TUGAS KECIL 2 IF2211 STRATEGI ALGORITMA:

IMPLEMENTASI CONVEX HULL untuk VISUALISASI TES *LINEAR SEPARABILITY DATASET* dengan ALGORITMA *DIVIDE AND CONQUER*

Dibuat oleh :

Rizky Akbar Asmaran (13520111)

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

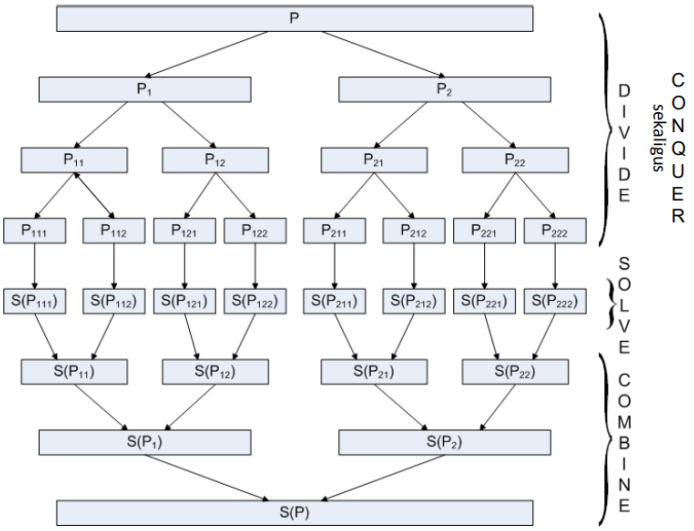
BANDUNG

2022

A. Algoritma Divide and Conquer

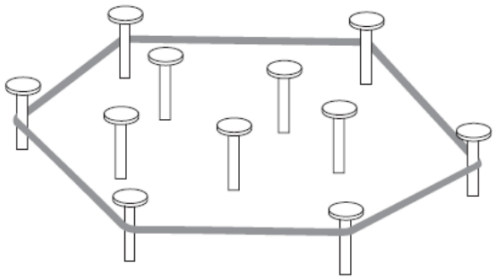
1. Definisi

Divide and Conquer adalah suatu algoritma yang membagi sebuah persoalan menjadi beberapa sub-persoalan yang memiliki kemiripan dengan persoalan semua namun berukuran lebih kecil. Algoritma ini melakukan dua buah aksi penting, yaitu conquer(solve) dan combine. Conquer menyelesaikan masing masing sub-persoalan dan combine adalah menggabungkan solusi masing masing sub-persoalan sehingga membentuk solusi persoalan semula. Berikut ilustrasi dari algoritma Divide and Conquer.



2. Implementasi Algoritma Divide and Conquer untuk menyelesaikan masalah *Convex Hull*

Convex hull adalah salah satu permasalahan yang berkaitan dengan bidang geometri komputasional. Convex hull ini didefinisikan sebagai himpunan titik terkecil yang mengandung koordinat yang sifatnya convex. Suatu koordinat bisa dikatakan convex apabila 2 buah titik sembarang pada semua segmen garis berakhir pada kedua titik tersebut terdapat pada koordinat tersebut.



Pada permasalahan ini, terdapat beberapa algoritma yang dapat digunakan seperti, algoritma *Graham Scan,* algoritma, *Jarvis's March*, dan algoritma *Quickhull*. Pada tugas kecil 2 ini, saya menggunakan algoritma *Quickhull* dan menggunakan konsep strategi Divide and Conquer. Berikut merupakan langkah langkah untuk mencari *Convex Hull* dengan menggunakan algoritma *Quickhull* :

1. Cari titik maksimum(kanan) dan minimum(kiri) dari suatu set of titik.
2. Setelah didapatkan titik titik yang saling berkaitan maka partisi set of titik dibagi menjadi dua dengan menggunakan determinan, apabila nilai determinan bernilai minus, maka set titik tersebut berada di segmen kiri(bawah) S1 dan apabila nilai determinan bernilai positif, maka set titik tersebut berada di segmen kanan(atas) S2.
3. Kumpulan titik pada S1 akan membentuk convex hull bagian bawah, dan kumpulan titik pada S2 akan membentuk convex hull bagian atas. S merupakan hasil gabungan (merge) dari S1 dan S2.
4. Sebelum dilakukan penggabungan pada S1 dan S2. diterapkan algoritma divide and conquer pada S1 dan S2, dalam program saya menggunakan algoritma quickhull dengan menggunakan fungsi quickhull_kiri dan quickhull_kanan, secara rekursif sampai tidak ada lagi titik yang bisa diproses lagi.
5. Hasil dari langkah ke-4 dilakukan penggabungan antara hasil rekursif S1 dan hasil rekursif S2 yang menjadi hasil penggabungan S.
6. Setelah proses 5 selesai, maka S akan menghasilkan bentuk Convex Hull yang terbentuk dari titik-titik pembentuk convex hull

B. Kode Program

<table>
<tr><th>Fungsi Determinan</th><th>Fungsi Maxmin</th></tr>
<tr><td>

```
# fungsi untuk menghitung
determinan dari 3 titik
def determinan(titik1, titik2,
titik3):
    return (titik1[0] * titik2[1])
+ (titik1[1] * titik3[0]) +
(titik2[0] * titik3[1]) -
(titik3[0] * titik2[1]) -
(titik3[1] * titik1[0]) -
(titik2[0] * titik1[1])
```

</td><td>

```
# fungsi untuk mengembalikan nilai minimum dan maksimum dari suatu list titik
def maxmin(list_titik):
    minimum = list_titik[0]
    maximum = list_titik[0]

    for titik in list_titik:
        if(titik[0] <= minimum[0]):
            minimum = titik
        if(titik[0]) >= maximum[0]:
            maximum = titik
    return minimum,maximum
```

</td></tr>
<tr><th>Fungsi Angle</th><th>Fungsi bagiSisi</th></tr>
</table>

```python
# fungsi untuk mencari sudut
def angle(A,B,C):
    Ax, Ay = A[0]-B[0], A[1]-B[1]
    Cx, Cy = C[0]-B[0], C[1]-B[1]
    a = atan2(Ay, Ax)
    c = atan2(Cy, Cx)
    return (c - a)
```

```python
# fungsi untuk membagi 2 sisi kiri atau kanan dimana:
# kalo determinannya < 0 -> berada di sisi kiri
# kalo determinannya > 0 -> berada di sisi kanan
def bagiSisi(points,min,max):
    kiri = []
    kanan = []
    for titik in points:
        if(not((titik[0] == min[0]) and (titik[1] == min[1])) and not((titik[0] == max[0]) and (titik[1] == max[1]))):

if(determinan(min,max,titik)>0):
                kanan.append(titik)

if(determinan(min,max,titik)<0):
                kiri.append(titik)

    return kiri,kanan
```

Fungsi cariIndeks

Fungsi merge

```python
def cariIndeks(titik,coordinate):
    i = 0
    for a in titik:
        if(coordinate[0] == a[0] and coordinate[1] == a[1]):
            return i
        else:
            i += 1
```

```python
# Fungsi untuk menyatukan 2 buah list
def merge(list1, list2):
    for x in list2:
        list1.append(x)
    return list1
```

Fungsi quickhull_kiri

Fungsi quickhull_kanan

```python
# setelah dibagi sisi maka tiap sisi melakukan quickhull
def quickhull_kiri(kiri,min_absis,max_absis,titik):
    if(len(kiri) == 0):
        result = [[cariIndeks(titik,min_absis),cariIndeks(titik,max_absis)]]
        return result
    else:
        sudut = 0
        temp = kiri[0]
        for coordinate in kiri:
            if(sudut < angle(coordinate,min_absis,max_absis)):
                sudut = angle(coordinate,min_absis,max_absis)
                temp = coordinate
        kiri_baru1,_ = bagiSisi(kiri,min_absis,temp)
        kiri_baru2,_ = bagiSisi(kiri,temp,max_absis)
        # bagian yang rekursi
        result1 = quickhull_kiri(kiri_baru1,min_absis,temp,titik)
        result2 = quickhull_kiri(kiri_baru2,temp,max_absis,titik)
        return merge(result1, result2)
```

```python
def quickhull_kanan(kanan,min_absis,max_absis,titik):
    if(len(kanan)==0):
        result = [[cariIndeks(titik,min_absis),cariIndeks(titik,max_absis)]]
        return result
    else:
        sudut = 0
        temp = kanan[0]
        for coordinate in kanan:
            if(sudut < angle(max_absis,min_absis,coordinate)):
                sudut = angle(max_absis,min_absis,coordinate)
                temp = coordinate
        _,kanan_baru1 = bagiSisi(kanan,min_absis,temp)
        _,kanan_baru2 = bagiSisi(kanan,temp,max_absis)
        # bagian yagn rekursi
        result1 = quickhull_kanan(kanan_baru1,min_absis,temp,titik)
        result2 = quickhull_kanan(kanan_baru2,temp,max_absis,titik)
        return merge(result1, result2)
```

Fungsi myhull

```python
# setelah melakukan quickhull pada masing masing sisi, maka dilakukan penggabungan antara
# quickhull kiri dan kanan
def myHull(titik):
    result = []
    min_absis,max_absis = maxmin(titik)
    kiri,kanan = bagiSisi(titik,min_absis,max_absis)
    result_kiri = quickhull_kiri(kiri,min_absis,max_absis,titik)
    result_kanan = quickhull_kanan(kanan,min_absis,max_absis,titik)
    result = merge(result_kanan, result_kiri)
    return result
```

convexhull.py
Keseluruhan:

convexhull.py > angle

```python
from math import atan2, pi

# fungsi untuk menghitung determinan dari 3 titik
def determinan(titik1, titik2, titik3):
    return (titik1[0] * titik2[1]) + (titik1[1] * titik3[0]) + (titik2[0] * titik3[1]) - (titik3[0] * titik2[1]) - (titik3[1] * titik1[0]) - (titik2[0] * titik1[1])

# fungsi untuk mengembalikan nilai minimum dan maksimum dari suatu list titik
def maxmin(list_titik):
    minimum = list_titik[0]
    maximum = list_titik[0]

    for titik in list_titik:
        if(titik[0] <= minimum[0]):
            minimum = titik
        if(titik[0]) >= maximum[0]:
            maximum = titik
    return minimum,maximum

# fungsi untuk mencari sudut
def angle(A,B,C):
    Ax, Ay = A[0]-B[0], A[1]-B[1]
    Cx, Cy = C[0]-B[0], C[1]-B[1]
    a = atan2(Ay, Ax)
    c = atan2(Cy, Cx)
    return (c - a)

# fungsi untuk membagi 2 sisi kiri atau kanan dimana:
# kalo determinannya < 0 -> berada di sisi kiri
# kalo determinannya > 0 -> berada di sisi kanan
def bagiSisi(points,min,max):
    kiri = []
    kanan = []
    for titik in points:
        if(not((titik[0] == min[0]) and (titik[1] == min[1])) and not((titik[0] == max[0]) and (titik[1] == max[1]))):
            if(determinan(min,max,titik)>0):
                kanan.append(titik)
            if(determinan(min,max,titik)<0):
                kiri.append(titik)

    return kiri,kanan

def cariIndeks(titik,coordinate):
    i = 0
    for a in titik:
        if(coordinate[0] == a[0] and coordinate[1] == a[1]):
            return i
        else:
            i += 1

# Fungsi untuk menyatukan 2 buah list
def merge(list1, list2):
    for x in list2:
        list1.append(x)
    return list1

# melakukan Algoritma Divide and Conquer
# dengan membagi 2 sisi, kiri dan kanan
# lalu dilakukan fungsi quickhull kiri dan kanan (menyelesaikan permasalahan di dua sisi, kiri dan kanan)
def quickhull_kiri(kiri,min_absis,max_absis,titik):
```

```python
    i += 1

def merge(list1, list2):
    for x in list2:
        list1.append(x)
    return list1

def quickhull_kiri(kiri,min_absis,max_absis,titik):
    #print("min_absis",min_absis,"max_absis",max_absis)
    if(len(kiri) == 0):
        result = [[cariIndeks(titik,min_absis),cariIndeks(titik,max_absis)]]
        return result
    else:
        sudut = 0
        temp = kiri[0]
        for coordinate in kiri:
            #print("sudut untuk koordinat",angle(coordinate,min_absis,max_absis),coordinate)
            if(sudut < angle(coordinate,min_absis,max_absis)):
                sudut = angle(coordinate,min_absis,max_absis)
                temp = coordinate
        kiri_baru1,_ = bagiSisi(kiri,min_absis,temp)
        kiri_baru2,_ = bagiSisi(kiri,temp,max_absis)
        #print("ini temp",temp)
        #print("ini kiri baru")
        #print(kiri_baru1,kiri_baru2)
        result1 = quickhull_kiri(kiri_baru1,min_absis,temp,titik)
        result2 = quickhull_kiri(kiri_baru2,temp,max_absis,titik)
        return merge(result1, result2)

def quickhull_kanan(kanan,min_absis,max_absis,titik):
    if(len(kanan)==0):
        result = [[cariIndeks(titik,min_absis),cariIndeks(titik,max_absis)]]
        return result
    else:
        sudut = 0
        temp = kanan[0]
        for coordinate in kanan:
            if(sudut < angle(max_absis,min_absis,coordinate)):
                sudut = angle(max_absis,min_absis,coordinate)
                temp = coordinate
        _,kanan_baru1 = bagiSisi(kanan,min_absis,temp)
        _,kanan_baru2 = bagiSisi(kanan,temp,max_absis)
        result1 = quickhull_kanan(kanan_baru1,min_absis,temp,titik)
        result2 = quickhull_kanan(kanan_baru2,temp,max_absis,titik)
        return merge(result1, result2)

def myHull(titik):
    result = []
```

```python
def myHull(titik):
    result = []
    min_absis,max_absis = maxmin(titik)
    kiri,kanan = bagiSisi(titik,min_absis,max_absis)
    result_kiri = quickhull_kiri(kiri,min_absis,max_absis,titik)
    result_kanan = quickhull_kanan(kanan,min_absis,max_absis,titik)
    result = merge(result_kanan, result_kiri)
    return result
```

main.py

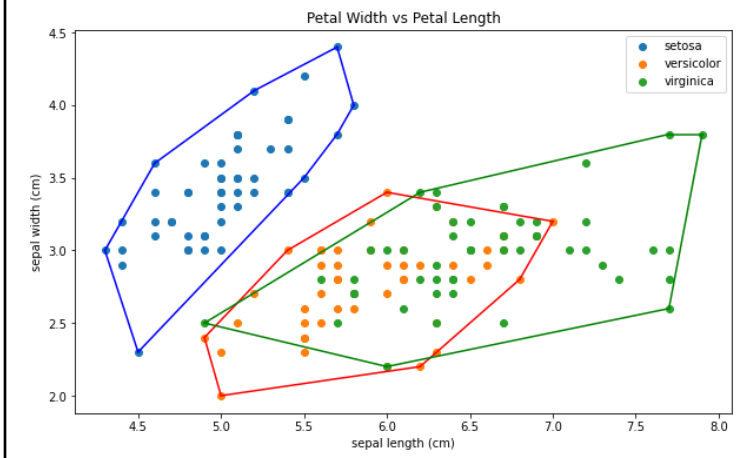
main.py > ...

```python
import matplotlib.pyplot as plt
from sklearn import datasets
from convexhull import myHull
import pandas as pd

x = True
while x:
    print("Berikut daftar dataset yang bisa digunakan: ")
    print("1. Iris")
    print("2. Breast Cancer")
    print("3. Wine")
    pilihan = int(input("Pilihan Dataset: "))

    if(pilihan >= 1 and pilihan<= 3):
        if(pilihan == 1):
            data = datasets.load_iris()
        elif(pilihan == 2):
            data = datasets.load_breast_cancer()
        elif(pilihan == 3):
            data = datasets.load_wine()

        df = pd.DataFrame(data.data, columns=data.feature_names)
        df['Target'] = pd.DataFrame(data.target)
        jumlahAttribut = len(data.feature_names)

        print(" ")
        print("Berikut daftar attribute: ")
        for i in range(jumlahAttribut):
            print(str(i+1) + "." +data.feature_names[i])
        x = int(input("Pilihan attribute x: "))
        y = int(input("Pilihan attribute y: "))

        if ((x>=1 and x<= jumlahAttribut) and (y>=1 and y <= jumlahAttribut) and (x != y)):
            plt.figure(figsize = (10, 6))
            label = len(df['Target'].unique())
            colors = ['b','r','g']

            plt.title(str(data.feature_names[x-1])+" vs "+str(data.feature_names[y-1]))
            plt.xlabel(data.feature_names[x-1])
            plt.ylabel(data.feature_names[y-1])

            for i in range(label):
                bucket = df[df['Target'] == i]
                bucket = bucket.iloc[:,[x-1,y-1]].values
                hull = myHull(bucket)   #implementasi convexhull
                plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i], color=colors[i])
                for simplex in hull:
                    plt.plot(bucket[simplex, 0], bucket[simplex, 1], color=colors[i])
            plt.legend()
            plt.show()
            plt.close('all')
        else:
            print("invalid attribute")
    else:
        print("Masukan Salah")

    endState = input("\nMau coba dataset yang lain? (Y/N): ")
    if endState.upper() != "Y":
        x = False

```

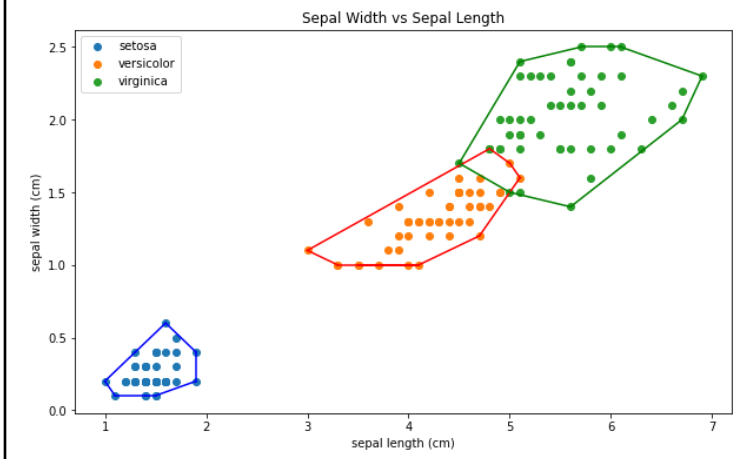
C. Screenshot

Dataset Iris:

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

data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
print(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 = myHull(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()
```

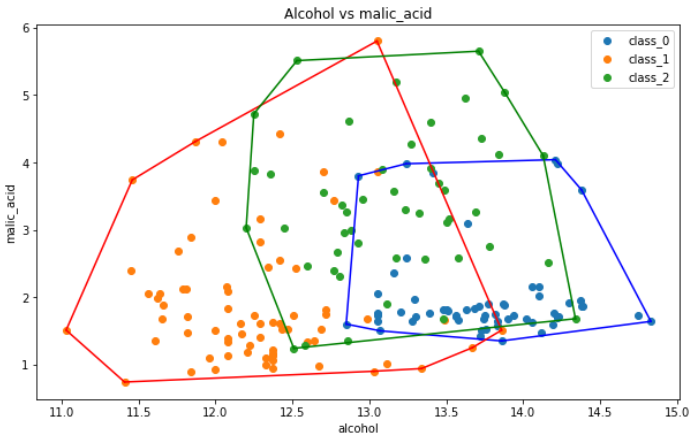



```
import matplotlib.pyplot as plt
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

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

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Sepal Width vs Sepal 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[:,[2,3]].values
    hull = myHull(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()
```

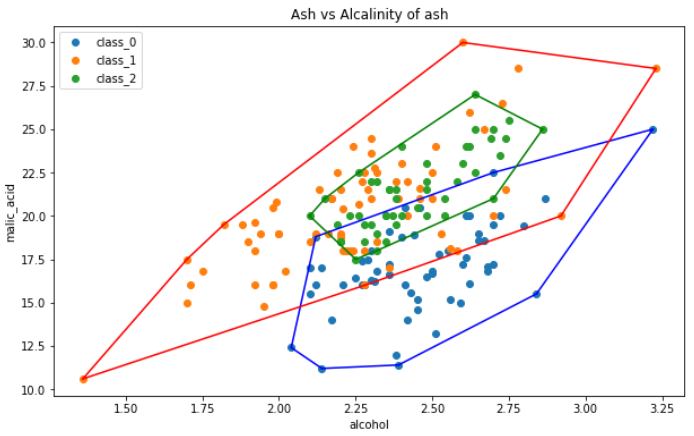


Dataset Wine:

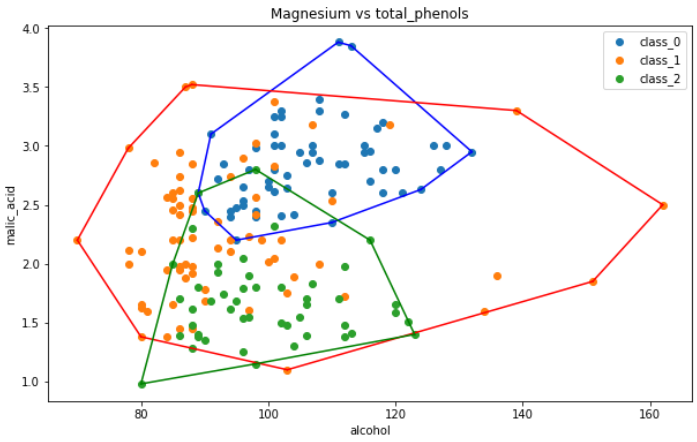
```
import matplotlib.pyplot as plt
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

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

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
    hull = myHull(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()
```



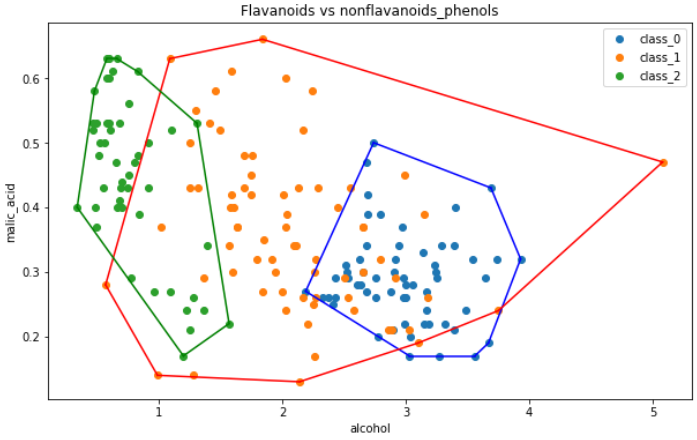

```
import matplotlib.pyplot as plt
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

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

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Ash vs Alcalinity of ash')
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[:,[2,3]].values
    hull = myHull(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()
```



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

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

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Magnesium vs total_phenols')
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[:,[4,5]].values
    hull = myHull(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()
```

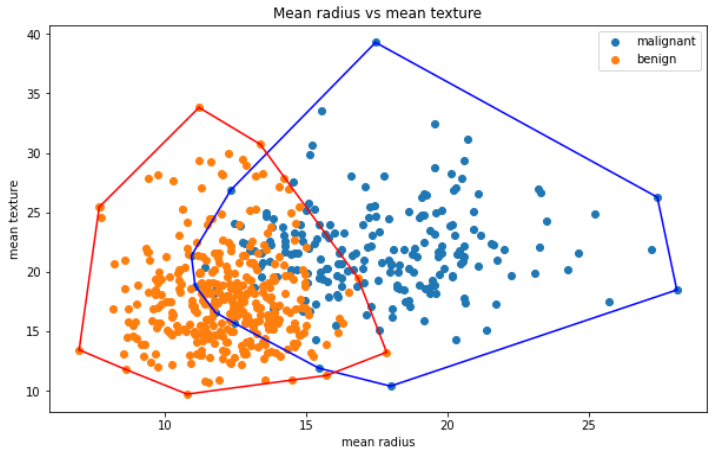



```
import matplotlib.pyplot as plt
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

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

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Flavanoids vs nonflavanoids_phenols')
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[:,[6,7]].values
    hull = myHull(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()
```

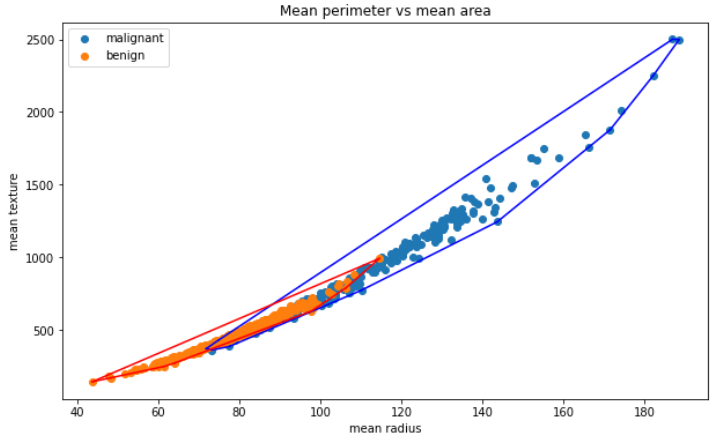


Dataset Cancerbreast:

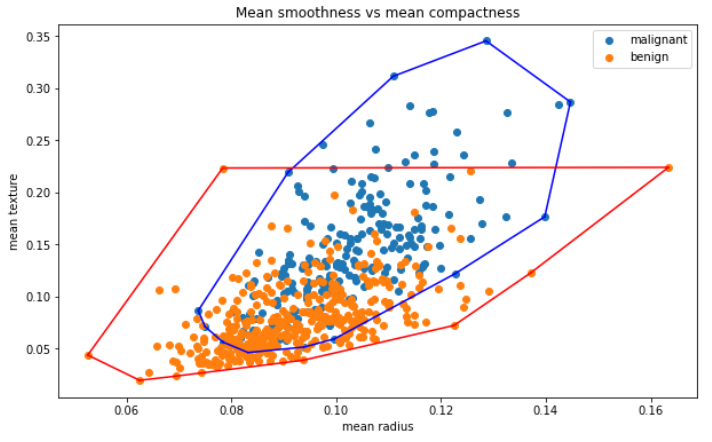
```
import matplotlib.pyplot as plt
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

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

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
    hull = myHull(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()
```



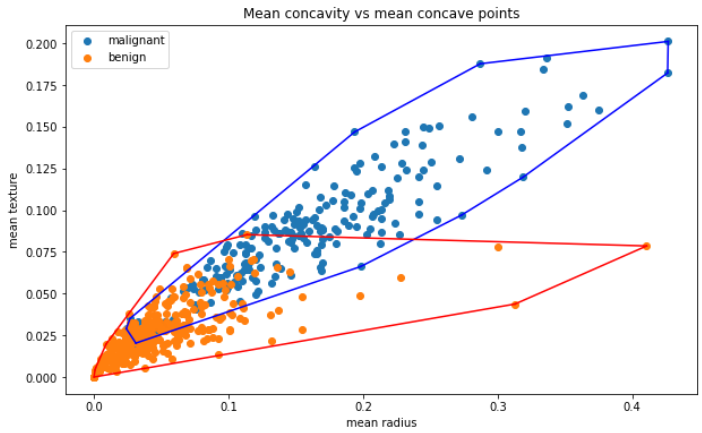

```
import matplotlib.pyplot as plt
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

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

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Mean perimeter vs mean area')
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[:,[2,3]].values
    hull = myHull(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()
```




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

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

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Mean smoothness vs mean compactness')
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[:,[4,5]].values
    hull = myHull(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()
```



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

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

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Mean concavity vs mean concave points')
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[:,[6,7]].values
    hull = myHull(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()
```



D. Alamat Drive dan Link Github
Link Drive :
Link Github: https://github.com/kibare/13520111_Tucil2

E. CHECKLIST

Point	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	✔	