IF2211 STRATEGI ALGORITMA

LAPORAN TUGAS KECIL 2
Implementasi Convex Hull untuk Visualisasi Tes *Linear Separability Dataset* dengan Algoritma *Divide and Conquer*

Oleh:

Tri Sulton Adila 13520033

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2022

A. Langkah-Langkah Algoritma *Divide and Conquer*

1. Lakukan pengurutan pada masukan array. Pengurutan didasarkan pada nilai absis menaik, apabila nilai absis sama, urutkan berdasarkan nilai ordinat menaik.
2. Ambil elemen pertama P_1 dan terakhir P_2 dari array yang telah diurut sebagai sudut pembentuk convex hull. Masukkan garis P_1P_2 dan P_2P_1 ke dalam himpunan solusi.
3. Partisi array menjadi bagian kiri (atas) dan kanan (bawah) dengan menggunakan konsep determinan.
4. Untuk setiap partisi, lakukan pencarian titik titik pembentuk convex hull dengan cara mencari titik P_3 yang merupakan titik dengan jarak terjauh dari garis P_1P_2 pada partisi kiri atau P_2P_1 pada partisi kanan.
5. Misalkan pada partisi kiri, setelah ditemukan titik P_3, masukkan P_1P_3 dan P_3P_2 ke dalam himpunan solusi dan hapuslah P_1P_2 dari himpunan solusi.
6. Lakukan pencarian convex hull pada partisi sebelah kiri dari garis P_1P_3 dan P_3P_2 dengan cara yang sama mulai dari langkah ke-3.
7. Lakukan proses yang sama pada partisi kanan dari garis P_2P_1.
8. Pencarian titik-titik convex hull pada tiap partisi dilakukan secara rekursif.

B. Source Code Program

1. myfunctions.py

```
class myConvexHull:
 """
 Instantiate a convex hull from a given set of points.
 """

 EPSILON = 1e-10

 def __init__(self, ndarray):
  self.simplices = [] # himpunan solusi yang nantinya berisi garis
dari dua titik pembentuk convex hull
  self.array = ndarray.tolist()
  self.array.sort() # convert array to sorted list
  self.convex_hull()

 def convex_hull(self):
  """
  I.S. : himpunan solusi masih kosong
  Proses: mencari titik pembentuk convex hull
  F.S. : himpunan solusi berisi garis dari dua titik pembentuk
convex hull
  """
  # dua titik ekstrem terkiri dan terkanan misalkan A dan B
  left_point = self.array[0]
  right_point = self.array[len(self.array) - 1]
  # masukkan garis AB dan BA ke dalam himpunan solusi
  self.simplices.append([left_point, right_point])
  self.simplices.append([right_point, left_point])
  # membagi array menjadi dua partisi yang berisi titik yang berada
di kiri dan kanan garis AB dan BA
  left_partition = self.partisi(left_point, right_point,
self.array)
  right_partition = self.partisi(right_point, left_point,
self.array)
  # cari titik pembentuk convex hull dari setiap partisi
```

```
  self.find_hull(left_partition, left_point, right_point)
  self.find_hull(right_partition, right_point, left_point)

 def find_hull(self, array, pangkal, ujung):
  """
  I.S. : array berisi titik yang berada di kiri / atas garis
relatif terhadap pangkal dan ujungnya misalkan pangkal A dan ujung
B
  Proses: mencari titik terjauh dari garis relatif tersebut
misalkan titik terjauh C lalu garis AC dan CB ditambahkan ke
himpunan solusi
    lalu melakukan partisi untuk titik yang berada pada kiri garis
AC dan CB, pencarian convex hull selanjutnya
    dilakukan dengan rekursif untuk setiap partisi
  F.S. : himpunan solusi telah berisi semua garis pembentuk convex
hull

  :param array: titik titik yang berada di kiri / atas garis
relatif terhadap pangkal dan ujungnya
  :param pangkal: titik pangkal garis
  :param ujung: titik ujung garis
  """
  if(len(array) == 0): # base case, titik C sebelum pemanggilan
fungsi ini merupakan titik pembentuk convex hull terakhir pada
partisi
   return
  else:
   # titik terjauh dari garis relatif tersebut misalkan C
   farrest_point = self.farrest_point(array, pangkal, ujung)
   # ganti garis AB dengan AC dan CB
   self.simplices = [x for x in self.simplices if x != [pangkal,
ujung]]
   self.simplices.append([pangkal, farrest_point]) # garis AC
   self.simplices.append([farrest_point, ujung])   # garis CB
   # hapus titik C sebelum di partisi karena sudah merupakan titik
pembentuk convex hull
   array = [x for x in array if x != farrest_point]
   # partisi array
   left_partition = self.partisi(pangkal, farrest_point, array) #
titik titik di kiri garis AC
   right_partition = self.partisi(farrest_point, ujung, array) #
titik titik di kanan garis BC atau di kiri CB
   # rekursif untuk setiap partisi
   self.find_hull(left_partition, pangkal, farrest_point)
   self.find_hull(right_partition, farrest_point, ujung)

 def farrest_point(self, array, p1, p2):
  """
  :param array: array 2 dimensi berisi banyak titik
  :param p1: titik p1
  :param p2: titik p2
  :return: titik terjauh yang berada di array dari garis p1p2
  """
  max_distance = 0
  farrest_point = None
  for p in array:
   d = self.distance(p, p1, p2)
   if(d > max_distance):
    max_distance = d
    farrest_point = p
  return farrest_point
```

```
 def partisi(self, pangkal, ujung, array):
  """
  :param pangkal: titik pangkal garis misalkan A
  :param ujung: titik ujung garis misalkan B
  :param array: array 2 dimensi berisi banyak titik di antara garis
AB

  mengembalikan array 2 dimensi yang berisi titik yang berada di
kiri / atas garis relatif terhadap AB
  """
  partition_array = []
  for i in range(len(array)):
   det = self.determinant(pangkal, ujung, array[i])
   if(det > 0 and abs(det) > self.EPSILON):
    partition_array.append(array[i])
  return partition_array

 def distance(self, p0, p1, p2):
  """
  return distance from p0 to line formed by p1 and p2
  """
  X = 0
  Y = 1
  return abs((p2[X] - p1[X]) * (p1[Y] - p0[Y]) - (p1[X] - p0[X]) *
(p2[Y] - p1[Y])) / (((p2[X] - p1[X]) ** 2 + (p2[Y] - p1[Y]) ** 2)
** 0.5)

 def determinant(self, p1, p2, p3):
  """
  :param p1: array 2 dimensi
  :param p2: array 2 dimensi
  :param p3: array 2 dimensi
  :return: determinant
  """
  X = 0
  Y = 1
  return (p1[X] * p2[Y] + p2[X] * p3[Y] + p3[X] * p1[Y]) - (p1[Y] *
p2[X] + p2[Y] * p3[X] + p3[Y] * p1[X])
```

2. test_myConvexHull.ipynb

Berikut merupakan salah satu source code untuk melakukan pengujian myConvexHull. Tes ini untuk menampilkan convex hull grafik sepal widht vs sepal length pada dataset iris. Adapun untuk tes lainnya tidak akan ditampilkan karena source code yang digunakan sama saja, hanya mengubah dataset dan indeks kolom.

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

data = datasets.load_iris()

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

#visualisasi hasil myConvexHull
plt.figure(figsize = (10, 6))
```

```
colors = ['b','r','g']
plt.title('Sepal Width vs Sepal Length Using myConvexHull')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])

# bucket adalah array yang berisi data yang akan dihitung
# MyConvexHull menerima input berupa array 2 dimensi yaitu bucket
# hull.simplices adalah array yang berisi ordianat dan absis
antardua titik yang membentuk convex hull

for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1],
label=data.target_names[i])
    # simplex berisi ordinat dan absis antar titik yang membentuk
convex hull
    for simplex in hull.simplices:
        plt.plot([simplex[0][0], simplex[1][0]], [simplex[0][1],
simplex[1][1]], colors[i])
plt.legend()

#visualisasi hasil ConvexHull
from scipy.spatial import ConvexHull

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Sepal Width vs Sepal Length Using scipy')
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])
    # simplex berisi ordinat dan absis antar titik yang membentuk
convex hull
    for simplex in hull.simplices:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

C. Screenshoot Hasil Uji

Datasets yang digunakan pada uji ini adalah sebagai berikut.

1. Iris datasets
2. Wine datasets
3. Breast Cancer datasets

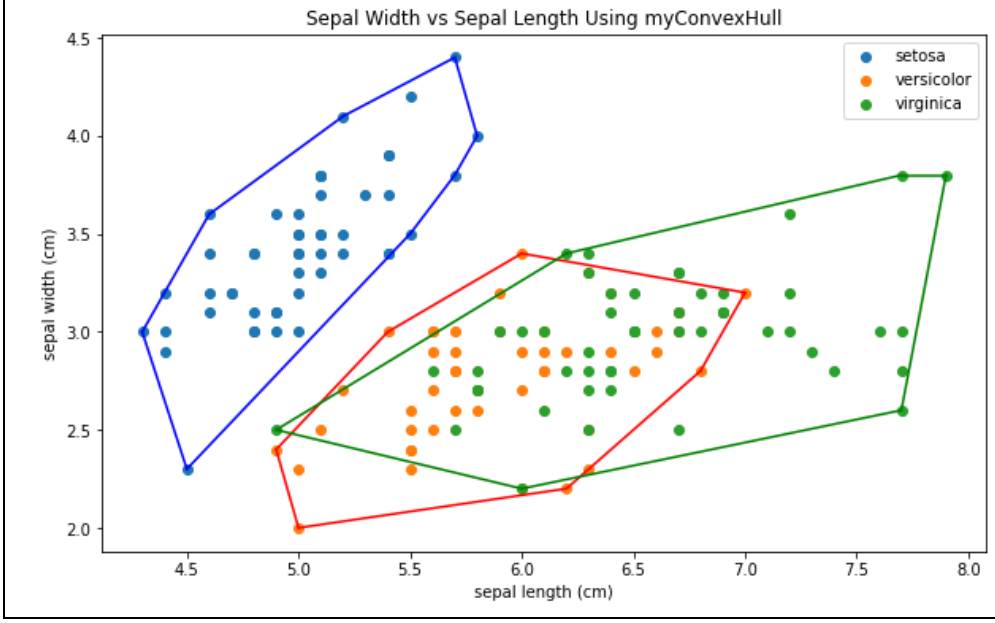
Untuk setiap dataset akan diberikan satu atau dua test convex hull dan untuk setiap test akan diberikan plot convex hull dengan menggunakan library myConvexHull dilanjutkan dengan menggunakaan library spicy sebagai perbandingan.

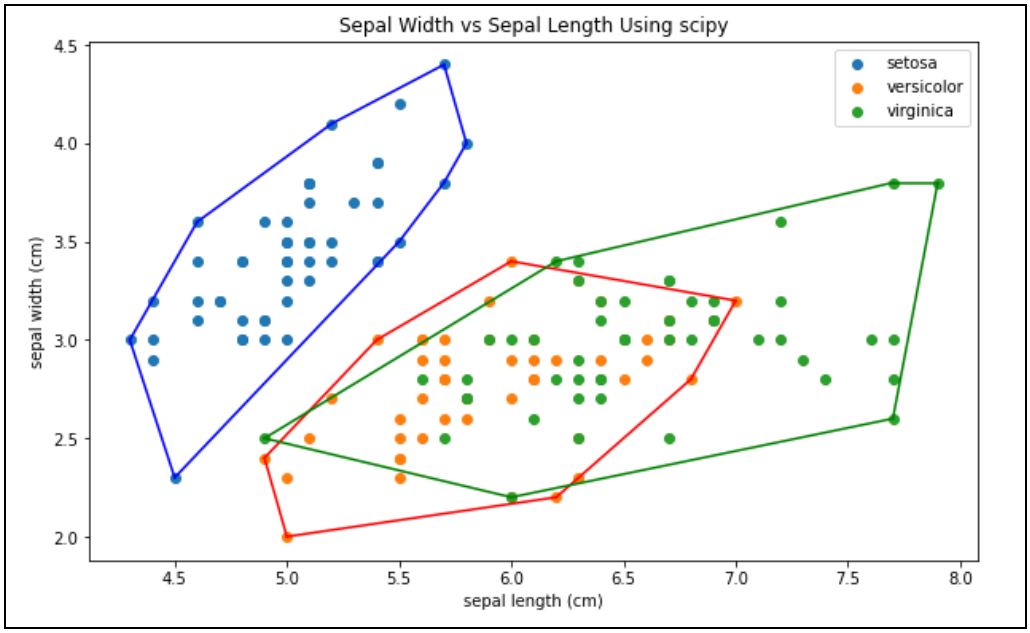
1. Iris Datasets
   - sepal width vs sepal length

masukan

```
data = datasets.load_iris()
bucket = bucket.iloc[:,[0,1]].values
```

keluaran





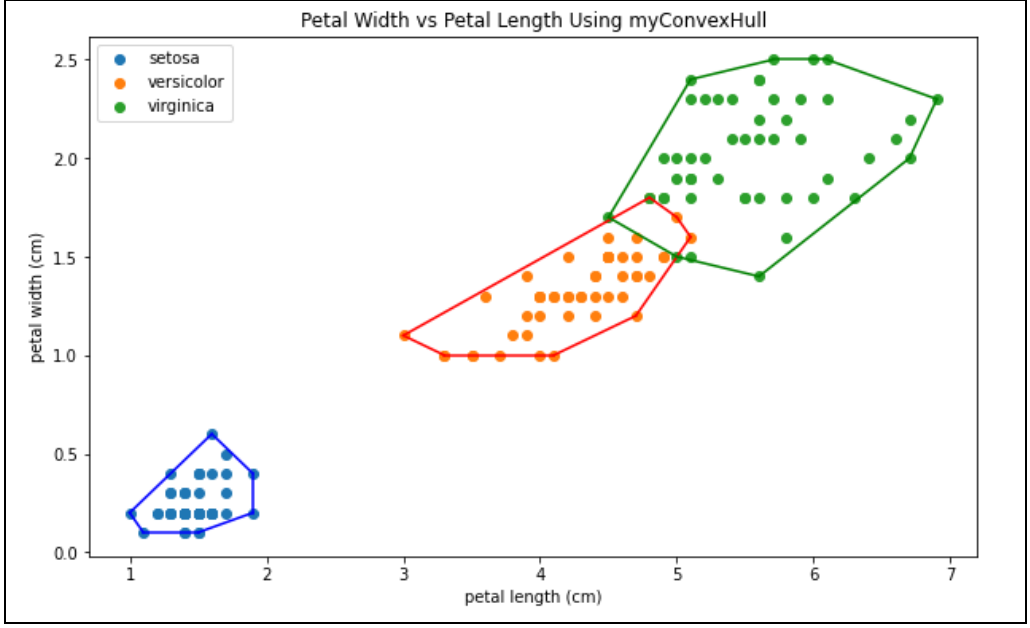
- petal width vs petal length

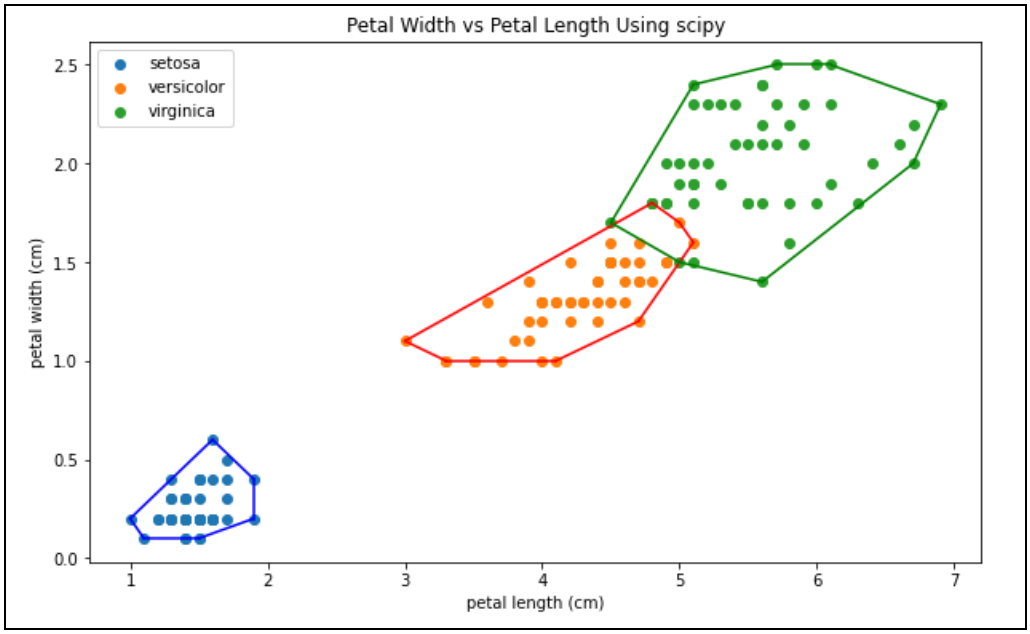
masukan

```
data = datasets.load_iris()
```

```
bucket = bucket.iloc[:,[2,3]].values
```

keluaran





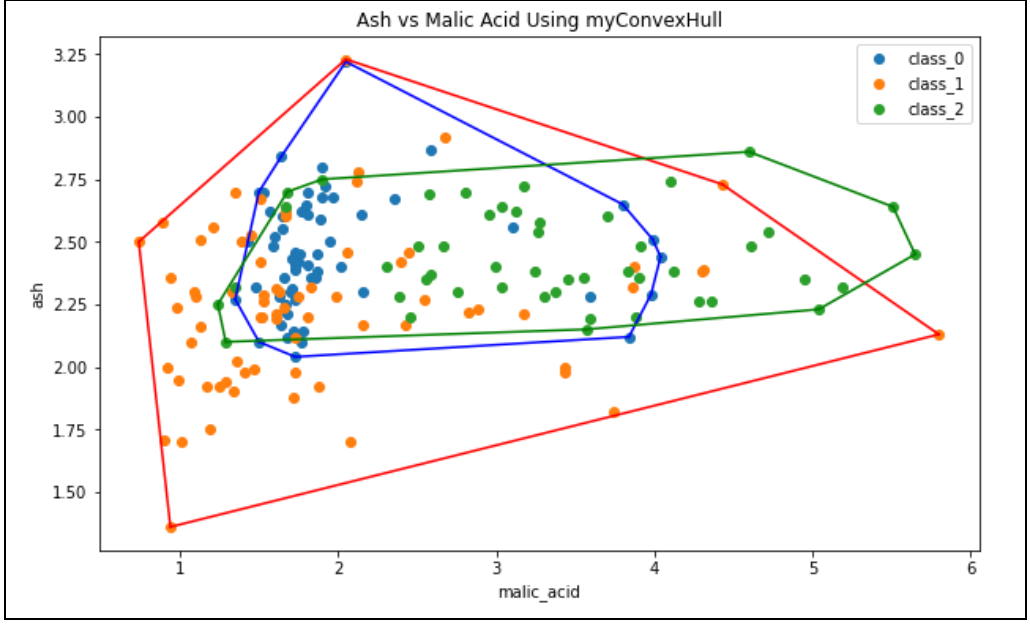
2. Wine Datasets

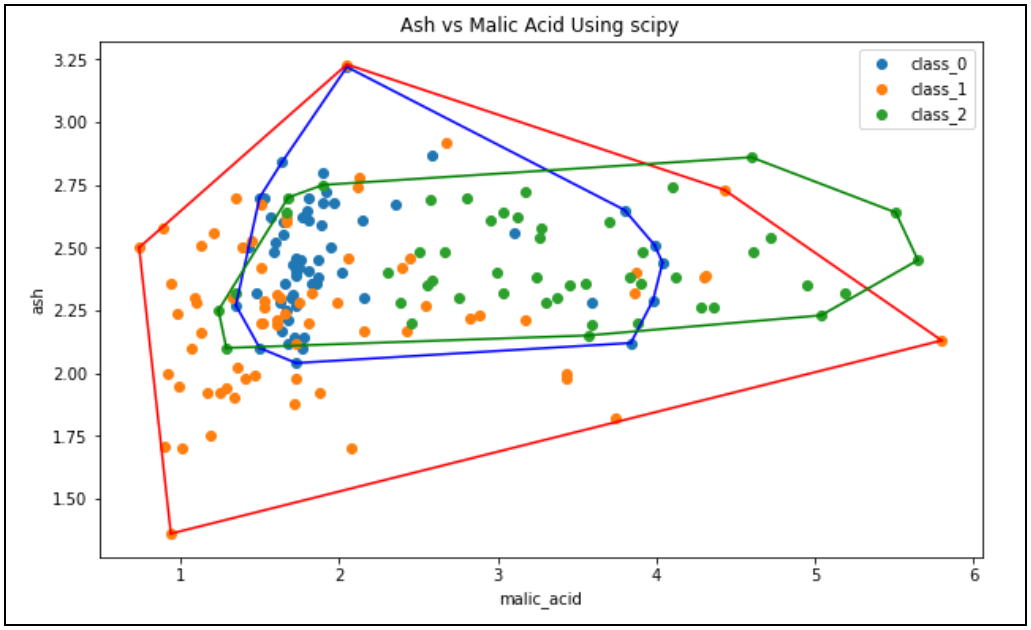
- ash vs malic acid

masukan

```
data = datasets.load_wine()
bucket = bucket.iloc[:,[1,2]].values
```

keluaran



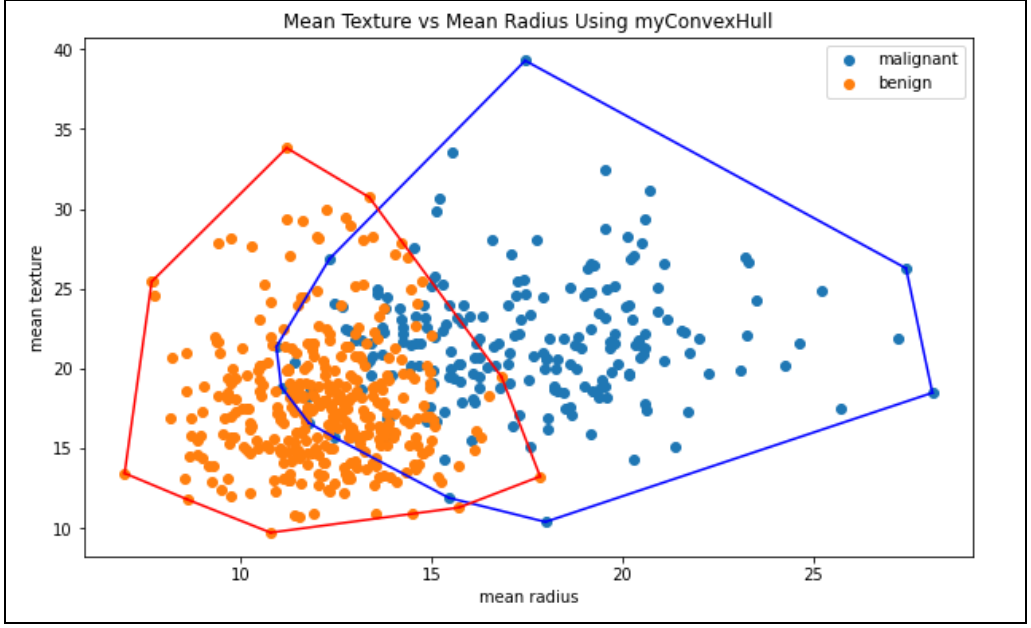


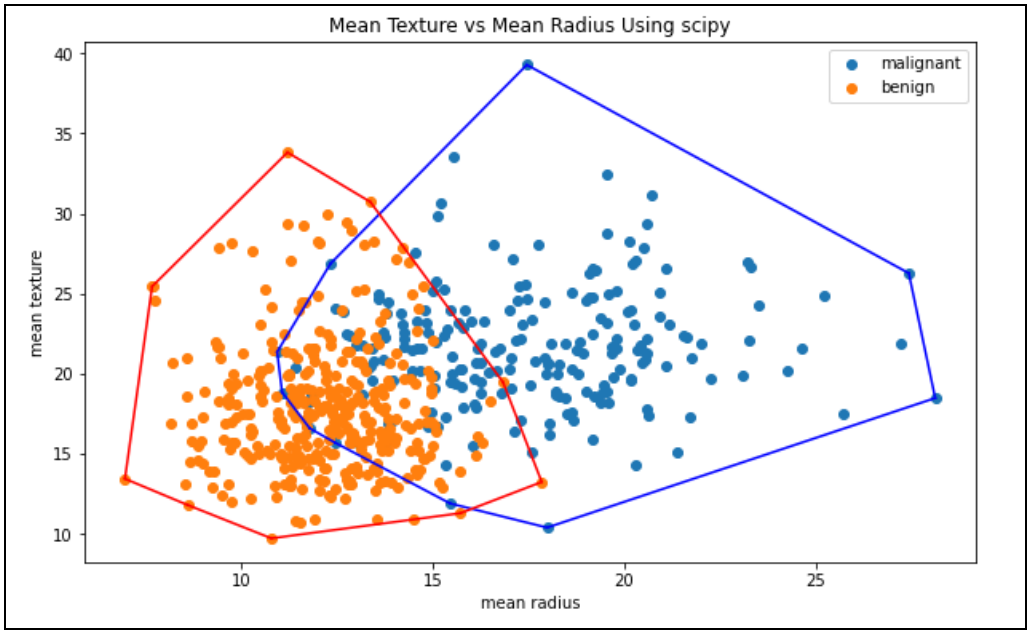
3. Breast Cancer Datasets
   - mean texture vs mean radius

masukan

```
data = datasets.load_breast_cancer()
bucket = bucket.iloc[:,[0,1]].values
```

keluaran





D. Alamat Github

https://github.com/3sulton/myConvexHull

E. Tabel

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	√	