

plt.title(str(data.feature_names[a]) + ' vs ' +
str(data.feature_names[b]))
plt.xlabel(data.feature_names[a])
plt.ylabel(data.feature_names[b])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[a,b]].values
    arrBucket = bucket.tolist() # Mengubah ndarray menjadi array of
points (list)

    hull = convexHull(arrBucket) #bagian ini diganti dengan hasil
implementasi ConvexHull Divide & Conquer
    absis = []
    ordinat = []
    plt.scatter(bucket[:, 0], bucket[:, 1],
label=data.target_names[i])
    for j in range(len(hull)):
        point = hull[j]
        absis.append(point[0])
        ordinat.append(point[1])
        plt.plot(absis, ordinat, colors[i])
plt.legend()
```

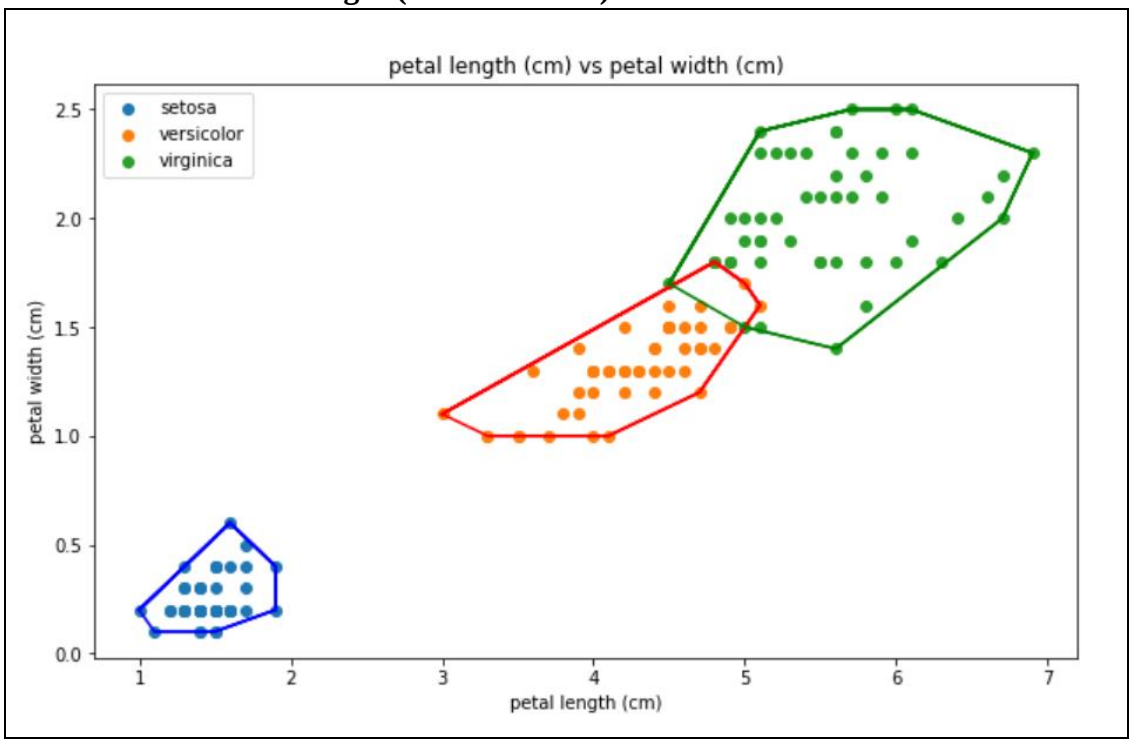
Bab 3 – Screenshot Hasil Pengujian

3.1 Dataset Iris

3.1.1 Sepal Width vs Sepal Length (kolom 0 dan 1)

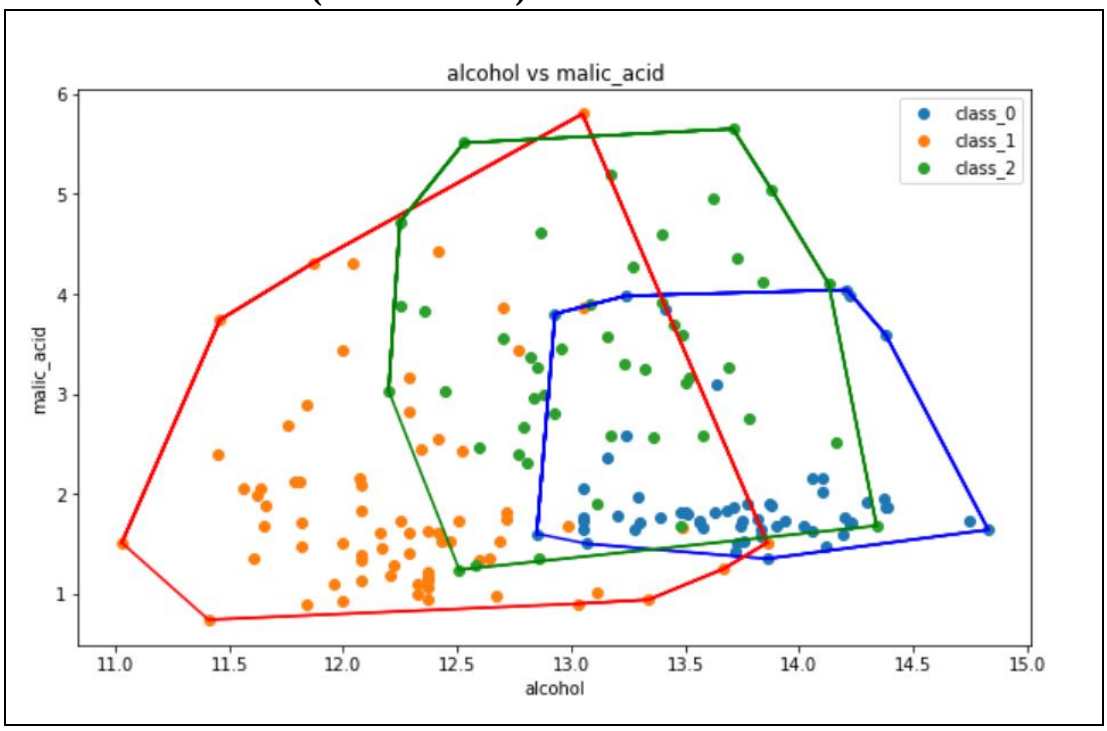


3.1.2 Petal Width vs Petal Length (kolom 2 dan 3)

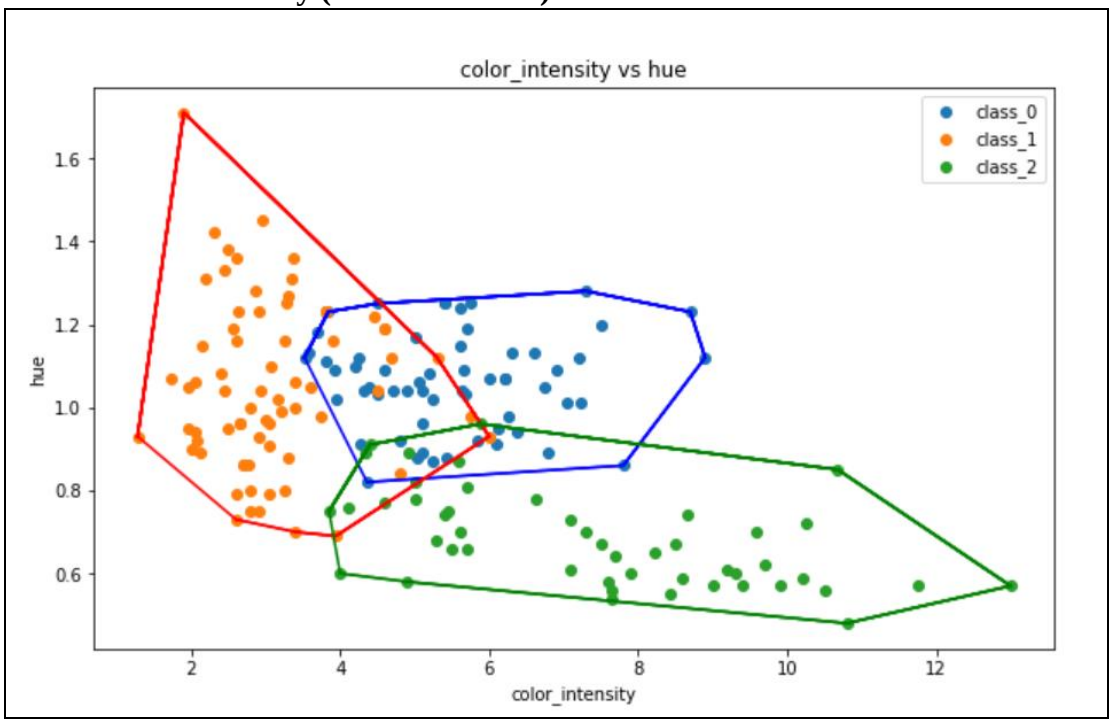


3.2 Dataset Wine

3.2.1 Malic Acid vs Alcohol (kolom 0 dan 1)

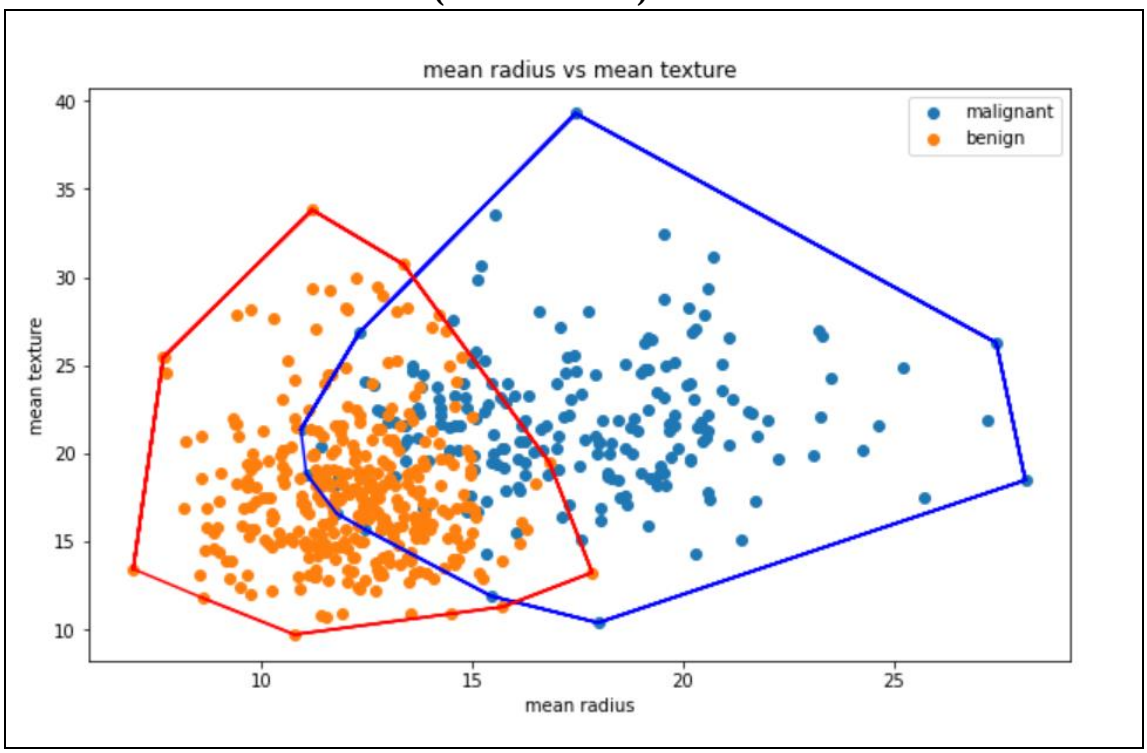


3.2.2 Hue vs Color Intensity (kolom 9 dan 10)

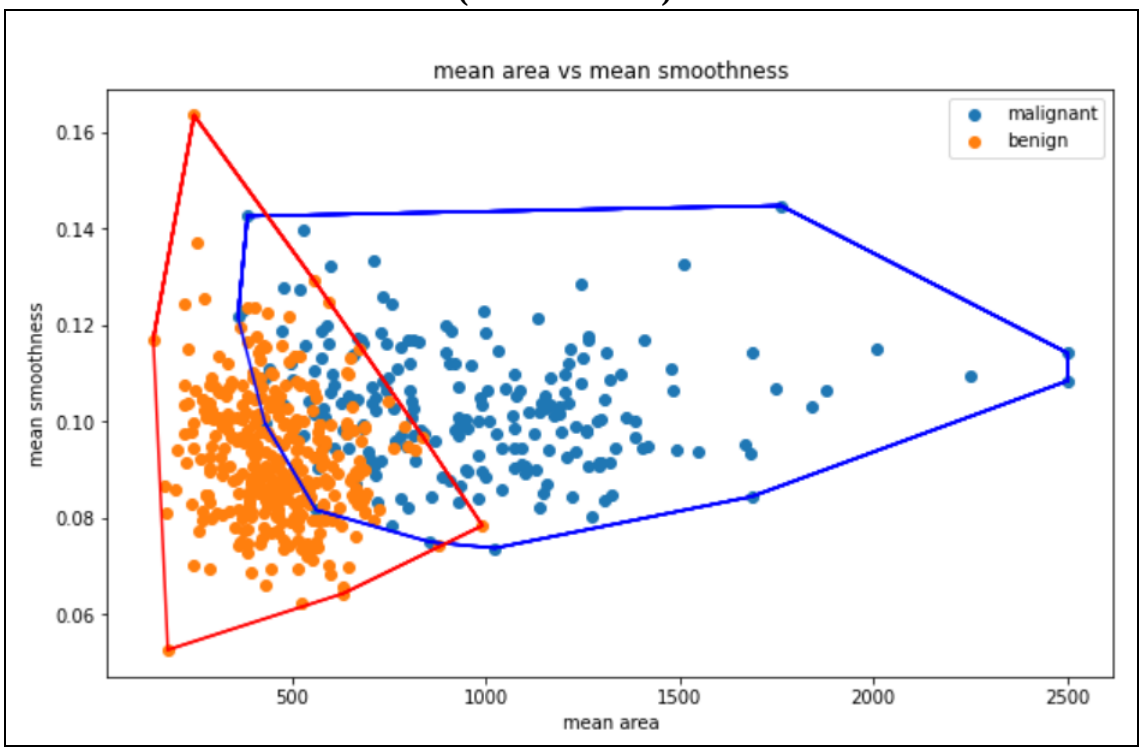


3.3 Dataset Breast Cancer

3.3.1 Mean Texture vs Mean Radius (kolom 0 dan 1)



3.3.2 Mean Smoothness vs Mean Area (kolom 3 dan 4)



Bab 4 – Lampiran

4.1 Alamat Drive Program

https://github.com/lyorafelicya/myConvexHull

4.2 Tabel *Checklist*

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.	√	

Referensi

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf