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:

Hilda Carissa 13520164

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

DAFTAR ISI

Algoritma Divide and Conquer

Pada implementasi convex hull ini, digunakan algoritma divide and conquer. Algoritma *divide and conquer* sendiri memiliki tiga poin utama. Ada *divide*, *conquer,* dan *combine*. *Divide*, seperti artinya yaitu bagi, berarti membagi persoalan menjadi upa-persoalan yang memiliki jenis persoalan yang mirip dengan persoalan semula, namun memiliki ukurang yang lebih kecil. Kemudian ada *conquer*, yaitu menyelesaikan masing-masing upa-persoalan yang ada secara langsung jika berukuran kecil dan secara rekursif jika berukuran besar, Terakhir ada *combine,* langkah terakhir yang berarti menyatukan solusi-solusi dari upa-persoalan sehingga membentuk sebuah solusi utama yang dapat menyelesaikan persoalan semula.

Dalam program yang dibuat disini, algoritma ini tentu saja diterapkan. Pada awalnya, himpunan titik-titik yang diberikan akan diurutkan terlebih dahulu dari paling kecil ke paling besar, setelah itu diambil dua titik paling ujung kiri dan kanan yang kemudian dinamakan p1 dan pn. Kedua titik ini akan menjadi titik utama yang membentuk garis. Setelah garis terbentuk, titik-titik lain yang ada di himpunan akan dibagi ke dua himpunan berbeda, satu yang berada di sebelah kiri garis utama, satu yang ada di sebelah kanan garis utama.

Setelah dibagi menjadi dua bagian, dicarilah Hull dari masing-masing bagian dengan menggunakan algoritma *divide and conquer*. Saat masuk ke fungsi findHull, pertama ditentukan terlebih dahulu titik terjauh dari garis, kemudian masukkan titik terjauh ini ke dalam himpunan solusi. Setelah itu, tarik garis dari titik p1 ke titik terjauh, dan titik terjauh ke pn. Hal ini kemudian akan membuat semacam segitiga pada titik-titik yang tersebar tersebut. Selanjutnya, masukkan titik-titik yang masih berada di luar segitiga yang terbetnk ke dalam sebuah himpunan misalnya S11 untuk titik di sebelah kiri garis p1-titik terjauh dan S12 untuk titik-titik di sebelah kanan pn-titik terjauh, lalu kemudian gunakan kembali fungsi findHull untuk kedua himpunan titik yang masih ada ini. Fungsi findHull akan terus dilakukan hingga sudah tidak ada lagi titik yang berada di luar segitiga.

Ketika titik yang berada di luar segitiga sudah habis, maka fungsi findHull akan mengembalikan titik-titik terjauh atau bisa dibilang titik terluar dari himpunan titik-titik ini. Maka didapatlah Hull untuk himpunan titik. Setelah itu, ditariklah garis dari p1 ke titik-titik terjauh atau Convex Hull dari himpunan titik awal. Pada akhirnya, garis yang dibentuk bisa dibilang melingkari titik-titik lain yang ada di dalamnya.

Kode Program

```python
def myConvexHull(sorted_arr):
    #sort the array
    p1 = 0
    pn = len(sorted_arr)-1

    #list index titik di sorted_arr yang bisa jadi convexhull
    toBeHull = []

    #eliminate index titik yang gamungkin convexhull (karena di garis p1pn)
    i = 1
    while(i != len(sorted_arr)-1):
        right = (sorted_arr[pn][1]-sorted_arr[p1][1])*(sorted_arr[i][0] - sorted_arr[p1][0])
        left = (sorted_arr[pn][0]-sorted_arr[p1][0])*(sorted_arr[i][1] - sorted_arr[p1][1])
        if(right != left):
            toBeHull.append(int(i))
        i += 1

    #sort list
    toBeHull.sort()
    #tentuin kanan kiri
    s1 = [] #di kiri
    s2 = [] #di kanan
    s1 = diKiri(p1, pn, toBeHull, sorted_arr)
    s2 = diKiri(pn, p1, toBeHull, sorted_arr)
    # inisialisasi array titik solusi
    S = [0, len(sorted_arr)-1]
    fh1 = findHull(s1, p1, pn, sorted_arr)
    fh2 = findHull(s2, pn, p1, sorted_arr)
```

Gambar 2.1 Algoritma Convex Hull Utama

```python
# jika hasil fungsi tidak ada maka tidak akan ditambahkan dalam array solusi
if fh1 != None:
    S += fh1
if fh2 != None:
    S += fh2

# inisialisasi array sebelah kiri (S1) dan kanan (S2)
S1 = []
S2 = []
for i in S:
    if isDiKiri(p1, pn, i, sorted_arr) and i != p1 and i != pn:
        S1.append(i)
    elif isDiKiri(pn, p1, i, sorted_arr) and i != p1 and i != pn:
        S2.append(i)
S1.sort()
S2.sort()

#inisialisasi array solusi final dengan titik p1 ke titik pertama di S1
#agar array final berisi kumpulan array of index yang menggambarkan ujung garis-garis
Sf = [[p1, S1[0]]]
Sf.append([p1, S2[0]])
# semua titik akan disambungkan ke titik selanjutnya, titik terakhir disambungkan ke pn
for i in range(len(S1)-1):
    Sf.append([S1[i], S1[i+1]])
for i in range(len(S2)-1):
    Sf.append([S2[i], S2[i+1]])
Sf.append([S1[-1], pn])
Sf.append([S2[-1], pn])
return Sf
```

Gambar 2.2 Algoritma Convex Hull Utama

```python
#cari Hullnya
def findHull(s, p1, pn, sorted_arr):
    if len(s) == 0:
        return s
    else:
        # simpan titik terjauh
        terjauh = titikTerjauh(s, p1, pn, sorted_arr)
        #inisialsasi array titik terjauh
        fs = [terjauh]

        #pisahkan titik yang berada di kiri garis p/pn-titik terjauh
        S11 = diKiri(p1, terjauh, s, sorted_arr)
        S12 = diKiri(terjauh, pn, s, sorted_arr)


        #cari lagi untuk himpunan titik di sebelah kiri garis p1-titik terjauh
        fh1 = findHull(S11, p1, terjauh, sorted_arr)
        fh2 = findHull(S12, terjauh, pn, sorted_arr)

        #jika ada hasilnya, tambahkan ke array final, jika tidak abaikan
        if fh1 != None:
            fs += fh1
        if fh2 != None:
            fs += fh2
        return fs
```

Gambar 2.3 Algoritma mencari hull (rekursif)

```python
# untuk membagi titik di kanan dan kiri garis p1pn
def diKiri(p1, pn, toBeHull, sorted_arr):
    s1 = [] #di kiri
    for i in toBeHull:
        if isDiKiri(p1, pn, i, sorted_arr) and not(isInLine(p1, pn, i, sorted_arr)): #kalau ada di kiri garis dan ga di garis
            s1.append(i)
    return s1

#cari terjauh
def titikTerjauh(s, p1, pn, sorted_arr):
    p1 = np.asarray(sorted_arr[p1])
    pn = np.asarray(sorted_arr[pn])
    pmax = -1
    dmax = -1
    for i in s:
        #menghitung jarak titik ke garis
        d = np.abs(np.cross(pn-p1, p1-sorted_arr[i])) / np.linalg.norm(pn-p1)

        #jika jarak lebih besar dari jarak maksimal
        if d > dmax:
            #ubah jarak maksimal dan ubah index titik terjauh
            dmax = d
            pmax = i
    return pmax
```

Gambar 2.4 Fungsi untuk memisahkan himpunan titik di kiri dan mencari titik terjauh dari garis yang dijadikan pivot

```python
#untuk mengecek apakah titik checked ada di garis p1-pn
def isInLine(p1, pn, checked, sorted_arr):
    right = (sorted_arr[pn][1]-sorted_arr[p1][1])*(sorted_arr[checked][0] - sorted_arr[p1][0])
    left = (sorted_arr[pn][0]-sorted_arr[p1][0])*(sorted_arr[checked][1] - sorted_arr[p1][1])
    if(right == left):
        return True
    return False

#untuk mengecek apakah titik checked berada di sebelah kiri (luar) garis p1-pn
def isDiKiri(p1, pn, checked, sorted_arr):
    det = (sorted_arr[p1][0]*sorted_arr[pn][1]) + (sorted_arr[checked][0]*sorted_arr[p1][1]) + (sorted_arr[pn][0]*sorted_arr[checked][1]) - (sort
    if det > 0:
        return True
    return False
```

Gambar 2.5 Fungsi untuk mengecek apakah titik berada di garis dan apakah titik berada di sebelah kiri garis

Screenshot Input - Output Program

1. Dataset Iris - Sepal Length vs Sepal Width

```python
#visualisasi hasil ConvexHull
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
data = datasets.load_iris()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
```

[1] ✓ 14.2s

```
(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

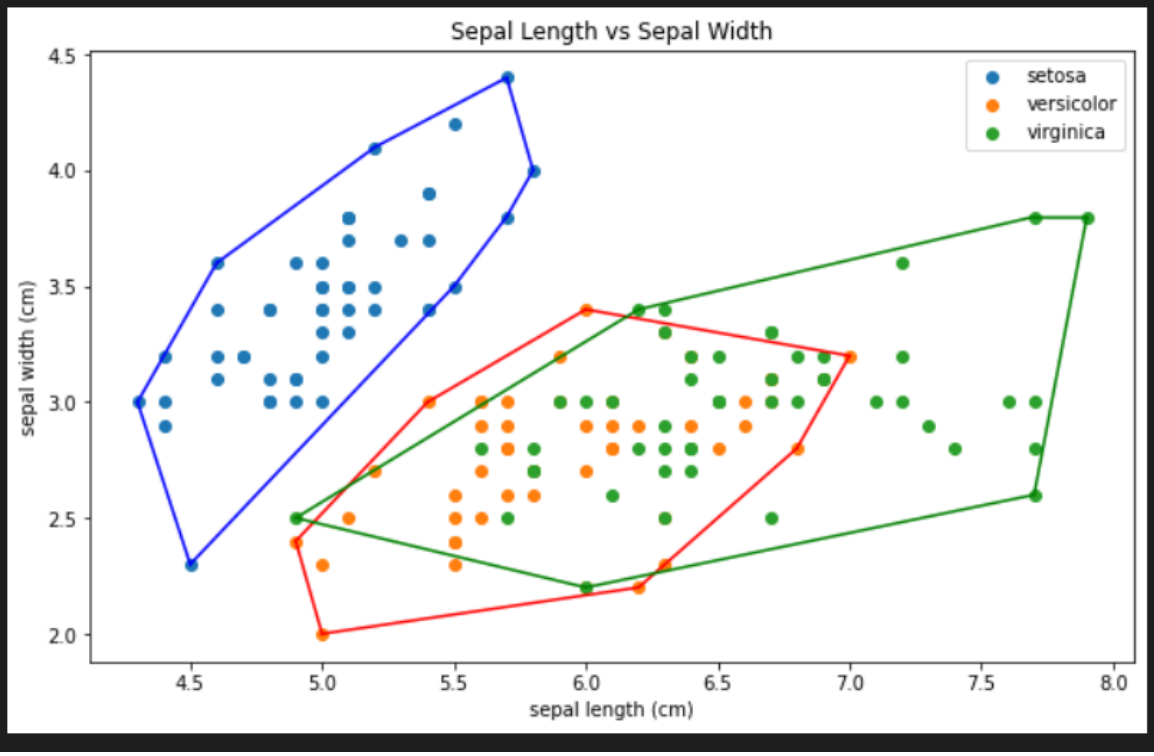
Gambar 3.1 Input dataset Iris

```
#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Sepal Length vs Sepal Width')
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
    bucket = bucket[np.lexsort((bucket[:, 1], (bucket[:,0])))]
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

[13] ✓ 0.7s

Gambar 3.1.1 Input dataset iris Sepal Length vs Sepal Width

```
<matplotlib.legend.Legend at 0x15f23bea340>
```



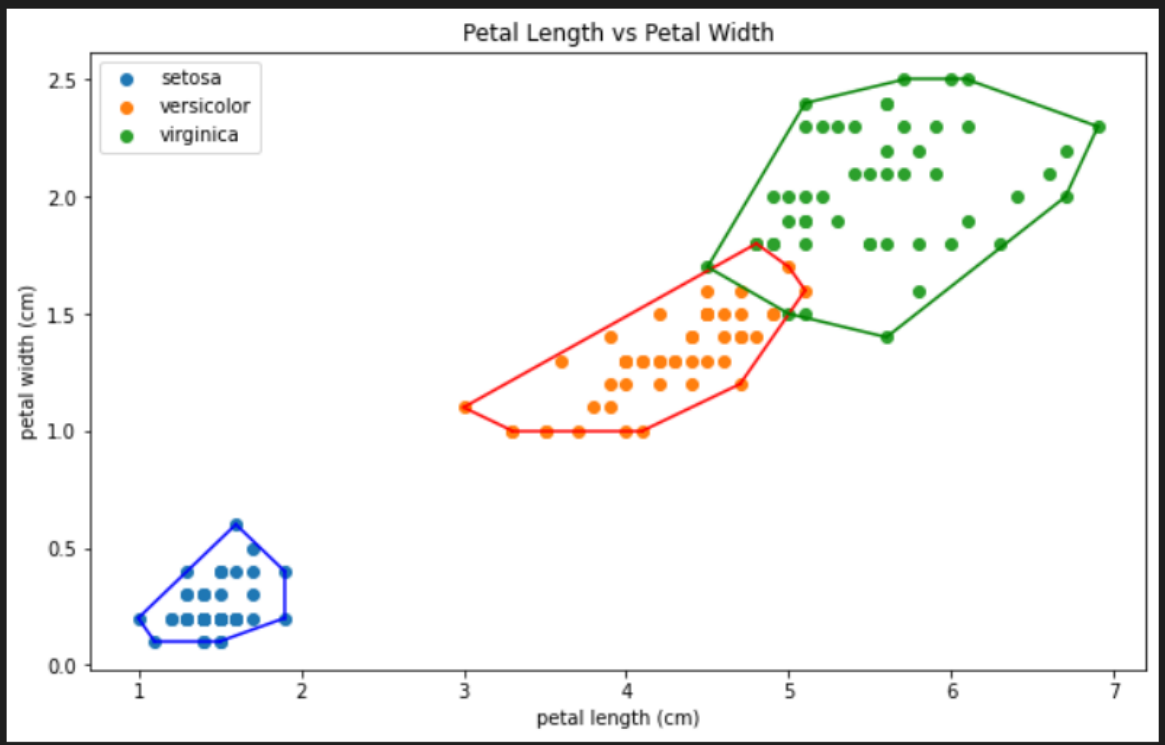
Gambar 3.1.2 Output dataset Iris Sepal Length vs Sepal Width

2. Dataset Iris - Petal Length vs Petal Width

```
#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Petal Length vs Petal Width')
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[2, 3]].values
    bucket = bucket[np.lexsort((bucket[:, 1], (bucket[:,0])))]
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

Gambar 3.2.1 Input dataset Iris Petal Length vs Petal Width

```
<matplotlib.legend.Legend at 0x15f23b222e0>
```



Gambar 3.2.2 Output dataset Iris Petal Length vs Petal Width

3. Dataset Wine - Alcohol vs Malic Acid

```python
#visualisasi hasil ConvexHull
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
data = datasets.load_wine()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
```

✓ 0.1s Python

```
(178, 14)
```

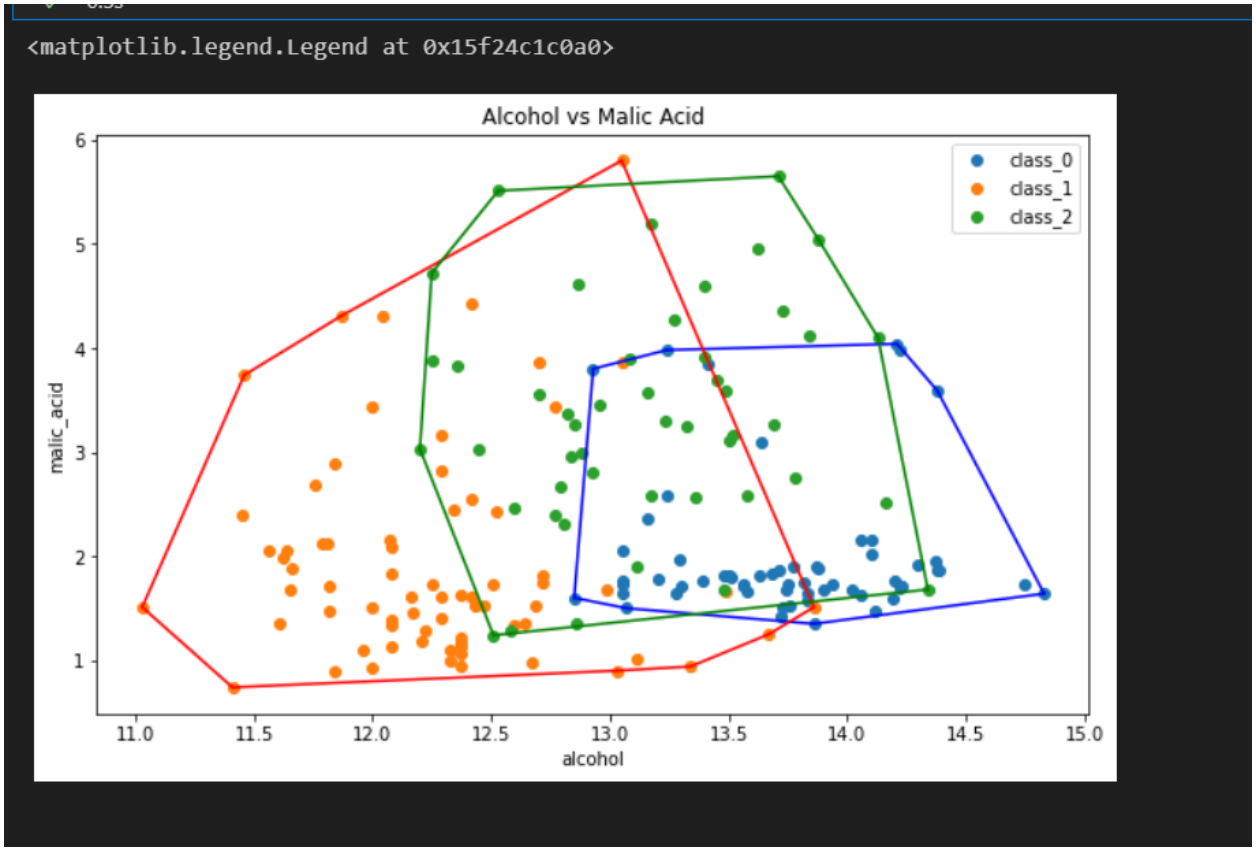
	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od3
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	

Gambar 3.3.1 Input Dataset Wine

```python
#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
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
    bucket = bucket[np.lexsort((bucket[:, 1], (bucket[:,0])))]
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

✓ 0.5s

3.3.2 Input Dataset Wine Alcohol vs Malic Acid



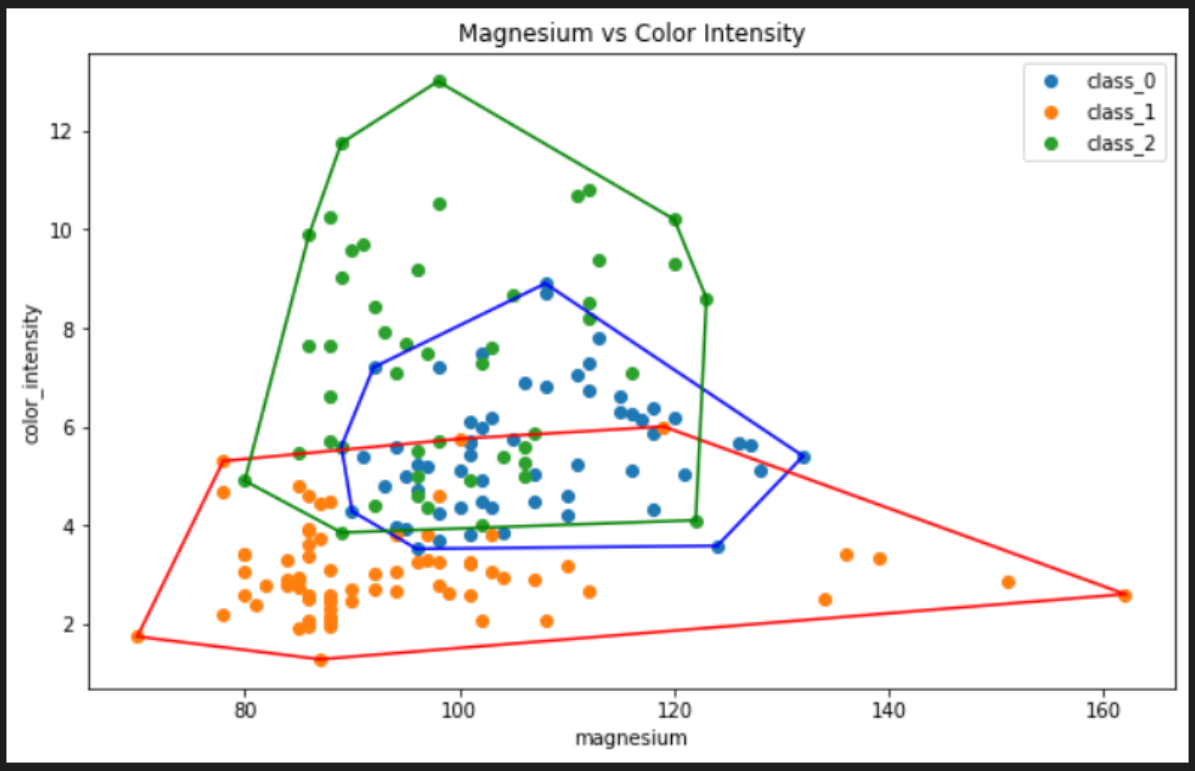
Gambar 3.3.3 Output Dataset Wine Alcohol vs Malic Acid

4. Dataset Wine - Magnesium vs Color Intensity

```python
#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Magnesium vs Color Intensity')
plt.xlabel(data.feature_names[4])
plt.ylabel(data.feature_names[9])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[4, 9]].values
    bucket = bucket[np.lexsort((bucket[:, 1], (bucket[:,0])))]
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

Gambar 3.4.1 Input Dataset Wine Magnesium vs Color Intensity

```
<matplotlib.legend.Legend at 0x15f24cc8550>
```



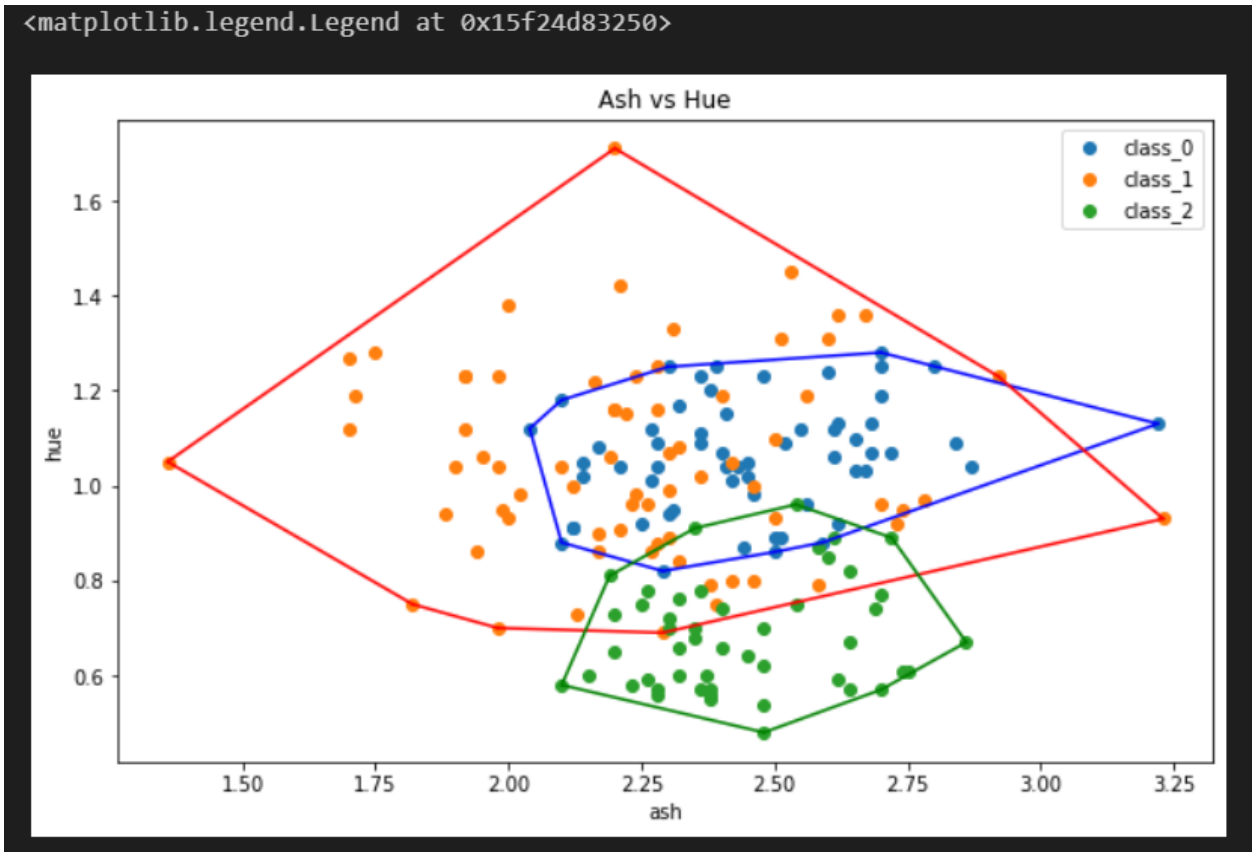
Gambar 3.4.2 Output Dataset Wine Magnesium vs Color Intensity

5. Dataset Wine - Ash vs Hue

```python
#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Ash vs Hue')
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[10])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[2, 10]].values
    bucket = bucket[np.lexsort((bucket[:, 1], (bucket[:,0])))]
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

✓ 0.4s

Gambar 3.5.1 Input Dataset Wine - Ash vs Hue



Gambar 3.5.2 Output Dataset Wine Ash vs Hue

6. Dataset Breast Cancer - Mean Radius vs Mean Texture

```
#visualisasi hasil ConvexHull
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
data = datasets.load_breast_cancer()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
```

✓ 0.2s Python

```
(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	worst area	worst smoothness	cc
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	2019.0	0.1622	
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	1956.0	0.1238	
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	1709.0	0.1444	
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	567.7	0.2098	
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	1575.0	0.1374	

5 rows × 31 columns

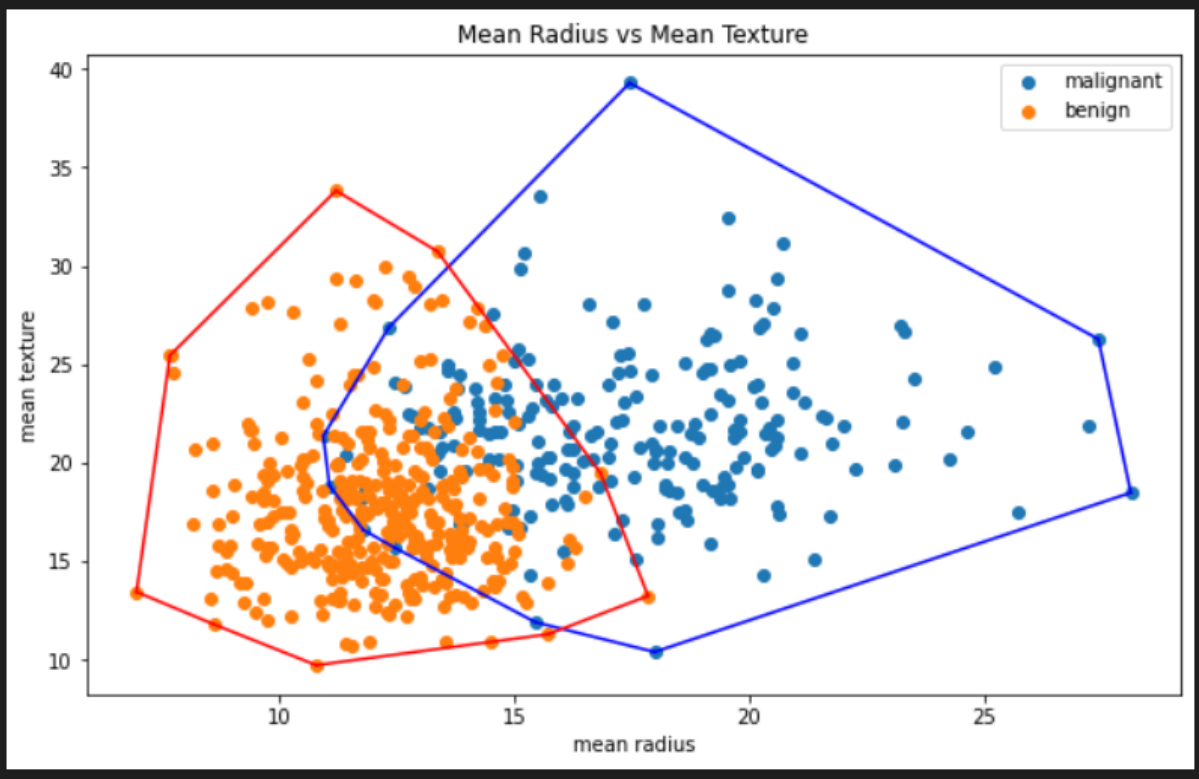
Gambar 3.6.1 Input Dataset Breast Cancer

```python
#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Mean Radius vs Mean Texture')
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
    bucket = bucket[np.lexsort((bucket[:, 1], (bucket[:,0])))]
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

✓ 0.4s

Gambar 3.6.2 Input Dataset Breast Cancer Mean Radius vs Mean Texture

```
<matplotlib.legend.Legend at 0x15f24e86100>
```



Gambar 3.6.3 Output Dataset Breast Cancer Mean Radius vs Mean Texture

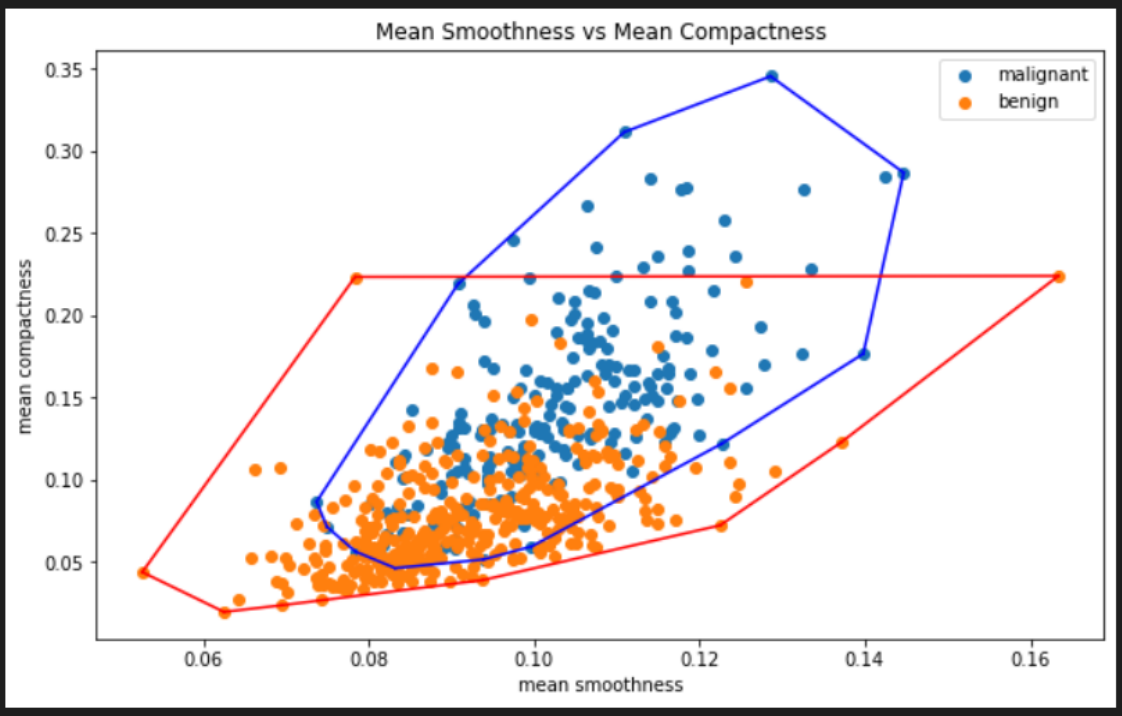
7. Dataset Breast Cancer - Mean Smoothness vs Mean Compactness

```python
#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Mean Smoothness vs Mean Compactness')
plt.xlabel(data.feature_names[4])
plt.ylabel(data.feature_names[5])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[4, 5]].values
    bucket = bucket[np.lexsort((bucket[:, 1], (bucket[:,0])))]
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

✓ 0.6s

Gambar 3.7.1 Input Dataset Breast Cancer Mean Smoothness vs Mean Compactness

```
... <matplotlib.legend.Legend at 0x15f24f1a700>
```



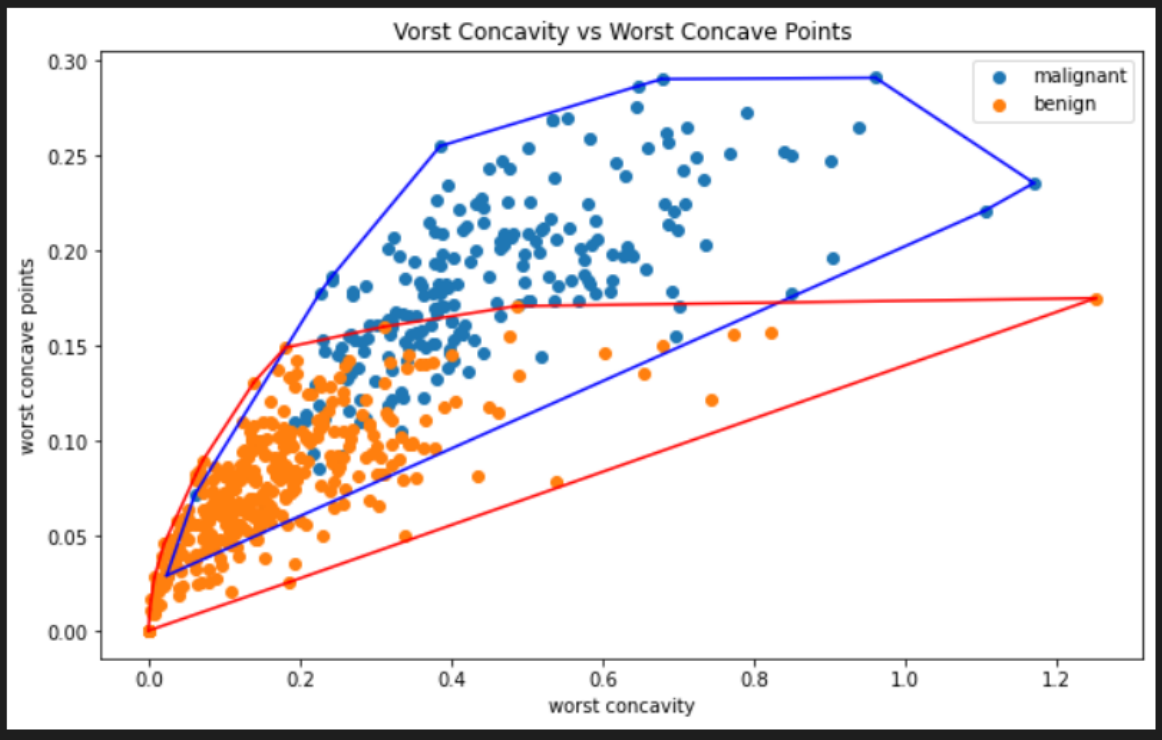
Gambar 3.7.2 Output Dataset Breast Cancer Mean Smoothness vs Mean Compactness

8. Dataset Breast Cancer - Worst Concavity vs Worst Concave Points

```
#visualisasi hasil ConvexHull
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Vorst Concavity vs Worst Concave Points')
plt.xlabel(data.feature_names[26])
plt.ylabel(data.feature_names[27])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[26, 27]].values
    bucket = bucket[np.lexsort((bucket[:, 1], (bucket[:,0])))]
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
✓ 0.6s
```

Gambar 3.8.1 Input Dataset Breast Cancer - Worst Concavity vs Worst Concave Points

```
<matplotlib.legend.Legend at 0x15f25015cd0>
```



Gambar 3.8.2 Output Dataset Breast Cancer - Worst Concavity vs Worst Concave Points

Link Kode Program

https://github.com/hcarissa/Tucil2_13520164

Checklist

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	✔	

Daftar Referensi

Algoritma Divide and Conquer. Munir, R.. 2022. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf

Algoritma Divide and Conquer. Munir, R., Maulidevi, N. 2022. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf

Penentuan Daerah Evakuasi Minimum Akibat Kebocoran Radiasi Nuklir dengan Memanfaatkan Convex Hull. Prabowo, P. 2017. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2016-2017/Makalah2017/Makalah-IF2211-2017-002.pdf