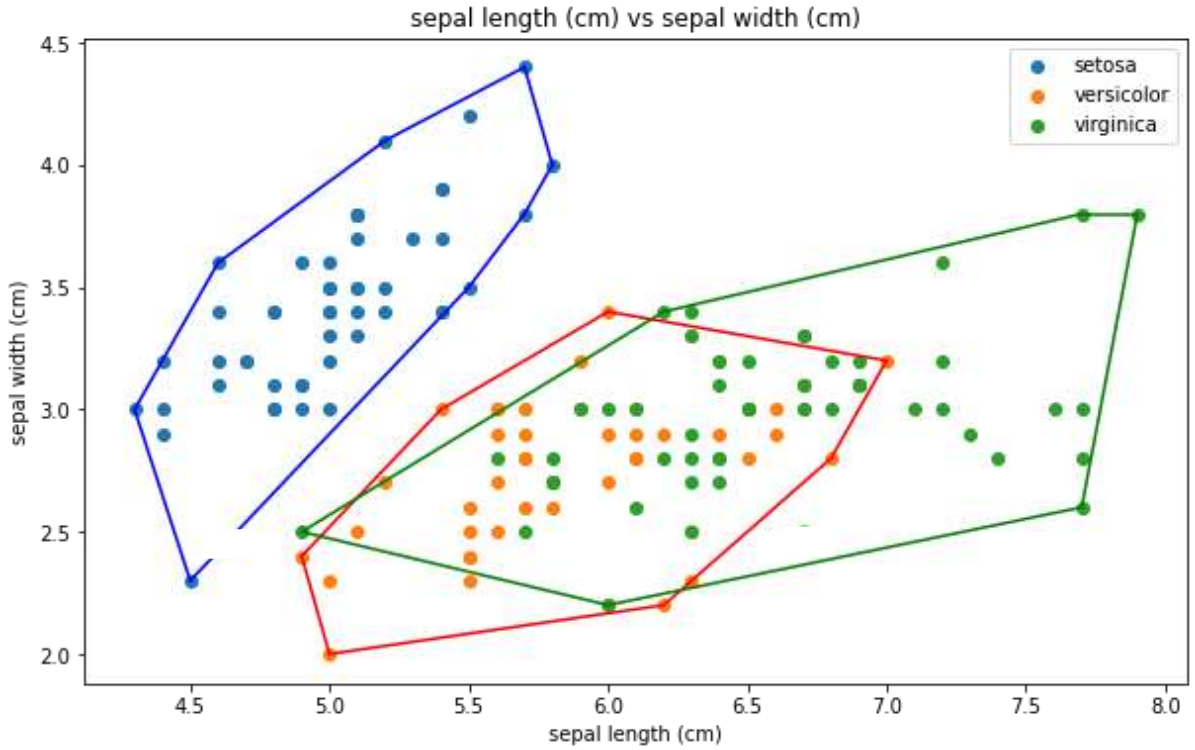
LAPORAN TUGAS KECIL 2

Mata Kuliah IF2211 Strategi Algoritma



Nama Penulis:

Aji Andhika Falah 13520012

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

BAB I

Algoritma *Divide and Conquer*

Definisi dari *divide* adalah membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya berukuran hampir sama). Definisi dari *conquer* adalah menyelesaikan masing-masing upa-persoalan (secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar). Jika digabung, definisi dari algoritma *divide and conquer* adalah mengabungkan solusi masing-masing upa-persoalan sehingga membentuk solusi persoalan semula.

Untuk mencari convex hull dari sekumpulan titik, akan digunakan algoritma *divide and conquer* dengan langkah-langkah:

1. Mencari titik tertinggi dan terendah dari sekumpulan titik
2. Membentuk sebuah garis dari kedua garis tersebut
3. Membagi titik-titik tersebut menjadi di kiri garis dan di kanan garis. Jika ada titik yang ada di garis tersebut, maka dapat diabaikan. letak garis dapat dihitung dengan rumus

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

   dengan (x_1,y_1) adalah titik terbawah, (x_2,y_2) adalah titik teratas, dan (x_3,y_3) adalah titik yang dicari
4. Untuk sebuah bagian (misal bagian kiri), ada dua kemungkinan:
   a) Jika tidak ada titik di bagian tersebut, maka garis pembaginya adalah pembentuk convex hull
   b) Jika ada titik di bagian tersebut, maka dilihat jumlah titiknya. Jika ada 1, maka garis dari salah satu ujung garis pembagi ke titik tersebut dan garis dari titik tersebut ke ujung garis pembagi lainnya adalah pembentuk convex hull. Jika lebih dari 1, dicari titik terjauh. Misal p1 adalah titik teratas garis pembagi, p2 adalah titik terbawah garis pembagi, dan p3 adalah titik terjauh tersebut. Akan dibentuk garis p1p3 dan p2p3. Titik di dalam segitiga p1p2p3 bisa diabaikan. Kumpulkan titik yang ada di bagian atas p1p3 dan di bawah p2p3.

5. Terapkan langkah 4 untuk kedua bagian tersebut dan bagian kanan sampai titik diluar "kosong"
6. Kumpulkan garis-garis pembentuk convex hull untuk membentuk bidang convex hull

BAB 2

Source Program

Program ini ditulis dalam bahasa Python yang menggunakan Jupiter Notebook. Terdapat 2 file di program ini, main.ipynb dan myConvexHull.py. File main.ipynb adalah file Jupiter Notebook untuk mengubah dataset menjadi titik-titik yang dapat diolah myConvexHull, serta menampilkan hasil dari myConvexHull tersebut. File myConevHull adalah library berisi algoritma *divide and conquer* untuk mencari convex hull dari titik-titik yang diterima, dan menghasilkan garis-garis convex hull.

File main.ipynb

```
# Import library yang dibutuhkan
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
import matplotlib.pyplot as plt

#Algoritma Convex Hull buatan
from myConvexHull import ConvexHull as cv
```

```
# Convex Hull Sepal Length vs Sepal Width dari Dataset Iris Plant
#load dataset dari scikit
data = datasets.load_iris()

#membentuk dataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

#membentuk convex hull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[0]) + " vs " + str(data.feature_names[1]))
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
    hull = cv(bucket)
    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()
```

```
# Convex Hull Petal Length vs Petal Width dari Dataset Iris Plant
#load dataset dari scikit
data = datasets.load_iris()

#membentuk dataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

#membentuk convex hull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[2]) + " vs " + str(data.feature_names[3]))
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
    hull = cv(bucket)
    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()
```

```
# Convex Hull Alcohol vs Malic_Acid dari Dataset Wine Recognition
#load dataset dari scikit
data = datasets.load_wine()

#membentuk dataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

#membentuk convex hull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[0]) + " vs " + str(data.feature_names[1]))
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
    hull = cv(bucket)
    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()
```

```
# Convex Hull mean radius vs mean texture dari Dataset Wine Recognition
#load dataset dari scikit
data = datasets.load_breast_cancer()

#membentuk dataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

#membentuk convex hull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[0]) + " vs " + str(data.feature_names[1]))
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
    hull = cv(bucket)
    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()
```

File myConvexHull.py

```
import numpy as np

#Fungsi Convex Hull
def ConvexHull(points):
    left = []
    right = []
    det = 0
    detMax = 0
    detMin = 0
    add = []
    result = np.array([[]], dtype=int)

    #Mencari titik teratas dan terbawah
    top = np.append(points[0], [int(0)])
    bottom = np.append(points[0], [0])
    i = 0
    for point in points:
        if point[1] > top[1]:
            add = np.append(point, [int(i)])
```

```
            top = add
        if point[1] < bottom[1]:
            add = np.append(point, [i])
            bottom = add
        i += 1

    #Mengumpulkan titik-titik di bagian kiri dan kanan dari garis yang
terbentuk dari titik teratas dan terbawah,
    #juga mencari titik terjauh di kiri dan kanan garis tersebut
    i = 0
    for point in points:
        if ((point[0] == top[0] and point[1] == top[1]) or (point[0] ==
bottom[0] and point[1] == bottom[1])):
            pass
        else:
            det = bottom[0]*top[1]+top[0]*point[1]+point[0]*bottom[1]-
bottom[0]*point[1]-point[0]*top[1]-top[0]*bottom[1]
            if det > 0:
                add = np.append(point, [i])
                left.append(add)
                if det > detMax:
                    detMax = det
                    pointMax = add
            if det < 0:
                add = np.append(point, [i])
                right.append(add)
                if det < detMin:
                    detMin = det
                    pointMin = add
        i += 1

    #Mencari Convex Hull bagian kiri dan kanan
    result = ConvexHullLeft(pointMax, top, bottom, left)
    result = np.append(result, ConvexHullRight(pointMin, top, bottom, right),
axis=0)
    return result


#Fungsi Convex Hull Bagian Kiri
def ConvexHullLeft(left, top, bottom, points):
    above = []
    below = []
    detMax = 0
    pointMaxAbove = []
    pointMaxBelow = []
    result = np.array([[]], dtype=int)

    #basis, membentuk garis convex hull
    if(len(points)==0):
```

```
        return(np.array([[top[2], bottom[2]]], dtype=int))
    if(len(points)==1):
        return(np.array([[top[2], left[2]], [left[2], bottom[2]]], dtype=int))
    else:

        #Mengumpulkan titik-titik di atas dari garis yang terbentuk dari titik
teratas dan terjauh di kiri,
        #juga mencari titik terjauh yang diatas garis tersebut
        for point in points:
            if (point[0] == left[0] and point[1] == left[1]):
                pass
            else:
                det = left[0]*top[1]+top[0]*point[1]+point[0]*left[1]-
left[0]*point[1]-point[0]*top[1]-top[0]*left[1]
                if det > 0:
                    above.append(point)
                    if det > detMax:
                        detMax = det
                        pointMaxAbove = point

        #Mengumpulkan titik-titik di bawah dari garis yang terbentuk dari
titik terbawah dan terjauh di kiri,
        #juga mencari titik terjauh yang dibawah garis tersebut
        detMax = 0
        for point in points:
            if (point[0] == left[0] and point[1] == left[1]):
                pass
            else:
                det = bottom[0]*left[1]+left[0]*point[1]+point[0]*bottom[1]-
bottom[0]*point[1]-point[0]*left[1]-left[0]*bottom[1]
                if det > 0:
                    below.append(point)
                    if det > detMax:
                        detMax = det
                        pointMaxBelow = point

        #Mencari titik convex hull lain
        result = ConvexHullLeft(pointMaxAbove, top, left, above)
        result = np.append(result, ConvexHullLeft(pointMaxBelow, left, bottom,
below), axis=0)
        return result


#Fungsi Convex Hull Bagian Kanan
def ConvexHullRight(right, top, bottom, points):
    above = []
    below = []
    detMin = 0
    pointMinAbove = []
```

```
    pointMinBelow = []
    result = np.array([[]], dtype=int)

    #basis, membentuk garis convex hull
    if(len(points)==0):
        return(np.array([[top[2], bottom[2]]], dtype=int))
    if(len(points)==1):
        return(np.array([[top[2], right[2]], [right[2], bottom[2]]],
dtype=int))
    else:

        #Mengumpulkan titik-titik di atas dari garis yang terbentuk dari titik
teratas dan terjauh di kanan,
        #juga mencari titik terjauh yang diatas garis tersebut
        for point in points:
            if (point[0] == right[0] and point[1] == right[1]):
                pass
            else:
                det = right[0]*top[1]+top[0]*point[1]+point[0]*right[1]-
right[0]*point[1]-point[0]*top[1]-top[0]*right[1]
                if det < 0:
                    above.append(point)
                    if det < detMin:
                        detMin = det
                        pointMinAbove = point

        #Mengumpulkan titik-titik di bawah dari garis yang terbentuk dari
titik terbawah dan terjauh di kanan,
        #juga mencari titik terjauh yang dibawah garis tersebut
        detMin = 0
        for point in points:
            if (point[0] == right[0] and point[1] == right[1]):
                pass
            else:
                det = bottom[0]*right[1]+right[0]*point[1]+point[0]*bottom[1]-
bottom[0]*point[1]-point[0]*right[1]-right[0]*bottom[1]
                if det < 0:
                    below.append(point)
                    if det < detMin:
                        detMin = det
                        pointMinBelow = point

        #Mencari titik convex hull lain
        result = ConvexHullRight(pointMinAbove, top, right, above)
        result = np.append(result, ConvexHullRight(pointMinBelow, right,
bottom, below), axis=0)
        return result
```

BAB 3

Screenshot input dan *output*

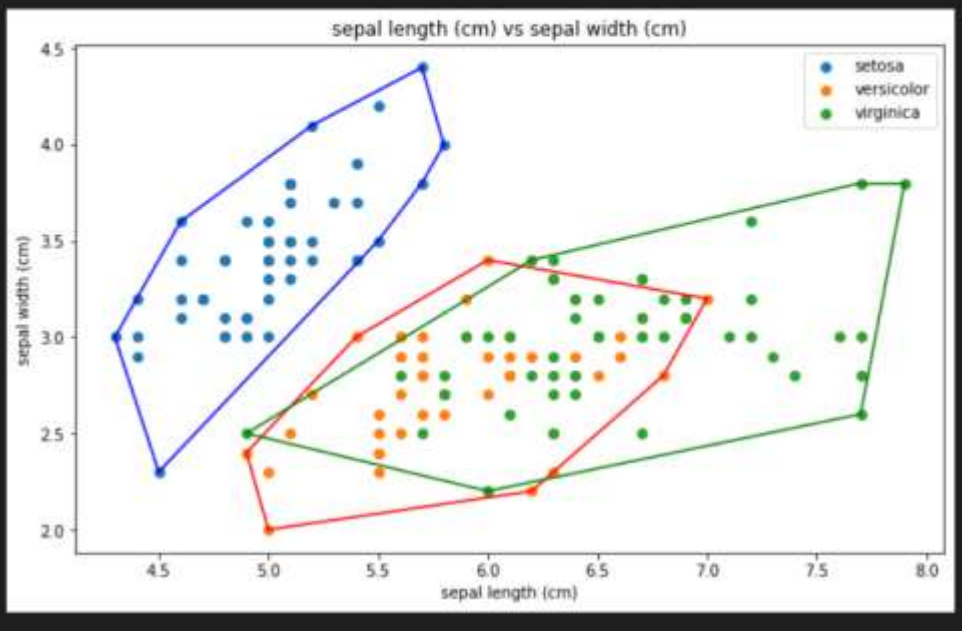
1. Convex Hull Sepal Length vs Sepal Width dari Dataset Iris Plant

input:

```
#load dataset dari scikit
data = datasets.load_iris()

#membentuk dataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

#membentuk convex hull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[0]) + " vs " + str(data.feature_names[1]))
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
    hull = cv(bucket)
    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()
```

output:

```
            plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
    plt.legend()
[6]  ✓ 0.2s
...  (150, 5)
     <matplotlib.legend.Legend at 0x2a6b3ffb850>
```



2. Convex Hull Petal Length vs Petal Width dari Dataset Iris Plant

input:

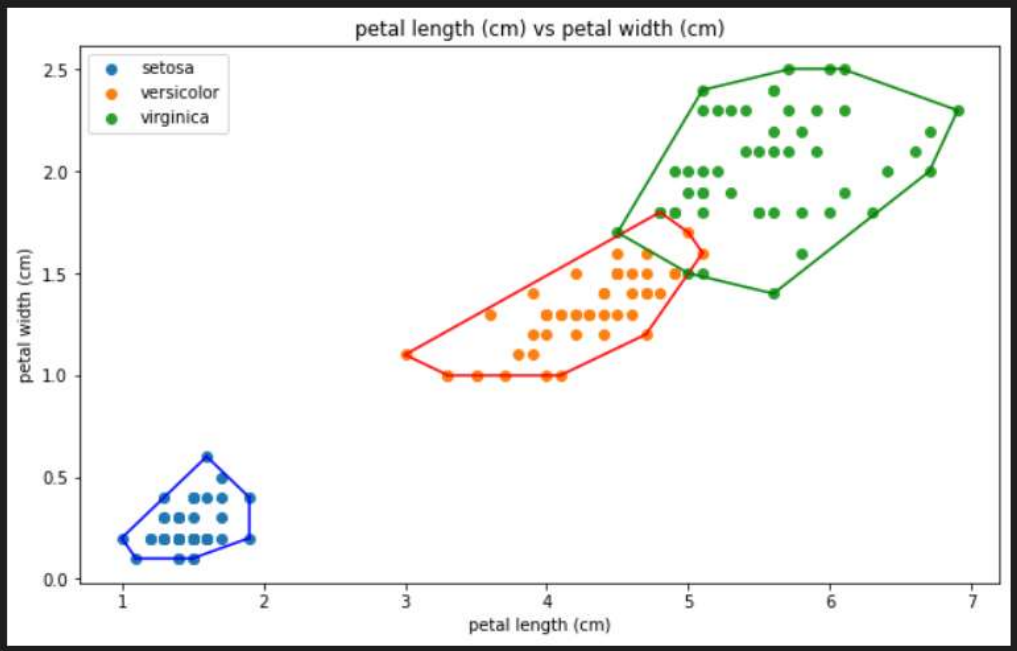
```
#load dataset dari scikit
data = datasets.load_iris()

#membentuk dataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

#membentuk convex hull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[2]) + " vs " + str(data.feature_names[3]))
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
    hull = cv(bucket)
    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()
```

output:

```
            plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
    plt.legend()
```

[3]

```
(150, 5)
<matplotlib.legend.Legend at 0x1d01d5b1e50>
```



3. Convex Hull Alcohol vs Malic_Acid dari Dataset Wine Recognition

input:

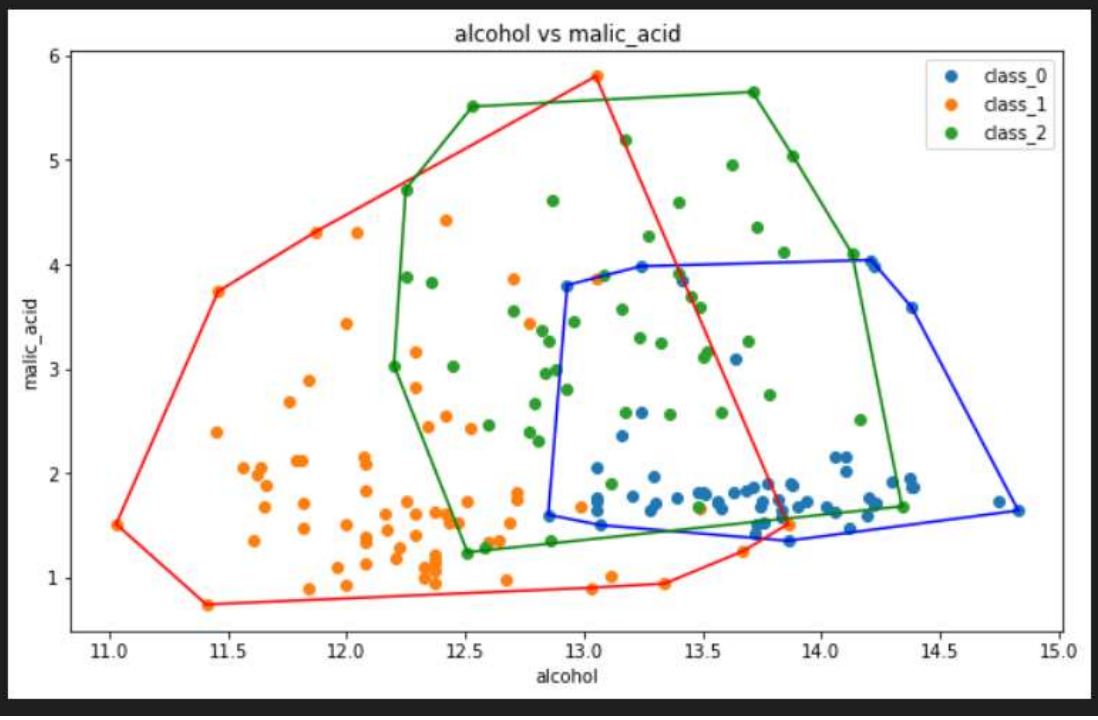
```
#load dataset dari scikit
data = datasets.load_wine()

#membentuk dataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

#membentuk convex hull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[0]) + " vs " + str(data.feature_names[1]))
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
    hull = cv(bucket)
    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()
```

output:

```
            plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
    plt.legend()
```

[5]

```
(178, 14)
<matplotlib.legend.Legend at 0x1d01d5caf70>
```



4. Convex Hull mean radius vs mean texture dari Dataset Wine Recognition

input:

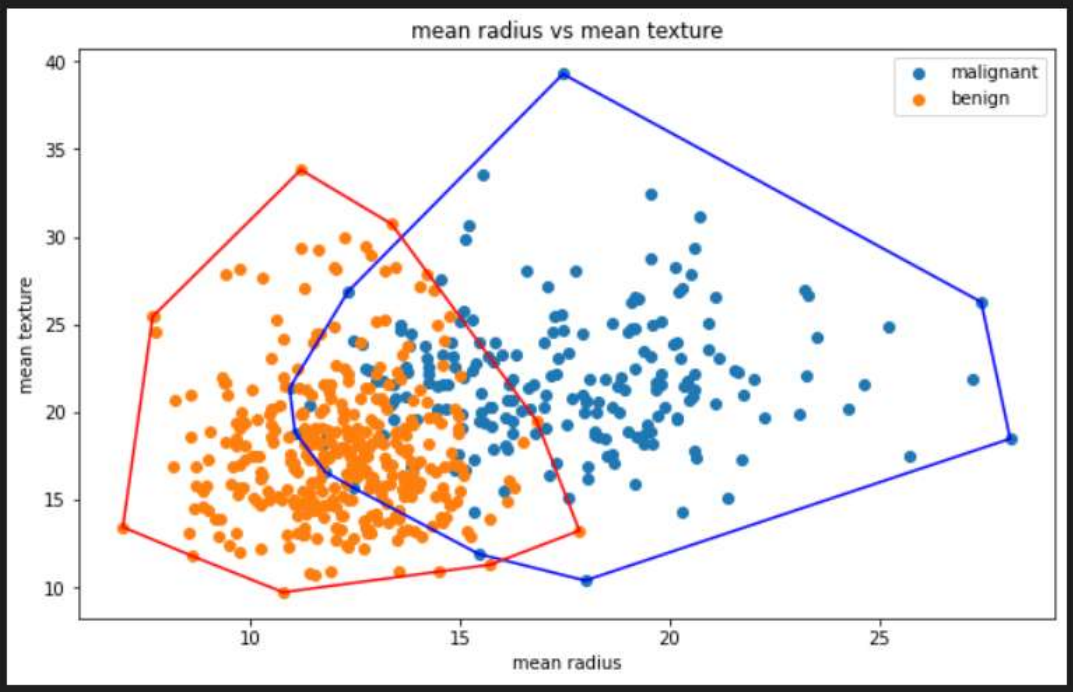
```
#load dataset dari scikit
data = datasets.load_breast_cancer()

#membentuk dataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

#membentuk convex hull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[0]) + " vs " + str(data.feature_names[1]))
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
    hull = cv(bucket)
    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()
```

output:

```
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
    plt.legend()
```

[19]

```
(569, 31)
<matplotlib.legend.Legend at 0x1373c3441f0>
```



BAB 4

GitHub program

Untuk program saya, bisa dilihat di link berikut:

https://github.com/Enderageous/Tucil2_13520012

Lampiran

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.	√	