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*

Dibuat oleh:

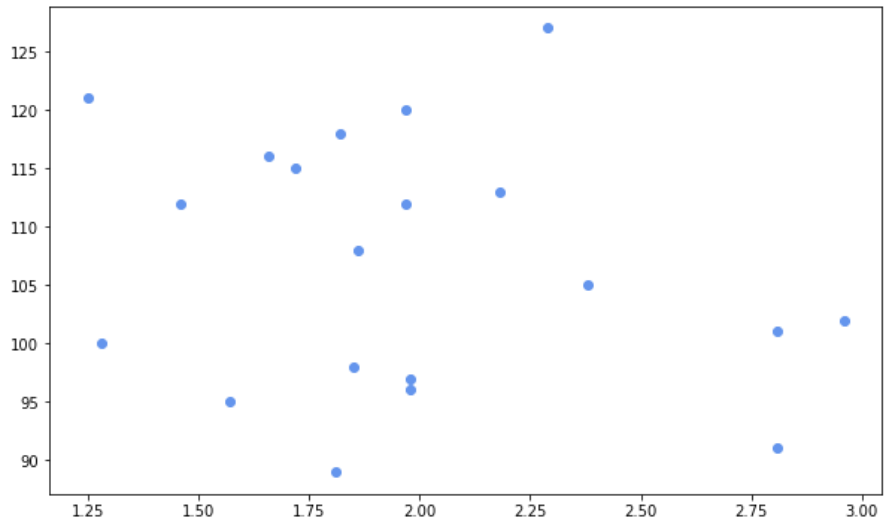
Felicia Sutandijo – 13520050

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022

A. Algoritma *Divide and Conquer*

Berikut adalah algoritma *divide and conquer* yang digunakan dalam pustaka myConvexHull dalam pembuatan *convex hull* dari sekumpulan titik:

1. *Input* atau masukan yang diterima berupa *bucket* yang merupakan sebuah *array* berisi titik-titik (dalam bentuk *array* dengan dua anggota, yaitu x dan y) yang akan diproses. Untuk memudahkan penjelasan, titik-titik ini akan digambarkan menggunakan grafik.



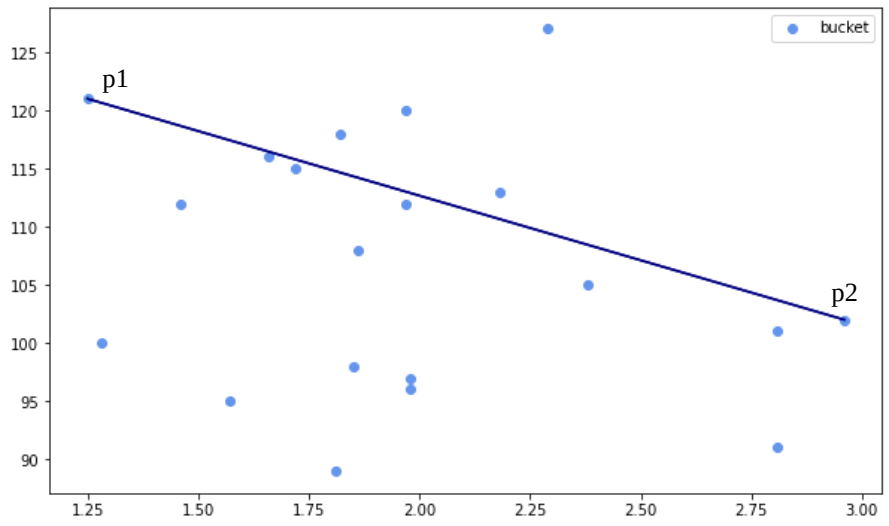

```
2.29, 127.0
1.28, 100.0
2.81, 101.0
2.18, 113.0
1.82, 118.0
1.97, 112.0
1.98, 96.0
1.25, 121.0
1.98, 97.0
1.85, 98.0
2.38, 105.0
1.57, 95.0
1.81, 89.0
2.81, 91.0
2.96, 102.0
1.46, 112.0
1.97, 120.0
1.72, 115.0
1.86, 108.0
1.66, 116.0
```

Gambar 1. *Bucket* yang berisi dua puluh buah titik sembarang

2. Karena *output* atau keluaran yang dihasilkan berupa *hull*, maka pertama-tama *hull* diinisiasi sebagai *array* kosong yang siap menampung indeks-indeks (dari *bucket*) titik pembentuk *convex hull*. *IndexList*, yang merupakan kumpulan indeks seluruh titik yang masih harus diproses, juga diinisiasi dengan seluruh indeks *valid* dari elemen di dalam *bucket.*

3. Selanjutnya, program akan memproses titik-titik di dalam *bucket* berdasarkan indeks urutannya saja, sedangkan titik-titik di dalam *bucket* itu sendiri hanya ‘dilihat’ sebagai acuan tanpa diubah.

4. Untuk memulai algoritma *divide and conquer*, program mencari terlebih dahulu titik-titik ekstrem dari kumpulan titik di dalam *bucket*, yaitu titik-titik dengan absis (bila absis sama, ordinat dibandingkan) minimum dan maksimum. Kedua titik tersebut merupakan titik-titik pertama yang akan membentuk *convex hull*, sehingga indeks kedua titik tersebut ditambahkan ke dalam *hull.* Selanjutnya, titik minimum akan diberi nama p1 dan titik maksimum diberi nama p2.




```
2.29, 127.0
1.28, 100.0
2.81, 101.0
2.18, 113.0
1.82, 118.0
1.97, 112.0
1.98, 96.0
1.25, 121.0
1.98, 97.0
1.85, 98.0
2.38, 105.0
1.57, 95.0
1.81, 89.0
2.81, 91.0
2.96, 102.0
1.46, 112.0
1.97, 120.0
1.72, 115.0
1.86, 108.0
1.66, 116.0
```

Gambar 2. Pencarian p1 dan p2

5. Indeks kedua titik tersebut kemudian dihapus dari *indexList* karena sudah tidak perlu diproses lagi.
6. Titik-titik di dalam *bucket* dipartisi menjadi dua bagian, yaitu titik yang berada di sebelah kiri garis p1p2, dan titik yang berada di sebelah kanan garis p1p2. Partisi menggunakan rumus di bawah:

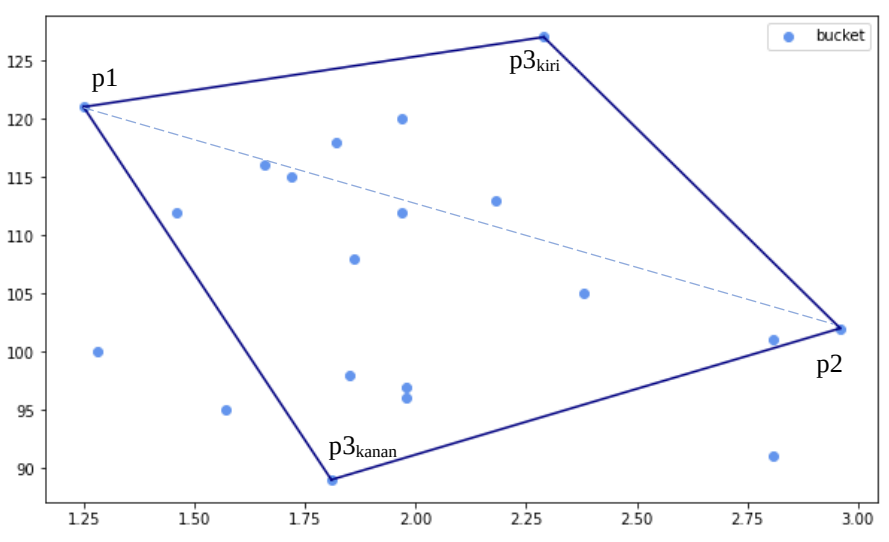
$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = x_1y_2 + x_3y_1 + x_2y_3 - x_3y_2 - x_2y_1 - x_1y_3$$

Titik (x3,y3) berada di sebelah kiri dari garis ((x1,y1), (x2,y2)) jika hasil determinan positif, dan berada di sebelah kanan garis jika hasil determinan negatif.

7. Untuk kedua partisi tersebut, algoritma rekursif *partConvexHull* dijalankan.
8. Setiap kali algoritma *partConvexHull* dijalankan, program mencari satu buah titik dengan jarak terjauh dari garis acuan, yang pada kasus ini merupakan garis p1p2. Indeks titik tersebut ditambahkan ke dalam *hull* dan dihapus dari daftar *indexList*, dan titik diberi nama p3. Jarak dari titik ke garis dapat dihitung menggunakan rumus:

$$\text{distance}(P_1, P_2, (x_0, y_0)) = \frac{|(x_2 - x_1)(y_1 - y_0) - (x_1 - x_0)(y_2 - y_1)|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$

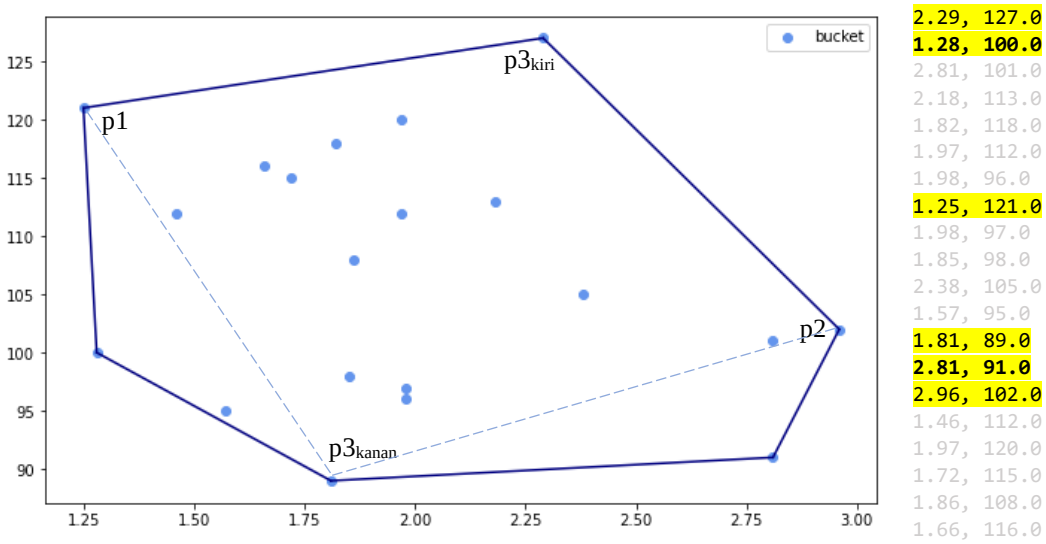
(x1,y1) dan (x2,y2) merupakan dua titik pembentuk garis dan (x0,y0) adalah titik sembarang.




```
2.29, 127.0
1.28, 100.0
2.81, 101.0
2.18, 113.0
1.82, 118.0
1.97, 112.0
1.98, 96.0
1.25, 121.0
1.98, 97.0
1.85, 98.0
2.38, 105.0
1.57, 95.0
1.81, 89.0
2.81, 91.0
2.96, 102.0
1.46, 112.0
1.97, 120.0
1.72, 115.0
1.86, 108.0
1.66, 116.0
```

Gambar 3. Pencarian p3

9. Indeks yang masih tersisa di *indexList* dipartisi ke dalam dua bagian yang diproses, yaitu titik yang berada di sebelah kiri garis p1p3 dan titik yang berada di sebelah kiri p3p2. Perhatikan bahwa urutan pemrosesan garis dibuat secara *clockwise* sehingga algoritma selalu hanya perlu mengambil partisi kiri, karena partisi kanan merupakan kumpulan titik-titik yang berada ‘di dalam’ segitiga yang terbentuk.
10. Kemudian, untuk kedua partisi tersebut, algoritma *partConvexHull* dijalankan kembali dengan garis acuan yang baru (p1p3 dan p3p2).



Gambar 3. Rekursi hingga *convex hull* terbentuk

11. Algoritma berhenti ketika tidak ada lagi titik yang tersisa pada partisi, atau ketika hanya tinggal 1 titik saja (titik tersebut langsung dimasukkan ke dalam *hull*).
12. *Hull* telah berisi seluruh indeks titik pembentuk *convex hull* dan siap untuk digambar.

B. *Source Code* Program dalam Python

1. Pustaka myConvexHull (myConvexHull.py)

```
from math import sqrt
import numpy as np


def distance(p1, p2, p3):
  # Fungsi untuk menghitung jarak dari titik ke garis.
  # Fungsi menerima titik p1 dan p2 pembentuk garis,
  # serta sebuah titik p3.
  # Fungsi mengembalikan jarak dari p3 ke garis p1p2.

  # ALGORITMA
  return (abs((p2[0]-p1[0])*(p1[1]-p3[1])-(p1[0]-p3[0])*(p2[1]-
p1[1])))/(sqrt((p2[0]-p1[0])**2+(p2[1]-p1[1])**2))


def partition(bucket, indexList, p1, p2):
  # Fungsi untuk membagi kumpulan titik-titik menjadi dua partisi.
  # Fungsi menerima kumpulan titik-titik yang ditampung di dalam bucket dan
  # indeksnya ditandai indexList,
  # serta titik p1 dan p2 pembentuk garis.
  # Fungsi mengembalikan dua partisi kiri dan kanan;
  # titik yang berada di kiri garis p1p2 dimasukkan ke partisi kiri,
  # dan titik yang berada di kanan garis p1p2 dimasukkan ke partisi kanan,
  # serta titik yang berada tepat pada garis p1p2 tidak diperhitungkan.

  # KAMUS
  left = [] # partisi kiri
  right = [] # partisi kanan

  # ALGORITMA
  for i in indexList:
    p3 = bucket[i] # titik yang diuji
    d = p1[0]*p2[1]+p3[0]*p1[1]+p2[0]*p3[1]-p3[0]*p2[1]-p2[0]*p1[1]-
p1[0]*p3[1]
    if (d > 0): # titik berada di kiri garis
      left.append(i)
    elif (d < 0): # titik berada di kanan garis
      right.append(i)

  return (left, right)
```

```
def partConvexHull(hull, bucket, indexList, p1, p2):
  # Prosedur untuk menentukan titik-titik pembentuk convex hull secara
rekursif
  # I.S. hull sembarang
  # F.S. hull ditambahkan indeks-indeks titik (yang ditampung dalam bucket)
  # yang membentuk convex hull secara terurut

  # ALGORITMA
  # Titik-titik yang diperiksa adalah titik-titik yang indeksnya berada di
dalam indexList
  if (len(indexList) == 0): # basis 0
    pass
  elif (len(indexList) == 1): # basis 1
    hull.append(indexList[0])
  else: # rekursi
    # mencari titik dengan jarak terbesar ke garis p1p2 untuk ditambahkan
ke dalam hull
    maxIdx = -1
    maxD = 0
    for i in indexList:
      p3 = bucket[i]
      d = distance(p1, p2, p3)
      if (d >= maxD):
        maxIdx = i
        maxD = d
    # titik dengan jarak terbesar telah ditemukan dan siap ditambahkan ke
dalam hull

    # menghapus titik terjauh dari kumpulan titik
    indexList.remove(maxIdx)

    # partisi titik-titik untuk menentukan titik mana saja yang masih
berada 'di luar' convex hull,
    # partisi yang akan digunakan keduanya merupakan partisi kiri dari
kedua garis yang baru terbentuk
    # karena algoritma berjalan secara clockwise
    left, right = partition(bucket, indexList, p1, bucket[maxIdx])
    left1, right1 = partition(bucket, right, bucket[maxIdx], p2)

    # memanggil rekursi secara terurut agar memudahkan plotting titik
    partConvexHull(hull, bucket, left1, bucket[maxIdx], p2) # kiri
    hull.append(maxIdx) # tengah (titik yang jaraknya paling jauh tadi)
    partConvexHull(hull, bucket, left, p1, bucket[maxIdx]) # kanan
```

```
'''FUNGSI UTAMA'''
def convexHull(bucket):
  # Fungsi untuk menentukan titik-titik pembentuk convex hull dari
sekumpulan titik.
  # Fungsi menerima kumpulan titik yang ditampung dalam bucket (array of
array).
  # Fungsi mengeluarkan hull (array) yang berisi indeks-indeks titik
pembentuk convex hull
  # yang berurutan, serta titik pertama (indeks 0) dituliskan lagi di akhir
array (indeks terakhir)
  # untuk memudahkan plotting grafik (agar convex hull menjadi utuh,
nyambung semua dari awal balik ke awal)

  # KAMUS
  hull = [] # penampung titik-titik pembentuk convex hull
  indexList = [i for i in range(len(bucket))] # daftar indeks yang valid
dari bucket
  # pada program ini, pemrosesan dan 'pencatatan' titik-titik menggunakan
indeksnya, bukan langsung titiknya

  # ALGORITMA
  # mencari titik-titik ekstrem dalam bucket, sesuai absisnya, kemudian
bila absis sama, baru ordinat dibandingkan
  # titik-titik ini dipakai sebagai acuan pertama dalam partisi divide and
conquer
  minIdx = np.argmin(bucket, axis=0)[0] # p1
  maxIdx = np.argmax(bucket, axis=0)[0] # pn

  # menghapus titik-titik ekstrem dari kumpulan titik
  indexList.remove(minIdx)
  indexList.remove(maxIdx)

  # membagi titik-titik menjadi bagian kiri dan kanan dari garis p1pn
  left, right = partition(bucket, indexList, bucket[minIdx],
bucket[maxIdx])

  # memanggil rekursi secara terurut agar memudahkan plotting titik
  hull.append(maxIdx) # pn
  partConvexHull(hull, bucket, left, bucket[minIdx], bucket[maxIdx]) #
partisi kiri
  hull.append(minIdx) # p1
  partConvexHull(hull, bucket, right, bucket[maxIdx], bucket[minIdx]) #
partisi kanan
  hull.append(maxIdx) # pn lagi, supaya grafik utuh

  return hull
```

2. Program Utama untuk Pengujian Pustaka (main.py)

```
# arguments
import argparse
parser = argparse.ArgumentParser(description='Program untuk membuat convex
hull.')
parser.add_argument('dataset', metavar='dataset', choices=['iris', 'wine',
'breast_cancer'], help='name of dataset')
parser.add_argument('x', help='first attribute', type=int)
parser.add_argument('y', help='second attribute', type=int)
parser.add_argument('-o', dest='output',
                    default='output.png',
                    help='output file name (default: output.png)')
args = parser.parse_args()

# imports
from myConvexHull import convexHull
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets

# load data based on argument
if args.dataset == 'iris':
  data = datasets.load_iris()
elif args.dataset == 'wine':
  data = datasets.load_wine()
elif args.dataset == 'breast_cancer':
  data = datasets.load_breast_cancer()

# create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)

# configure which columns to use
x = args.x
y = args.y

# plot data and convex hull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(f'{data.feature_names[x].title()} vs
{data.feature_names[y].title()}')
plt.xlabel(data.feature_names[x])
plt.ylabel(data.feature_names[y])
for i in range(len(data.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:,[x,y]].values
  hull = convexHull(bucket) #hasil implementasi convexHull
```

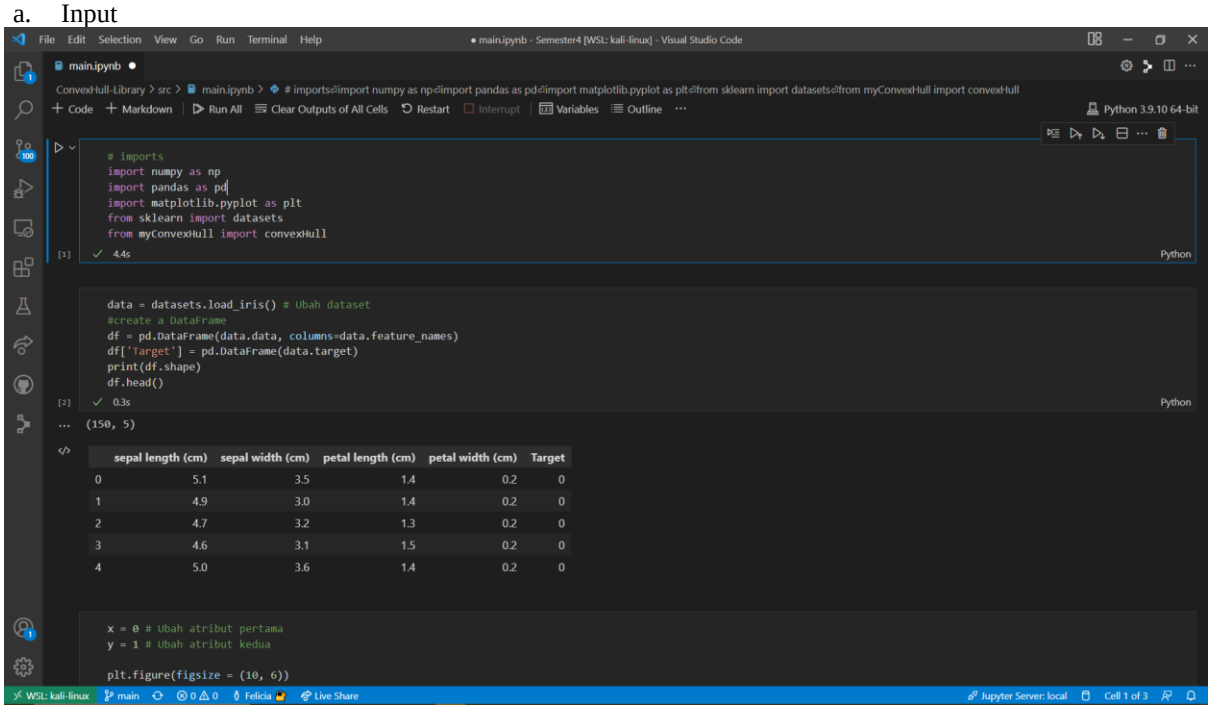
```
  #ConvexHull Divide & Conquer
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
  for j in range (len(hull)-1):
    plt.plot(bucket[[hull[j], hull[j+1]], 0], bucket[[hull[j], hull[j+1]],
1], colors[i])
plt.legend()
plt.savefig(args.output) # save plot

print(f'Convex hull berhasil dibuat. Output diberi nama {args.output}')
```

C. *Screenshot* Input dan Output

1. Sepal Length – Sepal Width (Iris)

a. Input



```
# imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import convexHull
```

```
data = datasets.load_iris() # Ubah dataset
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
```

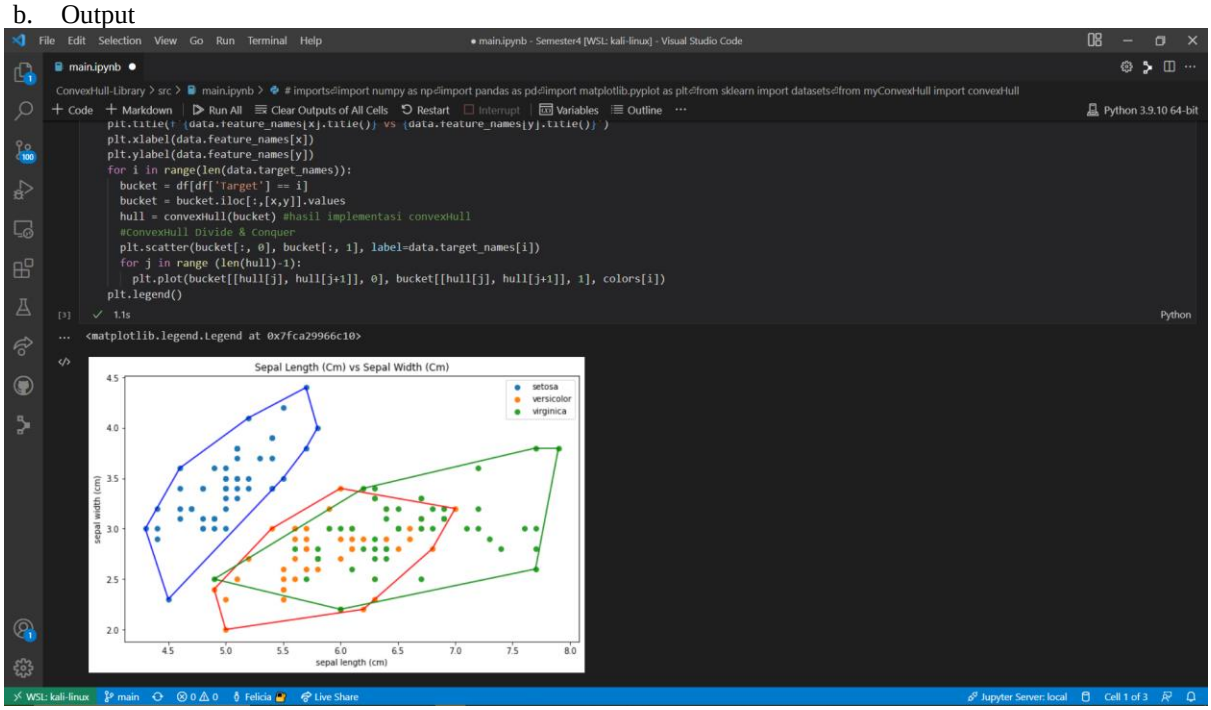
```
(150, 5)
```

	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

```
x = 0 # Ubah atribut pertama
y = 1 # Ubah atribut kedua

plt.figure(figsize = (10, 6))
```

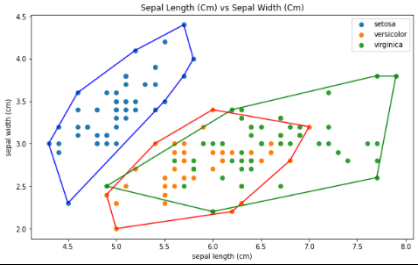
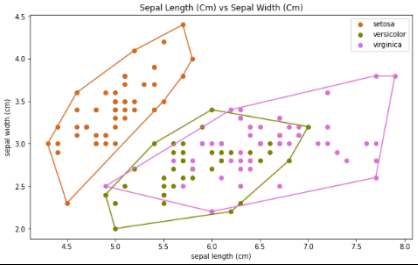
b. Output



```
plt.title(f'{data.feature_names[x].title()} vs {data.feature_names[y].title()}')
plt.xlabel(data.feature_names[x])
plt.ylabel(data.feature_names[y])
for i in range(len(data.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:,[x,y]].values
  hull = convexHull(bucket) #hasil implementasi convexHull
  #ConvexHull Divide & Conquer
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
  for j in range (len(hull)-1):
    plt.plot(bucket[[hull[j], hull[j+1]], 0], bucket[[hull[j], hull[j+1]], 1], colors[i])
plt.legend()
```

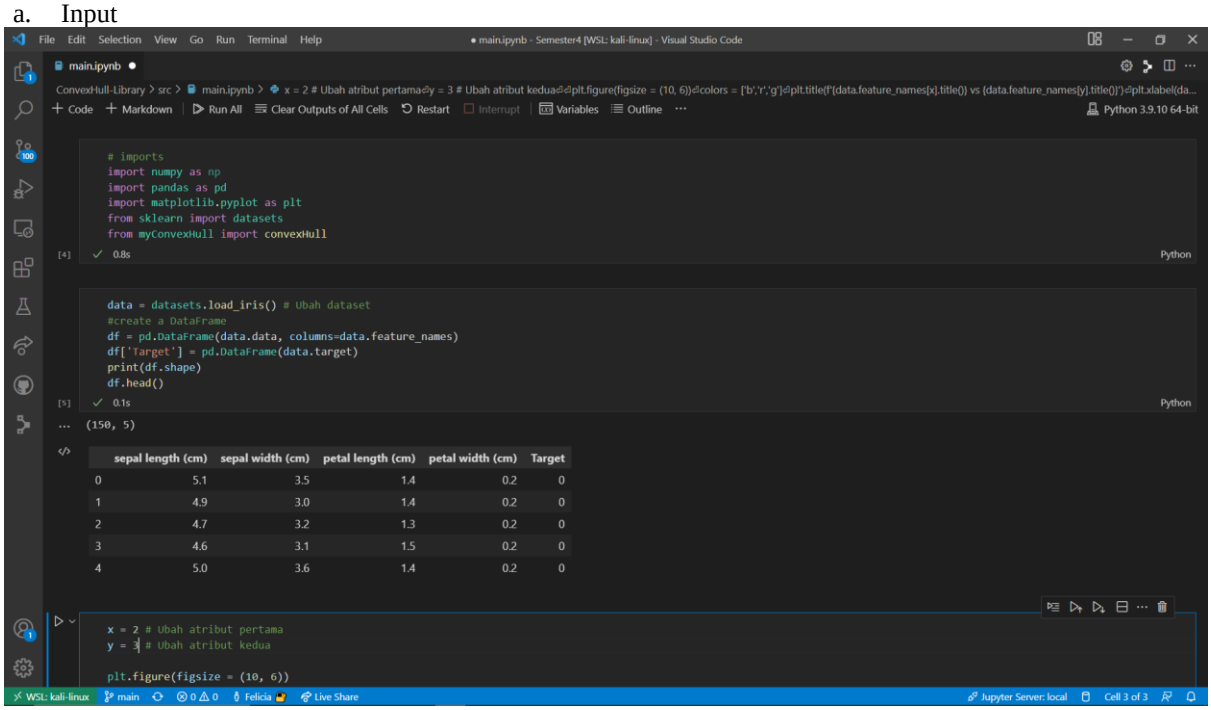
```
<matplotlib.legend.Legend at 0x7fca29966c10>
```


c. Perbandingan Hasil Output Pustaka myConvexHull dengan scipy.spatial

	
Hasil output myConvexHull	Hasil output scipy.spatial

2. Petal Length – Petal Width (Iris)

a. Input



```
# imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import convexHull
```

```
data = datasets.load_iris() # Ubah dataset
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
```

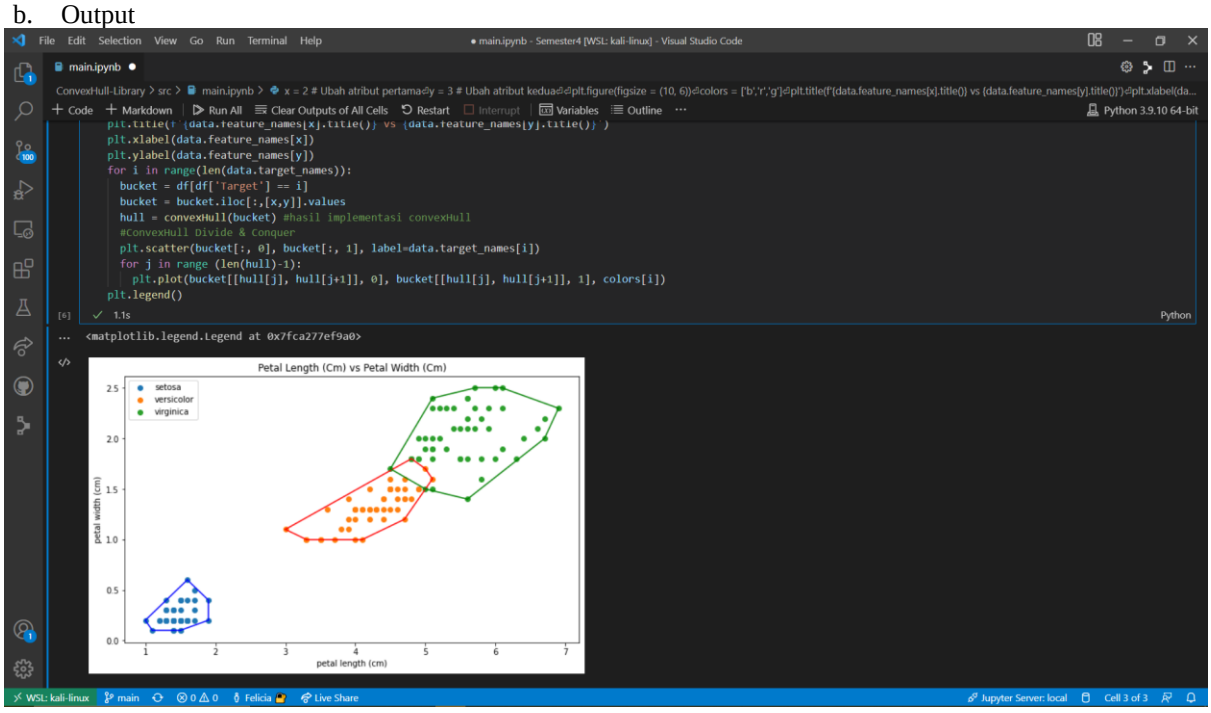
```
(150, 5)
```

	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

```
x = 2 # Ubah atribut pertama
y = 3 # Ubah atribut kedua

plt.figure(figsize = (10, 6))
```

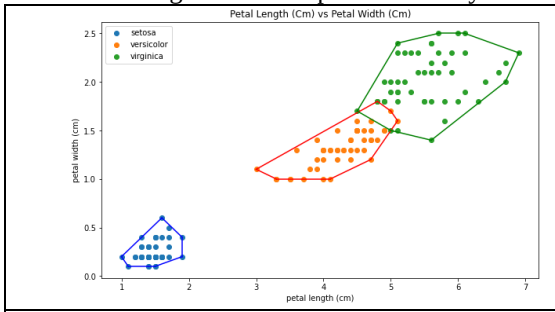
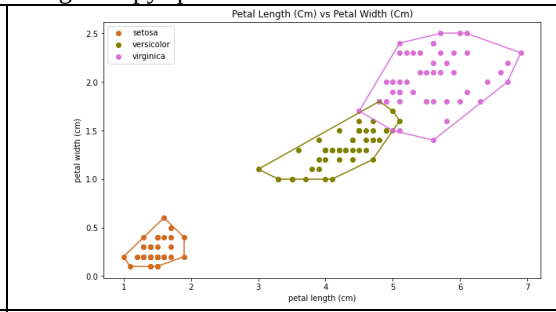
b. Output



```
plt.xlabel(data.feature_names[x])
plt.ylabel(data.feature_names[y])
for i in range(len(data.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:,[x,y]].values
  hull = convexHull(bucket) #hasil implementasi convexHull
  #ConvexHull Divide & Conquer
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
  for j in range (len(hull)-1):
    plt.plot(bucket[[hull[j], hull[j+1]], 0], bucket[[hull[j], hull[j+1]], 1], colors[i])
plt.legend()
```

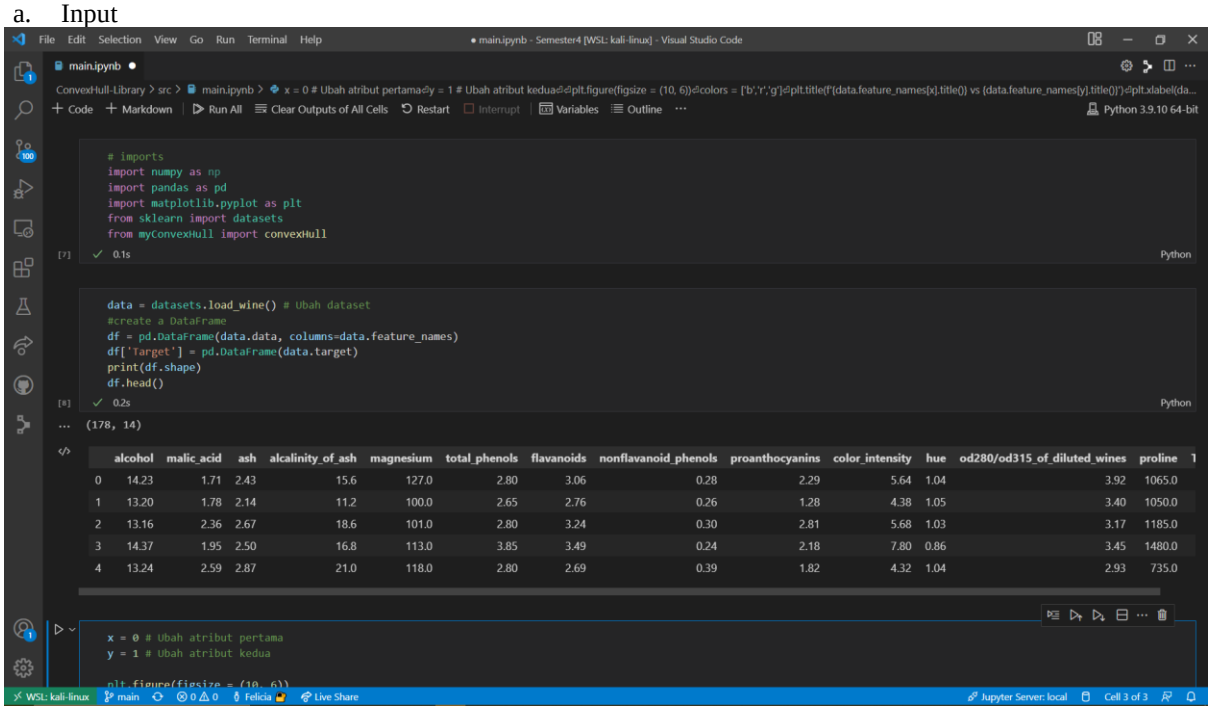
```
<matplotlib.legend.Legend at 0x7fca277ef9a0>
```


c. Perbandingan Hasil Output Pustaka myConvexHull dengan scipy.spatial

	
Hasil output myConvexHull	Hasil output scipy.spatial

3. Alcohol – Malic Acid (Wine)

a. Input



```
# imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import convexHull
```

```
data = datasets.load_wine() # Ubah dataset
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
```

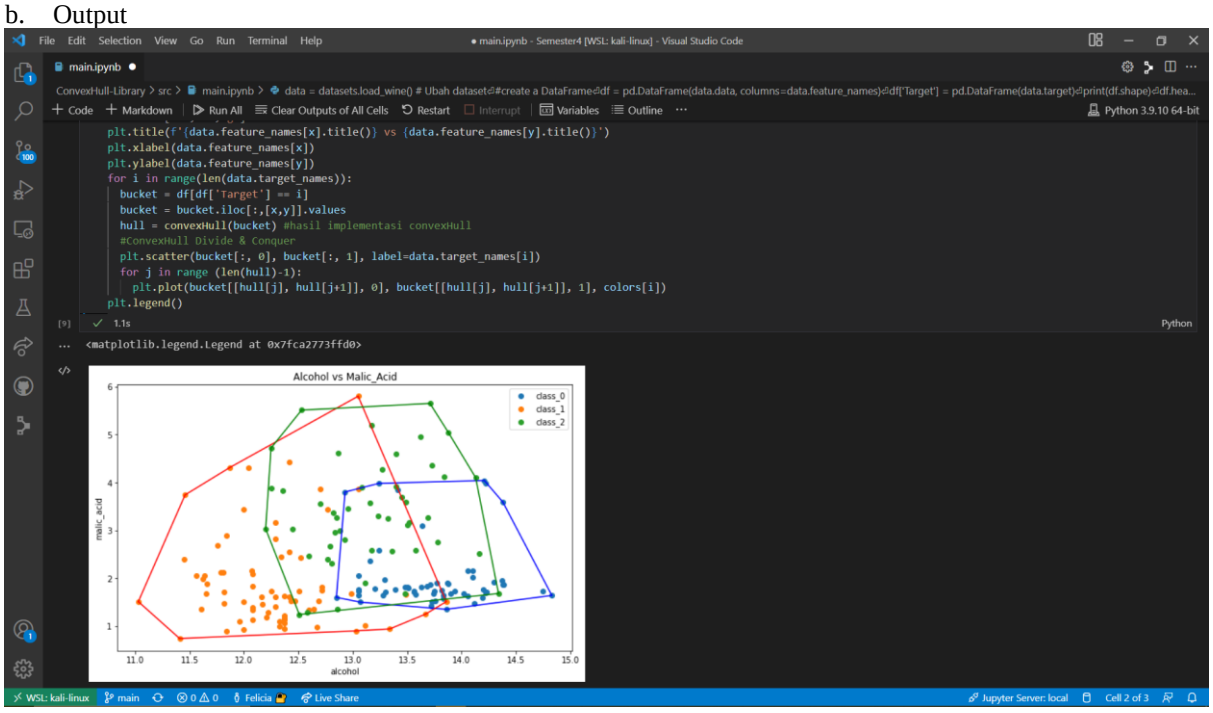
```
(178, 14)
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065.0
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050.0
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185.0
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480.0
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735.0

```
x = 0 # Ubah atribut pertama
y = 1 # Ubah atribut kedua

plt.figure(figsize = (10, 6))
```

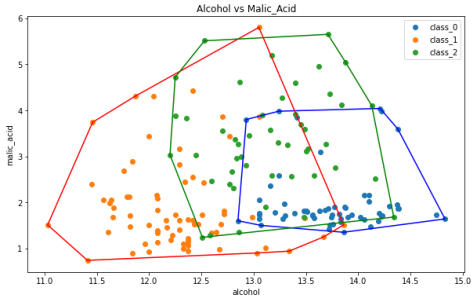
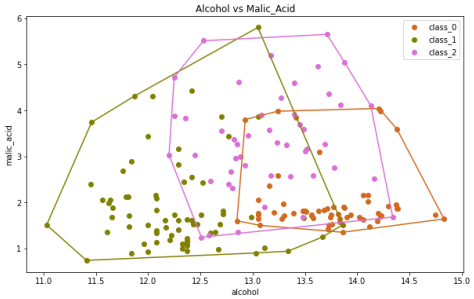
b. Output



```
plt.title(f'{data.feature_names[x].title()} vs {data.feature_names[y].title()}')
plt.xlabel(data.feature_names[x])
plt.ylabel(data.feature_names[y])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[x,y]].values
    hull = convexHull(bucket) #hasil implementasi convexHull
    #ConvexHull Divide & Conquer
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for j in range (len(hull)-1):
        plt.plot(bucket[[hull[j], hull[j+1]], 0], bucket[[hull[j], hull[j+1]], 1], colors[i])
plt.legend()
```

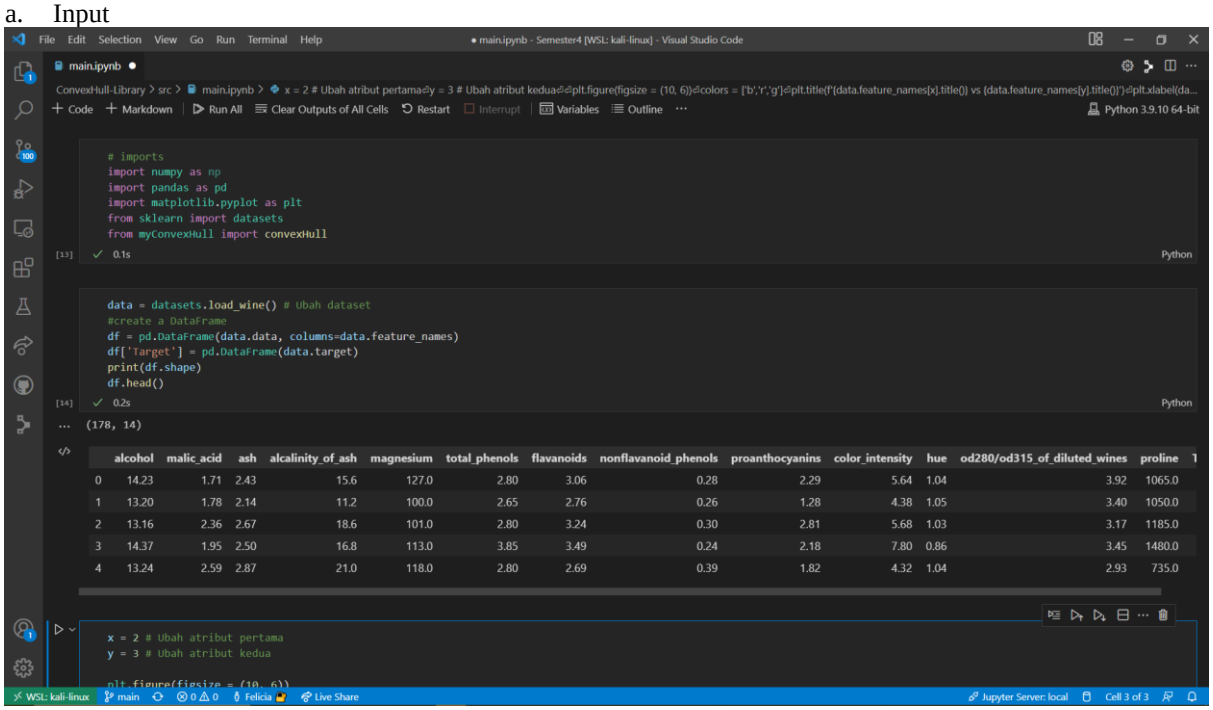
```
<matplotlib.legend.Legend at 0x7fca2773ffd0>
```


c. Perbandingan Hasil Output Pustaka myConvexHull dengan scipy.spatial

	
Hasil output myConvexHull	Hasil output scipy.spatial

4. Ash – Alcalinity of Ash (Wine)

a. Input



```
# imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import convexHull
```

[13] ✓ 0.1s Python

```
data = datasets.load_wine() # Ubah dataset
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
```

[14] ✓ 0.2s Python

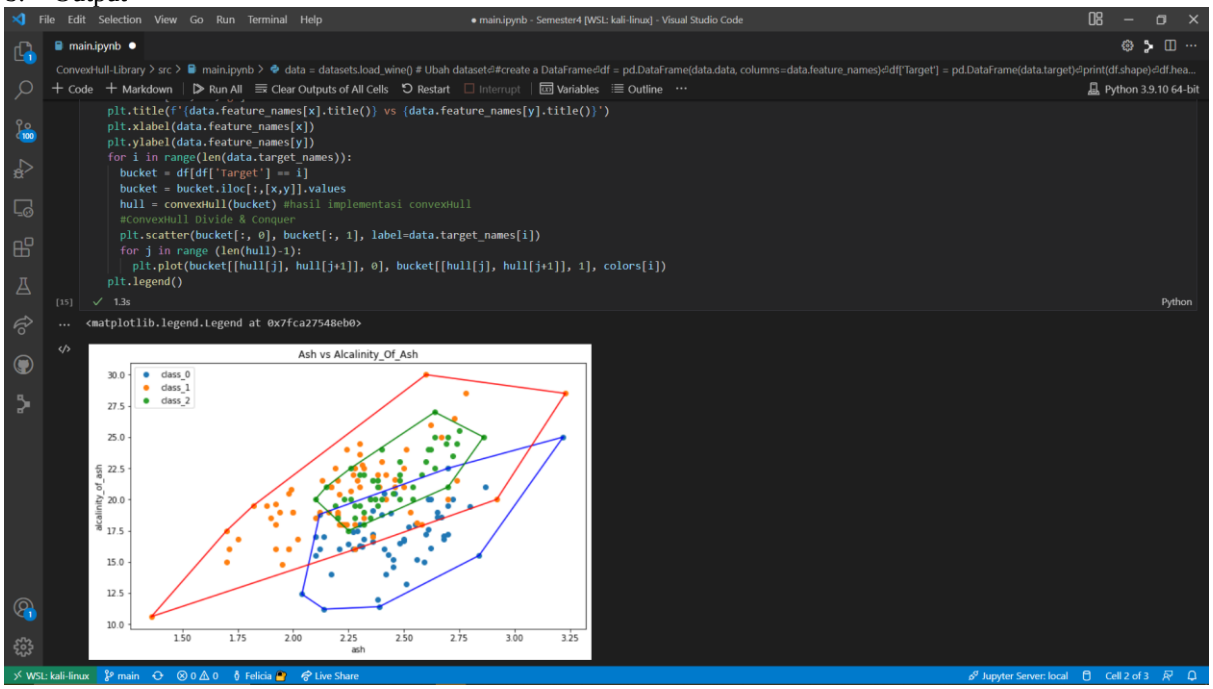
```
(178, 14)
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065.0
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050.0
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185.0
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480.0
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735.0

```
x = 2 # Ubah atribut pertama
y = 3 # Ubah atribut kedua

plt.figure(figsize = (10, 6))
```

b. Output

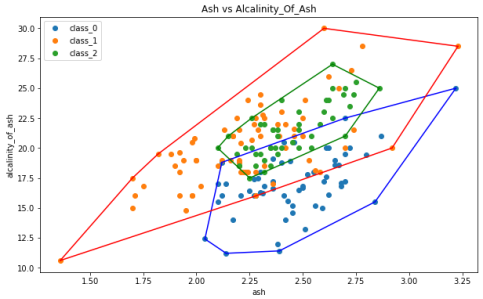
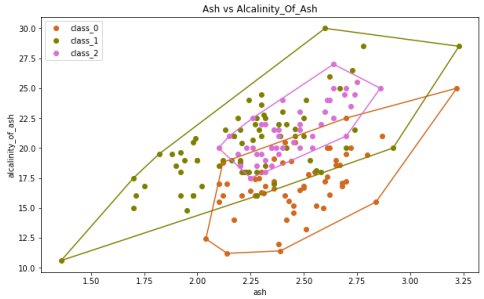


```
plt.title(f'{data.feature_names[x].title()} vs {data.feature_names[y].title()}')
plt.xlabel(data.feature_names[x])
plt.ylabel(data.feature_names[y])
for i in range(len(data.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:,[x,y]].values
  hull = convexHull(bucket) #hasil implementasi convexHull
  #ConvexHull Divide & Conquer
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
  for j in range (len(hull)-1):
    plt.plot(bucket[[hull[j], hull[j+1]], 0], bucket[[hull[j], hull[j+1]], 1], colors[i])
plt.legend()
```

[15] ✓ 1.3s Python

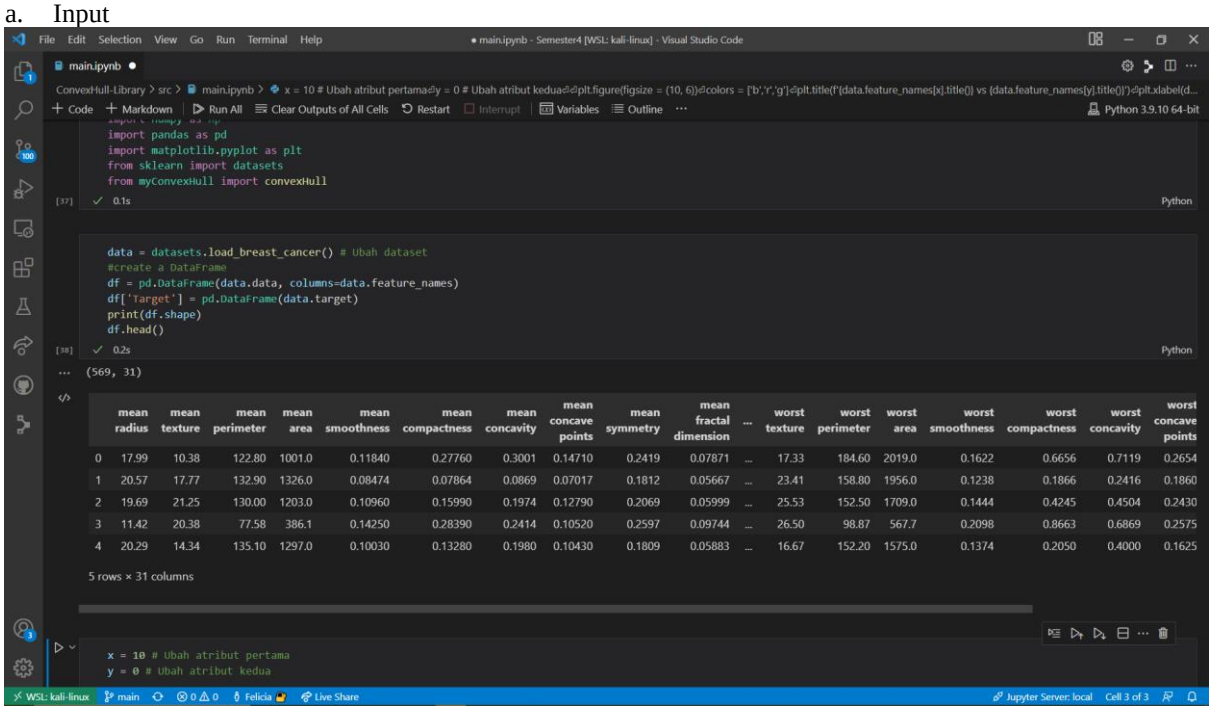
```
<matplotlib.legend.Legend at 0x7fca27548eb0>
```


b. Perbandingan Hasil Output Pustaka myConvexHull dengan scipy.spatial

	
Hasil output myConvexHull	Hasil output scipy.spatial

5. Radius Error – Mean Radius (Breast Cancer)

a. Input



```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import convexHull
```

```
data = datasets.load_breast_cancer() # Ubah dataset
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
```

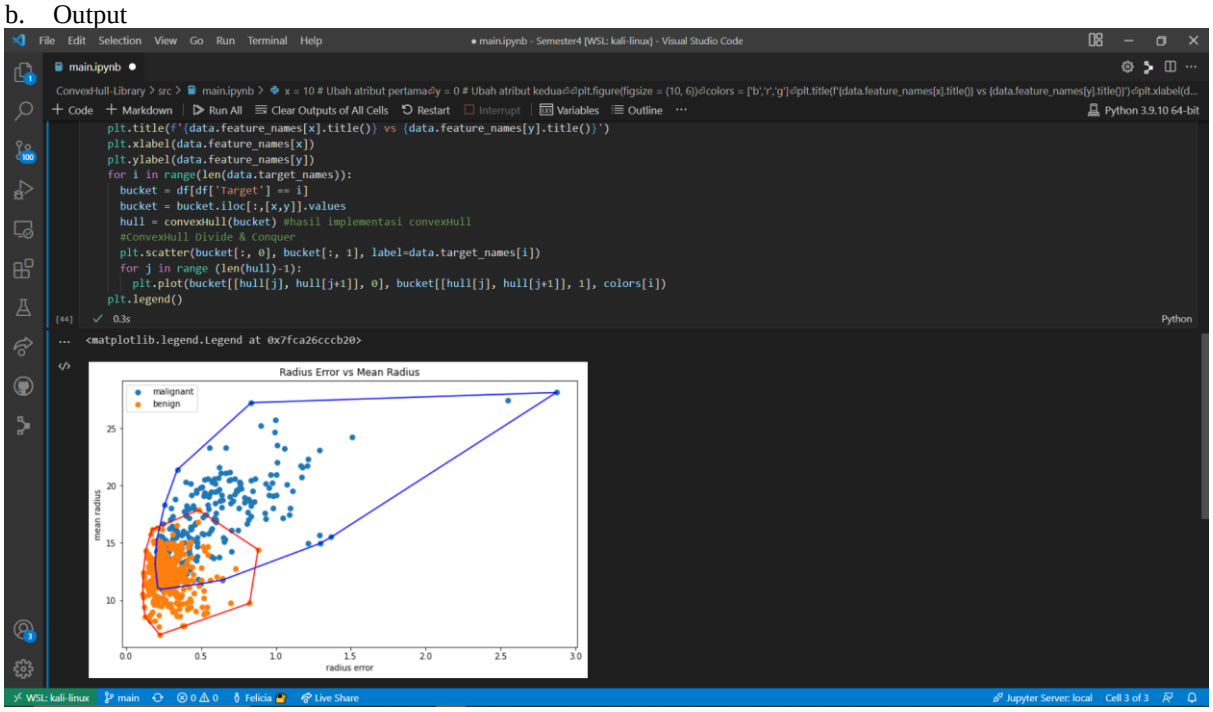
```
(569, 31)
```

	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	worst compactness	worst concavity	worst concave points
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	0.6656	0.7119	0.2654
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	0.1866	0.2416	0.1860
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	0.4245	0.4504	0.2430
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	0.8663	0.6869	0.2575
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	0.2050	0.4000	0.1625

5 rows × 31 columns

```
x = 10 # Ubah atribut pertama
y = 0 # Ubah atribut kedua
```

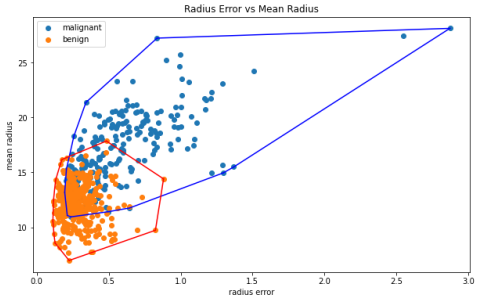
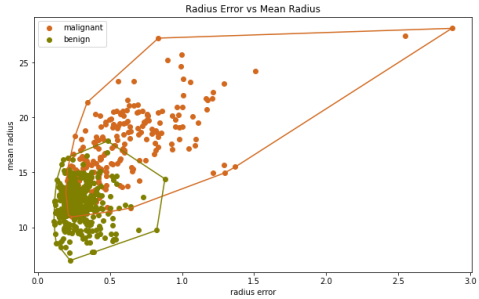
b. Output



```
plt.title(f'{data.feature_names[x].title()} vs {data.feature_names[y].title()}')
plt.xlabel(data.feature_names[x])
plt.ylabel(data.feature_names[y])
for i in range(len(data.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:,[x,y]].values
  hull = convexHull(bucket) #hasil implementasi convexHull
  #ConvexHull Divide & Conquer
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
  for j in range (len(hull)-1):
    plt.plot(bucket[[hull[j], hull[j+1]], 0], bucket[[hull[j], hull[j+1]], 1], colors[i])
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fca26cccb20>
```


b. Perbandingan Hasil Output Pustaka myConvexHull dengan scipy.spatial

	
Hasil output myConvexHull	Hasil output scipy.spatial

D. Alamat *Drive* Kode Program

Kode program dapat diakses menggunakan link *Google Drive* berikut:
https://drive.google.com/drive/folders/1xYNac4Svy0tlZIDwFx8CsKPjpCJOvzpX?usp=sharing

Atau menggunakan *Github*:
https://github.com/FelineJTD/ConvexHull-Library

E. Tabel Penilaian

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.	√	