LAPORAN TUGAS KECIL 2 IF2211 Strategi Algoritma

Semester II tahun 2021/2022

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

Steven Gianmarg Haposan Siahaan
13520145

Program Studi Teknik Informatika
Sekolah Tinggi Elektro dan Informatika
Institut Teknologi Bandung
2022

Daftar Isi

BAB I

Landasan Teori

1.1 Algoritma Divide and Conquer

Divide and Conquer merupakan salah satu strategi dalam pemecahan masalah. Algoritma ini berprinsip memecah permasalahan yang terlalu besar menjadi beberapa bagian kecil sehingga lebih mudah untuk diselesaikan. Algoritma ini dimulai dengan membagi masalah menjadi beberapa masalah yang memiliki kemiripan dengan masalah semula namun berukuran lebih kecil, bahkan idealnya berukuran hampir sama(tahap Divide). Lalu dilanjutkan dengan memecahkan masing-masing masalah tadi secara rekursif (Conquer). Lalu setiap solusi sub masalah tadi akan digabungkan sehingga membentuk solusi masalah semula.

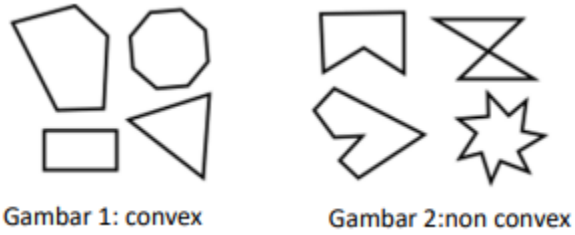
Objek masalah yang di bagi adalah masukan (input) atau instances yang berukuran n: tabel (larik), matriks, dan sebagainya, bergantung pada masalahnya. Tiap-tiap upa-masalah mempunyai karakteristik yang sama (the same type) dengan karakteristik masalah asal, sehingga metode Divide and Conquer lebih natural diungkapkan dalam skema rekursif.

Strategi ini seringkali lebih baik daripada strategi brute force yang sudah dipelajari sebelumnya. Jika menggunakan algoritma Brute force akan mencoba semua kemungkinan yang ada. Algoritma brute force terbilang straight forward,langsung menyelesaikan masalah berdasarkan problem statement dan konsep yang dilibatkan. Hal ini yang membuat divide and conquer seringkali lebih efektif, karena strategi divide and conquer membagi persoalan menjadi sub persoalan yang lebih kecil yang relatif lebih mudah untuk dipecahkan.

1.2 Convex Hull

Convex hull merupakan salah satu permasalahan yang bisa diselesaikan dengan strategi divide and conquer. Convex hull sendiri merupakan sebuah himpunan titik pada bidang planar adalah himpunan titik – titik yang membentuk sebuah poligon konveks yang melingkupi seluruh himpunan titik tersebut. Pada permasalahan ini, kita diminta untuk membentuk convex hull dengan diwali sebuah garis yang terbentuk dari titik dengan absis terkecil dan terbesar(2 titik ini akan menjadi titik convex hull),lalu himpunan titik itu akan terbagi menjadi 2 bagian yang berada di atas garis dan di bawah garis. Lalu pada setiap bagian atas dan bawah(sebut saja s1 dan s2) dari garis tersebut akan mendapat perlakuan yang sama, dimulai dengan mencari titik terjauh dari garis tersebut yang akan membentuk segitiga. Titik terjauh tersebut juga akan menjadi salah satu convex hull bersama dengan 2 titik lainnya tadi. Setelah terbentuk segitiga akan terdapat 3 bagian titik pada setiap s1 maupun s2, yakni bagian dalam segitiga,bagian kiri atas dan bagian kanan atas. Setelah terbentuk 3 bagian tersebut, akan ada 3 garis yang akan digunakan sebagai acuan, yakni garis yang terbentuk dari titik dengan absis minimum dan tititk dengan absis maksimum,garis yang terbentuk dari titik dengan absis minimum dengan titik terjauh, serta garis yang terbentuk dari titik terjauh dan tititk dengan absis maksimum. Hal pertama kita mengecek

apakah titik terdapat di dalam bagian segitiga, karena jika terdapat di dalam segitiga titik tersebut tidak perlu untuk diperiksa dan lanjut ke titik lainnya. Lalu ulangi hal yang sama untuk mencari hull pada bagian kiri atas (garis yang terbentuk dari titik dengan absis minimum dengan titik terjauh) dan kanan atas(garis yang terbentuk dari titik terjauh dan titik dengan absis maksimum)



Gambar 1: convex Gambar 2:non convex

Gambar 1.1 Convex dan Non convex

BAB II

Program dan Spesifikasi Tugas

2.2 Spesifikasi Tugas

Pada tugas ini akan dibuat implementasi dalam bahasa python dari algoritma yang dijelaskan pada bab I dan algoritma tersebut akan dibandingkan dengan metode pencarian convex hull yang dimiliki library scipy. Poligon convex hull tersebut akan diterapkan pada kumpulan dataset milik library sklearn. Akan diambil dua atribut dan dibentuk convex hull untuk masing-masing kelompok data dengan nilai target yang sama. Kumpulan data tersebut beserta convex hull nya akan disimpan dalam bentuk gambar. Gambar inilah yang akan dibandingkan antara algoritma yang dibuat sendiri dengan algoritma milik library scipy.

Buatlah sebuah pustaka (library) myConvexHull dalam bahasa Python yang dapat mengembalikan convex hull dari kumpulan data 2 dimensi (dapat dianggap kumpulan titik 2 dimensi). Himpunan titik pada bidang planar disebut convex jika untuk sembarang dua titik pada bidang tersebut (misal p dan q), seluruh segmen garis yang berakhir di p dan q berada pada himpunan tersebut. Contoh gambar 1 adalah poligon yang convex, sedangkan gambar 2 menunjukkan contoh yang non-convex. Penjelasan lebih detail dapat dilihat pada materi kuliah Divide & Conquer bagian 4.

2.2 Implementasi Divide and Conquer pada Convex Hull

Langkah pembuatahn ConvexHull :

1. Bagi himpunan titik menjadi 2 bagian yakni atas dan bawah, dengan terlebih dahulu mencari titik dengan absis minimum dan titik dengan absis minimum(akan menjadi 2 hull pertama) yang akan menjadi garis pembatas kedua bagian tersebut.
2. Masukkan titik yang berada pada bagian atas dan bagian bawah ke 2 sub set yang berbeda. Setelah itu cari titik terjauh(akan menjadi hull berikutnya) agar terbentuk 2 segitiga,yakni segitiga atas dan segitiga bawah.
3. Pada masing-masing bagian lakukan hal yang sama pada 2 garis baru yang berhasil terbentuk, cari titik yang terletak di atas garis kiri dan diatas garis kanan, dan kembali masukkan ke dalam subset yang berbeda. Lalu cari kembali titik terjauh(titik ini akan menjadi hull juga) untuk kembali membentuk segitiga dan ulangi kembali tahap 2. Proses akan terus diulangi sampai sudah tidak ada titik yang berada diatas garis yang terbentuk.
4. Setelah ditemukan titik-titik yang akan membentuk convex hull, masuk ke tahap visualisasi dengan referensi code untuk visualisasi disertakan pada spesifikasi tugas,namun ada beberapa yang saya rombak untuk menyesuaikan dengan pustaka myConvexHull milik saya.
5. Penjelasan lebih lengkap akan dijelaskan pada setiap fungsinya.

2.3 Source Code

```
import numpy as np
import pandas as pd
```

```
import math
import matplotlib.pyplot as plt
from sklearn import datasets
import matplotlib.pyplot as plt
def sortByXCoord(points):
# Mengurutkan absis dari yang terkecil #
    points=sorted(points,key=lambda x:x[0])
    return points
def sortByYCoord(points):
#Mengurutkan Y dari yang paling kecil #
    points=sorted(points,key=lambda x:x[1])
    return points
def jarakTitik(p1,p2):
# Menghitung jarak titik p1 dan p2 #
    return (math.sqrt((p2[0]-p1[0])**2 + (p2[1]-p1[1])**2))
def jarakTitikKeGaris(point,line) :
# Menghitung jarak titik terhadap garisi #
    line=sortByXCoord(line)
    slope=gradien(line[0],line[1])
    yInt = line[0][1]-(slope*line[0][0]) #mencari konstanta pers garis
    dist= abs(-slope*point[0]+1*point[1]-yInt)/(math.sqrt(1**2+(slope**2)))
    return (dist)
def gradien(point1,point2) :
# Menghitung gradien #
    if(point2[0]!=point1[0]):
        slope = (point2[1]-point1[1])/(point2[0]-point1[0])
        return slope
    else :
        return False
def sisiTitik(point1,point2,point3):
# Menghitung nilai determinan untuk membantu menentukan posisi titik terhadap
garis #
    det=point1[0]*point2[1]+
point3[0]*point1[1]+point2[0]*point3[1]-point3[0]*point2[1]-point2[0]*point1[1]-
point1[0]*point3[1]
    return det
def upperSide(point1,point2,point3):
# Memeriksa apakah point 3 terletak dibagian atas garis yang dibentuk point 2
dan point 3 #
    if(sisiTitik(point1, point2, point3)>0.00) :
```

```
        return True
    else :
        return False
def lowerSide(point1, point2, point3):
# Memeriksa apakah point 3 terletak dibagian bawah garis yang dibentuk point 2
dan point 3 #
    if(sisiTitik(point1, point2, point3)<0.00):
        return True
    else :
        return False
def segaris(point1, point2, point3):
# Memeriksa apakah point 3 terletak pada garis yang dibentuk point 2 dan point 3
#
    if(sisiTitik(point1, point2, point3)==0):
        return True
    else:
        return False
def LuasSegitiga(point1, point2, point3) :
# Menghitung luas segitiga yang dibentuk oleh point1,point2,point3 #
    if(isTriangle(point1, point2, point3) and segaris(point1, point2, point3) is
not True) :
        luas=0

s=(jarakTitik(point1,point2)+jarakTitik(point2,point3)+jarakTitik(point1,
point3))/2
        luas =
s*(s-jarakTitik(point1,point2))*(s-jarakTitik(point2,point3))*(s-jarakTitik(poin
t3,point1))
    return math.sqrt(luas)
def isTriangle(point1, point2, point3) :
# Memeriksa apakah point1,point2,point3 akan membentuk segitiga #
    a=jarakTitik(point1, point2)
    b=jarakTitik(point2, point3)
    c= jarakTitik(point1,point3)
    s=[a,b,c]
    s.sort()
    if(segaris(point1, point2, point3) is True):
        return False
    if(segaris(point1, point2, point3) is not True and (s[0]+s[1])>s[2]) :
        return True
```

```
    else :
        return False
def isInsideTriangle(point1, point2, point3,point4):
# Memeriksa apakah point4 terletak di dalam segitiga #
    if(isTriangle(point1, point2, point3) is True):
        s=[point1, point2, point3]
        s=sortByXCoord(s)
        if(sisiTitik(s[0], s[1],point4 )>=0.00 and sisiTitik(s[0], s[2],
point4)<=0.00 and sisiTitik(s[1], s[2], point4)>=0.00):
            return True
        else :
            return False
def titik_terjauh(s,point1,point2):
# s=kumpulan titik,point1=titik minimumabsis,point2=titik maximum absis #
# Mencari titik terjauh dari garis yang dibentuk oleh 2 titik #
    s=sortByXCoord(s)
    titikTerjauh = s[0]
    patokan=0
    for i in range(0,len(s)):
        if(jarakTitikKeGaris(s[i], [point1, point2])>=patokan):
            patokan=jarakTitikKeGaris(s[i], [point1, point2])
            titikTerjauh.append(s[i])
    return titikTerjauh[-1]
def firstSecondHull(points):#points=list of points
# Mencari hull pertama dan kedua yang akan membagi himpunan menjadi 2 bagian #
    points=sorted(points)
    palingKanan=[]
    firstSecondHull=[points[0]]
    if(points[-1][0]==points[-2][0]):
        for i in range(len(points)):
            if(points[i][0]==points[-1][0]) :
                palingKanan.append(points[i])
                break
    else :
        palingKanan.append(points[-1])
    firstSecondHull.append(palingKanan[0])
    return(firstSecondHull)
def myConvexHull(s) :
    hull=[]
    if s is not list :
```

```
        s=s.tolist()
    first_Second_Hull=firstSecondHull(s)
    hull.append(first_Second_Hull[0])
    hull.append(first_Second_Hull[-1])
    s1=[]#kiri/atas
    s2=[]#kanan/bawah
# Membagi himpunan menjadi 2 bagian yang dibatasi oleh garis yang dibentuk 2
titik hull pertama yang ditemukan(titik paling kiri dan paling kanan)
    for z in range(0,len(s)):
        if(s[z]!=first_Second_Hull[0] and s[z]!=first_Second_Hull[-1]):
            if(lowerSide(first_Second_Hull[0], first_Second_Hull[-1], s[z])):
                s2.append(s[z]) #sisi kanan/bawah untuk variabel x
            if(upperSide(first_Second_Hull[0],
first_Second_Hull[-1],s[z])>0.00):
                s1.append(s[z]) #sisi kiri/atas untuk variabel x
# Mencari hull dari 2 sub bagian tersebut #
    findHull(s1,first_Second_Hull[0], first_Second_Hull[-1],hull)
    findHull(s2,first_Second_Hull[-1],first_Second_Hull[0],hull)
    return hull
def findHull(sk,p,q,hull):
# Mencari hull dari 2 bagian yang sudah dibuat #
    sk=sortByXCoord(sk)
    sub_s1=[]#kiri/atas yang bagian kiri
    sub_s2=[]#kiri/atas yang bagian kanan
    if(len(sk)==0): return
    else :
# Menemukan hull dengan fungsi titik terjauh #
        a=titik_terjauh(sk,p,q)
        sk.remove(a)
        hull.insert(1,a)
# Membagi kembali menjadi sub bagian yang lebih kecil #
        for j in range(0,len(sk)):
            if(isInsideTriangle(p,q,a,sk[j])==False) :
                if(upperSide(p,a,sk[j] )):
                    (sub_s1).append(sk[j])
                if(upperSide(a,q,sk[j])):
                    (sub_s2).append(sk[j])
# Mencari kembali hull dari sub bagian tersebut #
    findHull(sub_s1,p,a,hull)
    findHull(sub_s2,a,q,hull)
```

```
def splitandMergeHull(hull):
# Memecah dan menggabungkan namun secara terurut untuk membantu proses
visualisasi #
    first_Second_Hull=firstSecondHull(hull)
    upHull=[]
    downHull=[]
    for j in range(1,len(hull)-1):
        if(upperSide(first_Second_Hull[0],first_Second_Hull[-1],hull[j]) is
True):
            upHull.append(hull[j])
        if(lowerSide(first_Second_Hull[0],first_Second_Hull[-1],hull[j]) is
True):
            downHull.append(hull[j])
    downHull.append(first_Second_Hull[0])
    upHull.append(first_Second_Hull[0])
    downHull.append(first_Second_Hull[-1])
    upHull.append(first_Second_Hull[-1])
    upHull=sorted(upHull)
    downHull=sorted(downHull,reverse=True,key=lambda x:x[0])
    combine=upHull+downHull
    plot=[getX(combine)]+[getY(combine)]
    return(plot)
def getX(hull) :
# Mengambil nilai x dari (x,y) #
    hull_x=[]
    for e in hull :
        hull_x.append(e[0])
    return hull_x
def getY(hull):
# Mengambil nilai y dari (x,y) #
    hull_y=[]
    for e in hull :
        hull_y.append(e[1])
    return hull_y
def displayConvexHull(df,a,b,hull):
# Visualisasi Convex Hull dengan pustaka myConvexHull
  plt.figure(figsize = (10, 6))
  colors = ['b','r','g','c','m','y','k','w']
  plt.title(data.feature_names[a] + ' vs ' + data.feature_names[b])
  plt.xlabel(data.feature_names[a])
```

```
  plt.ylabel(data.feature_names[b])
  for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[a,b]].values
    hull=myConvexHull(bucket)
    hull= splitandMergeHull(hull)
    print(hull)
    plt.scatter(getX(bucket), getY(bucket), label=data.target_names[i])
    plt.plot(hull[0],hull[-1],colors[i%8])
  plt.legend()
hull=[]
data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
displayConvexHull(df,0,1,hull)
```

2.4 Penjelasan Fungsi-fungsi yang terdapat pada Program

2.4.1 Fungsi Utama

Fungsi utama yang dimaksud disini adalah yang menggambarkan divide and conquer,yakni ada yang bagian memecah persoalan menjadi beberapa persoalan,lalu fungsi yang memecahkan persoalan,lalu fungsi yang mengkombinasi persoalan tersebut.

2.4.1.1 Fungsi myConvexHull dengan parameter s berupa list of points

Fungsi ini akan menerima inputan berupa list of list yang berisikan point-point(himpunan dari titik) yang akan dibuat convex hullnya. Pada fungsi ini himpunan point yang mungkin saja berbentuk array numpy akan di konversi dulu menjadi bentuk yang sesuai dengan parameter dengan bantuan fungsi bawaan array.tolist(). Langkah kerja awal dari fungsi ini adalah dengan mencari titik dengan absis paling kecil dan titik dengan absis paling besar, yakni dengan bantuan fungsi sortByXCoord(). Setelah ditemukan 2 titik tersebut, 2 titik tersebut akan dimasukan ke list yang akan memuat semua hull,karena dipastikan 2 titik tersebut merupakan hull dari himpunan tersebut. Lalu dilanjutkan dengan mencari hull berikutnya, yakni dengan memetakan apakah titik lainnya berada di atas atau di bawah garis yang terbentuk diantara 2 titik yang tadi ditemukan. Proses mencari ini dengan memanfaatkan fungsi upperSide() dan lowerSide() untuk memeriksa letak titik. Lalu titik yang berada di bawah garis tersebut akan diletakkan di list baru yang akan menampung,begitu juga dengan yang berada di atas garis tersebut. Selanjutnya 2 array baru itu akan masuk ke dalam proses pencarian hull berikutnya, pada langkah ini akan memanggil fungsi findHull(). Proses yang terjadi pada findHull() akan dijelaskan pada bagian fungsi terkait.

2.4.1.2 Fungsi FindHull dengan parameter sk merupakan list of points, p : titik dengan absis terdekat, q: titik dengan absis terjauh

Fungsi ini merupakan fungsi yang akan mencari semua hull yang akan membentuk polygon dimana semua titik akan berada di dalam polygon tersebut. Pada fungsi ini, hal pertama yang akan dilakukan adalah mencari titik terjauh dari subset titik yang terletak dibagian atas maupun bagian bawah. Tujuan mencari titik terjauh adalah agar bisa menemukan hull yang merupakan titik terjauh tersebut, serta agar dapat membentuk segitiga untuk kembali memecah menjadi subset terkecil dan akan terus berlanjut hingga set yang dimasukan pada fungsi findHull merupakan list kosong(panjang elemen==0). Mencari titik terjauh cukup dengan memanggil fungsi titik_terjauh(). Selanjutnya setelah menemukan 3 titik, kita akan mengecek apakah 3 titik itu akan membentuk segitiga, selanjutnya kita akan memeriksa apakah titik yang terdapat dalam set yang dijadikan parameter tersebut terletak di dalam atau diluar segitiga. Pada algoritma ini,kedua proses itu dapat dilakukan dengan cukup memanggil fungsi isInsideTriangle(). Lalu akan dilanjutkan dengan mengelompokkan kembali ke subset berikutnya dengan memanfaatkan fungsi upperSide() dan lowerSide(), untuk memeriksa letak titik.

2.4.1.3 Fungsi splitandMergeHull dengan parameter list dari hull yang telah dikumpulkan

Fungsi ini akan membagi hull yang sudah terkumpul dalam satu list of list ke dalam 2 bagian,yakni upHull dan downHull(keduanya tetap merupakan list of list). Pada fungsi ini proses di awali dengan kembali memeriksa letak titik dengan memanfaatkan fungsi upperSide() dan lowerSide(). Lalu memasukkannya sesuai pengelompokkan ke upHull dan downHull. Selanjutnya,karena titik dengan maksimum absis dan minimum absis tadi belum masuk ke bagian manapun, masukkan keduanya ke tiap bagian(untuk mempermudah proses plotting). Selanjutnya, kita ingin memecah hull yang telah terurut tadi menjadi bagian absis dan bagian ordinat. Alasannya adalah agar lebih mudah saat melakukan plotting. Namun sebelum itu, karena pada saat plotting kita harus melakukan plotting dari satu titik ke titik lainnya yang sudah terurut,disini saya akan mengurutkan berdasarkan absis,yakni satu bagian diurutkan ascending berdasarkan absis dan satu bagian diurutkan descending berdasarkan absis. Saat memecah akan dimanfaatkan fungsi getX() dan getY(). Lalu setelah dipecah akan terbentuk bagian absis yang sudah terurut serta bagian ordinat yang sudah terurut,lalu masukkan kembali ke satu bagian yang akan di return oleh fungsi ini.

2.4.1.4 Fungsi visualisasiConvexHull dengan parameter df,a,b,hull

Fungsi ini akan membantu dalam visualisasi convex hull yang telah ditemukan solusinya. Fungsi ini memiliki parameter df merupakan data frame yang telah dibuat berdasarkan dataset yang telah ditentukan, a dan b merupakan nomor kolom yang akan digunakan,serta hull merupakan list of points dari convex hull yang telah ditemukan.

2.4.2 Fungsi Pendukung

2.4.2.1 Fungsi getX dengan parameter list dari hull yang telah dikumpulkan(hull)

Fungsi ini hanya akan membantu dalam memisahkan list of points menjadi list of absis.

2.4.2.2 Fungsi getY dengan parameter list dari hull yang telah dikumpulkan(hull)

Fungsi ini hanya akan membantu dalam memisahkan list of points menjadi list of ordinat.

2.4.2.3 Fungsi sisiTitik dengan parameter 2 titik (minimum absis dan maximum absis) dan 1 titik checking(s1,s2,z)

Fungsi ini akan mereturn nilai determinan dengan memanfaatkan materi aljabar geometri pada semester 3 sebelumnya.

2.4.2.4 Fungsi upperSide dengan parameter 2 titik (minimum absis dan maximum absis) dan 1 titik checking(s1,s2,z)

Fungsi ini akan memeriksa apakah titik terletak di atas 2 titik hull awal (titik minimum absis dan maximum absis). Proses pemeriksaan dilakukan dengan bantuan fungsi sisiTitik() yang akan mereturn nilai determinan.

2.4.2.5 Fungsi lowerSide dengan parameter 2 titik (minimum absis dan maximum absis) dan 1titik checking(s1,s2,z)

Fungsi ini akan memeriksa apakah titik terletak di atas 2 titik hull awal (titik minimum absis dan maximum absis). Proses pemeriksaan dilakukan dengan bantuan fungsi sisiTitik() yang akan mereturn nilai determinan.

2.4.2.6 Fungsi segaris dengan parameter 2 titik (minimum absis dan maximum absis) dan 1titik checking(s1,s2,z)

Fungsi ini akan memeriksa apakah titik terletak di antara 2 titik hull awal (titik minimum absis dan maximum absis). Proses pemeriksaan dilakukan dengan bantuan fungsi sisiTitik() yang akan mereturn nilai determinan.

2.4.2.7 Fungsi gradien dengan parameter 2 titik (s1,s2)

Fungsi ini akan menghitung gradien antara 2 titik yang akan membentuk garis dengan rumus jarak gradien seperti yang telah diketahui yakni deltay/deltax.

2.4.2.8 Fungsi jarakTitik dengan parameter 2 titik (p1,p2)

Fungsi ini akan menghitung jarak antara 2 titik dengan rumus jarak 2 titik seperti yang telah diketahui yakni akar dari jumlah kuadrat dari selisih absis dan selisih ordinat(sqrt((deltax)^2+(deltay)^2) .

2.4.3.9 Fungsi jarakTitikKeGaris dengan parameter titik dan garis(point,line)

Fungsi ini akan menghitung jarak terdekat antara titik ke garis([[a,b],[b,c]]) dengan memanfaatkna konsep kedua garis yang berpotongan. Namun setelah diturunkan rumusnya didapatkan bahwa rumusnya adalah jarak antara (x1,y1) dengan Ax+By+C=0 adalah (x1*A+y1*B+C) /sqrt(A^2+B^2).

2.4.3.10 Fungsi titik_terjauh dengan parameter list of points, 2 titik yang akan membentuk garis (s,p,q)

Fungsi ini akan mencari titik terjauh dari 2 titik yang akan membentuk garis. Proses ini diawali dengan mengurutkan list of points berdasarkan nilai absis(terurut membesar). Lalu akan dilakukan proses pencarian titik terjauh dengan memanfaatkan fungsi jarakTitikKeGaris().

2.4.3.11 Fungsi isTriangle dengan parameter 3 titik sudut(A,B,C)

Fungsi ini akan mengembalikan True jika 3 titik membentuk segitiga, dan false jika tidak. Dimulai dengan mengecek apakah 3 titik segaris atau tidak dan mengecek syarat segitiga, yakni sisi terpanjang<penjumlahan 2 sisi lainnya.

2.4.3.3 Fungsi LuasSegitiga dengan parameter 3 titik sudut(A,B,C)

Fungsi ini akan menghitung luas segitiga yang terbentuk oleh 3 titik sudut. Pada fungsi ini akan dimanfaatkan rumus heron yang bermanfaat untuk menghitung luas segitiga jika diketahui 3 titik sudut. Langkah awal adalah dengan menghitung keliling dan panjang 3 garis yang terbentuk oleh 3 titik tersebut. Rumus heron = sqrt (keliling/2 * (keliling/2-sisi1)* (keliling/2-sisi2)*(keliling/2-sisi3))

2.4.3.4 Fungsi isInsideTriangle dengan parameter 4 titik(3 titik segitiga serta 1 titik checking)

Fungsi ini hanya akan memeriksa apakah titik terletak di dalam segitiga atau tidak. Ide awalnya adalah dengan memeriksa nilai determinan titik terhadap 3 garis yang terbentuk oleh 3 titik.

2.4.3.5 Fungsi sortByXCoord dengan parameter list of points

Fungsi ini hanya akan membantu mengurutkan points yang berbentuk list of list berdasarkan absisnya terurut membesar.

2.4.3.6 Fungsi sortByYCoord

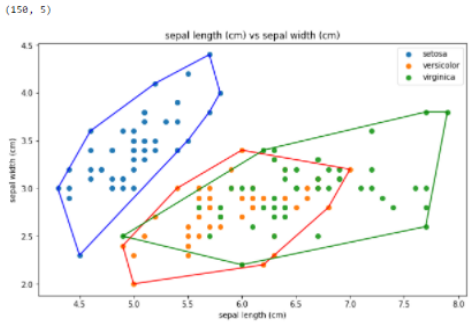
Fungsi ini hanya akan membantu mengurutkan points yang berbentuk list of list berdasarkan absisnya terurut membesar.

2.4.3.6 firstSecondHull dengan parameter list of points

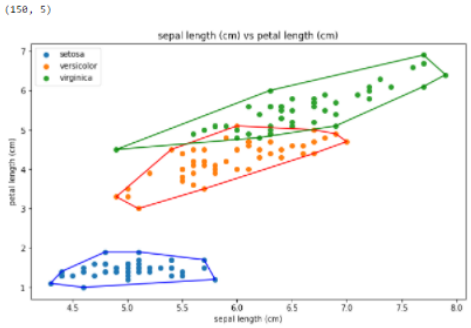
Fungsi ini akan membantu mencari 2 titik yang digunakan sebagai hull pertama dan kedua yang akan membagi himpunan menjadi 2 bagian.

2.5 Bukti dari Hasil Pustaka myConvexHull

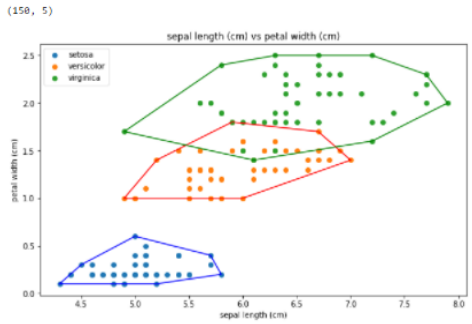
2.5.1 Dataset Iris



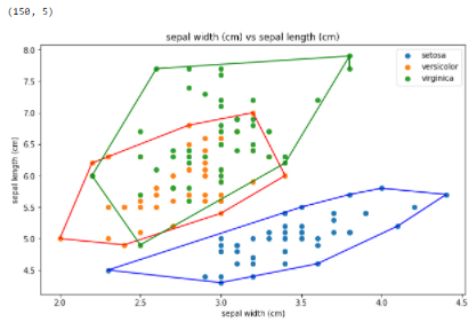
Gambar 2.1 Sepal length(cm) vs sepal width(cm)



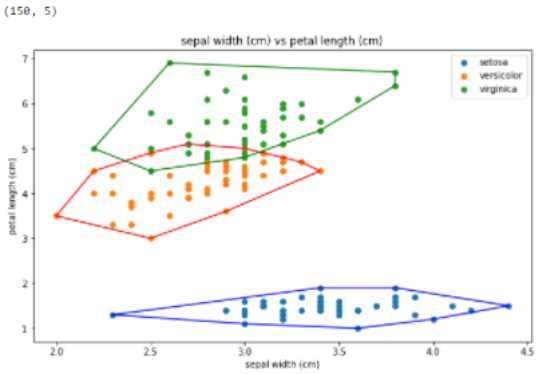
Gambar 2.2 Sepal length(cm) vs petal length(cm)



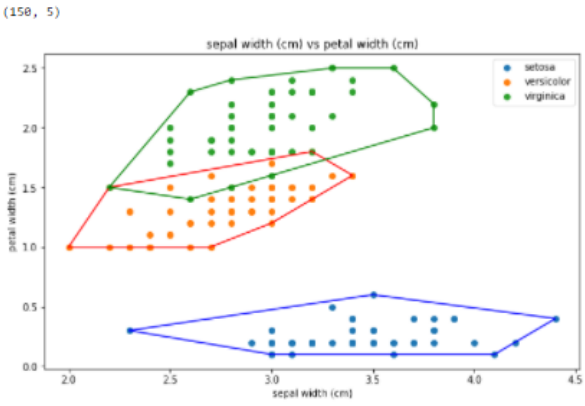
Gambar 2.3 Sepal length(cm) vs petal width(cm)



Gambar 2.4 Sepal width(cm) vs sepal length(cm)

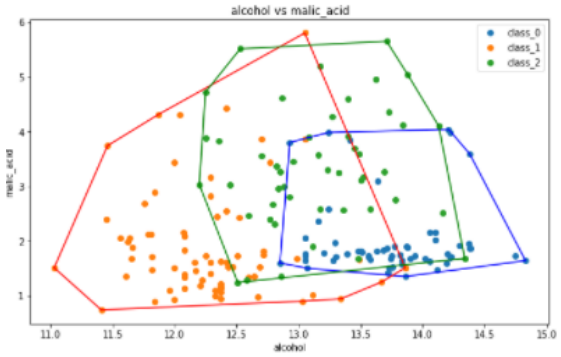


Gambar 2.5 Sepal width(cm) vs petal length(cm)

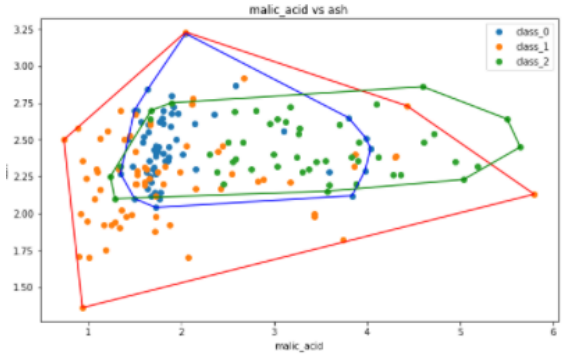


Gambar 2.6 Sepal width(cm) vs petal width(cm)

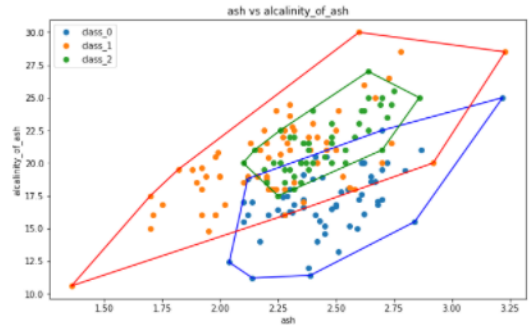
2.5.2 Dataset Wine



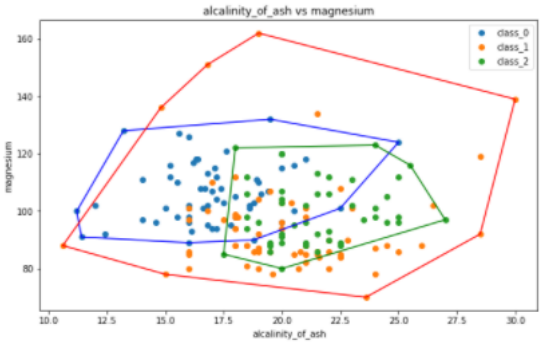
Gambar 2.7 alcohol vs malic_acid



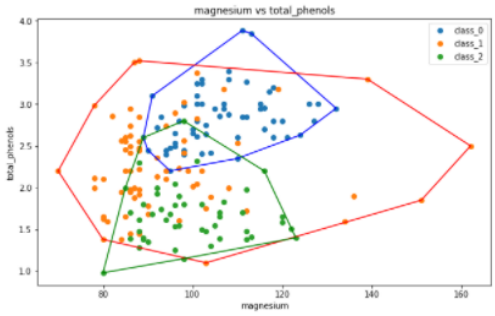
Gambar 2.8 malic acid vs ash



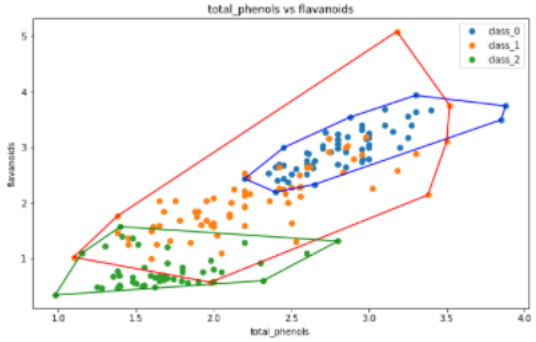
Gambar 2.9 ash vs alcalinity_of_ash



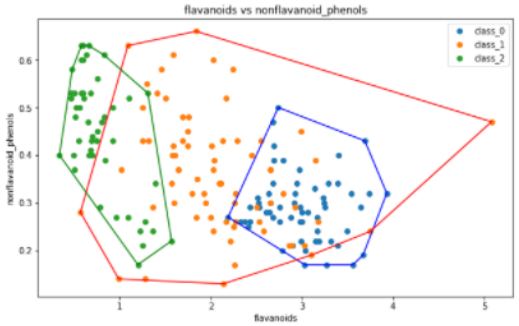
Gambar 2.13alcalinity_of_ash vs magnesium



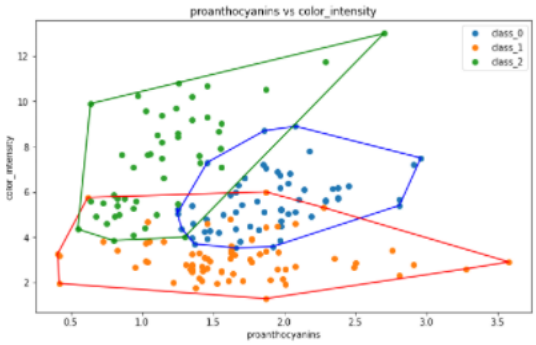
Gambar 2.14 magnesium vs total phenols



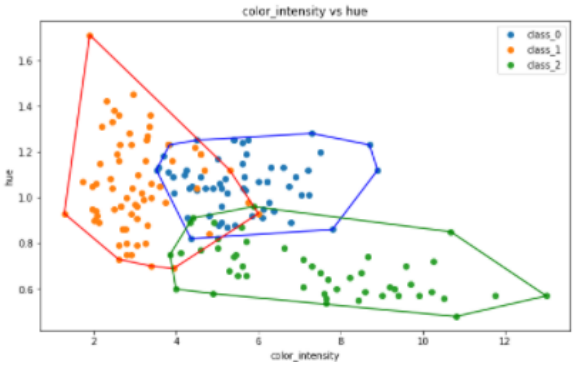
Gambar 2.15 total phenols vs flavanoids



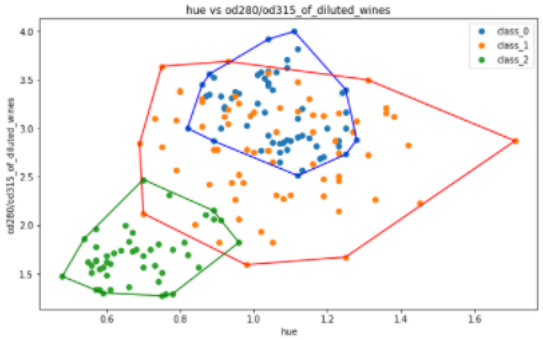
Gambar 2.10 flavanoids vs flavanoids_phenols



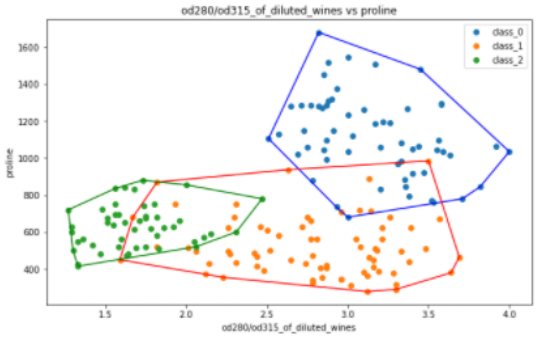
Gambar 2.11 proanthocyanins vs color_intensitiy



Gambar 2.11 color_intensitiy vs hue

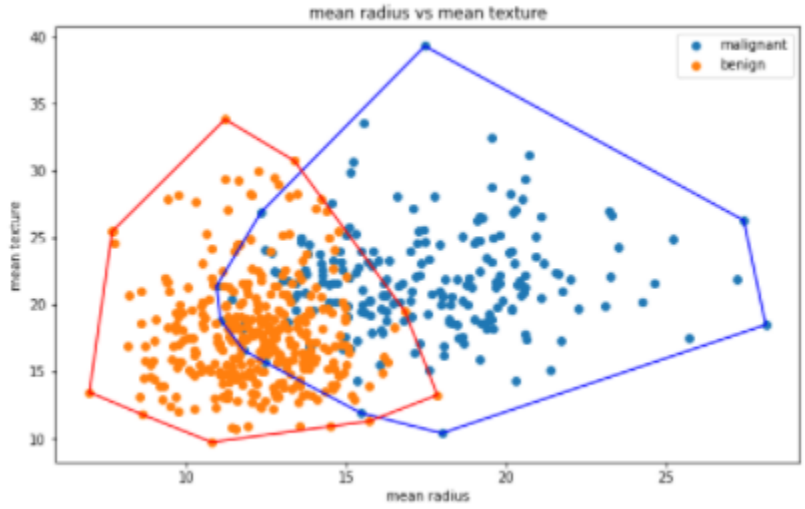


Gambar 2.12 hue vs of_diluted_wines

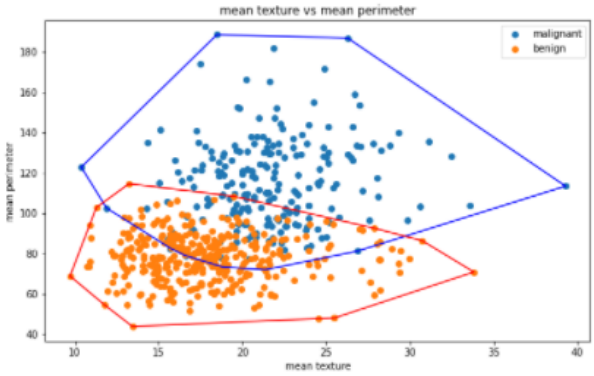


Gambar 2.17 of_diluted_wines vs proline

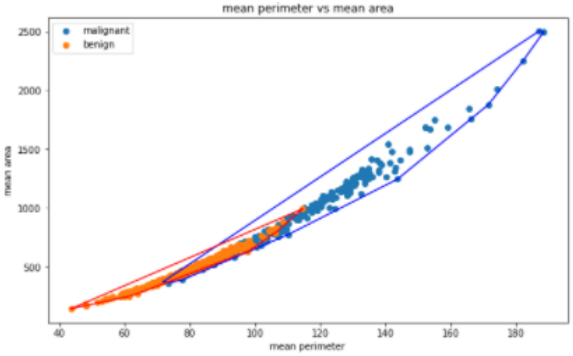
2.5.3 Dataset breast_cancer



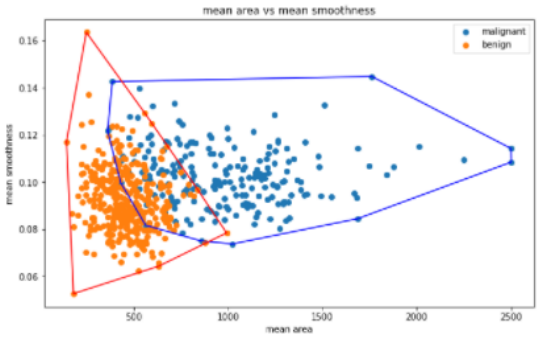
Gambar 2.18 mean radius vs mean texture



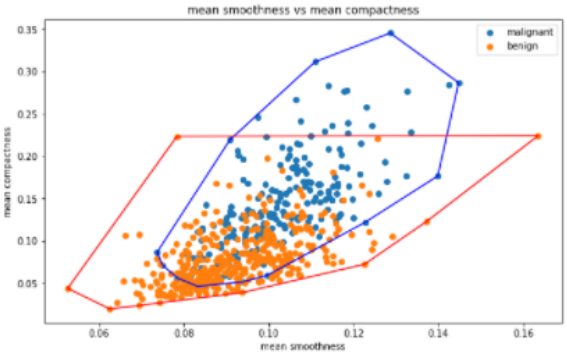
Gambar 2.19 mean texture vs mean perimeter



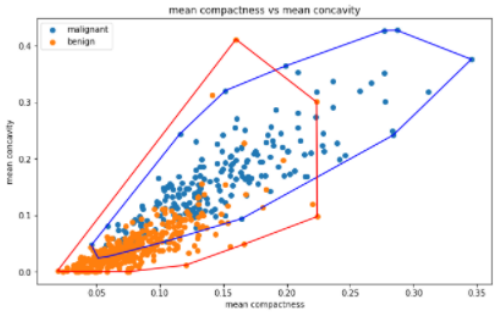
Gambar 2.20 mean perimeter vs mean area



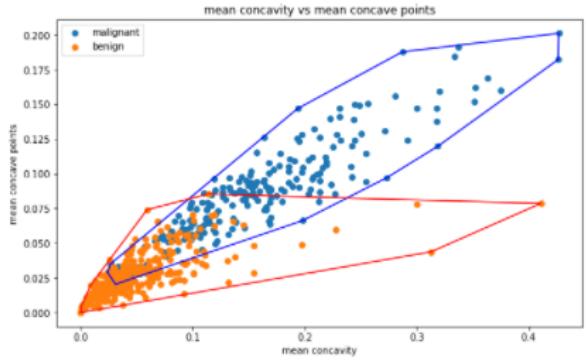
Gambar mean area vs mean smoothness



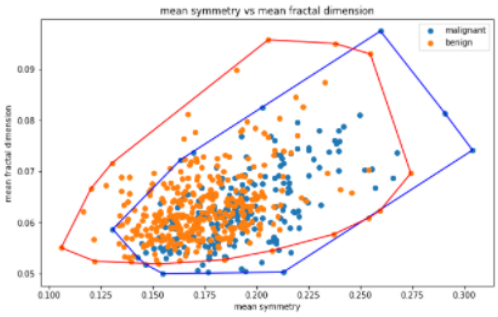
Gambar mean smoothness vs mean compactness



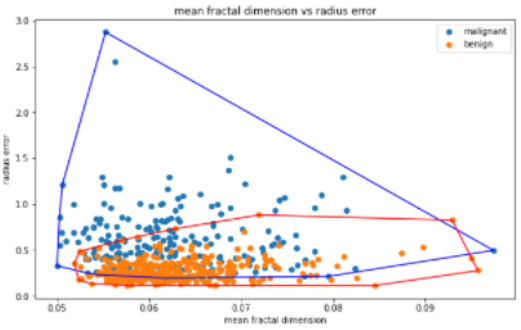
Gambar 2.21mean compactness vs mean concavity



Gambar 2.22 mean concavity vs mean concave points

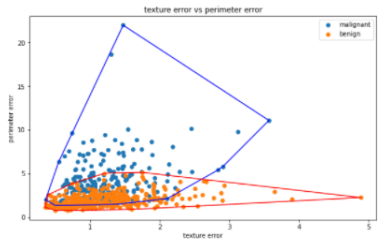


Gambar 2.23mean symmetry vs mean fractal dimension

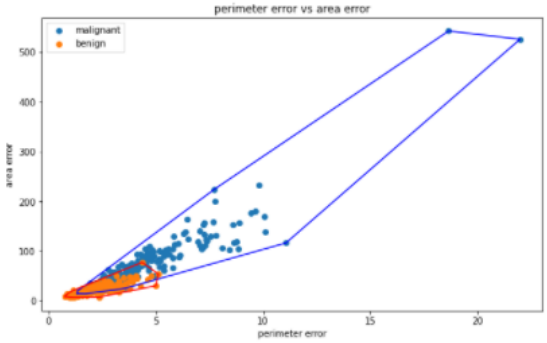


Gambar 2.24mean fractal dimension vs radius error

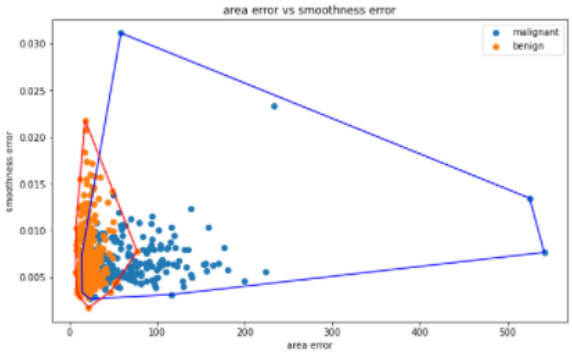
Gambar 2.25 radius error vs texture error



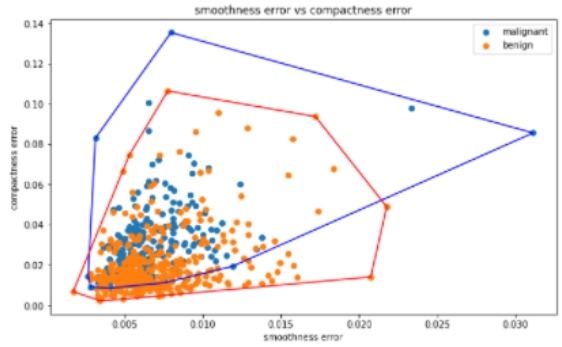
Gambar 2.26 texture error vs perimeter error



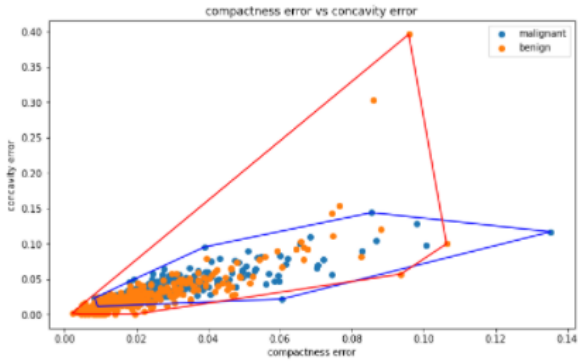
Gambar 2.27 perimeter error vs area error



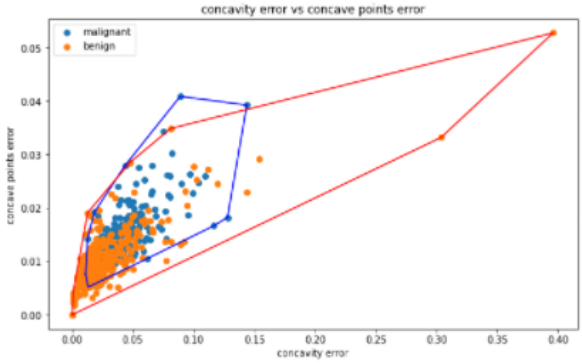
Gambar 2.28 area error vs smoothness error



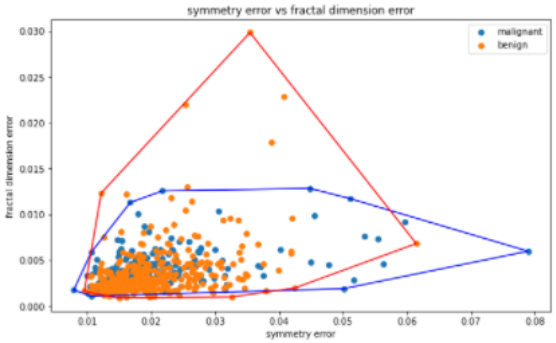
Gambar 2.29 smoothness error vs compactness error



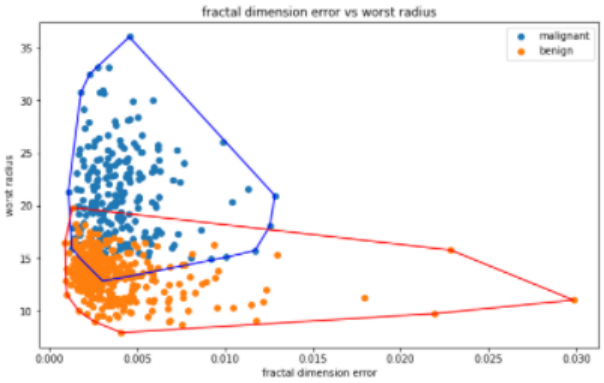
Gambar 2.30 compactness error vs concavity error



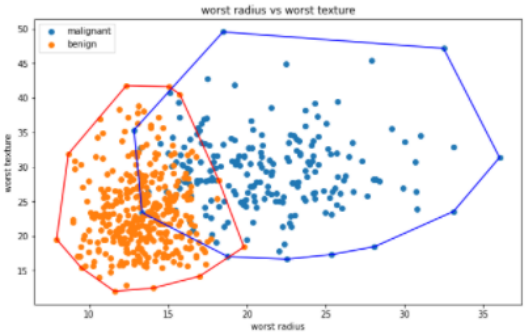
Gambar 2.31 concavity vs concave points error



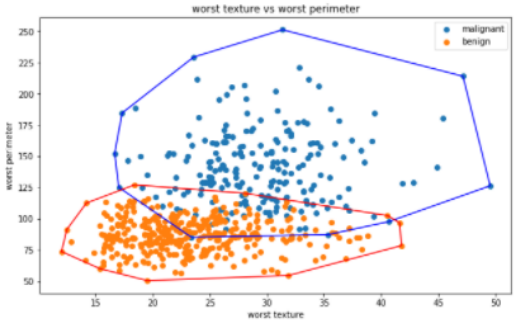
Gambar 2.32 symetri error vs fractal dimension error



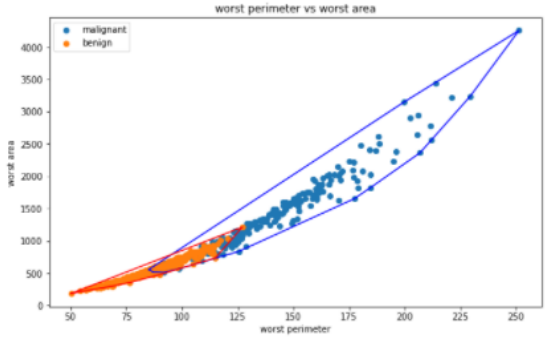
Gambar 2.33 fractal dimension error vs worst radius



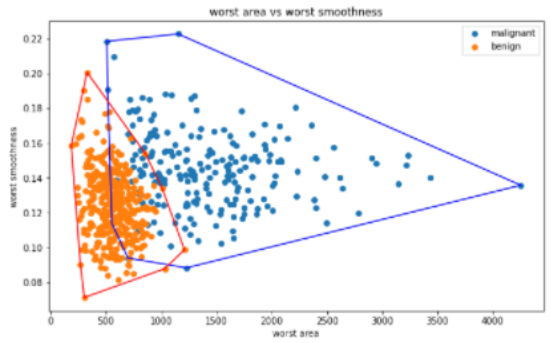
Gambar 2.34 worst radius vs worst texture



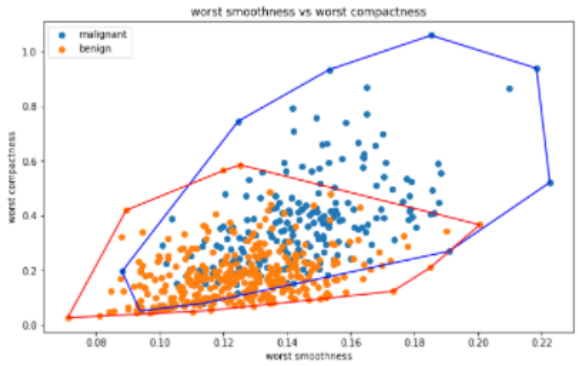
Gambar 2.35 worst radius vs worst texture



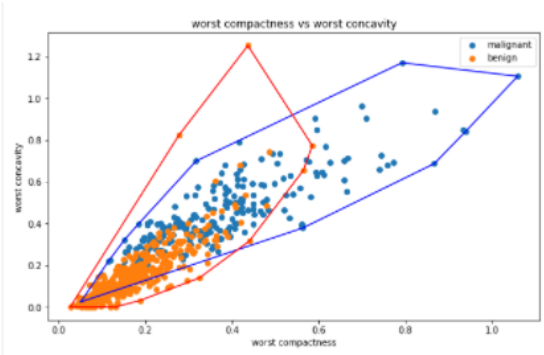
Gambar 2.36 worst perimeter vs worst area



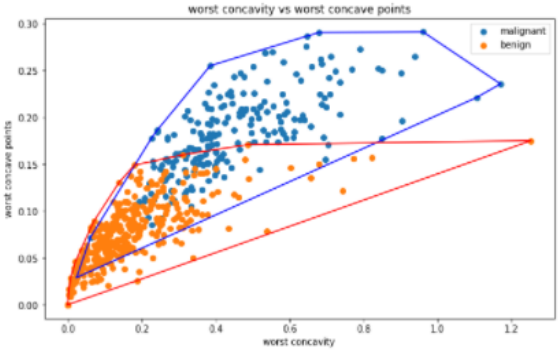
Gambar 2.37 worst area vs worst smoothness



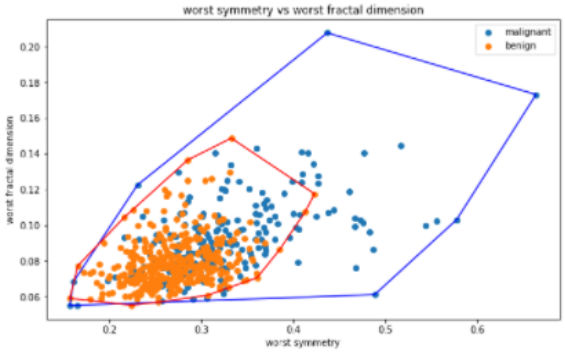
Gambar 2.38 worst smoothness vs worst compactness



Gambar 2.39 worst compactness vs worst concavity



Gambar 2.40 worst concavity vs worst concave points



Gambar 2.41 worst symmetry vs worst fractal dimensi

BAB III

Penutup

3.1 Saran

Tidak ada saran untuk tugas kecil 2 ini, spesifikasi sudah cukup, waktu pengerjaan sudah lebih dari cukup. Terimakasih saya ucapkan kepada tim pengajar,yakni dosen dan juga para asisten yang sudah bekerja dengan sangat baik.

3.2 Kesimpulan

Divide and conquer cukup efektif dalam menyelesaikan persoalan,yakni dengan membagi persoalan,menyelesaikan setiap sub persoalan dan menyatukan solusi sub persoalan

DAFTAR PUSTAKA

“Algoritma Divide and Conquer.” *Informatika*, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf. Accessed 24 February 2022.

“Algoritma Divide and Conquer.” *Informatika*, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian2.pdf. Accessed 24 February 2022.

“Algoritma Divide and Conquer.” *Informatika*, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian3.pdf. Accessed 24 February 2022.

“Algoritma Divide and Conquer.” *Informatika*, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian4.pdf. Accessed 24 February 2022.

“Homepage Rinaldi Munir.” *Informatika*, https://informatika.stei.itb.ac.id/~rinaldi.munir/. Accessed 24 February 2022.

“Matplotlib Colors.” *Matplotlib*, https://matplotlib.org/2.0.1/api/colors_api.html. Accessed 26

“NumPy Documentation.” *NumPy*, https://numpy.org/doc/. Accessed 28 February 2022.

February 2022.

Lampiran

Link Repository : https://github.com/StevenSiahaann/Tucil2_13520145

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