Laporan Tugas Kecil -2
IF2211 - Strategi Algoritma

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

Nama : Ghebyon Tohada Nainggolan

NIM : 13520079

Kelas : K-01

Bahasa : Python

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021

A. ALGORITMA DIVIDE AND CONQUER PADA *CONVEX HULL*

Divide and Conquer adalah algoritma yang mereduksi sebuah persoalan yang besar menjadi dua buah sub persoalan yang memiliki solusi yang mirip dengan persoalan semula, namun berukuran lebih kecil. Kemudian memecahkan masing-masing sub persoalan tersebut secara rekursif dan menggabungkan solusinya sehingga membentuk solusi masalah semula.

Pada Tugas Kecil II IF2211 Strategi Algoritma ini, permasalahan mencari kumpulan titik 'terluar' yang membentuk *Convex Hull* dapat dilakukan dengan menggunakan algoritma Divide and Conquer.

Langkah-langkah :

1. Inisialisasi
   - S : himpunan titik sebanyak n, dengan n > 1, yaitu titik p1(x1, y1) sehingga pn(xn,yn) pada bidang Cartesian dua dimensi
   - Urutkan titik-titik pada himpunan S berdasarkan nilai absis yang menaik, dan jika ada nilai absis yang sama, maka diurutkan dengan nilai ordinat yang menaik.
   - Tentukan dua titik yang memiliki jarak terjauh, kedua titik tersebut akan menjadi pondasi awal dalam pembentukan *convex hull*
2. Divide and Conquer
   - Garis yang menghubungkan dua titik ekstrim sebut saja p1 dan pn membagi titik S menjadi dua bagian yaitu S1(kumpulan titik di sebelah kiri garis p1pn) dan S2(kumpulan titik di sebelah kanan garis p1,pn)
   - Titik yang berada pada garis p1,pn tidak akan menjadi solusi dari *Convex Hull*
   - Untuk salah satu bagian (misal S1), terdapat dua kemungkinan :
     - Basis : Jika tidak ada titik pada S1, maka p1 dan pn menjadi pembentuk *Convex Hull*
     - Rekurens : Jika terdapat titik pada S1, pilih sebuah titik yang memiliki jarak terjauh dari garis p1pn (misalkan p max). Jika terdapat beberapa titik dengan jarak yang sama, pilih sebuah titik yang memaksimalkan sudut pmax p1 pn.

     note : Titik yang berada di dalam daerah segitiga diabaikan
   - Tentukan kumpulan titik yang berada di sebelah kiri garis p1 pmax menjadi bagian S1,1 dan di sebelah kiri pmax pn menjadi bagian S1,2
   - Lakukan hal yang sama pada(butir 4 dan 5) pada bagian S2, hingga didapat, bagian terluar dari garis yang dibentuk adalah kosong.
   - Hasil yang didapat adalah pasangan titik yang membentuk titik terluar dari *Convex Hull*

B. SOURCE CODE PROGRAM

Library MyConvexHull

```
import math
import numpy as np
import matplotlib.pyplot as plt

# Menghitung Jarak antara titik a dan b pada koordinat
def pythagoras(a,b):
    return math.sqrt(a**2+b**2)

# Menghitung jarak antara titik x0, y0 dengan garis yang dibentuk oleh dua titik x1,y1 dan
x2,y2
def distanceBtwnLineAndPoint(x1,y1,x2,y2,x0,y0):
    return abs((x2-x1)*(y1-y0) - (x1-x0)*(y2-y1)) / np.sqrt(np.square(x2-x1) +
np.square(y2-y1))

# Menghitung besar sudut yang dibentuk oleh titik B(x,y) A(x,y) C(x3,y3) dengan vertex
A(x,y)
def lawOfCosines(Ax,Ay,Bx,By,Cx,Cy):
    p12 = pythagoras(Ax-Bx,Ay-By)
    p13 = pythagoras(Ax-Cx,Ay-Cy)
    p23 = pythagoras(Bx-Cx,By-Cy)
    return math.degrees(math.acos((p12**2 + p13**2 - p23**2) / (2*p12*p13)))

# Kelas myConvexHull
# Mengandung method yang digunakan untuk menentukan titik mana saja pada atribut item yang
merupakan
# bagian dari convex hull.
class myConvexHull:
    def __init__(self, myPoints):
        self.myPoints = myPoints[np.lexsort((myPoints[:,1],myPoints[:,0]))]
        self.convexHullRelation = []
        distance = 0.0
        p1 = -1
        p2 = -1
        for i in range (self.myPoints.shape[0]):
            for j in range (i+1,len(self.myPoints)):
                currentDistance = pythagoras(self.myPoints[j][0] - self.myPoints[i][0],
                                             self.myPoints[j][1] - self.myPoints[i][1])
                if (currentDistance > distance):
                    distance = currentDistance
                    p1 = i
                    p2 = j

        topCandidate = []
        bottomCandidate = []
        x1 = self.myPoints[p1][0] ; y1 = self.myPoints[p1][1]
        x2 = self.myPoints[p2][0] ; y2 = self.myPoints[p2][1]
        for i in range (self.myPoints.shape[0]):
            x3 = self.myPoints[i][0] ; y3 = self.myPoints[i][1]
            tempMatrix = np.array([[x1,y1,1],
                                   [x2,y2,1],
                                   [x3,y3,1]])
```

```
            detTempMatrix = np.linalg.det(tempMatrix)
            if (detTempMatrix > 0):
                topCandidate.append(i)
            elif (detTempMatrix < 0):
                bottomCandidate.append(i)
        self.DnC(p1,p2,topCandidate)
        self.DnC(p2,p1,bottomCandidate)

    def DnC(self,point1, point2,candidateSet):

        topCandidate = []
        x1 = self.myPoints[int(point1)][0] ; y1 = self.myPoints[int(point1)][1]
        x2 = self.myPoints[int(point2)][0] ; y2 = self.myPoints[int(point2)][1]
        farthestDistance = -1
        farthestPoint = -1
        for i in candidateSet:
            if (i != point1 and i!= point2 and i != farthestPoint):
                x3 = self.myPoints[i][0] ; y3 = self.myPoints[i][1]
                tempMatrix = np.array([[x1,y1,1],
                                       [x2,y2,1],
                                       [x3,y3,1]])
                detTempMatrix = np.linalg.det(tempMatrix)
                if (detTempMatrix > 0):
                    topCandidate.append(i)
                    tempDistance = distanceBtwnLineAndPoint(x1,y1,x2,y2,x3,y3)
                    if (tempDistance > farthestDistance):
                        farthestDistance = tempDistance
                        farthestPoint = i
                    elif (tempDistance == farthestDistance and (x3 !=
                           self.myPoints[farthestPoint][0] and y3 !=
                           self.myPoints[farthestPoint][1])):
                        tempAngle = lawOfCosines(x1,y1,x3,y3,x2,y2)
                        farthestAngel = lawOfCosines(x1,y1,self.myPoints[farthestPoint][0],
                                                     self.myPoints[farthestPoint][1],x2,y2)
                        if (tempAngle > farthestAngel):
                            farthestDistance = tempDistance
                            farthestPoint = i

        if (farthestPoint == -1):
            self.convexHullRelation.append([int(point1),int(point2)])
        else:
            self.DnC(int(point1),farthestPoint,topCandidate)
            self.DnC(farthestPoint,int(point2),topCandidate)
```

Import dan Random Color

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

colors = ["#"+''.join([random.choice('0123456789ABCDEF') for j in range(6)])
          for i in range(100)]
```

Implementasi pada Dataset Iris

```
# Load Dataset
dataIris = datasets.load_iris()
df = pd.DataFrame(dataIris.data, columns=dataIris.feature_names)
df['Target'] = pd.DataFrame(dataIris.target)
```

```
# Plot
plt.figure(figsize = (10, 6))
plt.title('Sepal Width vs Sepal Length')
plt.xlabel(dataIris.feature_names[0])
plt.ylabel(dataIris.feature_names[1])
for i in range(len(dataIris.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    hull = myConvexHull(bucket)
    plt.scatter(hull.myPoints[:, 0], hull.myPoints[:, 1], label=dataIris.target_names[i],
color=colors[i])
    for simplex in hull.convexHullRelation:
        plt.plot(hull.myPoints[simplex, 0], hull.myPoints[simplex, 1], colors[i])
plt.legend()
```

```
# Plot
plt.figure(figsize = (10, 6))
plt.title('Petal Width vs Petal Length')
plt.xlabel(dataIris.feature_names[2])
plt.ylabel(dataIris.feature_names[3])
for i in range(len(dataIris.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[2,3]].values
    hull = myConvexHull(bucket)
    plt.scatter(hull.myPoints[:, 0], hull.myPoints[:, 1], label=dataIris.target_names[i],
color=colors[i])
    for simplex in hull.convexHullRelation:
        plt.plot(hull.myPoints[simplex, 0], hull.myPoints[simplex, 1], colors[i])
plt.legend()
```

Implementasi pada Dataset Digits

```
# Load dataset
dataDigits = datasets.load_digits()
df = pd.DataFrame(dataDigits.data, columns=dataDigits.feature_names)
df['Target'] = pd.DataFrame(dataDigits.target)
```

```
# Plot
plt.figure(figsize = (10, 6))
plt.title('Pixel 3 vs Pixel 4')
plt.xlabel(dataDigits.feature_names[3])
plt.ylabel(dataDigits.feature_names[4])
for i in range(len(dataDigits.target_names)):
    bucket = df[df['Target'] == i]
```

```
    bucket = bucket.iloc[:,[3,4]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=dataDigits.target_names[i],
color=colors[i])
    for simplex in hull.convexHullRelation:
        plt.plot(hull.myPoints[simplex, 0], hull.myPoints[simplex, 1], colors[i])
plt.legend()
```

```
# Plot
dataDigits = datasets.load_digits()
df = pd.DataFrame(dataDigits.data, columns=dataDigits.feature_names)
df['Target'] = pd.DataFrame(dataDigits.target)
plt.figure(figsize = (10, 6))
plt.title('Pixel 3 vs Pixel 5')
plt.xlabel(dataDigits.feature_names[3])
plt.ylabel(dataDigits.feature_names[5])
for i in range(len(dataDigits.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[3,5]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=dataDigits.target_names[i],
color=colors[i])
    for simplex in hull.convexHullRelation:
        plt.plot(hull.myPoints[simplex, 0], hull.myPoints[simplex, 1], colors[i])
plt.legend()
```

Implementasi pada Dataset Wine

```
# Load Dataset
dataWine = datasets.load_wine()
df = pd.DataFrame(dataWine.data, columns=dataWine.feature_names)
df['Target'] = pd.DataFrame(dataWine.target)
```

```
# Plot
plt.figure(figsize = (10, 6))
plt.title('Alcohol vs Malic_acid')
plt.xlabel(dataWine.feature_names[0])
plt.ylabel(dataWine.feature_names[1])
for i in range(len(dataWine.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=dataWine.target_names[i],
color=colors[i])
    for simplex in hull.convexHullRelation:
        plt.plot(hull.myPoints[simplex, 0], hull.myPoints[simplex, 1], colors[i])
plt.legend()
```

```
# Plot
plt.figure(figsize = (10, 6))
plt.title('Color Intensity vs Hue')
plt.xlabel(dataWine.feature_names[9])
plt.ylabel(dataWine.feature_names[10])
for i in range(len(dataWine.target_names)):
```

```
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[9,10]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=dataWine.target_names[i],
color=colors[i])
    for simplex in hull.convexHullRelation:
        plt.plot(hull.myPoints[simplex, 0], hull.myPoints[simplex, 1], colors[i])
plt.legend()
```

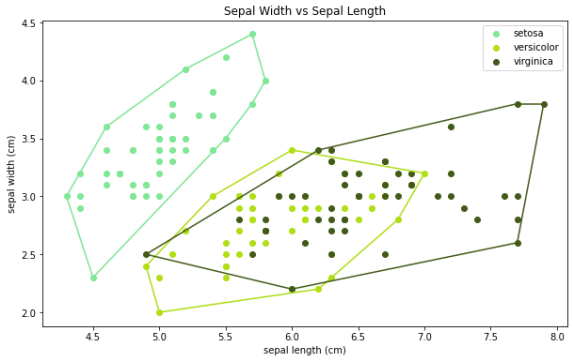
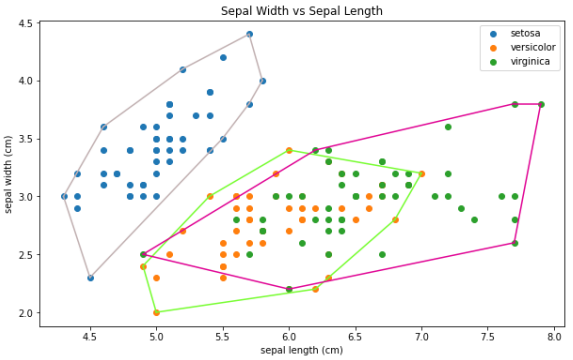
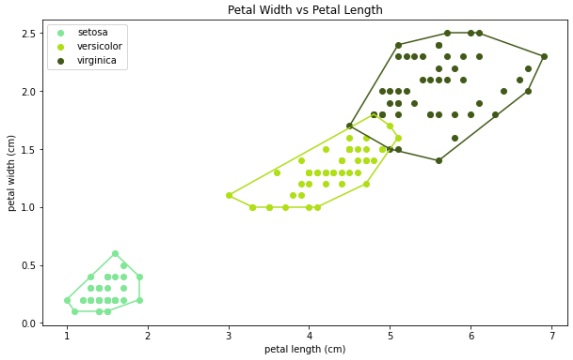
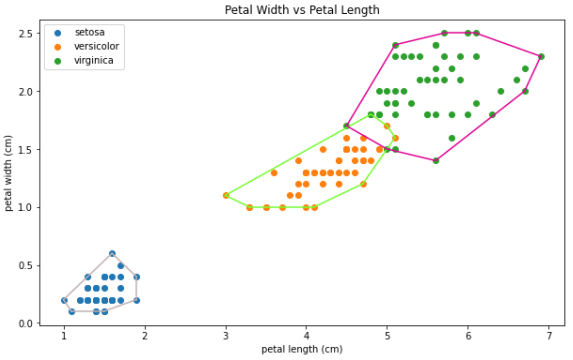
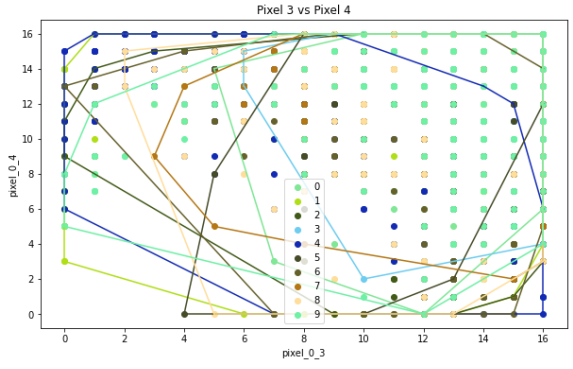
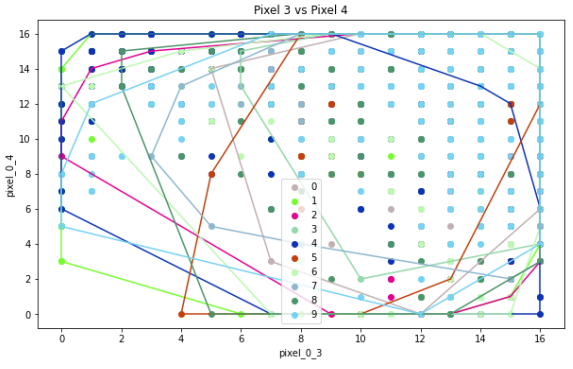
Implementasi pada Dataset Breast Cancer

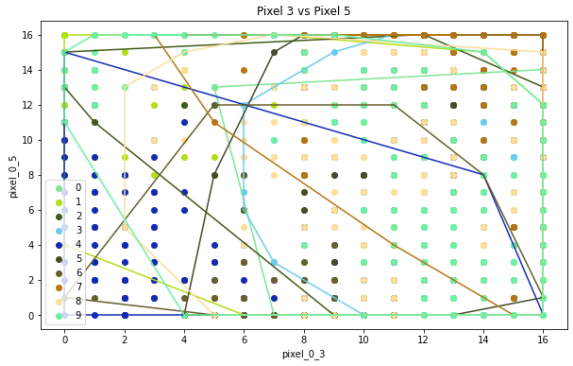
```
# Load Dataset
dataBreast = datasets.load_breast_cancer()
df = pd.DataFrame(dataBreast.data, columns=dataBreast.feature_names)
df['Target'] = pd.DataFrame(dataBreast.target)
```

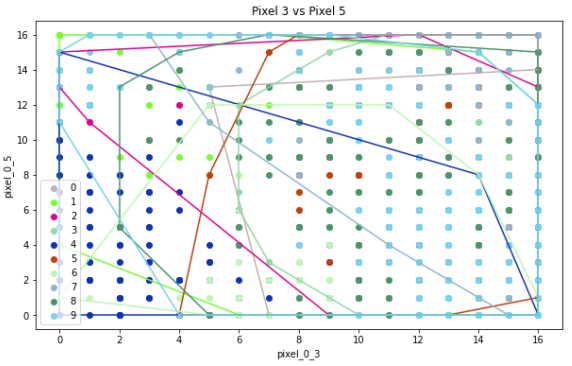
```
# Plot
plt.figure(figsize = (10, 6))
plt.title('Mean Radius vs Mean Texture')
plt.xlabel(dataBreast.feature_names[0])
plt.ylabel(dataBreast.feature_names[1])
for i in range(len(dataBreast.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=dataBreast.target_names[i],
color=colors[i])
    for simplex in hull.convexHullRelation:
        plt.plot(hull.myPoints[simplex, 0], hull.myPoints[simplex, 1], colors[i])
plt.legend()
```

```
# Plot
plt.figure(figsize = (10, 6))
plt.title('Mean Textur vs Mean Smoothness')
plt.xlabel(dataBreast.feature_names[1])
plt.ylabel(dataBreast.feature_names[4])
for i in range(len(dataBreast.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[1,4]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=dataBreast.target_names[i],
color=colors[i])
    for simplex in hull.convexHullRelation:
        plt.plot(hull.myPoints[simplex, 0], hull.myPoints[simplex, 1], colors[i])
plt.legend()
```

C. INPUT DAN OUTPUT

<table>
<tr><th colspan="2">Input : Dataset Iris</th></tr>
<tr><th>Output MyConvexHull</th><th>Output ConvexHull</th></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><th colspan="2">Input : Dataset Digits</th></tr>
<tr><th>Output MyConvexHull</th><th>Output ConvexHull</th></tr>
<tr><td> 
</td><td> 
</td></tr>
</table>

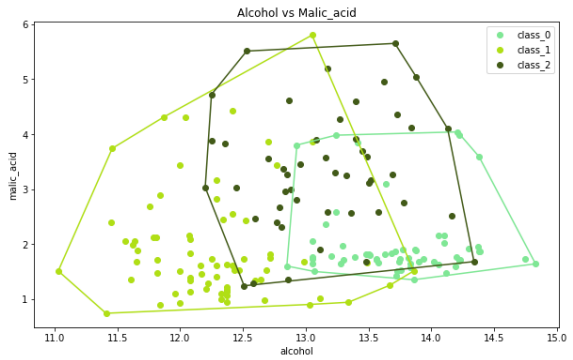


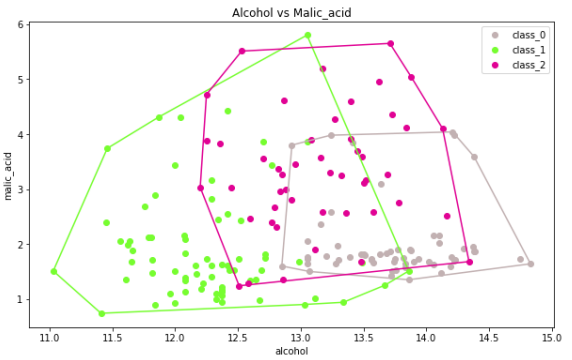


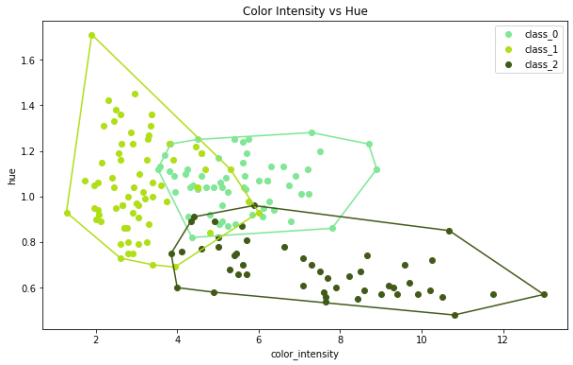
Input : Dataset Wine

Output MyConvexHull

Output ConvexHull



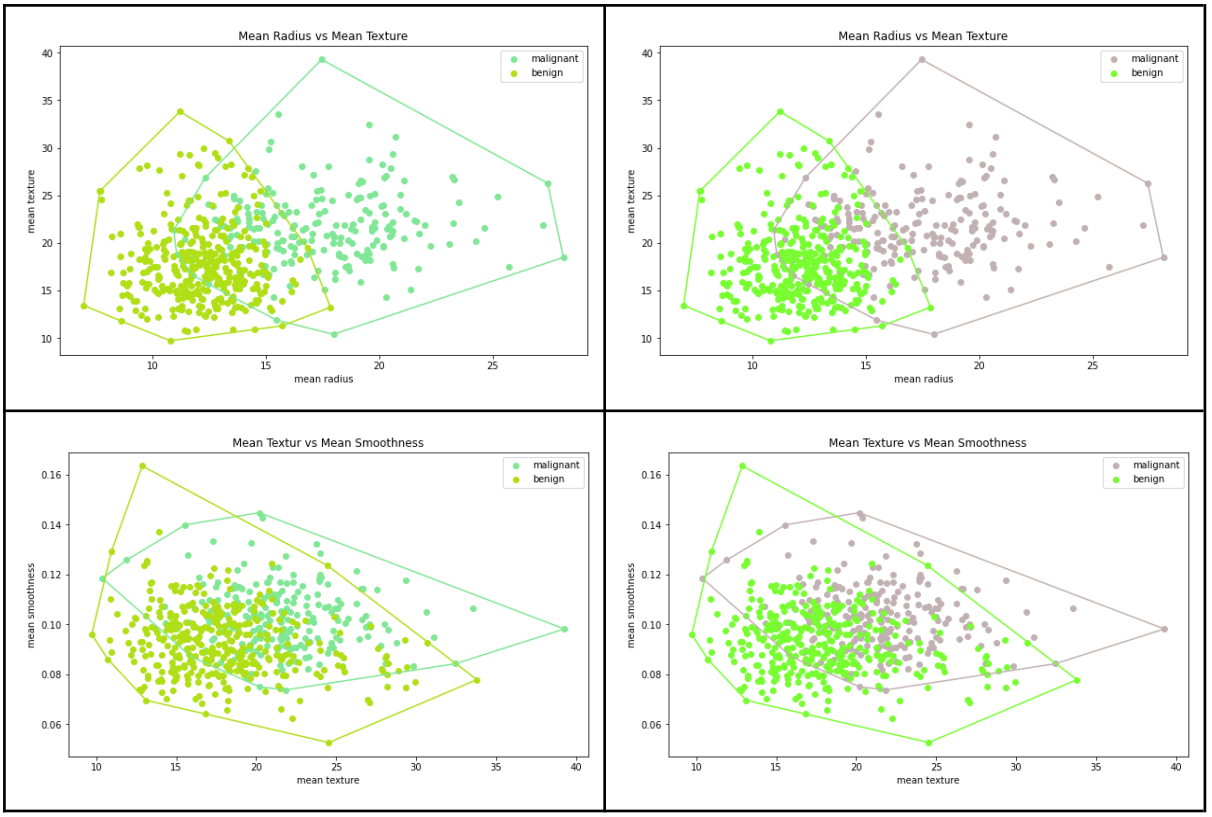




Input : Dataset Breast Cancer

Output MyConvexHull

Output ConvexHull


Mean Radius vs Mean Texture
malignant
benign
mean texture
mean radius
Mean Radius vs Mean Texture
malignant
benign
mean texture
mean radius
Mean Textur vs Mean Smoothness
malignant
benign
mean smoothness
mean texture
Mean Texture vs Mean Smoothness
malignant
benign
mean smoothness
mean texture

D. GITHUB

https://github.com/ghebyon/Tucil02-Stima

E. PENILAIAN MANDIRI

Poin	Ya	Tidak
1. Pustaka *myConvexHull* berhasil dibuat dan tidak ada kelasahan	✓	
2. Convex hull yang dihasilkan sudah benar	✓	
3. Pustaka *myConvexHull* dapat digunkan untuk menampilkan *convex hull* setiap label dengan warna yang berbeda	✓	
4. Bonus : program dapat menerima input dan menuliskan output untuk dataset lainnya	✓	