Tugas Kecil 2 IF2211 Strategi Algoritma

Semester II tahun 2021/2022

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

Disusun oleh:

Brianaldo Phandiarta 13520113

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

I. Algoritma *Divide and Conquer*

Algoritma *divide and conquer* adalah algoritma yang membagi persoalan menjadi beberapa upapersoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil. Kemudan, algoritma akan menyelesaikan masing-masing upapersoalan, yaitu dengan secara langsung ataupun secara rekursif. Setelah ditemukan solusi untuk masing-masing upapersoalan, solusi-solusi tersebut akan digabung hingga membentuk solusi untuk persoalan semula.

Dalam menyelesaikan persoalan pencarian *convex hull*, penulis menggunakan sumber https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf.

Algoritma *divide and conquer* untuk pencarian *convex hull*:

1. Mula-mula, algoritma akan menerima masukan himpunan titik (absis, ordinat) yang sudah terurut berdasarkan absis.
2. Kemudian, algoritma akan mengambil 2 titik, yaitu p_1 dan p_2. Titik p_1 merupakan titik paling kiri (absis terkecil) dan titik p_2 merupakan titik paling kanan (absis terbesar).
3. Titik yang diambil akan membentuk garis p_1p_2 dan p_2p_1 sehingga membentuk dua daerah, yaitu S_1 (daerah disebelah kanan garis p_1p_2) dan S_2 (daerah disebelah kanan garis p_2p_1.
4. Himpunan titik akan ditentukan termasuk pada daerah S_1 atau S_2 dengan menggunakan penentuan determinan, yaitu titik p_3 berada di sebelah kanan garis p_1p_2 (S_1) jika hasil dari determinan positif. Begitu pula sebaliknya untuk garis p_2p_1 (S_2). Rumus penentuan determinan adalah sebagai berikut.
$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$
dengan x_n dan y_n merupakan absis dan ordinat dari p_n.
5. Kumpulan titik pada S_1 dapat membentuk *convex hull* bagian atas, dan kumpulan titik pada S_2 dapat membentuk *convex hull* bagian bawah.
6. Dari masing-masing daerah, akan dilakukan algoritma *divide and conquer*.
7. Jika pada suatu bagian S tidak terdapat titik lain selain p_1 dan p_2, maka titik p_1 dan p_2 akan menjadi *convex hull* pada bagian S.
8. Jika pada suatu bagian S tidak kosong, algoritma akan memilih titik dengan jarak terjauh dari garis p_1p_2, yaitu p_{max}. Jika terdapat beberapa titik dengan jarak yang sama, pilih sebuat titik yang memaksimalkan sudut $p_{max}p_1p_2$.
9. Kemudian, himpunan titik akan ditentukan termasuk pada daerah $S_{1,1}$ atau $S_{1,2}$ dengan menggunakan penentuan determinan, yaitu titik p_3 berada di sebelah kanan garis p_1p_{max} ($S_{1,1}$) jika hasil determinan positif. Begitu pula sebalikanya untuk garis $p_{max}p_2$ ($S_{1,2}$).
10. Langkah 6, 7, 8 ,dan 9 akan dilakukan secara rekursif hingga langkah 7 terpenuhi, yaitu hingga seluruh upabagian telah kosong.
11. Setelah ditemukan solusi-solusi untuk setiap upadaerah (upapersoalan), masing-masing solusi akan digabung dan dikembalikan sebagai pasangan titik (*simplices*).

II. *Source Code/Library* myConvexHull dengan Bahasa Python

1. Include library

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

2. Fungsi Antara dan Variabel Konstan

   a. Variabel Konstan (EPSILON)

```
# EPSILON digunakan untuk menghandle kasus rounding error
EPSILON = 0.00001
```

   b. Fungsi checkPosition

```
# function checkPosistion (p1, p2, p3 : point) -> real
# Mengembalikan suatu angka untuk mengetahui di sebelah mana p3
# menurut garis yang dibentuk p1 dan p2 menggunakan determinan
def checkPosition(p1, p2, p3):
    mat = np.array([
        np.append(p1, 1),
        np.append(p2, 1),
        np.append(p3, 1),
    ])
    return np.linalg.det(mat)
```

   c. Fungsi getAngle

```
# function getAngle (p1, p2, p3 : point) -> real
# Mengembalikan sudut yang dibentuk oleh p1p2p3
def getAngle(p1, p2, p3):
    ba = p1 - p3
    bc = p2 - p3

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

    return np.degrees(angle)
```

   d. Fungsi getMaxDistancePoint

```
# function getMaxDistancePoint (points : array of point, p1, p2 :
point) -> point
# Mengembalikan point dengan jarak terjauh dari garis p1p2
# Jika terdapat jarak yang sama, akan dibandingkan dari besar sudut
```

```
def getMaxDistancePoint(points, p1, p2):
    maxPoint = points[0]
    maxDis = np.linalg.norm(np.cross(p2-p1, p1-
points[0]))/np.linalg.norm(p2-p1)
    for point in points:
        dis = np.linalg.norm(np.cross(p2-p1, p1-
point))/np.linalg.norm(p2-p1)
        if (dis > maxDis):
            maxDis = dis
            maxPoint = point
        elif (dis == maxDis):
            if (getAngle(point, p2, p1) > getAngle(maxPoint, p2,
p1)):
                maxDis = dis
                maxPoint = point
    return maxPoint
```

e. Fungsi merge

```
# function merge (simplices1, simplices2 : simplices) -> simplices
# Mengembalikan hasil gabungan dari simplices1 dan simplices2
def merge(simplices1, simplices2):
    if (simplices1.size == 0):
        return simplices1
    else:
        return np.vstack([simplices1, simplices2])
```

3. Fungsi Utama convexHull Menggunakan Algoritma *Divide and Conquer*

a. Fungsi convexHull

```
# function convexHull (points : array of point) -> simplices
# Mengembalikan simplices dari points
# Menggunakan Algoritma Divide and Conquer
def convexHull(points):
    # sort
    points.view('i8,i8').sort(order=['f0','f1'], axis=0)

    # get p1 and p2
    p1 = points[0]
    p2 = points[-1]

    # partiton
    S1 = np.array([np.array([0,0])]) # Set of points right to the
                                     line p1p2
    S2 = np.array([np.array([0,0])]) # Set of points right to the
                                     line p2p1

    for point in points:
        if (checkPosition(p1, p2, point) > EPSILON):
            S1 = np.vstack([S1, point])
        if (checkPosition(p2, p1, point) > EPSILON):
            S2 = np.vstack([S2, point])
```

```
    S1 = np.delete(S1, 0, 0)
    S2 = np.delete(S2, 0, 0)

    simplices = merge(findHull(S1, p1, p2), findHull(S2, p2, p1))

    return simplices
```

b. Fungsi findHull

```
# function findHull (points : array of point, p1, p2 : point) ->
simplices
# Mengembalikan simplices dari points
# Menggunakan fungsi rekursif
def findHull(points, p1, p2):
    if (points.size == 0):
        return np.array([np.array([p1, p2])])
    else:
        pMax = getMaxDistancePoint(points, p1, p2)

        S1 = np.array([np.array([0,0])]) # Set of points right to the
                                           line p1pMax
        S2 = np.array([np.array([0,0])]) # Set of points right to the
                                           line pMaxp2

        for point in points:
            if (checkPosition(p1, pMax, point) > EPSILON):
                S1 = np.vstack([S1, point])
            if (checkPosition(pMax, p2, point) > EPSILON):
                S2 = np.vstack([S2, point])

        S1 = np.delete(S1, 0, 0)
        S2 = np.delete(S2, 0, 0)

        simplices = merge(findHull(S1, p1, pMax), findHull(S2, pMax,
p2))

        return simplices
```

III. *Screenshot* dari *Input-Output Test Cases*

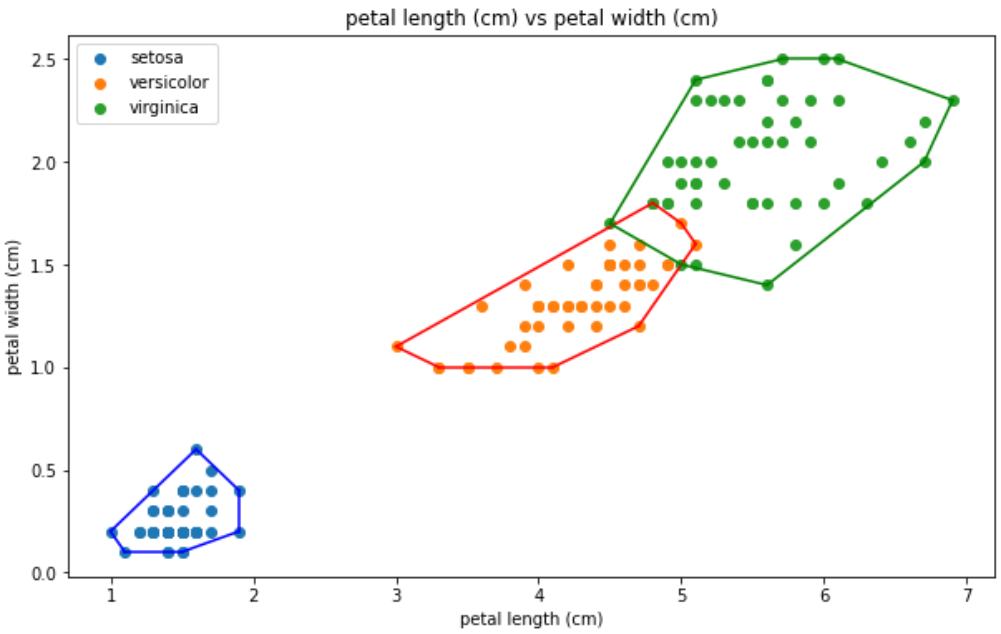
1. Data iris

   a. *petal-length* dan *petal-width*

      i. *Input*

```
data = datasets.load_iris()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
#visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[2] + ' vs ' + data.feature_names[3]))
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[2,3]].values
    hull = convexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        x_values = [simplex[0][0], simplex[1][0]]
        y_values = [simplex[0][1], simplex[1][1]]
        plt.plot(x_values, y_values, colors[i])
plt.legend()
```

      ii. *Output*

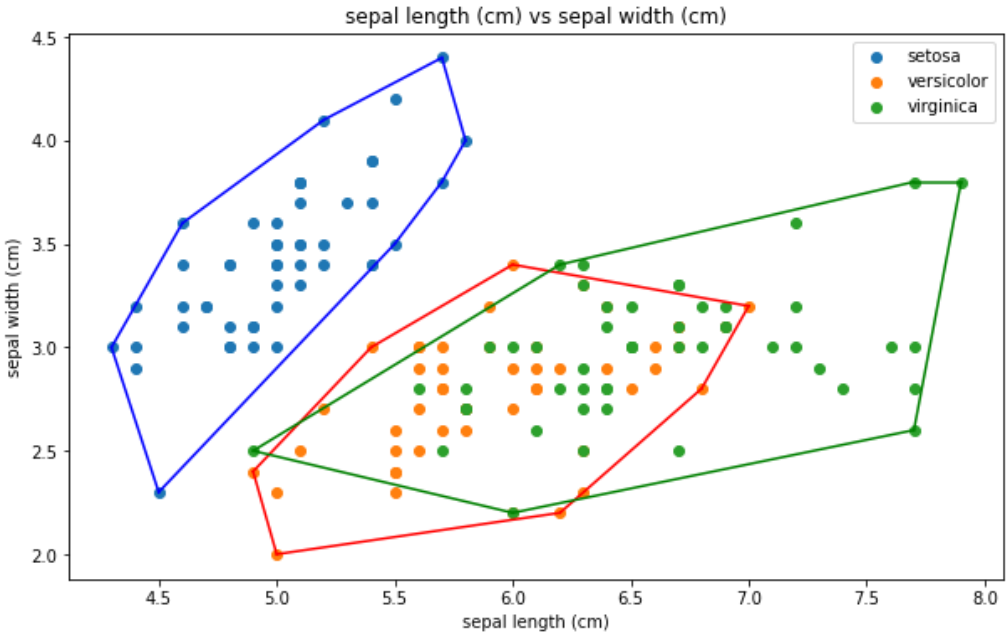


b. *sepal-length* dan *sepal-width*

i. *Input*

```
data = datasets.load_iris()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
#visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[0] + ' vs ' + data.feature_names[1]))
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) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        x_values = [simplex[0][0], simplex[1][0]]
        y_values = [simplex[0][1], simplex[1][1]]
        plt.plot(x_values, y_values, colors[i])
plt.legend()
```

ii. *Output*



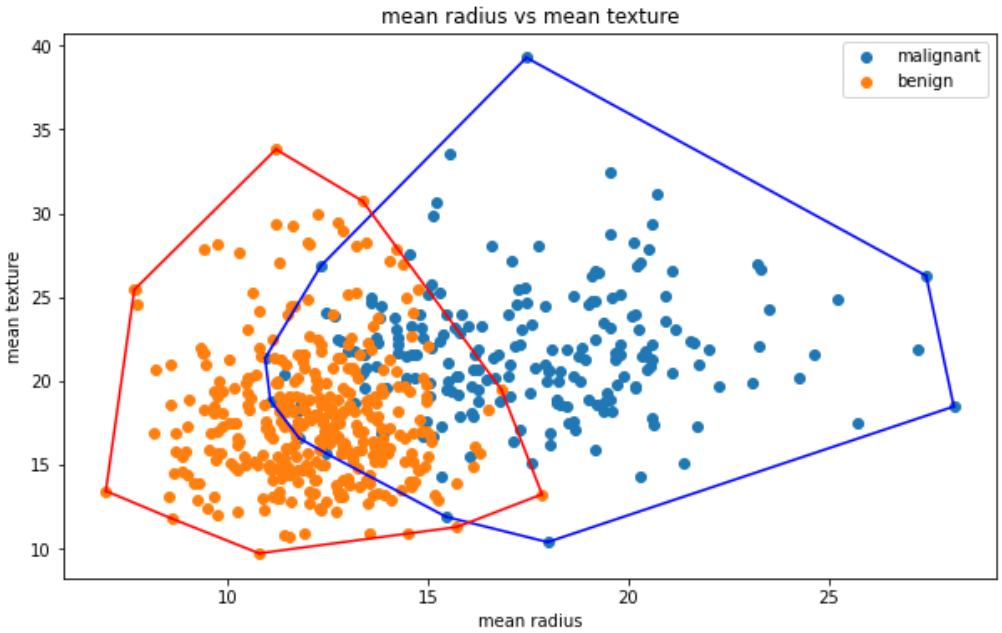
2. Data breast_cancer (*bonus*)

   a. *mean radius* dan *mean texture*

      i. *Input*

```python
data = datasets.load_breast_cancer()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
#visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r']
plt.title(str(data.feature_names[0] + ' vs ' + data.feature_names[1]))
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) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        x_values = [simplex[0][0], simplex[1][0]]
        y_values = [simplex[0][1], simplex[1][1]]
        plt.plot(x_values, y_values, colors[i])
plt.legend()
```

      ii. *Output*

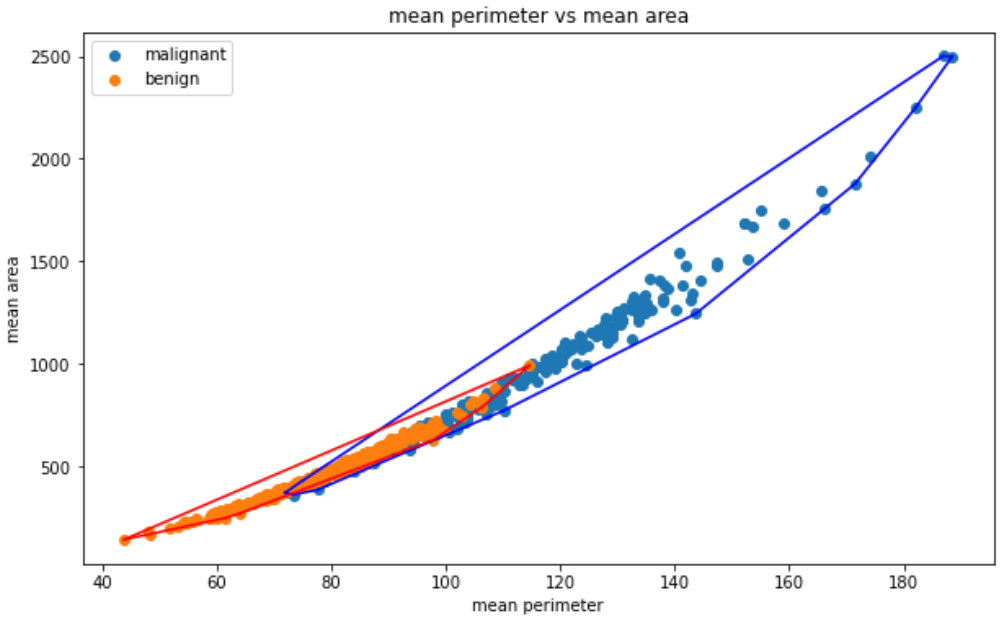


b. *mean perimeter* dan *mean area*

i. *Input*

```python
data = datasets.load_breast_cancer()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
#visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r']
plt.title(str(data.feature_names[2] + ' vs ' + data.feature_names[3]))
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[2,3]].values
    hull = convexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        x_values = [simplex[0][0], simplex[1][0]]
        y_values = [simplex[0][1], simplex[1][1]]
        plt.plot(x_values, y_values, colors[i])
plt.legend()
```

ii. *Output*

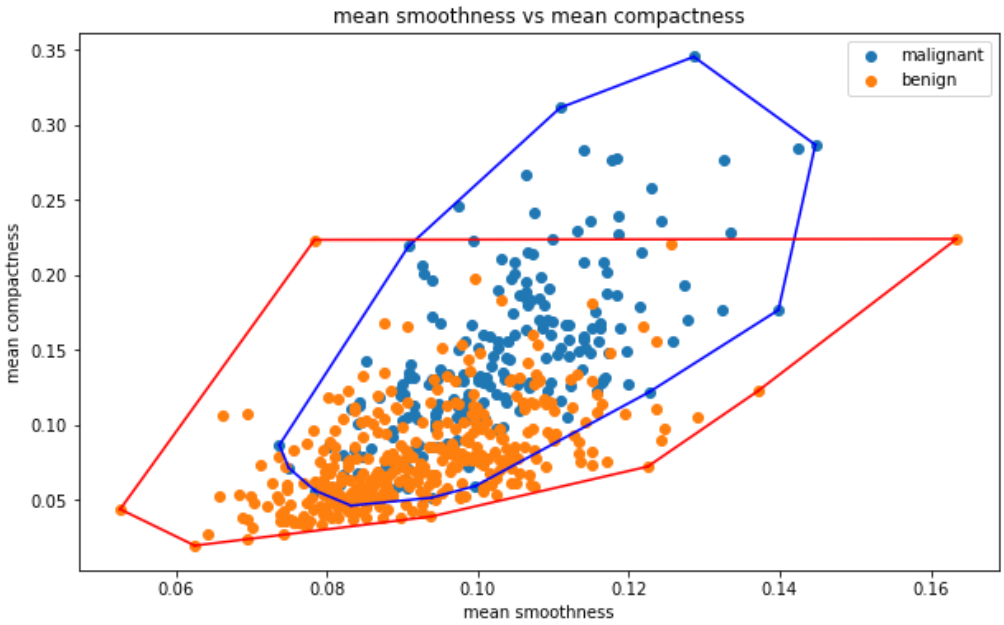


c. *mean smoothness* dan *mean compactness*

i. *Input*

```
data = datasets.load_breast_cancer()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
#visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r']
plt.title(str(data.feature_names[4] + ' vs ' + data.feature_names[5]))
plt.xlabel(data.feature_names[4])
plt.ylabel(data.feature_names[5])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[4,5]].values
    hull = convexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        x_values = [simplex[0][0], simplex[1][0]]
        y_values = [simplex[0][1], simplex[1][1]]
        plt.plot(x_values, y_values, colors[i])
plt.legend()
```

ii. *Output*

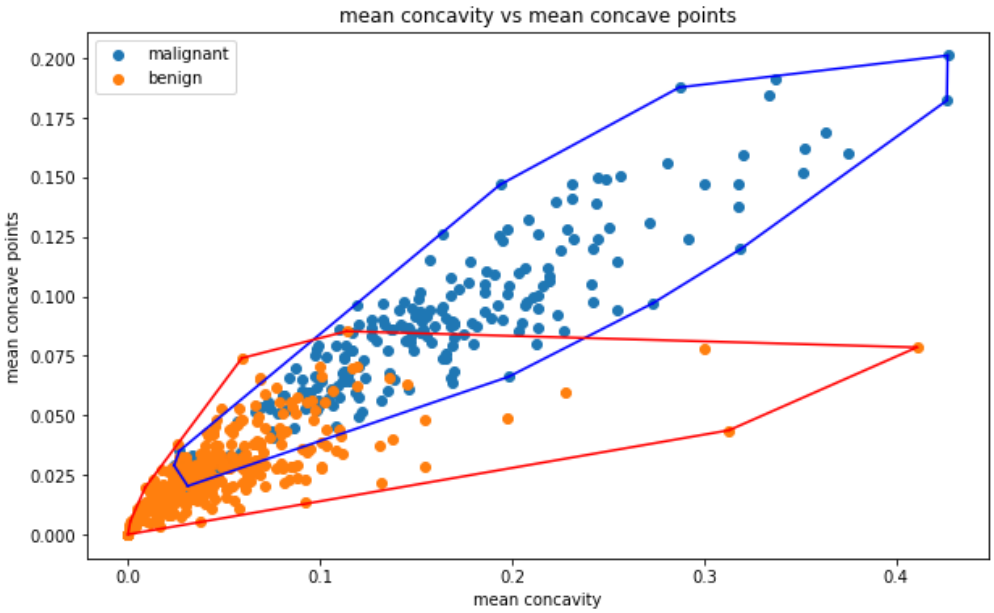


d. *mean concavity* dan *mean concave points*

i. *Input*

```
data = datasets.load_breast_cancer()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
#visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r']
plt.title(str(data.feature_names[6] + ' vs ' + data.feature_names[7]))
plt.xlabel(data.feature_names[6])
plt.ylabel(data.feature_names[7])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[6,7]].values
    hull = convexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        x_values = [simplex[0][0], simplex[1][0]]
        y_values = [simplex[0][1], simplex[1][1]]
        plt.plot(x_values, y_values, colors[i])
plt.legend()
```

ii. *Output*

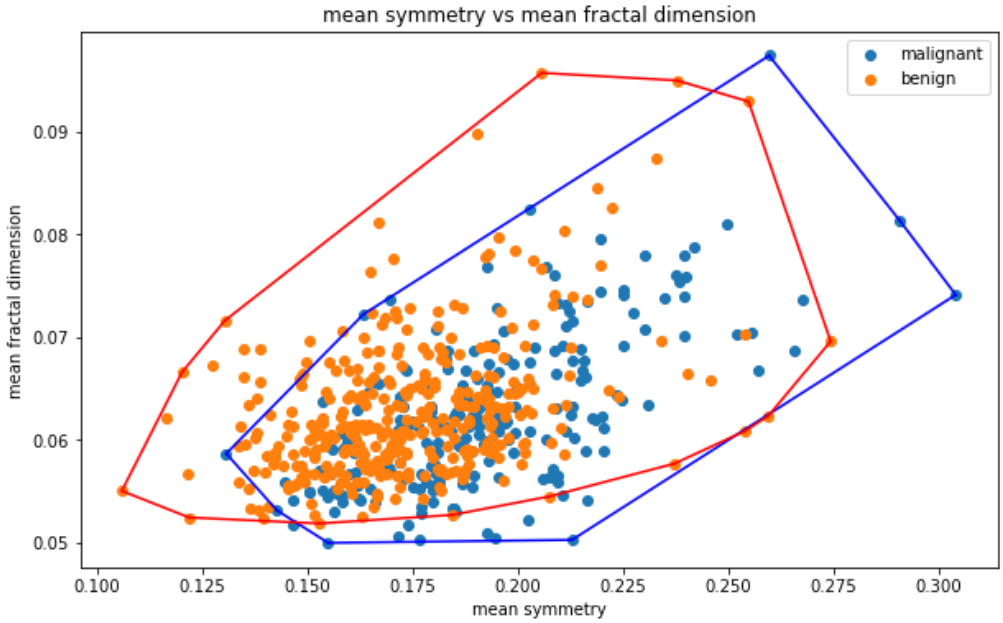


e. *mean symmetry* dan *fractal dimension*

i. *Input*

```
data = datasets.load_breast_cancer()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
#visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r']
plt.title(str(data.feature_names[8] + ' vs ' + data.feature_names[9]))
plt.xlabel(data.feature_names[8])
plt.ylabel(data.feature_names[9])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[8,9]].values
    hull = convexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        x_values = [simplex[0][0], simplex[1][0]]
        y_values = [simplex[0][1], simplex[1][1]]
        plt.plot(x_values, y_values, colors[i])
plt.legend()
```

ii. *Output*

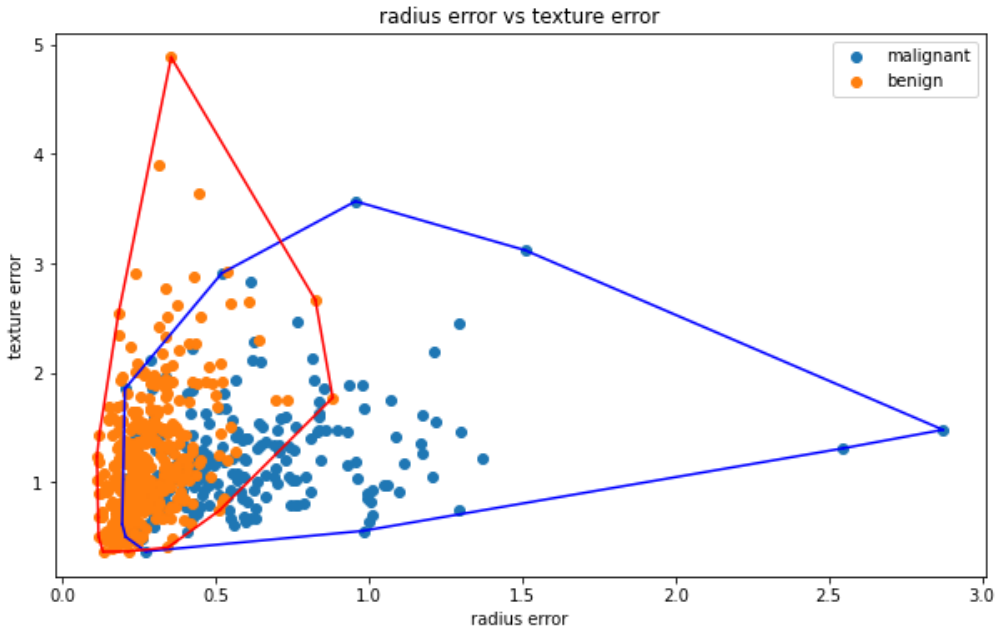


f. *radius error* dan *texture error*

i. *Input*

```
data = datasets.load_breast_cancer()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
#visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r']
plt.title(str(data.feature_names[10] + ' vs ' + data.feature_names[11]))
plt.xlabel(data.feature_names[10])
plt.ylabel(data.feature_names[11])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[10,11]].values
    hull = convexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        x_values = [simplex[0][0], simplex[1][0]]
        y_values = [simplex[0][1], simplex[1][1]]
        plt.plot(x_values, y_values, colors[i])
plt.legend()
```

ii. *Output*



IV. *Link* Kode Program

1. **Google Drive**

   https://drive.google.com/drive/folders/1wBQ3Hy8E_hLHtLQ962NgWjxjCSVMrGo1?usp=sharing

2. **Repository GitHub**

   https://github.com/Brianaldo/Tucil2_13520113

V. *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 menampilkan *convex hull* setiap label dengan warna yang berbeda	✓	
4. **Bonus**: program dapat menerima input dan menuliskan output untuk dataset lainnya.	✓	