Laporan Tugas Kecil 2 IF2211 Strategi Algoritma

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

DIBUAT OLEH

William Manuel Kurniawan

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

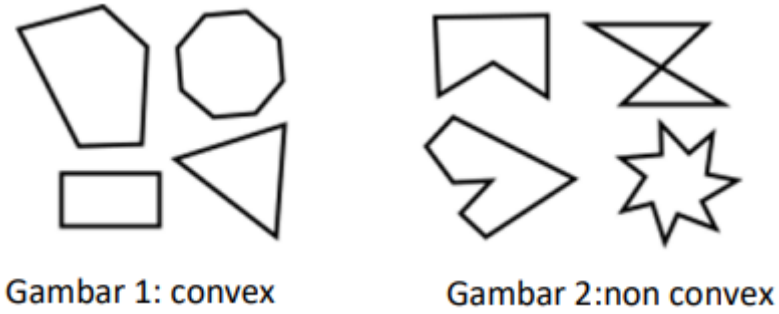
INSTITUT TEKNOLOGI BANDUNG

Semester II Tahun 2021/2022

Daftar Isi

Bab 1 Pendahuluan

Sebuah bidang planar akan disebut sebagai bidang convex (*convex hull*) jika pada bidang tersebut, semua garis yang dibuat dari 2 titik sembarang bidang akan berada di bagian dalam bidang tersebut. Contoh gambar 1 adalah poligon yang convex, sedangkan gambar 2 menunjukkan contoh yang non-convex.



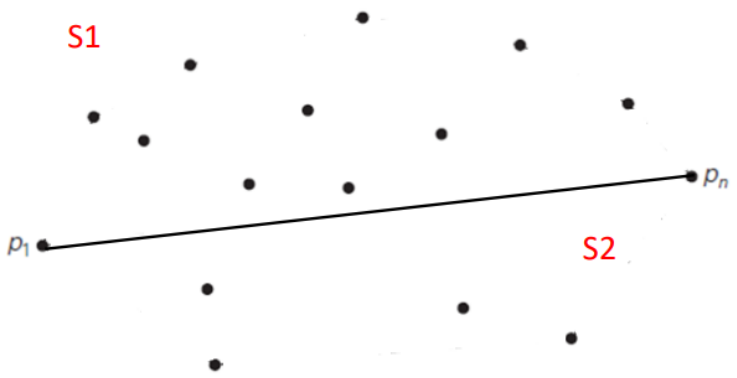
Gambar 1.1. Gambar bidang convex dan bidang non convex

Dari gambar 1, dapat dilihat bahwa jika dibuat garis dari 2 titik mana pun dalam bidang tersebut, hasil garisnya akan berada di dalam bidang sedangkan dari gambar 2, terdapat beberapa titik dimana garis yang dibentuk akan berada di luar bidang.

Metode algoritma *divide and conquer* bekerja dengan memecah sebuah masalah besar menjadi masalah yang lebih kecil sampai masalah menjadi mudah untuk diselesaikan. Setelah solusi ditemukan untuk masing-masing masalah yang dipecahkan, hasil tersebut akan kembali digabungkan menjadi solusi masalah awal. Dalam program, penerapan teknik tersebut akan dilakukan menggunakan rekursi dimana dibuat fungsi yang akan memanggil diri sendiri dengan kondisi terminasi tertentu dan dipanggil ulang dengan parameter tertentu yang diubah sehingga tidak terjadi pengulangan tak terhingga. Dalam praktek, fungsi rekursi akan mengeluarkan hasil dari masalah besar dari penyelesaian masalah kecil yang sudah disatukan secara otomatis.

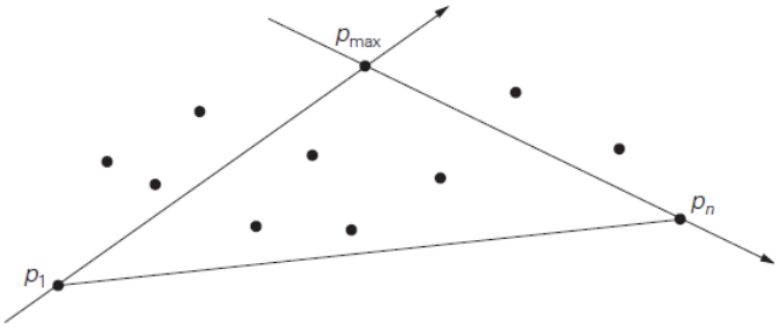
Bab 2 Teori Dasar Cara Kerja Program

Untuk membentuk *convex hull*, hal pertama yang dilakukan adalah mencari titik yang berada pada paling kiri dan yang berada pada paling kanan. Jika ada titik dengan nilai absis yang sama, ambil titik kiri dengan ordinat paling rendah dan titik kanan dengan ordinat paling tinggi. Dari 2 titik tersebut, bentuk sebuah garis lalu pisahkan titik lain dengan menentukan jika titik tersebut ada di sebelah kiri garis atau di sebelah kanan garis.



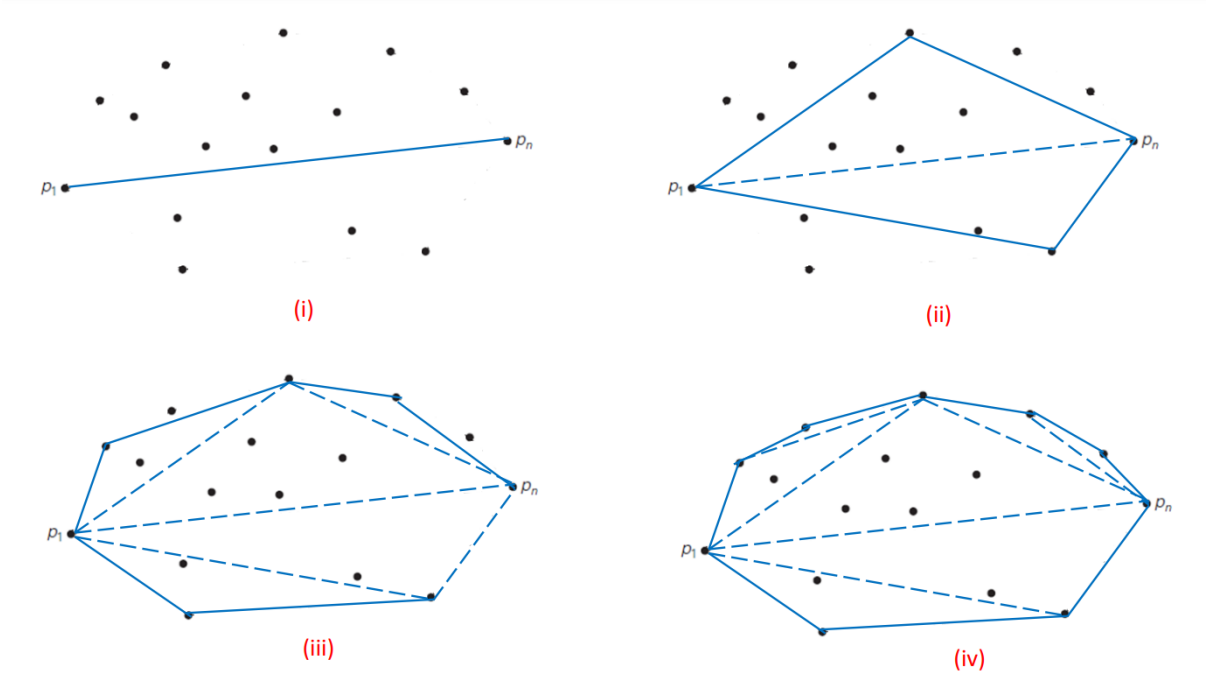
Gambar 2.1. Gambar hasil pembentukan garis dan pemisahan titik

Setelah titik dipisahkan, masing-masing sisi yang dipisahkan akan menjadi bagian dari *convex hull* akhir. Kemudian, cari titik yang memiliki jarak terjauh dari garis awal yang dibentuk. Jika tidak ditemukan, maka 2 titik dari garis awal tersebut akan disimpan sebagai bagian dari hasil *convex hull*. Jika ditemukan titik lain, maka akan dibentuk garis baru seperti gambar berikut.



Gambar 2.2. Gambar hasil pembentukan garis baru yang memiliki jarak maksimum dari garis awal

Dari garis baru tersebut, cari titik yang berada sisi kiri dan kanan garis serta berada di luar segitiga yang dibentuk. Lakukan ulang proses untuk mencari titik dengan jarak maksimum sampai semua titik sudah ditemukan dan tidak ada lagi titik di luar garis.



Gambar 2.3. Contoh penyelesaian *convex hull* dengan metode *divide and conquer*

Bab 3 Source Program

Algoritma penyelesaian *convex hull* dibuat menggunakan bahasa pemrograman Python dan visualisasi hasil *convex hull* akan dilakukan menggunakan jupyter notebook dengan bantuan library numpy dan matplotlib untuk visualisasi hasil *convex hull* dan library scikit-learn sebagai sumber data untuk *convex hull*. Dataset yang digunakan dari library scikit-learn adalah *iris*, *wine*, dan *breast_cancer*. Berikut adalah hasil program dari file convexhull.ipynb yang mengeluarkan hasil *convex hull* dari dataset *iris*.

```
import numpy as np

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

# Partisi digunakan untuk membantu melakukan quicksort
def partisi(bucket,i,j):
    p = i
    q = j
    pivot = bucket[i][0]
    temp_pivot = bucket[i]
    while (p < q):
        while (p<len(bucket) and bucket[p][0]<=pivot):
```

```
            p += 1
        while (bucket[q][0]>pivot):
            q -= 1
        if (p < q):
            temp = bucket[p]
            bucket[p] = bucket[q]
            bucket[q] = temp

    temp2 = bucket[q]
    bucket[q] = temp_pivot
    bucket[i] = temp2

    return q

# Quicksort dilakukan untuk nilai absis dari data
def quicksortX(bucket,i,j):
    if (i<j):
        temp = partisi(bucket,i,j)
        quicksortX(bucket,i,temp-1)
        quicksortX(bucket,temp+1,j)

# Selection sort dilakukan untuk nilai ordinat dari data
def sortY(bucket):
    for i in range (0,len(bucket)):
        start = i
        for j in range (i, len(bucket)):
            if(bucket[start][0] == bucket[j][0]):
                end = j

        kecil = bucket[start]
        tempvar = start
        for k in range (start,end+1):
            if(bucket[k][1]<kecil[1]):
                kecil = bucket[k]
                tempvar = k

        bucket[tempvar] = bucket[start]
        bucket[start] = kecil

# Penggabungan quicksort untuk absis dan selection sort untuk ordinat
def sortfinal(bucket,i,j):
    quicksortX(bucket,i,j)
    sortY(bucket)

# Mencari jika titik3 berada di sisi kiri / sisi kanan garis yang dibuat dari
titik1 dan titik2
def sisi(titik1,titik2,titik3):
```

```
    temp = titik1[0]*titik2[1] + titik3[0]*titik1[1] + titik2[0]*titik3[1]-
titik3[0]*titik2[1]-titik2[0]*titik1[1]-titik1[0]*titik3[1]
    if (temp > 0):
        return 1
    if (temp < 0):
        return -1
    return 0

# Mencari jarak titik3 dari garis yang dibuat dari titik1 dan titik2
def jarak(titik1,titik2,titik3):
    hasil = abs(((titik3[1]-titik1[1])*(titik2[0]-titik1[0])) - ((titik2[1]-
titik1[1])*(titik3[0]-titik1[0])))

    return hasil

# Mencari titik yang memiliki jarak terjauh dari titik1 dan titik2
def maxhull (bucket, titik1, titik2):

    max = 0
    maxtitik = -999
    for i in range (0, len(bucket)):
        temp = jarak(titik1,titik2,bucket[i])
        if (temp > max):
            max = temp
            maxtitik = i

    if (maxtitik == -999):
        return []


    return bucket[maxtitik]

# Bagian pencarian convex hull yang akan direkursi
def hull2(bucket, titik1, titik2, titik3):
    # Jika tidak ditemukan titik di luar convex hull maka titik1 dan titik2
dimasukkan ke hasil akhir
    if (len(bucket) == 0 or titik3 == []):
        return [titik1,titik2]

    else :

        left = []
        right = []

        # Melihat jika titik berada di sisi kiri / kanan dan menyimpan titik
tersebut di 2 list
        for j in range (0,len(bucket)):
            sisi1 = sisi(titik1,titik3,bucket[j])
```

```
            sisi2 = sisi(titik2,titik3,bucket[j])
            if (sisi1 == 1 and sisi2 == 1):
                left.append(bucket[j])
            if (sisi1 == -1 and sisi2 == -1):
                right.append(bucket[j])

        # Mencari titik terjauh dari list hasil pembagian
        nilai1 = maxhull(left,titik1,titik3)
        nilai2 = maxhull(right,titik2,titik3)

        return hull2(left,titik1,titik3,nilai1) +
hull2(right,titik3,titik2,nilai2)


# Bagian awal dari convex hull
def hull(bucket, titik1, titik2):
    # Jika tidak ditemukan titik di luar convex hull maka titik1 dan titik2
dimasukkan ke hasil akhir
    if (len(bucket) == 0):
        return [titik1,titik2]

    else :

        left = []
        right = []

        # Melihat jika titik berada di sisi kiri / kanan dan menyimpan titik
tersebut di 2 list
        for j in range (0,len(bucket)):
            sisi1 = sisi(titik1,titik2,bucket[j])
            if (sisi1 == 1):
                left.append(bucket[j])
            if (sisi1 == -1):
                right.append(bucket[j])

        # Mencari titik terjauh dari list hasil pembagian
        nilai1 = maxhull(left,titik1,titik2)
        nilai2 = maxhull(right,titik1,titik2)

        return hull2(left,titik1,titik2,nilai1) +
hull2(right,titik2,titik1,nilai2)

# Pemrosesan data menjadi beberapa titik yang akan digambar di convex hull
def hullfinal(bucket):
    sortfinal(bucket,0,len(bucket)-1)
    hasil3 = []
    temp1 = hull(bucket,bucket[0],bucket[len(bucket)-1])
    hasil3 = hasil3 + temp1
```

```
    return hasil3

data = datasets.load_iris()
# Mengambil data dari database
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal Length')
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
    # Mengubah numpy array menjadi list yang dapat diproses oleh fungsi
hullfinal
    temp = bucket.tolist()
    hasil = hullfinal(temp)
    # Mengubah list kembali menjadi numpy agar bisa digambar
    hasil = np.array(hasil)
    print(" ")
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for j in range (0,len(hasil)-1):
        plt.plot([hasil[j][0],hasil[j+1][0]], [hasil[j][1],hasil[j+1][1]],
colors[i])
    plt.legend()
```

untuk penyelesaian dataset *wine*, ditambahkan program sebagai berikut.

```
data = datasets.load_wine()
# Mengambil data dari database
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Alcohol vs Malic Acid')
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
```

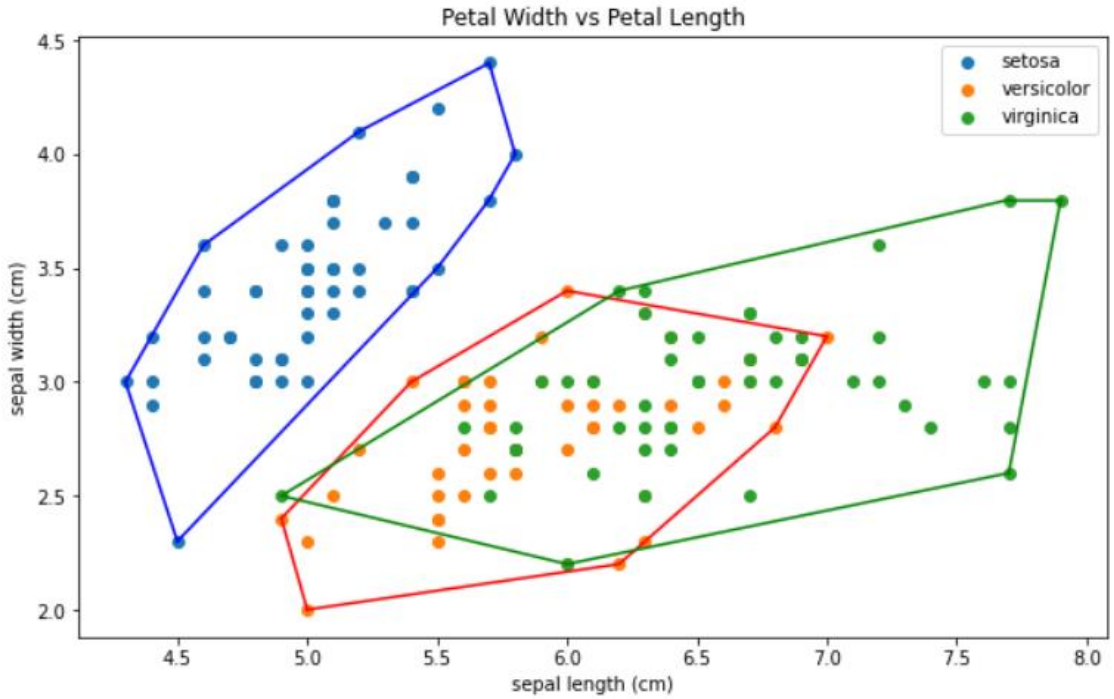
```
    # Mengubah numpy array menjadi list yang dapat diproses oleh fungsi
hullfinal
    temp = bucket.tolist()
    hasil = hullfinal(temp)
    # Mengubah list kembali menjadi numpy agar bisa digambar
    hasil = np.array(hasil)
    print(" ")
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for j in range (0,len(hasil)-1):
        plt.plot([hasil[j][0],hasil[j+1][0]], [hasil[j][1],hasil[j+1][1]],
colors[i])
plt.legend()
```

untuk penyelesaian dataset *breast_cancer*, ditambahkan program berikut.

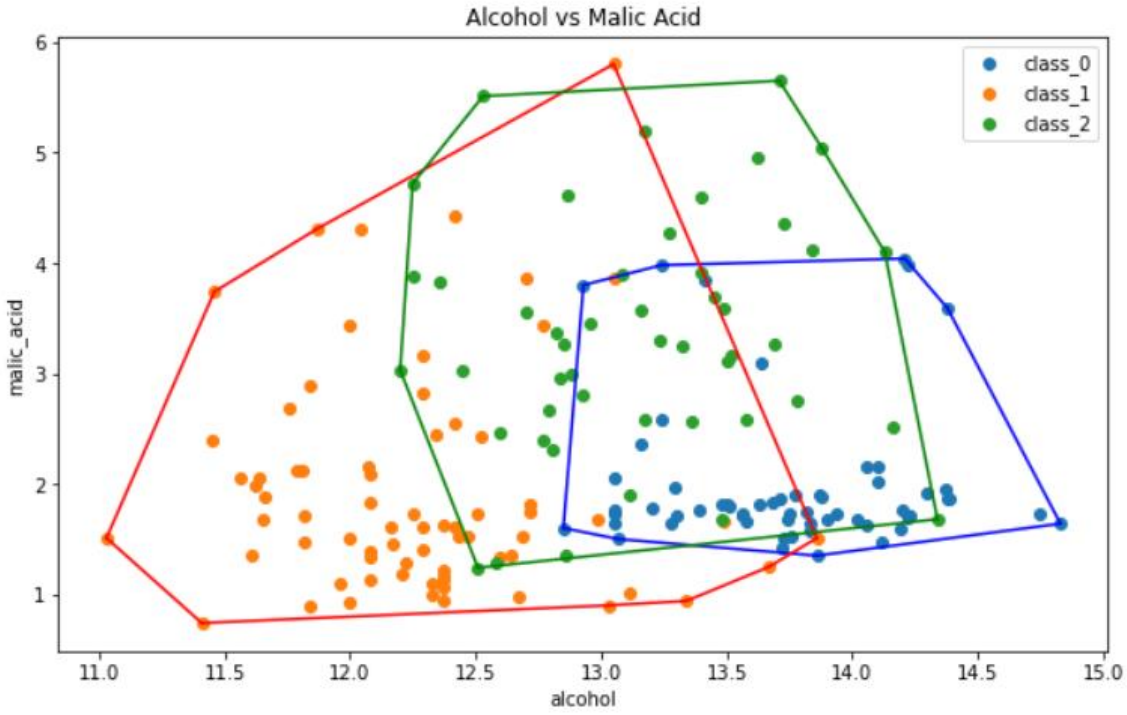
```
data = datasets.load_breast_cancer()
# Mengambil data dari database
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Texture VS Radius')
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
    # Mengubah numpy array menjadi list yang dapat diproses oleh fungsi
hullfinal
    temp = bucket.tolist()
    hasil = hullfinal(temp)
    # Mengubah list kembali menjadi numpy agar bisa digambar
    hasil = np.array(hasil)
    print(" ")
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for j in range (0,len(hasil)-1):
        plt.plot([hasil[j][0],hasil[j+1][0]], [hasil[j][1],hasil[j+1][1]],
colors[i])
    plt.legend()
```

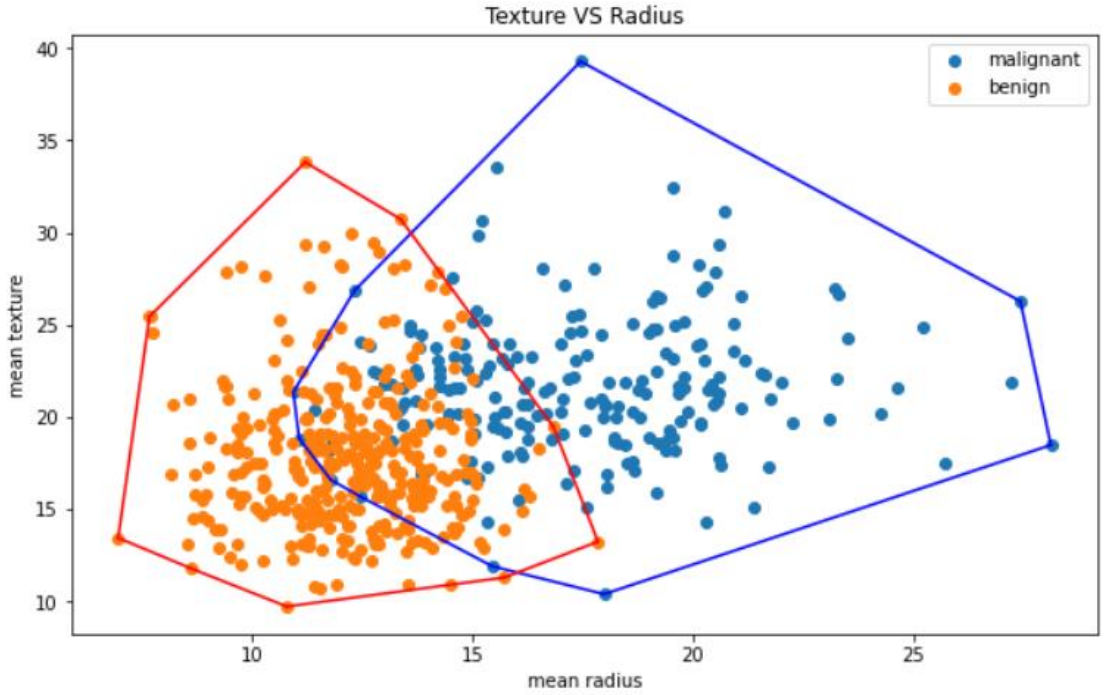
Bab 4 Hasil Pengujian



Gambar 4.1. Hasil *convex hull* dari dataset *iris*



Gambar 4.2. Hasil *convex hull* dari dataset *wine*



Gambar 4.3. Hasil *convex hull* dari dataset *breast_cancer*

Tabel Cek List

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.	√	

Link untuk dokumentasi dan kode di Github :

https://github.com/wmk567/convex-hull.git