IMPLEMENTASI CONVEX HULL UNTUK VISUALISASI TES *LINEAR SEPARABILITY DATASET DENGAN ALGORITMA DIVIDE AND CONQUER*

Fikri Ihsan Fadhiilah (13520148)

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jalan Ganesha 10 Bandung

13520148@std.stei.itb.ac.id

ALGORITMA *Divide and Conquer*

Divide artinya membagi persoalan menjadi beberapa up-masalah yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya berukuran hamper sama pada setiap upa masalah). *Conquer (solve)* artinya menyelesaikan masing-masing upa-masalah(secara langsung atau secara rekursif). *Combine* artinya menggabungkan solusi masing masin upa-masalah sehingga membentuk solusi perssoalan semula. Singkatnya, kita membagi permasalah menjadi beberapa sub-masalah lalu menyelesaikan sub-masalah tersebut dan pada akhirnya kita menggabungkan semua solusi dari setiap sub-masalah tersebut.

SOURCE PROGRAM

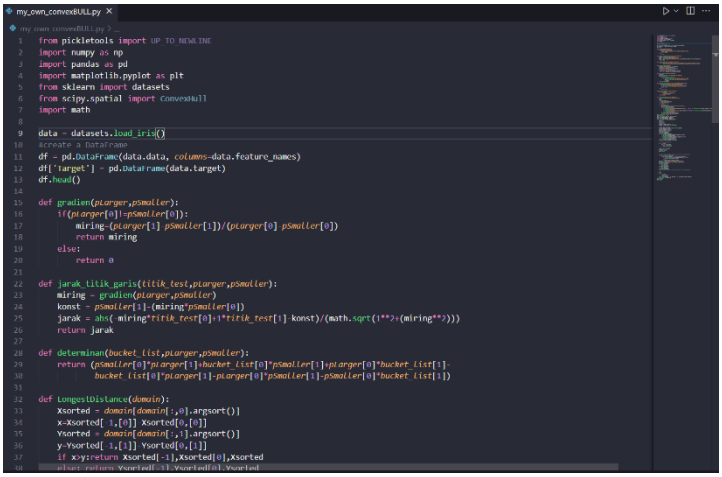
```
from pickletools import UP_TO_NEWLINE
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from scipy.spatial import ConvexHull
import math

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

def gradien(pLarger,pSmaller):
    if(pLarger[0]!=pSmaller[0]):
        miring=(pLarger[1]-pSmaller[1])/(pLarger[0]-pSmaller[0])
        return miring
    else:
```

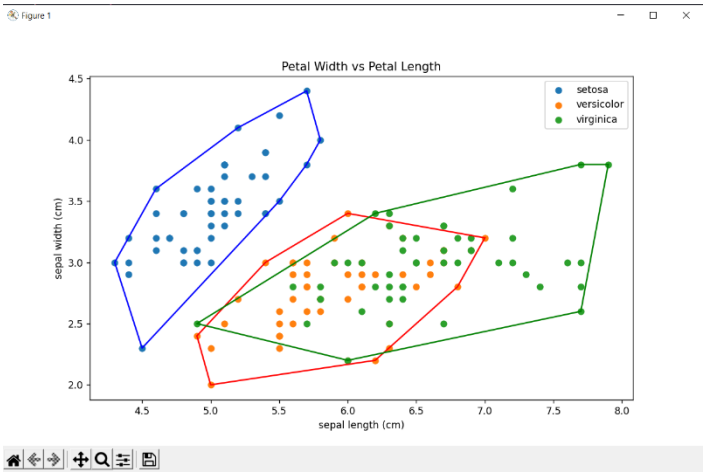
```
        return 0

def jarak_titik_garis(titik_test,pLarger,pSmaller):
    miring = gradien(pLarger,pSmaller)
    konst = pSmaller[1]-(miring*pSmaller[0])
    jarak = abs(-miring*titik_test[0]+1*titik_test[1]-
konst)/(math.sqrt(1**2+(miring**2)))
    return jarak

def determinan(bucket_list,pLarger,pSmaller):
    return
(pSmaller[0]*pLarger[1]+bucket_list[0]*pSmaller[1]+pLarger[0]*bucket_list[
1]-
            bucket_list[0]*pLarger[1]-pLarger[0]*pSmaller[1]-
pSmaller[0]*bucket_list[1])

def LongestDistance(domain):
    Xsorted = domain[domain[:,0].argsort()]
    x=Xsorted[-1,[0]]-Xsorted[0,[0]]
    Ysorted = domain[domain[:,1].argsort()]
    y=Ysorted[-1,[1]]-Ysorted[0,[1]]
    if x>y:return Xsorted[-1],Xsorted[0],Xsorted
    else: return Ysorted[-1],Ysorted[0],Ysorted

def titik_Jauh(area,pLarger,pSmaller):
    terjauh=0
    max=jarak_titik_garis(area[0],pLarger,pSmaller)
    if(len(area)>1):
        for i in range(1,len(area)):
            if jarak_titik_garis(area[i],pLarger,pSmaller) > max:
                max=jarak_titik_garis(area[i],pLarger,pSmaller)
                terjauh=i
    return area[terjauh]

def isInside_Triangle(titik_Jauh,pLarger,pSmaller,titik_test):
    if (determinan(titik_test,pLarger,pSmaller)>0.00
        and determinan(titik_test,titik_Jauh,pSmaller) <0.00
        and determinan(titik_test,pLarger,titik_Jauh)<0.00):
        return True
    else:
        return False

def deletePoint(bucket,pLarger,pSmaller):
    for i in bucket:
        if i==pLarger or i==pSmaller:
            bucket.remove(i)
    return bucket

def convexBull(bucket,pLarger,pSmaller):
    bucket=deletePoint(bucket,pLarger,pSmaller)
    s1=[]
    s2=[]
    if(len(bucket)==1):
        hull.append(bucket[0])
```

```
    elif(len(bucket)==0):
        return
    elif(len(bucket)>1):
        most_distance=titik_Jauh(bucket,pLarger,pSmaller)
        hull.append(most_distance)
        for i in range(len(bucket)):
            if determinan(bucket[i],most_distance,pSmaller) > 0.00 and not
isInside_Triangle(most_distance,pLarger,pSmaller,bucket[i]):
                s1.append(bucket[i])
            elif determinan(bucket[i],pLarger,most_distance) > 0.00 and
not isInside_Triangle(most_distance,pLarger,pSmaller,bucket[i]):
                s2.append(bucket[i])
        convexBull(s1,most_distance,pSmaller)
        convexBull(s2,pLarger,most_distance)
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)):
    hull=[]
    up_list=[]
    bot_list=[]
    hull_fix=[]
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values

    #Convex Hull Algorithm
    pLarger,pSmaller,bucket=LongestDistance(bucket)
    pLarger=pLarger.tolist()
    pSmaller=pSmaller.tolist()
    bucket=bucket.tolist()
    hull.append(pLarger)
    hull.append(pSmaller)
    for j in range(len(bucket)):
        if determinan(bucket[j],pLarger,pSmaller) > 0.00:
            up_list.append(bucket[j])
        elif determinan(bucket[j],pLarger,pSmaller) < 0.00:
            bot_list.append(bucket[j])
    convexBull(up_list,pLarger,pSmaller)
    convexBull(bot_list,pSmaller,pLarger)
    #Convex Hull Algorithm

    # avoid duplicates points
    for k in hull:
        if k not in hull_fix:
            hull_fix.append(k)
    # avoid duplicates points

    bucket=np.array(bucket)
    up_list.clear()
    bot_list.clear()
    hull.clear()
```

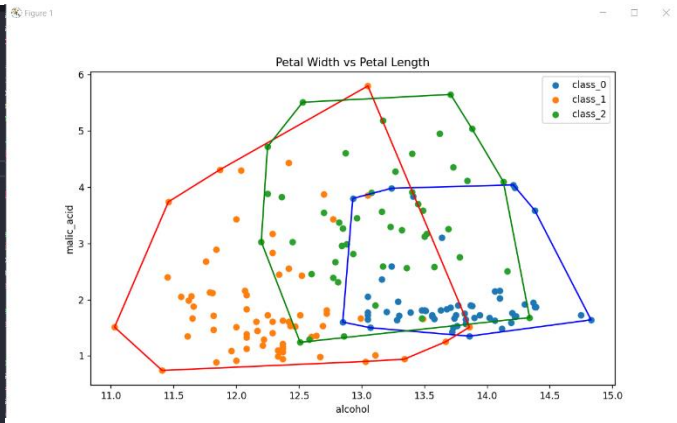
```
    #divide and sort the hull point to make the line correctly drawn
    for aa in range(len(hull_fix)):
        if hull_fix[aa]!=pLarger and hull_fix[aa]!=pSmaller:
            if determinan(hull_fix[aa],pLarger,pSmaller) > 0.00:
                up_list.append(hull_fix[aa])
            else:
                bot_list.append(hull_fix[aa])
    up_list.sort(key=lambda row:(row[1]),reverse=True)
    bot_list.sort(key=lambda row:(row[1]),reverse=False)
    hull.append(pLarger)
    for bb in up_list:
        hull.append(bb)
    hull.append(pSmaller)
    for bb in bot_list:
        hull.append(bb)
    hull.append(pLarger)
    #divide and sort the hull point to make the line correctly drawn

    a=[]
    b=[]
    for e in hull:
        a.append(e[0])
        b.append(e[1])
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    plt.plot(a,b, colors[i])
plt.legend()
plt.show()
```

Screenshoot Input-Output Program

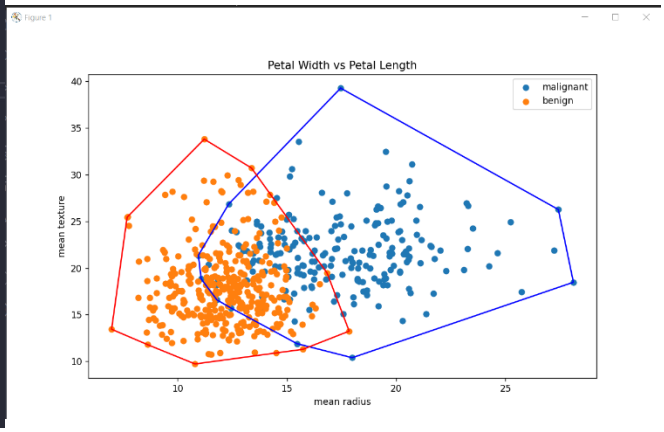




```python
from pickletools import UP_TO_NEWLINE
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from scipy.spatial import ConvexHull
import math

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

def gradien(pLarger,pSmaller):
    if(pLarger[0]!=pSmaller[0]):
        miring=(pLarger[1]-pSmaller[1])/(pLarger[0]-pSmaller[0])
        return miring
    else:
        return 0

def jarak_titik_garis(titik_test,pLarger,pSmaller):
    miring = gradien(pLarger,pSmaller)
    konst = pSmaller[1]-(miring*pSmaller[0])
    jarak = abs(-miring*titik_test[0]+1*titik_test[1]-konst)/(math.sqrt(1**2+(miring**2)))
    return jarak
```




```python
from pickletools import UP_TO_NEWLINE
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from scipy.spatial import ConvexHull
import math

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

def gradien(pLarger,pSmaller):
    if(pLarger[0]!=pSmaller[0]):
        miring=(pLarger[1]-pSmaller[1])/(pLarger[0]-pSmaller[0])
        return miring
    else:
        return 0

def jarak_titik_garis(titik_test,pLarger,pSmaller):
    miring = gradien(pLarger,pSmaller)
    konst = pSmaller[1]-(miring*pSmaller[0])
    jarak = abs(-miring*titik_test[0]+1*titik_test[1]-konst)/(math.sqrt(1**2+(miring**2)))
    return jarak

def determinan(bucket_list,pLarger,pSmaller):
    return (pSmaller[0]*pLarger[1]+bucket_list[0]*pSmaller[1]+pLarger[0]*bucket_list[1]-
            bucket_list[0]*pLarger[1]-pLarger[0]*pSmaller[1]-pSmaller[0]*bucket_list[1])
```



Poin	Ya	Tidak
Pustaka myConvexHull berhasil dibuat dan tidak ada kesalahan	v	
Convex hull yang dihasilkan sudah benar	v	
Pustaka myConvexHull dapat digunakan untuk menampilkan convex hull setiap label dengan warna yang berbeda.	v	
Bonus: program dapat menerima input dan menuliskan output untuk dataset lainnya.	v	

Link Program : https://github.com/Fikri-IF/STIMA-Tucil2