LAPORAN TUGAS KECIL 2

STRATEGI ALGORITMA IF2211

IMPLEMENTASI CONVEX HULL MENGGUNAKAN ALGORITMA DIVIDE AND CONQUER

Ziyad Dhia Rafi

13520064

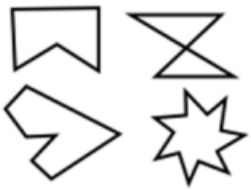
Algoritma Divide And Conquer

Algoritma Divide And Conquer adalah sebuah pendekatan pemrograman yang memecah masalah yang kompleks menjadi masalah-masalah kecil yang sederhana (basis) yang disebut sebagai Divide. Setelah dipecah dan diselesaikan, hasil perhitungan masalah-masalah kecil tersebut digabung menjadi satu kesatuan yang disebut sebagai conquer.

Convex Hull

Sebuah himpunan titik pada bidang planar disebut *convex* apabila untuk sembarang dua titik pada bidang tersebut (misalnya p dan q), seluruh segmen garis yang berakhir di p dan q berada pada himpunan tersebut

Gambar 1: convex



Gambar 2:non convex

Convex hull dari sebuah himpunan titik S adalah himpunan convex terkecil (convex polygon) yang mengandung S.

Algoritma Divide And Conquer Convex Hull

Pada implementasi Convex Hull kali ini, digunakan algoritma Divide And Conquer. Algoritma tersebut dapat dilihat dari Langkah-langkah berikut ini:

1. Dari sebuah himpunan titik S, dicari titik dengan x terkecil (apabila sama, cari y terkecil), dan x terbesar (apabila sama, cari y terbesar) yang kemudian kedua titik tersebut disimpan indeksnya dalam IdxMin dan IdxMax
2. Bentuk garis dari kedua titik tersebut yang memisahkan himpunan titik tersebut menjadi s1 dan s2 yang berisi indeks dari titik-titik tersebut. Posisi titik relatif terhadap garis dapat dihitung menggunakan fungsi determinan
3. Pada kedua himpunan tersebut lakukan pemanggilan fungsi rekursif hingga menuju basis.
   a. Apabila himpunan titik (indeks) memiliki besar 0 atau 1, maka kembalikan himpunan tersebut (himpunan kosong atau berukuran satu)
   b. Selain itu, cari titik yang memiliki jarak terjauh dari garis yang terbentuk sehingga membentuk segitiga (2 garis).
      Dari segitiga tersebut terbentuk 2 wilayah s1 dan s2.
      Selesaikan kedua wilayah tersebut dengan melakukan rekursi fungsi
      Gabung kedua hasil dua wilayah tersebut dan kembalikan hasilnya
4. Gabung kedua hasil Convex Hull dari s1 dan s2.
5. Pada akhirnya akan didapat himpunan Convex Hull yang akan menyimpan relasi titik-titik (indeks).

Kode Program

myConvexHull.py

```
import numpy as np
import math
# Starting Convex Hull
def myConvexHull(arrayPoints):
    s1 = np.array([])
    s2 = np.array([])
    idxMin, idxMax = myMinMax(arrayPoints)
    s1, s2 = divide(idxMin, idxMax, range(len(arrayPoints)), arrayPoints)
    hull1 = myConvexHullRec(idxMin, idxMax, s1, arrayPoints)
    hull2 = myConvexHullRec(idxMax, idxMin, s2, arrayPoints)
    tempHull = np.concatenate(([idxMin], hull1, [idxMax], hull2))
    HullObject = connectPoints(tempHull)
    HullObject = np.int_(HullObject)
    return HullObject
# make array of connected points (indexes)
def connectPoints(arrayIndexes):
    tempArray = []
    for i in range(len(arrayIndexes)-1):
        tempArray.append([int(arrayIndexes[i]), int(arrayIndexes[i+1])])
    tempArray.append([int(arrayIndexes[len(arrayIndexes)-1]),
int(arrayIndexes[0])])
    tempArray = np.array(tempArray)
    return tempArray

# Recursive function of ConvexHull
def myConvexHullRec(i1, i2, s, arrayPoints):
    # return array of indexes
    if len(s) == 0 or len(s) == 1:
        return s
    else:
        p1 = arrayPoints[i1]
        p2 = arrayPoints[i2]
        #farthest distance
        farIdx = s[0]
        fardistance = PointToLineDistance(p1, p2, arrayPoints[s[0]])
        for i in s:
            distance = PointToLineDistance(p1, p2, arrayPoints[i])
            if distance > fardistance:
                fardistance = distance
                farIdx = i
        s1, s1_dump = divide(i1, farIdx, s, arrayPoints)
        s2, s2_dump = divide(farIdx, i2, s, arrayPoints)
        hull1 = myConvexHullRec(i1, farIdx, s1, arrayPoints)
        hull2 = myConvexHullRec(farIdx, i2, s2, arrayPoints)
        hullR = np.concatenate((hull1, [farIdx], hull2))
```

```
        return hullR

def myMinMax(arrayPoints):
    idxMin = 0
    idxMax = 0
    for i in range(1, len(arrayPoints)):
        # Min check
        if arrayPoints[i, 0] < arrayPoints[idxMin, 0]:
            idxMin = i
        elif arrayPoints[i, 0] == arrayPoints[idxMin, 0]:
            if arrayPoints[i, 1] < arrayPoints[idxMin, 1]:
                idxMin = i
            # Max check
        if arrayPoints[i, 0] > arrayPoints[idxMax, 0]:
            idxMax = i
        elif arrayPoints[i, 0] == arrayPoints[idxMin, 0]:
            if arrayPoints[i, 1] > arrayPoints[idxMin, 1]:
                idxMin = i
    return idxMin, idxMax

# return left and right indexes
def divide(i1, i2, s, arrayPoints):
    s1 = []
    s2 = []
    p1 = arrayPoints[i1]
    p2 = arrayPoints[i2]
    for i3 in s:
        p3 = arrayPoints[i3]
        # left
        if determinant(p1, p2, p3) > 0.0000001: # avoid calculation error
            s1.append(i3)
        # right
        elif determinant(p1, p2, p3) < -0.0000001: # avoid calculation error
            s2.append(i3)
    s1 = np.array(s1)
    s2 = np.array(s2)
    return s1, s2

# Return determinant of 3 points (1 line & 1 point)
def determinant(p1, p2, p3):
    x1 = p1[0]
    x2 = p2[0]
    x3 = p3[0]
    y1 = p1[1]
    y2 = p2[1]
```

```
    y3 = p3[1]
    return x1*y2 + x3*y1 + x2*y3 - x3*y2 - x2*y1 - x1*y3

# Calculate distance of point to line
def PointToLineDistance(p1, p2, px):

    # equation of line
    a = p2[1] - p1[1]
    b = p1[0] - p2[0]
    c = -(a*(p1[0]) + b*(p1[1]))
    # ax + by + C = 0
    a /= b
    c /= b
    b = 1
    xx = px[0]
    yx = px[1]
    return abs(a*xx + b*yx + c)/math.sqrt(a**2 + b**2)
```

file visualizer dataset

*diubah2 sesuai kebutuhan dataset

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

data = datasets.load_iris()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.head())

#visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
idxA = 2
idxB = 3
plt.title((data.feature_names[idxA] + " vs " + data.feature_names[idxB]))
plt.xlabel(data.feature_names[idxA])
plt.ylabel(data.feature_names[idxB])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
```

```
    bucket = bucket.iloc[:,[idxA,idxB]].values
    hull = myConvexHull.myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])

plt.legend()
plt.show()
```

Skrinshut

Dataset Iris:

```
7   data = datasets.load_iris()
8   #create a DataFrame
9   df = pd.DataFrame(data.data, columns=data.feature_names)
10  df['Target'] = pd.DataFrame(data.target)
11  print(df.head())
```

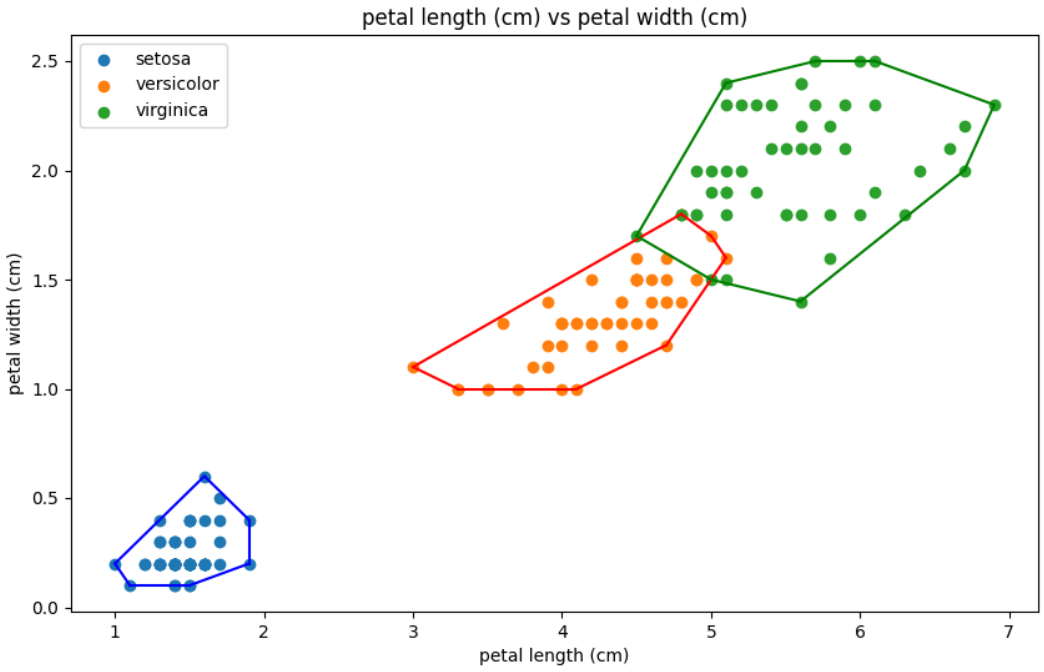
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  Target
0                5.1               3.5                1.4               0.2       0
1                4.9               3.0                1.4               0.2       0
2                4.7               3.2                1.3               0.2       0
3                4.6               3.1                1.5               0.2       0
4                5.0               3.6                1.4               0.2       0
PS D:\KULIAH WOY\AKADEMIK\Semester 4\Strategi Algoritma\Tucil 2>
```

1. Petal length vs Petal Width
   input:

```
#visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
idxA = 2
idxB = 3
plt.title((data.feature_names[idxA] + " vs " + data.feature_names[idxB]))
plt.xlabel(data.feature_names[idxA])
plt.ylabel(data.feature_names[idxB])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[idxA,idxB]].values
    hull = myConvexHull.myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
```

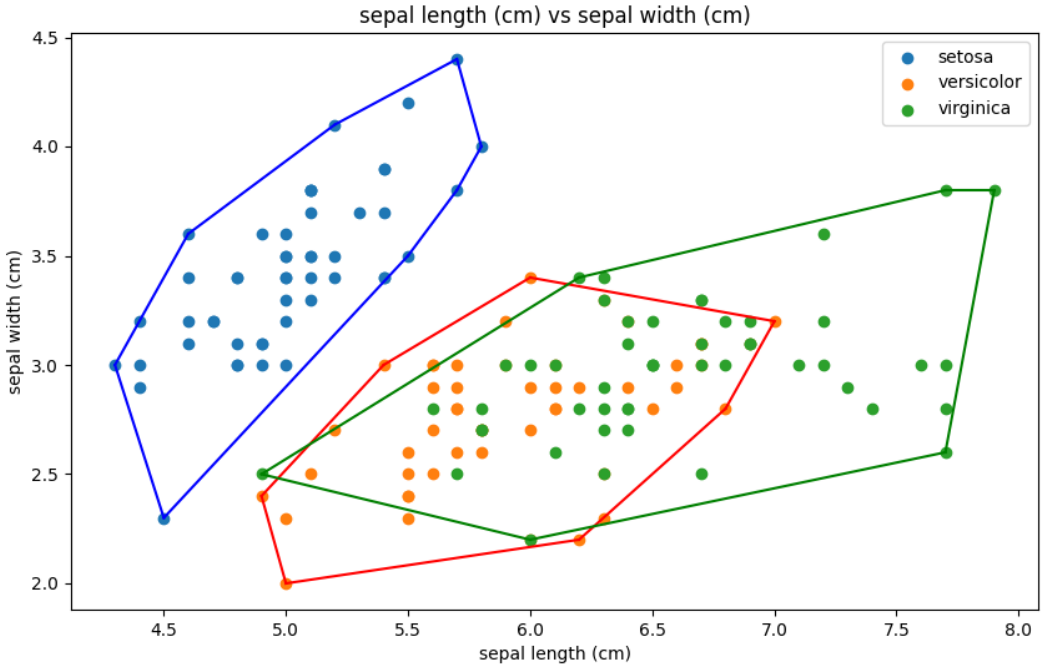
output:



2. Sepal Length vs Sepal Width
input:

```python
#visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
idxA = 0
idxB = 1
plt.title((data.feature_names[idxA] + " vs " + data.feature_names[idxB]))
plt.xlabel(data.feature_names[idxA])
plt.ylabel(data.feature_names[idxB])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[idxA,idxB]].values
    hull = myConvexHull.myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
```

output:



Dataset Wine:

```python
data = datasets.load_wine()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.head())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

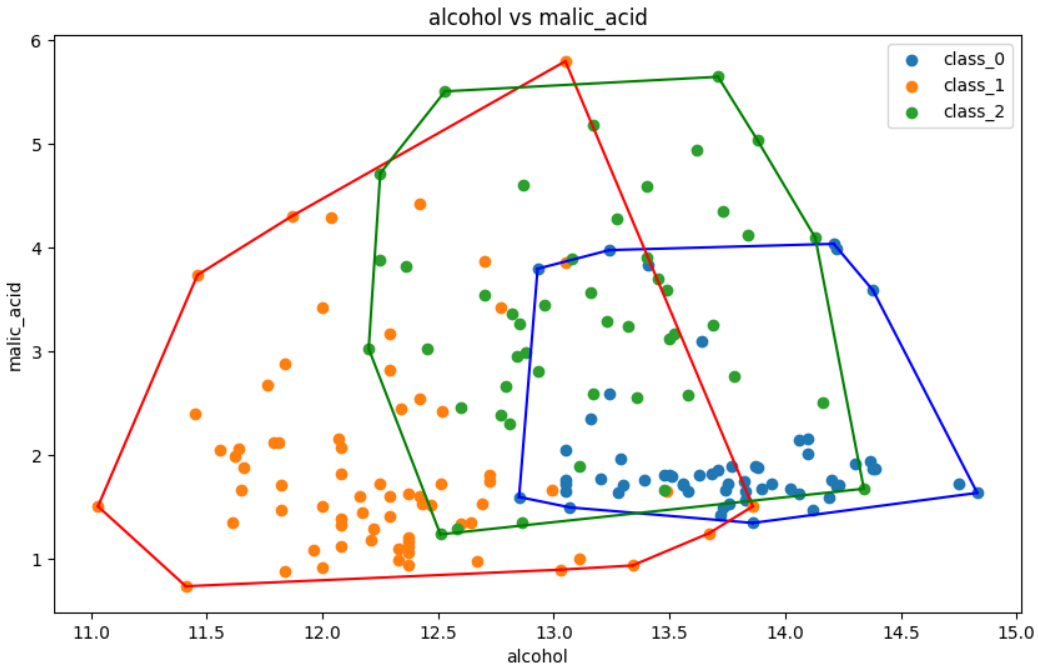
	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	...	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline	Target
0	14.23	1.71	2.43	15.6	127.0	2.80	...	2.29	5.64	1.04	3.92	1065.0	0
1	13.20	1.78	2.14	11.2	100.0	2.65	...	1.28	4.38	1.05	3.40	1050.0	0
2	13.16	2.36	2.67	18.6	101.0	2.80	...	2.81	5.68	1.03	3.17	1185.0	0
3	14.37	1.95	2.50	16.8	113.0	3.85	...	2.18	7.80	0.86	3.45	1480.0	0
4	13.24	2.59	2.87	21.0	118.0	2.80	...	1.82	4.32	1.04	2.93	735.0	0

1. Alcohol vs Malic Acid

input:

```
#visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
idxA = 0
idxB = 1
plt.title((data.feature_names[idxA] + " vs " + data.feature_names[idxB]))
plt.xlabel(data.feature_names[idxA])
plt.ylabel(data.feature_names[idxB])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[idxA,idxB]].values
    hull = myConvexHull.myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
```

output:

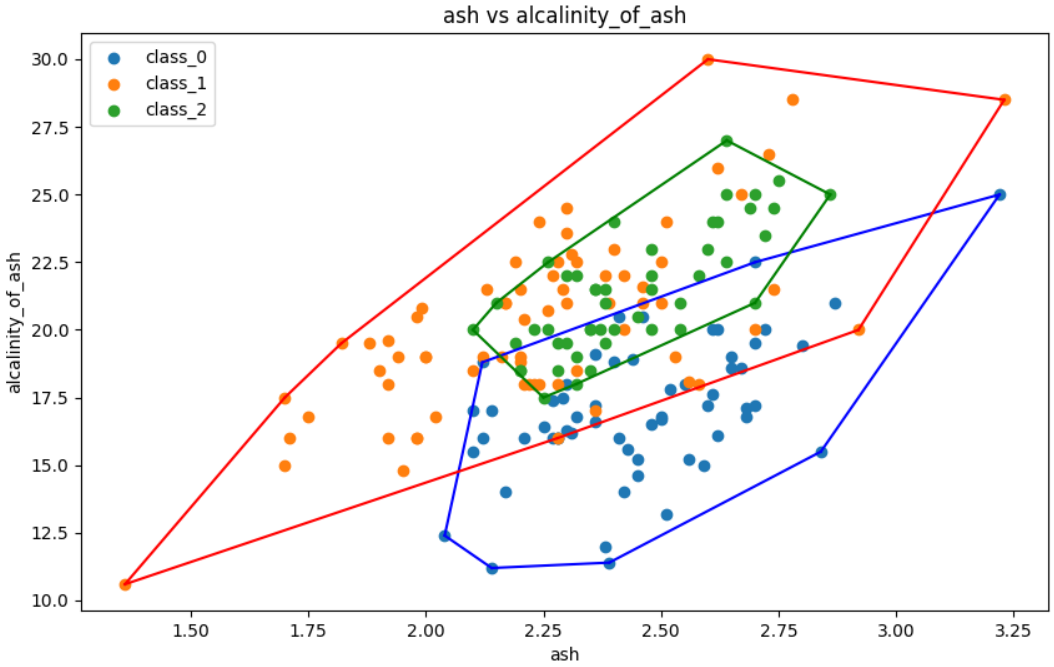


2. Ash vs Alcalinity of Ash

input:

```
#visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
idxA = 2
idxB = 3
plt.title((data.feature_names[idxA] + " vs " + data.feature_names[idxB]))
plt.xlabel(data.feature_names[idxA])
plt.ylabel(data.feature_names[idxB])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[idxA,idxB]].values
    hull = myConvexHull.myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
```

output:



Dataset Breast Cancer

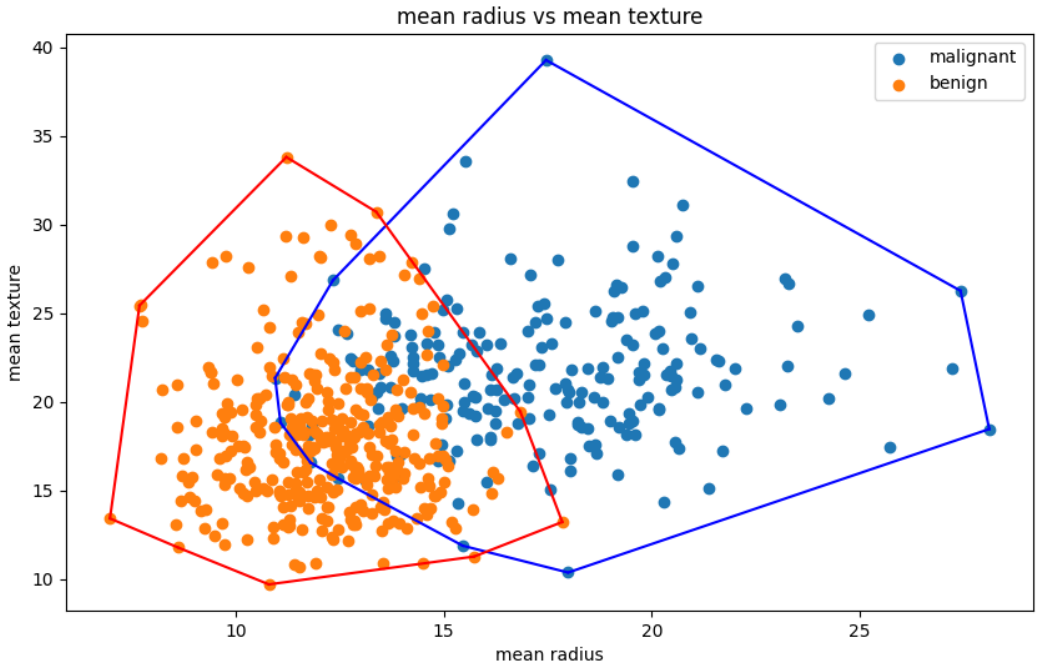
1. Mean radius vs Mean texture
   input:

```
data = datasets.load_breast_cancer()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.head())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	...	worst concavity	worst concave points	worst symmetry	worst fractal dimension	Target
0	17.99	10.38	122.80	1001.0	0.11840	...	0.7119	0.2654	0.4601	0.11890	0
1	20.57	17.77	132.90	1326.0	0.08474	...	0.2416	0.1860	0.2750	0.08902	0
2	19.69	21.25	130.00	1203.0	0.10960	...	0.4504	0.2430	0.3613	0.08758	0
3	11.42	20.38	77.58	386.1	0.14250	...	0.6869	0.2575	0.6638	0.17300	0
4	20.29	14.34	135.10	1297.0	0.10030	...	0.4000	0.1625	0.2364	0.07678	0

output:

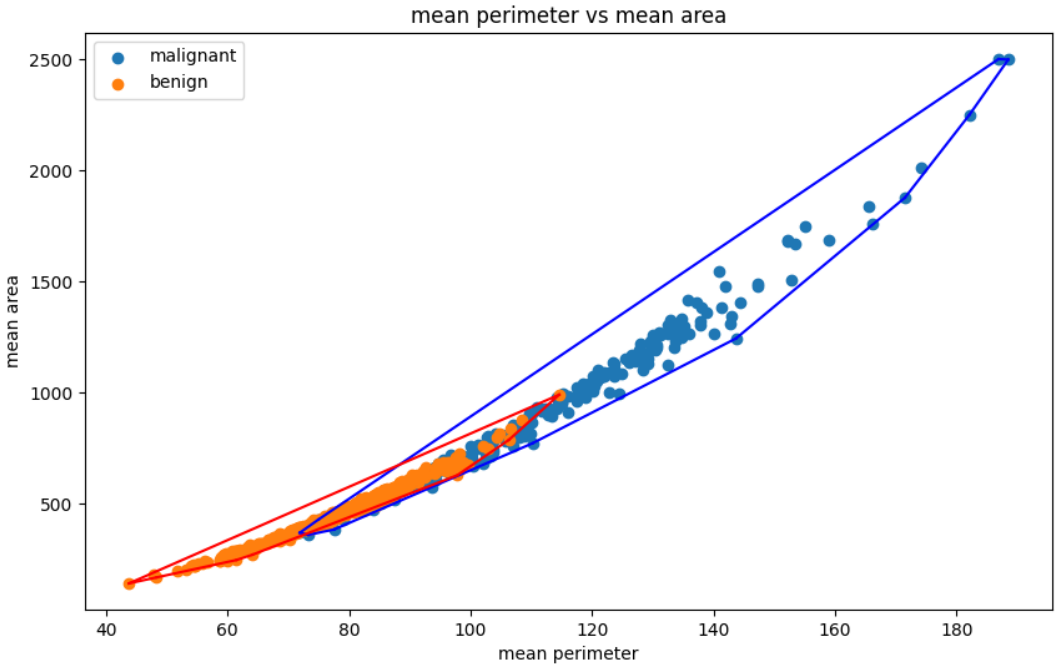


2. Mean perimeter vs Mean area

input:

```
#visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
idxA = 2
idxB = 3
plt.title((data.feature_names[idxA] + " vs " + data.feature_names[idxB]))
plt.xlabel(data.feature_names[idxA])
plt.ylabel(data.feature_names[idxB])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[idxA,idxB]].values
    hull = myConvexHull.myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
```

output:
Goog



Github Souurce Code

https://github.com/ziyaddr/TugasKecil2STIMA-13520064