Laporan Tugas Kecil 2 “Convex Hull”

IF2211 Strategi Algoritma

Disusun oleh:

Haidar Ihzaulhaq 13520150

Sekolah Tinggi Elektro dan Informatika

Institut Teknologi Bandung

2021/2022

1. Algoritma Divide & Conquer

Pada program ini, strategi algoritma yang digunakan adalah strategi *Divide and Conquer*. Strategi yang membagi permasalahan menjadi beberapa bagian, kemudian diselesaikan tiap bagiannya dan solusi dari tiap bagian kemudian disatukan untuk memperoleh solusi utuh persoalan. Dalam pencarian *convex hull* ini, strategi yang diterapkan yaitu:

a. Melakukan Sorting
   Langkah pertama yang dilakukan adalah dengan melakukan *sorting* pada dataset yang diterima, *sorting* dilakukan secara menaik pada nilai absisnya. Jika nilai absis sama, maka akan diurutkan menaik berdasarkan ordinatnya.
b. Menentukan garis pembagi utama, yaitu garis S
   Garis pembagi utama ini adalah garis yang membagi titik-titik menjadi dua bagian, yaitu bagian kanan dan kiri atau bagian atas dan bawah. Garis S ini adalah garis yang terbentuk dari dua titik yang berada di ujung-ujung koordinat berdasarkan absisnya (absis minimal dan absis maksimal) atau pada dataset dapat diakses dengan indeks ke-1 dan indeks ke-n (n: banyak dataset).
c. Membagi titik-titik menjadi dua bagian
   Titik-titik pada dataset kemudian dibagi menjadi dua bagian, yaitu bagian kanan dan kiri atau bagian atas dan bawah (tergantung bagaimana kemiringan garis).
d. Menentukan titik terjauh dari garis pada tiap bagian
   Di tiap bagian, dicari titik-titik yang memiliki jarak terjauh dengan garis S. Kemudian bentuk dua garis baru, yaitu garis S1 dan S2. Garis S1 adalah garis yang dibentuk oleh titik pertama dari garis sebelumnya dengan titik yang memiliki jarak terjauh dan S2 adalah titik yang dibentuk oleh titik terjauh dari garis sebelumnya dengan titik kedua pada garis sebelumnya (titik pertama adalah titik dengan absis lebih kecil dibanding titik kedua).
e. Membagi titik menjadi tiga bagian
   Setelah memiliki dua garis S1 dan S2, selanjutnya adalah membagi titik-titik pada bagian tersebut menjadi tiga bagian. Bagian pertama, atau misal data1, adalah titik-titik yang berada di luar garis S1. Bagian kedua atau misal data2 adalah titik-titik yang berada di dalam garis S1 dan S2, kemudian bagian ketiga atau misal data3 adalah titik-titik yang berada di luar garis S2.
f. Kemudian, ulangi langkah (d) dan (e) untuk data1 dan data3 (data2 tidak perlu) hingga tidak ada lagi titik-titik yang tersisa.
g. Ketika titik-titik sudah ada, maka garis yang terbentuk adalah garis yang menjadi bagian dari *convex hull*.

- Maksud dari di luar pada poin (e) adalah jika bagian yang dicek adalah bagian atas/kiri dari garis S, maka titik yang berada di luar adalah titik yang memiliki determinan > 0 sedangkan jika yang dicek adalah bagian bawah/kanan, maka titik yang berada di luar adalah titik yang memiliki determinan < 0. Nilai determinan yang dimaksud adalah jika garis dibentuk oleh titik X1 dan X2 serta titik tersebut adalah X3, maka yang dihitung adalah determinan dari matrix berikut:

$$\begin{vmatrix} x1 & y1 & 1 \\ x2 & y2 & 1 \\ x3 & y3 & 1 \end{vmatrix}$$

2. Source Program

a. myConvecHull.py

```
from cmath import sqrt
import numpy as np

simplices = np.array([[1,1]])       #for initial, later will be deleted
real_data = np.array([[0.0,0.0]])   #for initial, later will be deleted

def myConvexHull(dataset) :
   #start program
   #sort the data
   global real_data
   global simplices
   simplices = np.array([[1,1]])
   real_data = dataset
   data = dataset[np.lexsort((dataset[:,1],dataset[:,0]))]
   #set the first point and last point as line S (first point is min(x) and
last_point is max(x))
   first_point = data[0]
   last_point = data[len(data)-1]

   #divide points into two part, right and left, based on their determinant
   sLeft = np.array([[0.0,0.0]])
   sRight = np.array([[0.0,0.0]])
   #check every point from indeks 2 untill n-1
   for i in range(1,len(data)) :
      determinant = det(first_point,last_point,data[i])
      if (determinant > 0) :
         sLeft = np.vstack((sLeft,data[i]))
      elif (determinant < 0) :
         sRight = np.vstack((sRight,data[i]))
   #start checking
   sLeft = np.delete(sLeft,0,0)
   sRight = np.delete(sRight,0,0)
   #call the divide function
   divideUp(sLeft,first_point,last_point)
   divideDown(sRight,first_point,last_point)
   #edit the final result
   simplices = np.delete(simplices,0,0)
   simplices = simplices.astype('int')
   return simplices


def divideUp(data,point1,point2) :
   global simplices
   #check if there any point in the data
   if(len(data) == 0)  :
```

```
      #save the point1 and point2 as convex hull, save the idx of the point
      idx_p1 = -1
      idx_p2 = -1
      for i in range (len(real_data)) :
         if (real_data[i][0] == point1[0] and real_data[i][1] == point1[1]) :
            idx_p1 = i
         if (real_data[i][0] == point2[0] and real_data[i][1] == point2[1]) :
            idx_p2 = i
      idx_pair = np.array([idx_p1,idx_p2])
      simplices = np.vstack((simplices,idx_pair))

   else :
      #data still have some points, search points that have max distance with
the line
      max_dist = -1.0
      max_point = point1
      for i in range (len(data)) :
         determinant = det(point1,point2,data[i])
         dist = determinant/sqrt((point2[0]-point1[0])**2+(point2[1]-
point2[1])**2)
         dist = abs(dist)
         if (dist > max_dist) :
            max_point = data[i]
            max_dist = dist
         elif (dist == max_dist) :
            if (data[i][0] < max_point[0]) :
               max_point = data[i]

      #get the max point
      #check for the possible points in the left S1 (S1 = point1->max_point)
      S1 = np.array([[0.0,0.0]])
      for i in range (len(data)) :
         determinant = det(point1,max_point,data[i])
         if (determinant > 0 and (max_point[0] != data[i][0] or max_point[1]
!= data[i][1])) :
            #it's in the left of the line
            S1 = np.vstack((S1,data[i]))
      S1 = np.delete(S1,0,0)

      #check for the possible points in the right S2 (S2 = point2->max_point)
      S2 = np.array([[0.0,0.0]])
      for i in range (len(data)) :
         determinant = det(max_point,point2,data[i])
         if (determinant > 0 and (max_point[0] != data[i][0] or max_point[1]
!= data[i][1])) :
            #it's in the right of the line
            S2 = np.vstack((S2,data[i]))
      S2 = np.delete(S2,0,0)
```

```
      #call the divide function again for S1 and S2
      divideUp(S1,point1,max_point)
      divideUp(S2,max_point,point2)

def divideDown(data,point1,point2) :
   global simplices
   #check if there any point in the data
   if(len(data) == 0)  :
      #save the point1 and point2 as convex hull, save the index of the point
      idx_p1 = -1
      idx_p2 = -1
      for i in range (len(real_data)) :
         if (real_data[i][0] == point1[0] and real_data[i][1] == point1[1]) :
            idx_p1 = i
         if (real_data[i][0] == point2[0] and real_data[i][1] == point2[1]) :
            idx_p2 = i
      idx_pair = np.array([idx_p1,idx_p2])
      simplices = np.vstack((simplices,idx_pair))

   else :
      #data still have some points, search points that have max distance with
the line
      max_dist = -1.0
      max_point = point1
      for i in range (len(data)) :
         determinant = det(point1,point2,data[i])
         dist = determinant/sqrt((point2[0]-point1[0])**2+(point2[1]-
point2[1])**2)
         dist = abs(dist)
         if (dist > max_dist) :
            max_point = data[i]
            max_dist = dist
         elif (dist == max_dist) :
            if (data[i][0] < max_point[0]) :
               max_point = data[i]

      #get the max point
      #check for the possible points in the left S1 (S1 = point1->max_point)
      S1 = np.array([[0.0,0.0]])
      for i in range (len(data)) :
         determinant = det(point1,max_point,data[i])
         if (determinant < 0 and (max_point[0] != data[i][0] or max_point[1]
!= data[i][1])) :
            #it's in the left of the line
            S1 = np.vstack((S1,data[i]))
      S1 = np.delete(S1,0,0)
```

```
      #check for the possible points in the right S2 (S2 = point2->max_point)
      S2 = np.array([[0.0,0.0]])
      for i in range (len(data)) :
         determinant = det(max_point,point2,data[i])
         if (determinant < 0 and (max_point[0] != data[i][0] or max_point[1]
!= data[i][1])) :
            #it's in the right of the line
            S2 = np.vstack((S2,data[i]))
      S2 = np.delete(S2,0,0)

      #call the divide function again for S1 and S2
      divideDown(S1,point1,max_point)
      divideDown(S2,max_point,point2)

def det(point1, point2, point3) :
   result = 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 result
```

b. main.ipynb

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import myConvexHull

from sklearn import datasets
data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

plt.figure(figsize=(10,6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal Length')
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 = myConvexHull.myConvexHull(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()
```

3. Screenshot Input & Output

a. Dataset Iris: Sepal Width vs Sepal Length

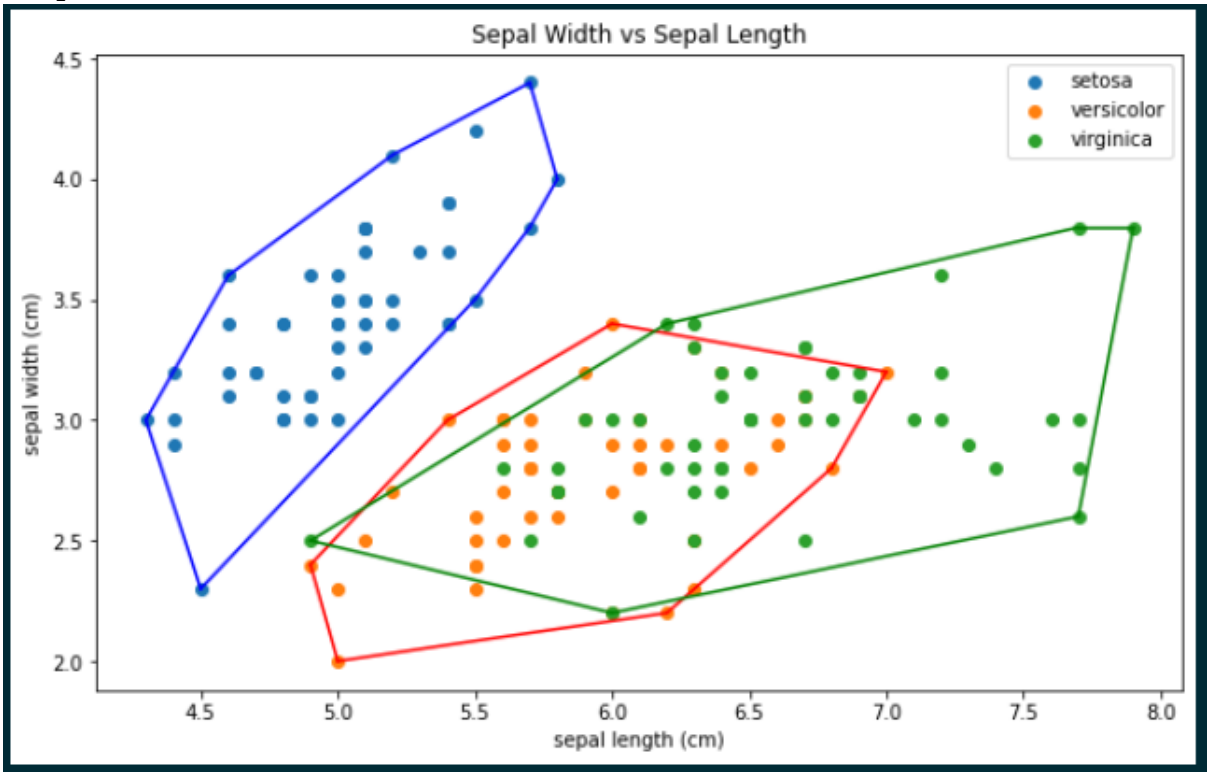
Input

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import myConvexHull

from sklearn import datasets
data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

plt.figure(figsize=(10,6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal Length')
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 = myConvexHull.myConvexHull(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



b. Dataset Iris: Petal Width vs Petal Length

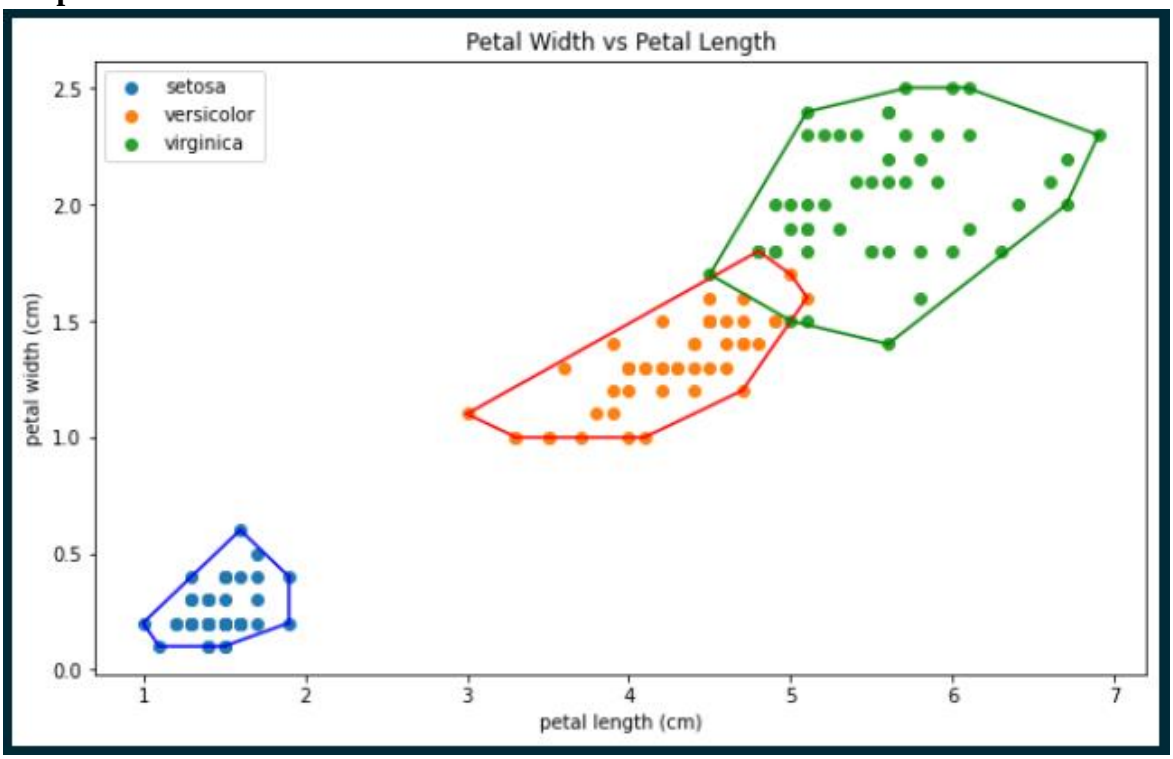
Input

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import myConvexHull

from sklearn import datasets
data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

plt.figure(figsize=(10,6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal Length')
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 = myConvexHull.myConvexHull(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



c. Dataset Wine: Ash vs Alcalinity of Ash

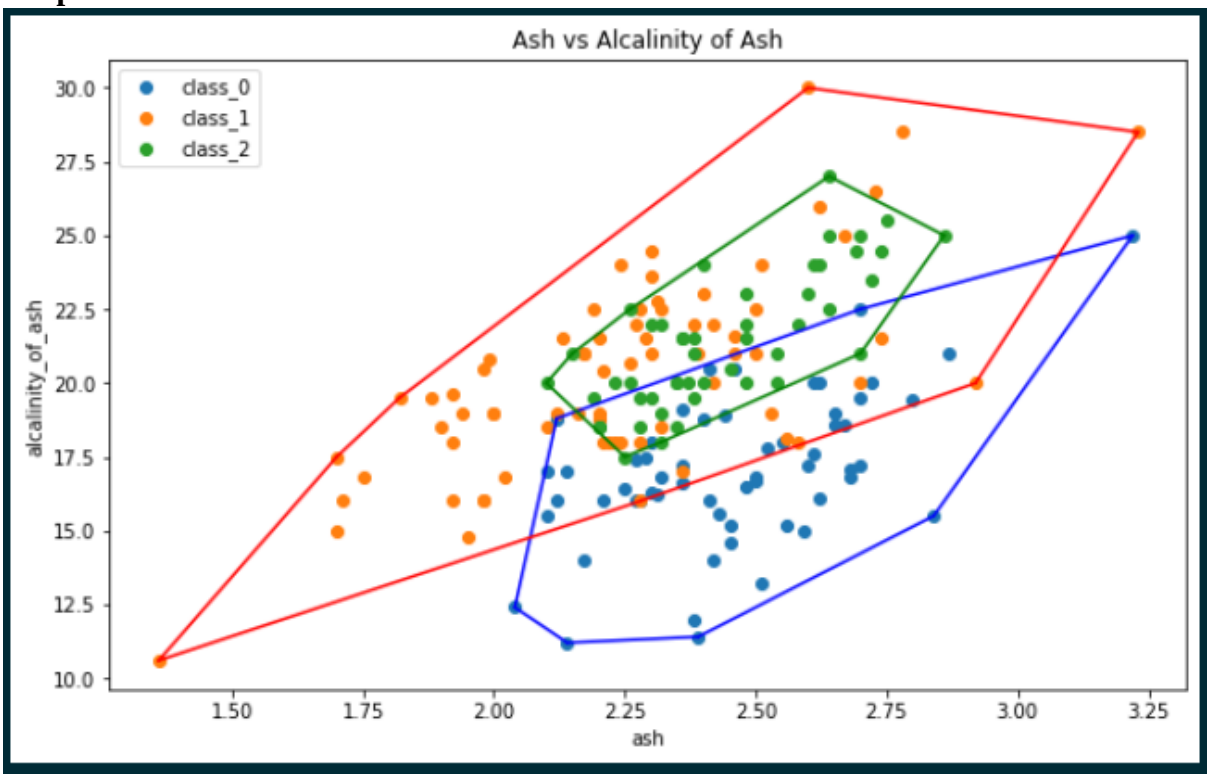
Input

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import myConvexHull

from sklearn import datasets
data = datasets.load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

plt.figure(figsize=(10,6))
colors = ['b','r','g']
plt.title('Ash vs Alcalinity of Ash')
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 = myConvexHull.myConvexHull(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



d. Dataset Wine: Alcohol vs Color Intensity

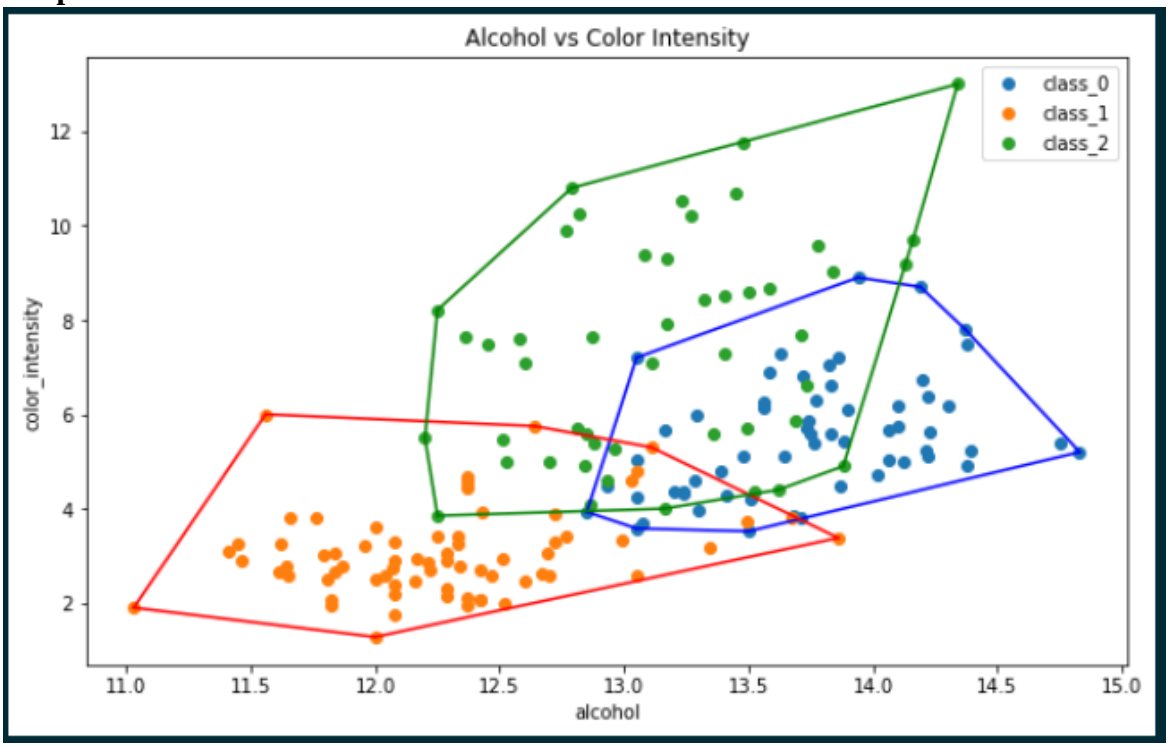
Input

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import myConvexHull

from sklearn import datasets
data = datasets.load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

plt.figure(figsize=(10,6))
colors = ['b','r','g']
plt.title('Alcohol vs Color Intensity')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[9])
for i in range (len(data.target_names)) :
    bucket = df[df['Target']==i]
    bucket = bucket.iloc[:,[0,9]].values
    hull = myConvexHull.myConvexHull(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



e. Dataset Breast Cancer Wisconsin: Mean Texture vs Worst Texture

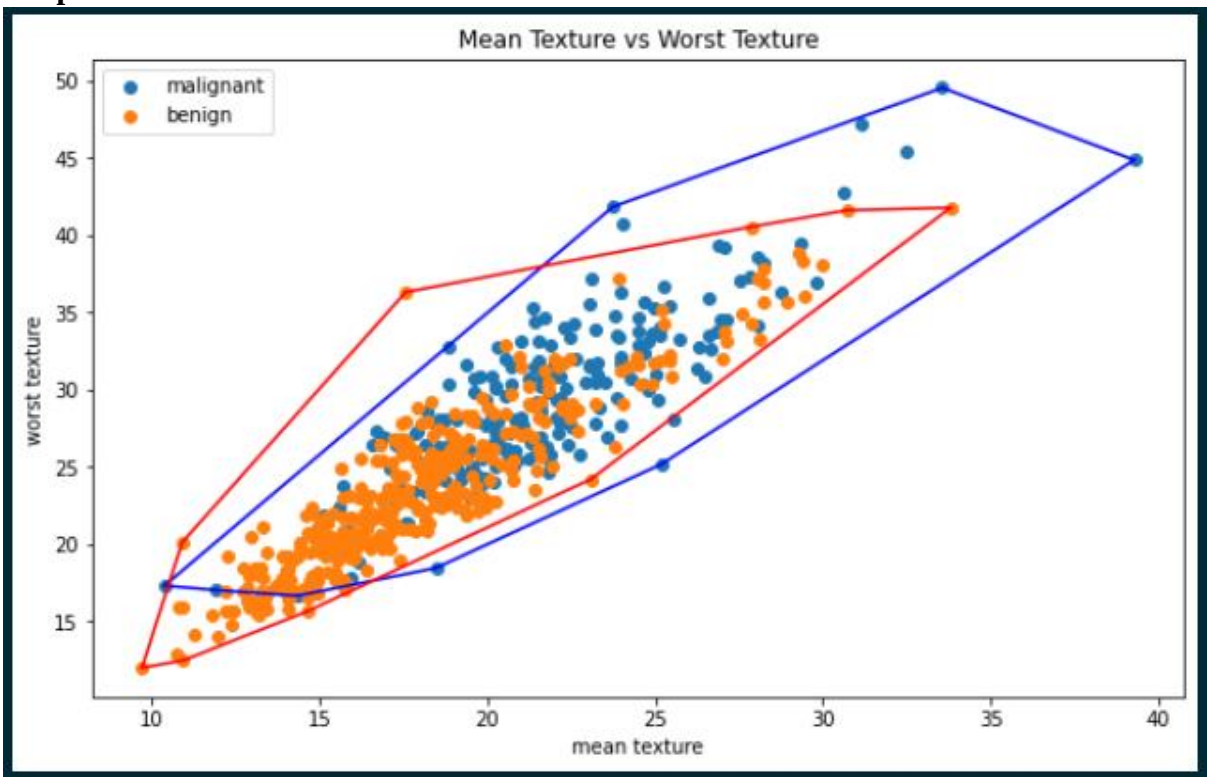
Input

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import myConvexHull

from sklearn import datasets
data = datasets.load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

plt.figure(figsize=(10,6))
colors = ['b','r','g']
plt.title('Mean Texture vs Worst Texture')
plt.xlabel(data.feature_names[1])
plt.ylabel(data.feature_names[21])
for i in range (len(data.target_names)) :
    bucket = df[df['Target']==i]
    bucket = bucket.iloc[:,[1,21]].values
    hull = myConvexHull.myConvexHull(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



f. Dataset Breast Cancer Wisconsin: Mean Perimeter vs Worst Perimeter

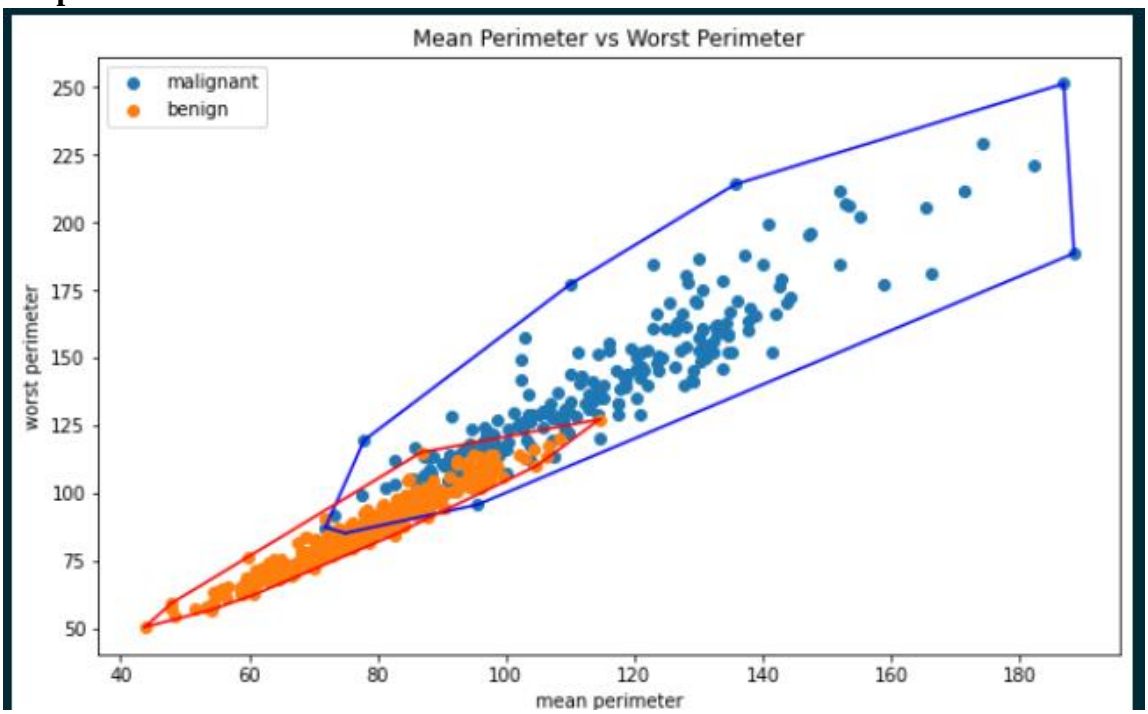
Input

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import myConvexHull

from sklearn import datasets
data = datasets.load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

plt.figure(figsize=(10,6))
colors = ['b','r','g']
plt.title('Mean Perimeter vs Worst Perimeter')
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[22])
for i in range (len(data.target_names)) :
    bucket = df[df['Target']==i]
    bucket = bucket.iloc[:,[2,22]].values
    hull = myConvexHull.myConvexHull(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



4. **Drive Program**
   https://github.com/haidarihza/myConvexHull

Checklist Poin:

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	✓	