LAPORAN TUGAS KECIL II

IMPLEMENTASI CONVEX HULL UNTUK VISUALISASI TES LINEAR SEPARABILITY DATASET DENGAN ALGORITMA DIVIDE AND CONQUER

Laporan dibuat untuk memenuhi salah satu tugas mata kuliah

IF2211 Strategi Algoritma

Disusun oleh:

Kristo Abdi Wiguna 13520058

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

DAFTAR ISI

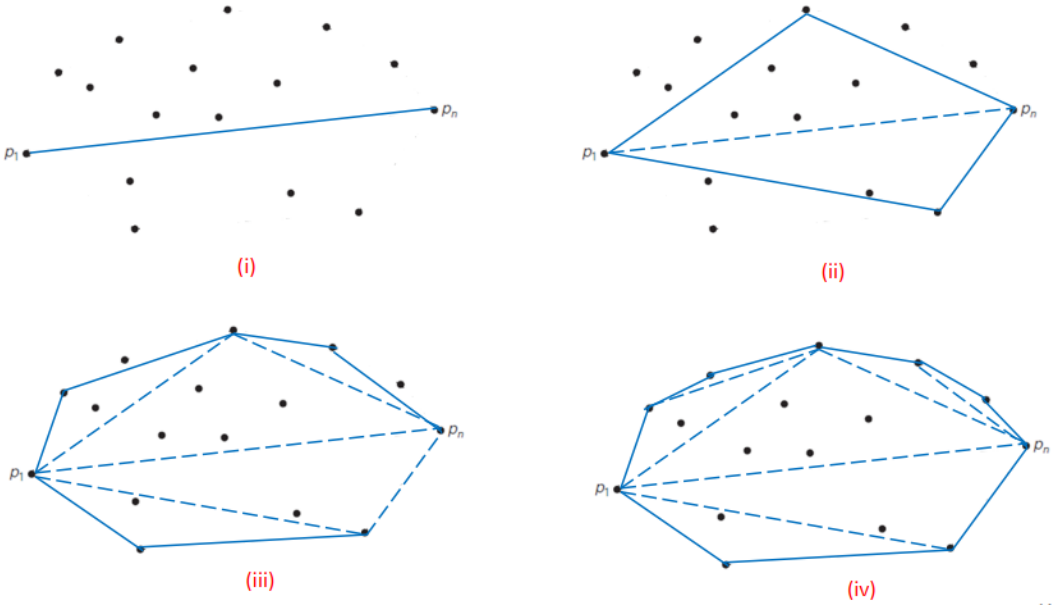
Algoritma Divide and Conquer

Algoritma penyelesaian convex hull dilakukan menggunakan algoritma Quickhull. Pertama – tama, untuk mencari titik P1 dan Pn sebagai pembagi dua wilayah S1 dan S2 maka perlu mengurutkan secara menaik titik – titik berdasarkan absisnya. Lalu mengambil titik pertama sebagai P1 dan titik terakhir sebagai Pn. Setelah itu memasuki langkah rekursif dimana membentuk garis P1 ke Pn dan mencari titik paling jauh dari garis tersebut di area titik – titik sisi kiri garis maupun area titik – titik sisi kanan garis.

Langkah tersebut dilakukan terus menerus hingga tidak ada titik lagi di sisi yang dicari. Ketika mencapai tidak ada lagi titik, maka akan menambahkan pasangan titik Pmax sebagai titik terjauh dengan P1 dan pasangan titik Pmax dengan Pn di *edgeList* serta menambahkan titik yang sudah dilalui di *res* dilakukan menaik secara rekursi yang telah dipanggil hingga kondisi akhir akan penuh dengan daftar pasangan titik yang koresponden membentuk sebuah garis sisi convex hull.

Setelah itu, edgeList yang sudah penuh dengan daftar pasangan sisi convex hull akan diconvert menjadi *edges* : sebuah List 1D berisi array indeks titik yang berpasangan menggunakan fungsi *getIndices* dan mempassing *edges* tersebut dan di plot oleh library matplotlib.

Langkah mengurutkan titik – titik secara menaik berdasarkan absis memiliki kompleksitas O(nlogn) dengan n sebagai jumlah titik yang ada. Mencari titik P1 dan Pn pada array terurut tersebut memiliki kompleksitas O(1). Membagi titik – titik menggunakan variabel *side* dan mengiterasi tiap titik sekaligus mencari titik Pmax dengan kompleksitas O(n) untuk average case. Untuk mencari indeks dari sisi dari titik yang koresponden memiliki kompleksitas waktu O(n^2) untuk worst case dimana setiap titik adalah bagian dari convex hull.



Source Program

Menggunakan bahasa pemrograman Python. Library yang digunakan adalah pandas, numpy, sklearn, dan matplotlib.

File main.py

```python
import numpy as np
from searchHull import *
from util import *

def ConvexHull(points)  :
    # Fungsi untuk mencari sisi yang koresponden di ConvexHull
    # Inisialisasi array
    res = [[0]*2]*2
    edgeList = [[0]*0]*0
    if (len(points) < 2) :
        # Cek minimal ada 2 titik di points
        return
    # Mencari leftmost dan rightmost titik di points (p1,pn), dan memasukkan p1 dan pn ke dalam res array
    sortedArr = sortPoints(points)
    res[0], res[1] = sortedArr[0], sortedArr[len(sortedArr)-1]
    # Mencari titik yang membentuk convex hull dan memasukkan sisi yang terhubung ke array edgeList
    findHull(sortedArr, res[0], res[1], res, edgeList, 1)
    findHull(sortedArr, res[0], res[1], res, edgeList, -1)

    edges = getIndices(edgeList, points)
    return edges
```

File searchHull.py

```python
import numpy as np
from util import *

def getIndices(edgeList, points) :
    # Fungsi yang mengembalikan list of array berisi indeks dimana edgeList berada di points
    res = np.zeros((len(edgeList),2), dtype=int)
    ind = 0
    for i in range(len(edgeList)) :
        foundX = False
        foundY = False
        for j in range(len(points)) :
            if (edgeList[i][0][0] == points[j][0] and edgeList[i][0][1] == points[j][1]) :
                res[ind][0] = j
                foundX = True
            if (edgeList[i][1][0] == points[j][0] and edgeList[i][1][1] == points[j][1]) :
                res[ind][1] = j
                foundY = True
        if (foundX and foundY) :
            ind+=1
    return res
```

```python
def findHull(s, p, q, res, edgeList, side) :
    # Fungsi rekursif yang mencari titik terjauh dari garis yang dibentuk p dan q di sisi side (1 jika di kiri, -1 jika di kanan)
    if (len(s) == 0) :
        # Base case
        return
    # Inisialisasi variabel
    indexToAdd = None
    dist = 0.0

    for i in range(len(s)) :
        # Mencari titik terjauh di sisi "side" dengan mengiterasi dan memiliki variabel temporary untuk mencatat jarak terjauh
        val = lineDistance(p, q, s[i])
        if ((whichSide(p, q, s[i]) == side)  and (val > dist) and (not isMember(s[i], res))) :
            # Mencatat indeks dari titik yang memiliki jarak terjauh
            dist = val
            indexToAdd = i

    # Jika tidak ada titik terjauh
    if (indexToAdd is None) :
        # Base case untuk menambahkan titik yang terhubung paling luar
        res.append(p)
        res.append(q)
        pair = [p, q]
        # Memasukkan sisi ke edgeList, sisi terbentuk dari titik terjauh dan kedua ujung garis
        edgeList.append(pair)
        return

    # Titik terjauh ditemukan
    # Rekursif DnC untuk mencari titik terjauh dari garis yang dibentuk param ke 2 dan ke 3 dengan sisi(side) area yang konsisten
    findHull(s, s[indexToAdd], p, res, edgeList, -whichSide(s[indexToAdd], p, q))
    findHull(s, s[indexToAdd], q, res, edgeList, -whichSide(s[indexToAdd], q, p))
```

File util.py

```python
def isMember(arrToCheck, arr) :
    # Mengecek apakah arrToCheck adalah anggota dari list of array arr
    for i in range(len(arr)) :
        if ((arr[i][0] == arrToCheck[0]) and (arr[i][1] == arrToCheck[1])) :
            return True
    return False

def sortPoints(points) :
    # Sort based on absis (first column), jika absis sama compare ordinatnya (second column)
    points = sorted(points, key=lambda x: (x[0], x[1]))
    return points

def lineDistance(p1, pn, ptest) :
    # Mengembalikan jarak dari titik ptest ke garis yang dibentuk p1 dan pn
    val = (ptest[1] - p1[1]) * (pn[0] - p1[0]) - (pn[1] - p1[1]) * (ptest[0] - p1[0])
    return abs(val)

def whichSide(p1, pn, ptest) :
    # Titik (x3,y3) berada di sebelah kiri dari garis ((x1,y1),(x2,y2)) jika hasil determinan positif
    x1, y1 = p1[0], p1[1]
    x2, y2 = pn[0], pn[1]
    x3, y3 = ptest[0], ptest[1]
    val = (x1*y2 + x3*y1 + x2*y3) - (x3*y2 + x2*y1 + x1*y3)
    # Jika ptest di kiri garis yang dibentuk p1 dan pn akan mengembalikan 1
    # Jika ptest di kanan garis yang dibentuk p1 dan pn akan mengembalikan -1
    if (val > 0) :
        return 1
    elif (val < 0) :
        return -1
    return 0
```

```python
import numpy as np

def isMember(arrToCheck, arr) :
    # Mengecek apakah arrToCheck adalah anggota dari list of array arr
    for i in range(len(arr)) :
        if ((arr[i][0] == arrToCheck[0]) and (arr[i][1] == arrToCheck[1])) :
            return True
    return False

def sortPoints(points) :
    # Sort based on absis (first column), jika absis sama compare ordinatnya (second column)
    points = sorted(points, key=lambda x: (x[0], x[1]))
    return points

def lineDistance(p1, pn, ptest) :
    # Mengembalikan jarak dari titik ptest ke garis yang dibentuk p1 dan pn
    val = (ptest[1] - p1[1]) * (pn[0] - p1[0]) - (pn[1] - p1[1]) * (ptest[0] - p1[0])
    return abs(val)

def whichSide(p1, pn, ptest) :
    # Titik (x3,y3) berada di sebelah kiri dari garis ((x1,y1),(x2,y2)) jika hasil determinan positif
    x1, y1 = p1[0], p1[1]
    x2, y2 = pn[0], pn[1]
    x3, y3 = ptest[0], ptest[1]
    val = (x1*y2 + x3*y1 + x2*y3) - (x3*y2 + x2*y1 + x1*y3)
    # Jika ptest di kiri garis yang dibentuk p1 dan pn akan mengembalikan 1
    # Jika ptest di kanan garis yang dibentuk p1 dan pn akan mengembalikan -1
    if (val > 0) :
        return 1
    elif (val < 0) :
        return -1
    return 0
```

File iris.ipynb

```
    #visualisasi hasil ConvexHull

    # Penggantian kolom
    a = 0
    b = 1

    plt.figure(figsize = (10, 6))
    colors = ['b','r','g']
    plt.title('Petal Width vs Petal Length')
    plt.xlabel(iris.feature_names[a])
    plt.ylabel(iris.feature_names[b])
    for i in range(len(iris.target_names)):
        bucket = df[df['Target'] == i]
        bucket = bucket.iloc[:,[a,b]].values
        edges = ConvexHull(bucket)
        plt.scatter(bucket[:, 0], bucket[:, 1], label=iris.target_names[i])
        for j in range(len(edges)):
            plt.plot(bucket[edges[j], 0], bucket[edges[j], 1], colors[i])

    plt.legend()
```

```
# Penggantian kolom
a = 2
b = 3

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Sepal Width vs Sepal Length')
plt.xlabel(iris.feature_names[a])
plt.ylabel(iris.feature_names[b])
for i in range(len(iris.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[a,b]].values
    edges = ConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=iris.target_names[i])
    for j in range(len(edges)):
        plt.plot(bucket[edges[j], 0], bucket[edges[j], 1], colors[i])

plt.legend()
```

File wine.ipynb

```
#visualisasi hasil ConvexHull

# Penggantian kolom
a = 0
b = 9

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Alcohol and Color Intensity')
plt.xlabel(wine.feature_names[a])
plt.ylabel(wine.feature_names[b])
for i in range(len(wine.target_names)):
    bucket = df1[df1['Target'] == i]
    bucket = bucket.iloc[:,[a,b]].values
    edges = ConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=wine.target_names[i])
    for j in range(len(edges)):
        plt.plot(bucket[edges[j], 0], bucket[edges[j], 1], colors[i])
plt.legend()
```

```
#visualisasi hasil ConvexHull

# Penggantian kolom
a = 2
b = 3

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Ash and Alcalinity of Ash')
plt.xlabel(wine.feature_names[a])
plt.ylabel(wine.feature_names[b])
for i in range(len(wine.target_names)):
    bucket = df1[df1['Target'] == i]
    bucket = bucket.iloc[:,[a,b]].values
    edges = ConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=wine.target_names[i])
    for j in range(len(edges)):
        plt.plot(bucket[edges[j], 0], bucket[edges[j], 1], colors[i])
plt.legend()
```

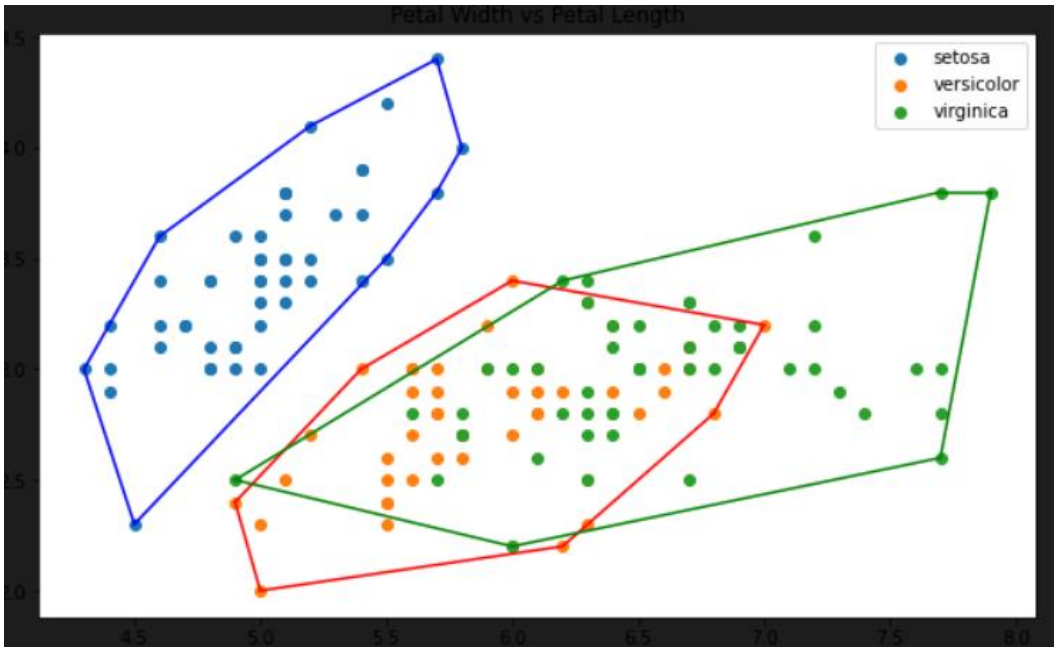
File breast_cancer.ipynb

```
#visualisasi hasil ConvexHull

# Penggantian kolom
a = 5
b = 8

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Mean Symetry vs Mean Compactness')
plt.xlabel(breast_cancer.feature_names[a])
plt.ylabel(breast_cancer.feature_names[b])
for i in range(len(breast_cancer.target_names)):
    bucket = df1[df1['Target'] == i]
    bucket = bucket.iloc[:,[a,b]].values
    edges = ConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=breast_cancer.target_names[i])
    for j in range(len(edges)):
        plt.plot(bucket[edges[j], 0], bucket[edges[j], 1], colors[i])
plt.legend()
```

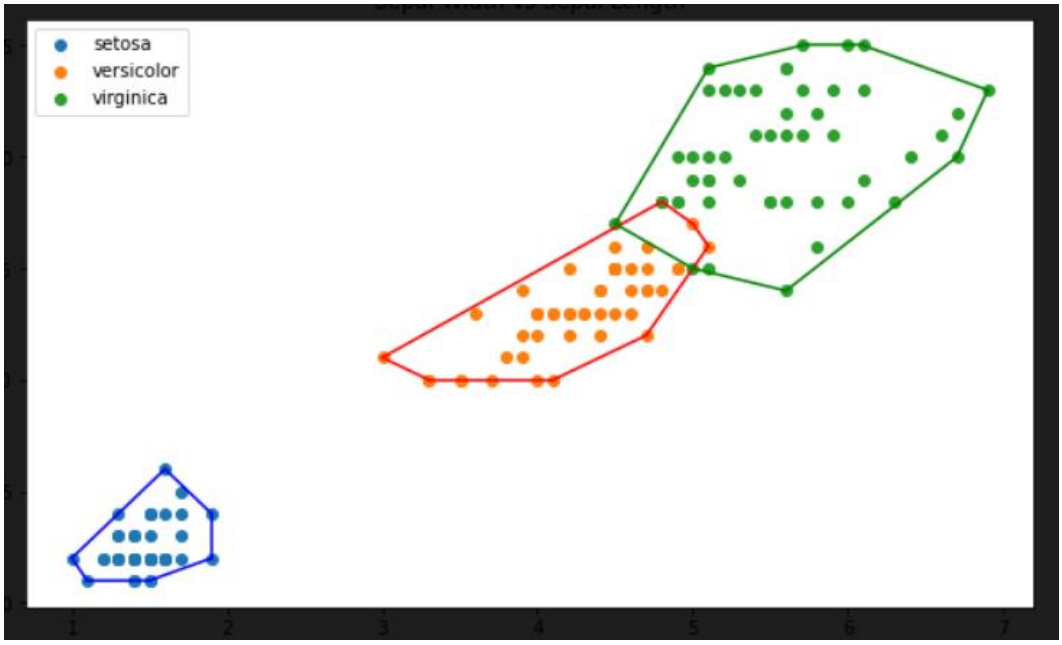
```
#visualisasi hasil ConvexHull

# Penggantian kolom
a = 0
b = 1

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Mean Radius vs Mean Texture')
plt.xlabel(breast_cancer.feature_names[a])
plt.ylabel(breast_cancer.feature_names[b])
for i in range(len(breast_cancer.target_names)):
    bucket = df1[df1['Target'] == i]
    bucket = bucket.iloc[:,[a,b]].values
    edges = ConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=breast_cancer.target_names[i])
    for j in range(len(edges)):
        plt.plot(bucket[edges[j], 0], bucket[edges[j], 1], colors[i])
plt.legend()
```

Screenshot Input dan Output

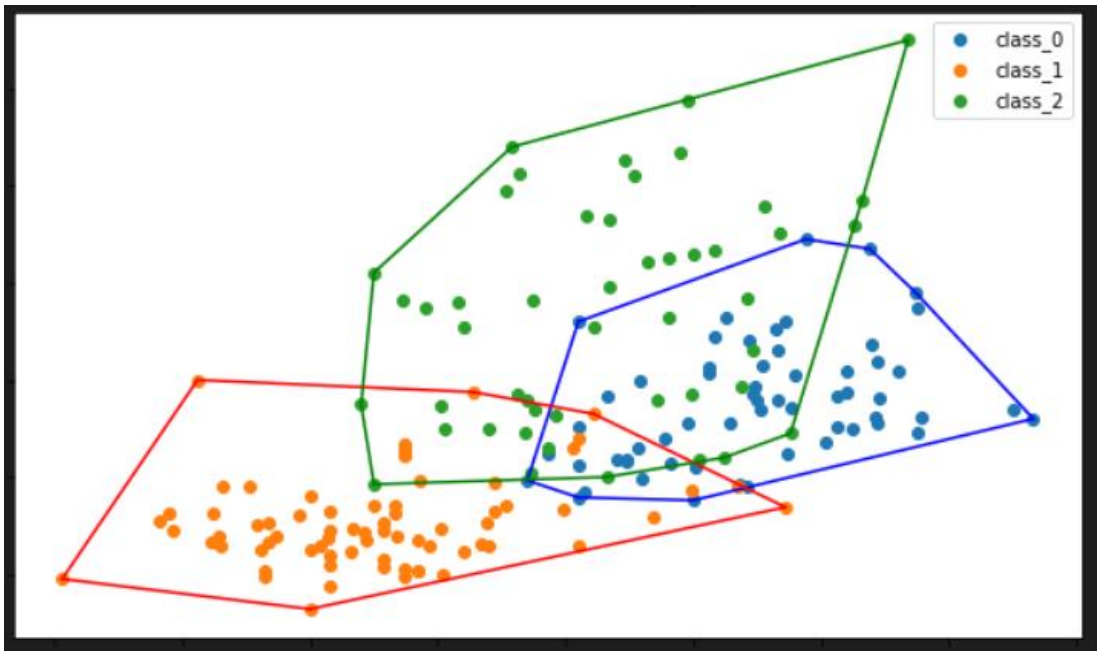
Dataset Iris (kolom 0, 1)



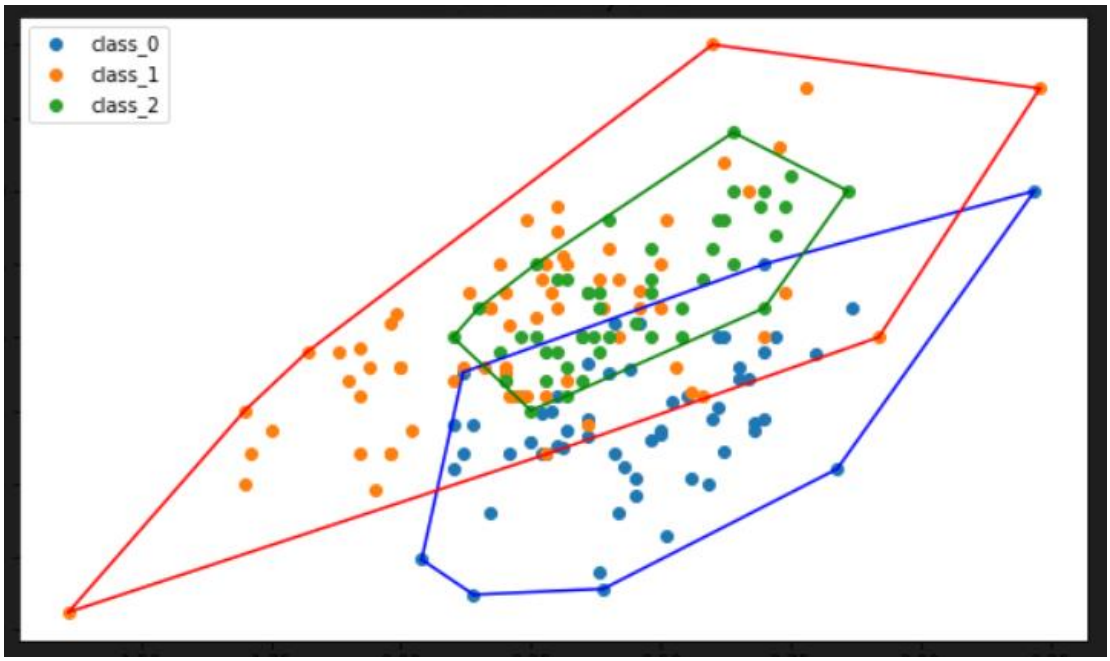
Dataset Iris (kolom 2,3)



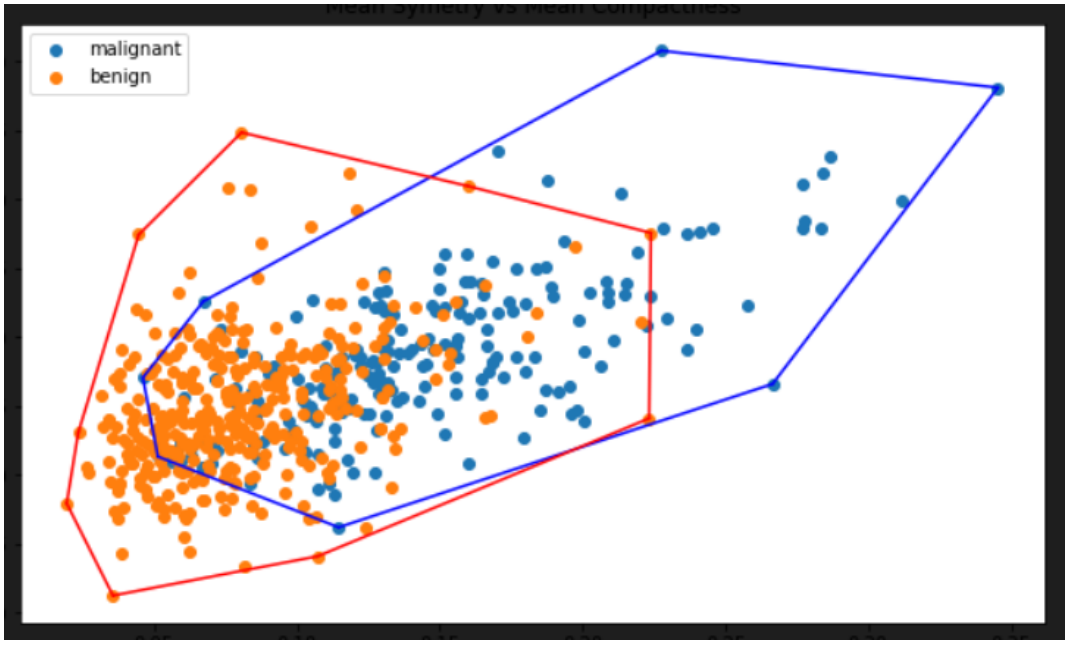
Dataset Wine (kolom 0, 9)



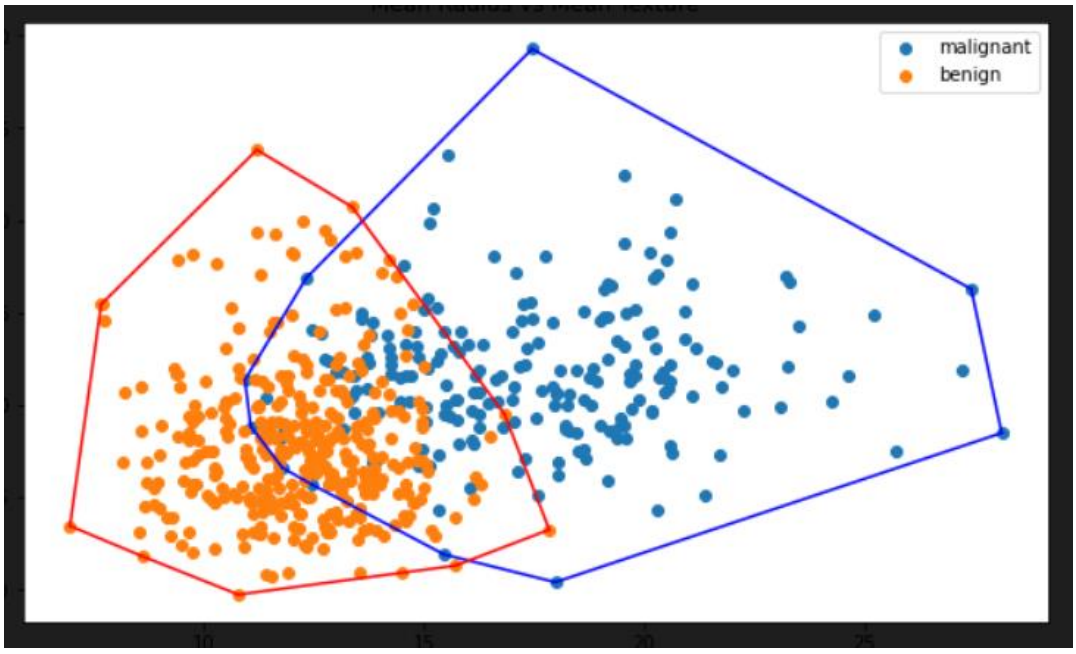
Dataset Wine (kolom 2, 3)



Dataset Breast Cancer (kolom 5,8)



Dataset Breast Cancer (kolom 0,1)



Link Kode Program

https://github.com/kristabdi/convex-hull

Checklist

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