Laporan Tugas Kecil Strategi Algoritma 2021/2022 Convex Hull

Disusun Oleh:

Andreas Indra Kurniawan 13520091/K-1

Algoritma Divide and Conquer

Secara garis besar algoritma divide and conquer membagi 2 sebuah data yang akan diproses(divide) dan menyelesaikannya satu persatu(conquer). Biasanya algoritma ini akan menggunakan rekursif karena tidak diketahui perlu dilakukan pembagian berapa kali untuk mencapai base case dimana data akan diproses lebih lanjut.

Langkah-Langkah Algoritma

1. Lakukan sort pada array titik(berdasarkan kolom 0 lalu kolom 1)
2. Elemen array pertama dan terakhir menjadi garis pertama
3. Garis pertama ini akan membagi 2 titik-titik ke array S1 dan S2
4. Dilakukan proses rekursi pada fungsi nextHull
5. Dicari titik terjauh dari garis
6. Buat 2 garis baru dengan titik kiri awal dan titik baru sebagai garis baru 1 dan titik kanan awal dan titik baru sebagai garis 2 baru
7. Hapus garis pertama dari array
8. Bagi 2 kembali titik-titik yang ada berdasarkan garis-garis baru dan ulangi proses rekursi.

Kode Program

```
import numpy as np
import numpy.linalg as lg
def myConvexHull(listTitik):    #MENENTUKAN TITIK AWAL HULL
    hull = []
    panjangList = len(listTitik)-1
    hull.append([[listTitik[0,0],listTitik[0,1]],[listTitik[panjangList,0],lis
tTitik[panjangList,1]]])      #MENENTUKAN GARIS AWAL BERDASARKAN SUMBU X
PALING KIRI DAN KANAN
    listTitik = np.delete(listTitik,0, axis = 0)
    listTitik = np.delete(listTitik,len(listTitik)-1, axis=0)
    S1 = []
    S2 = []

    for i in listTitik:          #MEMBAGI 2 SISI KIRI DAN KANAN TITIK
        kuadran = sisiKiri(hull[0][0],hull[0][1],[i[0,0],i[0,1]])
        if( kuadran == 1):
            S1.append([i[0,0],i[0,1]])
        elif(kuadran == -1):
            S2.append([i[0,0],i[0,1]])

    kiri = hull[0][0]
    kanan = hull[0][1]
    nextHull(S1,hull,kiri,kanan)
    nextHull(S2,hull,kanan,kiri)

    return hull

def nextHull(S,hull,kiri,kanan):    #MENENTUKAN TITIK SELANJUTNYA DARI HULL
DENGAN CARA REKURSIF
```

```
    if(len(S) != 0):
        kiri = np.array(kiri)
        kanan = np.array(kanan)
        cS = np.array(S)
        maksHull = cS[0]

        if(len(S) != 1):            #JIKA SISA SATU TITIK TIDAK PERLU
DIPERIKSA
            maks = jarakPerpLine(kiri,kanan,cS[0])  #MENGECEK TITIK YANG
JARAKNYA PALING JAUH DARI GARIS
            for i in cS:
                maksLok = jarakPerpLine(kiri,kanan,i)
                if(maks<maksLok):
                    maks = maksLok
                    maksHull = i

        maksHull = [maksHull[0],maksHull[1]]
        calonHapus = [[kiri[0],kiri[1]],[kanan[0],kanan[1]]]
        index = 0
        ketemu = False
        for i in hull:          #MENCARI INDEX DARI GARIS YANG AKAN DIGANTI
            if(calonHapus== i):
                ketemu = True
                break
            else:
                index += 1

        if(ketemu):             #JIKA KETEMU GARIS AKAN DITIMPA
            hull[index] = [[kiri[0],kiri[1]],maksHull]
            hull.append([maksHull,[kanan[0],kanan[1]]])
        else:
            hull.append([maksHull,[kanan[0],kanan[1]]])
            hull.append([[kiri[0],kiri[1]],maksHull])


        S1 = []
        S2 = []
        if(len(cS) != 1):
            for i in cS:                                #MEMBAGI 2 SISI KIRI
DAN KANAN TITIK LALU MENGAMBIL SISI KIRINYA SAJA
                iFunc = [i[0],i[1]]
                kuadran = sisiKiri([kiri[0],kiri[1]],maksHull,iFunc)     #S1
BERISI SISI KIRI DARI GARIS PERTAMA
                if( kuadran == 1):                      #S2 BERISI SISI KIRI
DARI GARIS KEDUA
                    S1.append(iFunc)                        #GARIS PERTAMA
TERBENTUK DARI TITIK KIRI DAN TITIK HULL YANG BARU
```

```
                kuadran =
sisiKiri(maksHull,[kanan[0],kanan[1]],iFunc)    #GARIS KEDUA TERBENTUK DARI
TITIK HULL BARU DAN TITIK KANAN
                if(kuadran == 1):
                    S2.append(iFunc)
        nextHull(S1,hull,kiri,maksHull)
        nextHull(S2,hull,maksHull,kanan)
        return
    else:
        return



def sisiKiri(a,b,c):
    det = a[0]*b[1] + c[0]*a[1] + b[0]*c[1] - c[0]*b[1] - b[0]*a[1] -
a[0]*c[1]
    if(det>0):
        return 1
    elif (det==0):
        return 0
    else:
        return -1

def jarakPerpLine(a,b,c):       #MENCARI JARAK ANTARA TITIK DAN GARIS
    jarak = lg.norm(np.cross(b-a,a-c)/lg.norm(b-a))
    return jarak
```

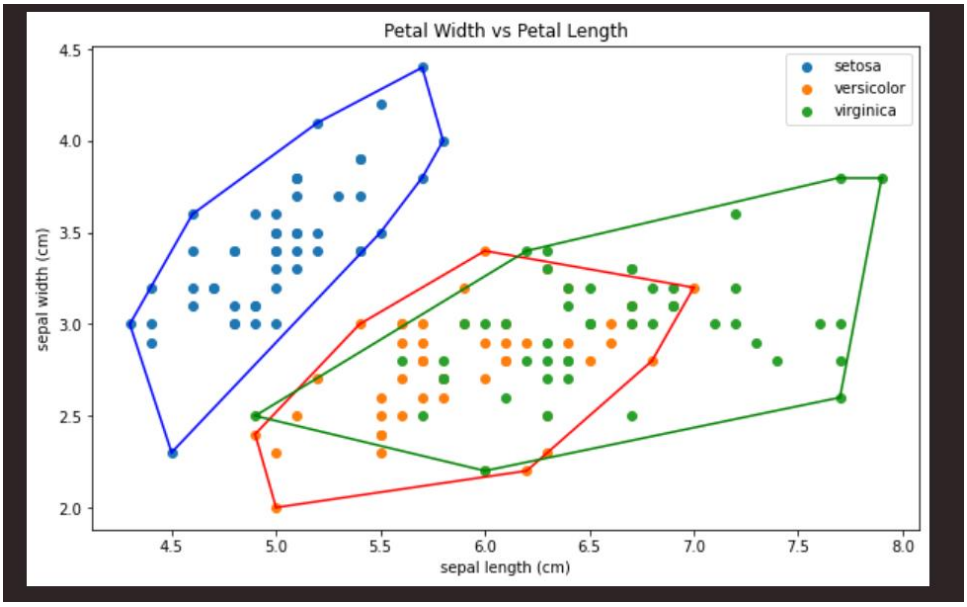
Skrinsut Input-Output Program

1. Input dataset iris(lebar petal dan panjang petal, total barisxkolom = 150x5)

```
Besar data:  (150, 5)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

2. Output dataset iris



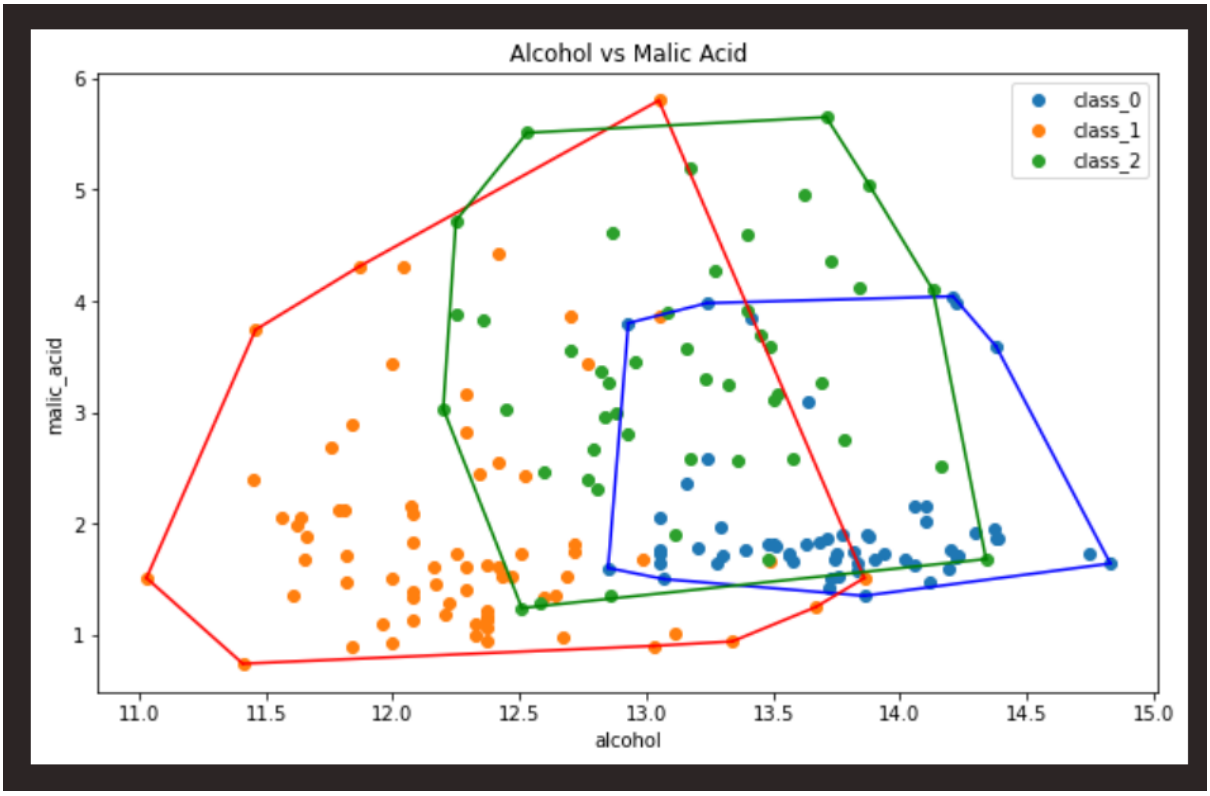
3. Input dataset wine(Alkohol dan Malic Acid, total barisxkolom = 178x14)

```
Besar data:  (178, 14)
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	co
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	

proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline	Target
2.29	5.64	1.04	3.92	1065.0	0
1.28	4.38	1.05	3.40	1050.0	0
2.81	5.68	1.03	3.17	1185.0	0
2.18	7.80	0.86	3.45	1480.0	0
1.82	4.32	1.04	2.93	735.0	0

4. Output dataset wine


Alcohol vs Malic Acid
class_0
class_1
class_2
malic_acid
alcohol
6
5
4
3
2
1
11.0
11.5
12.0
12.5
13.0
13.5
14.0
14.5
15.0

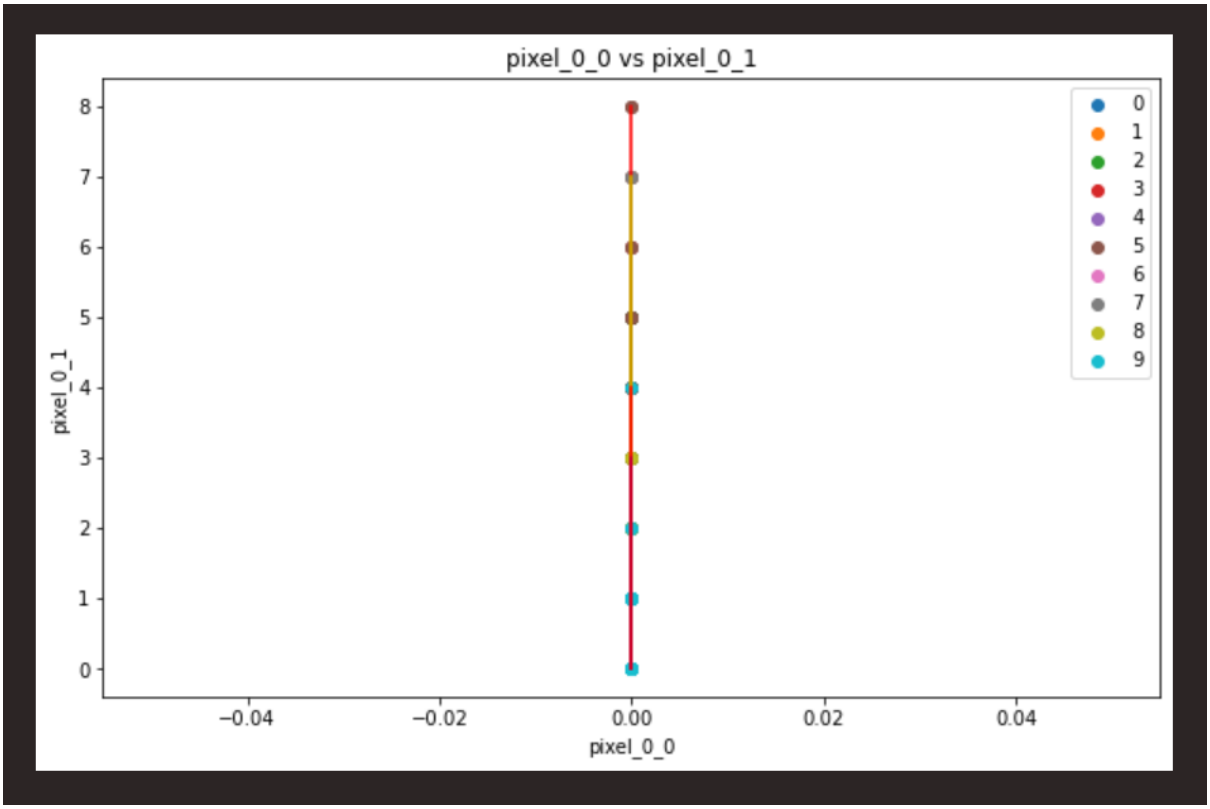
5. Input dataset digits(pixel_0_0 dan pixel_0_1, total barisxkolom = 1797x65)

```
Besar data:  (1797, 65)
```

	pixel_0_0	pixel_0_1	pixel_0_2	pixel_0_3	pixel_0_4	pixel_0_5	pixel_0_6	pixel_0_7	pixel_1_0	pixel_1_1	...	pixel_6_7	pixel_7_0
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	0.0	0.0
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	0.0	0.0
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0

pixel_7_0	pixel_7_1	pixel_7_2	pixel_7_3	pixel_7_4	pixel_7_5	pixel_7_6	pixel_7_7	Target
0.0	0.0	6.0	13.0	10.0	0.0	0.0	0.0	0
0.0	0.0	0.0	11.0	16.0	10.0	0.0	0.0	1
0.0	0.0	0.0	3.0	11.0	16.0	9.0	0.0	2
0.0	0.0	7.0	13.0	13.0	9.0	0.0	0.0	3
0.0	0.0	0.0	2.0	16.0	4.0	0.0	0.0	4

6. Output dataset digits



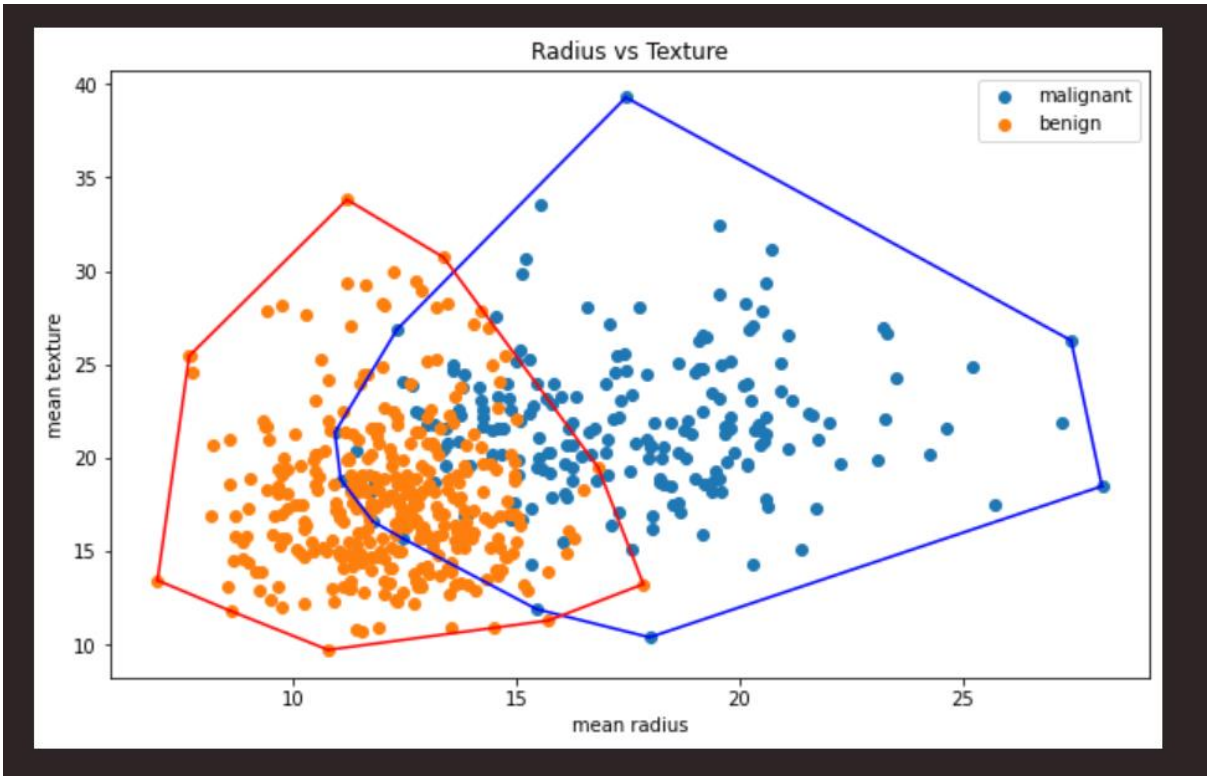
7. Input dataset breast cancer(radius dan texture, total barisxkolom = 569x31)

```
Besar data:  (569, 31)
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20

worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	Target
184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890	0
158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902	0
152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758	0
98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300	0
152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678	0

8. Output dataset breast cancer



Poin	Ya	Tidak
1. Pustaka myConvexHull berhasil dibuat dan 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	

Link github:

https://github.com/IMYELI/Tugas-Kecil-2-Stima-Convex-Hull.git