

**LAPORAN TUGAS KECIL 2**

**Implementasi Convex Hull untuk Visualisasi Tes Linear  
Separability Dataset  
dengan Algoritma Divide and Conquer**

Ditujukan untuk memenuhi salah satu tugas kecil mata kuliah IF2211 Strategi Algoritma (Stima)  
pada Semester II Tahun Akademik 2021/2022

Disusun oleh:

**Marcellus Michael Herman Kahari (K3) 13520057**



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

**2022**

## A. Algoritma *divide and conquer*

Algoritma *divide and conquer* dapat didefinisikan menjadi tiga bagian, yaitu *divisi*, *conquer*, dan *combine*. *Divide* berarti membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya berukuran hampir sama). *Conquer* berarti menyelesaikan masing-masing upa-persoalan (secara langsung jika sudah berukuran kecil atau secara *rekursif* jika masih berukuran besar). *Combine* berarti menggabungkan solusi masing-masing upa-persoalan sehingga membentuk solusi persoalan semula.

Tugas kecil Strategi Algoritma kedua ini membahas mengenai pembuatan *library* Convex Hull dengan menggunakan algoritma *divide and conquer*. Convex dapat disebut sebagai himpunan titik planar jika untuk sembarang dua titik pada bidang tersebut (misal p dan q), seluruh segmen garis yang berakhir di p dan q berada pada himpunan tersebut. Sementara Convex Hull dari suatu himpunan titik-titik S yang terdefinisi adalah kumpulan *convex* terkecil (*convex polygon*) yang mengandung S.

Cara kerja *convex hull* yang penulis buat adalah memanfaatkan skema *rekursif*. Pertama, algoritma akan masuk ke dalam fungsi `myConvexHull` yang menerima input data yang akan diolah dan menjadi sebuah *hull* yang merupakan array 2 dimensi yang berisi indeks-indeks data yang hendak dihubungkan. Data kemudian diubah menjadi *numpy array* dan ditambahkan indeks bantu pada kolom ketiganya. Kemudian, dengan menggunakan iterasi, dicari terlebih dahulu koordinat minimal dan maksimal yang jika dihubungkan dapat membagi titik-titik menjadi dua bagian. Iterasi untuk penentuan titik pertama cukup sederhana, yaitu menemukan titik dengan nilai x terkecil dan x terbesar. Kemudian, data, koordinat minimal dan maksimal, serta mark bertanda 0 dimasukkan ke fungsi *rekursif* `convexHullScratch`.

Dengan menggunakan bantuan fungsi *determinant*, titik-titik pada data dibagi menjadi dua yang ditempatkan pada array (**divide**), yaitu `aboveKoor` yang berisi titik-titik di atas garis dan `belowKoor` yang berisi titik-titik yang di bawah garis. Jika ternyata isi dari `aboveKoor` hanya 1 koordinat atau 0 koordinat, indeks titik tersebut dimasukkan ke dalam `result` dengan indeks minimal dan maksimal jika terdapat 0 koordinat dan indeks minimal dengan indeks 1 koordinat serta indeks 1 koordinat dengan indeks maksimal jika terdapat 1 koordinat (**conquer**). Begitu pula terjadi pada `belowKoor`. Untuk `aboveKoor`, program hanya akan melakukan pengecekan pada bagian atas garis. Sementara itu, untuk `belowKoor`, program hanya akan melakukan pengecekan pada bagian bawah garis. Hal ini didasarkan pada sifat dan cara kerja dari Convex Hull. Untuk menandai apakah itu `aboveKoor` atau `belowKoor`, digunakan mark jika `belowKoor` maka mark bernilai 1, jika `aboveKoor` mark bernilai 2. Penentuan titik baru untuk menjadi titik maksimal atau titik minimal berikutnya dicari menggunakan perhitungan dengan bantuan fungsi *sudut*. Dipilih suatu titik dengan sudut maksimal dengan maksimal sudut terletak pada titik minimal. Hasil dari setiap *rekursif* kemudian digabungkan dengan menggunakan fungsi `mergeHasil` (**combine**).

## B. Source Program

Source code program dituliskan dapat bahasa Python

```
# Nama : Marcellus Michael Herman Kahari
# NIM : 13520057
# Kelas : K03

# Set up awal
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
from sklearn import datasets

# Fungsi Bantu

# Fungsi getAngle digunakan untuk mendapatkan sudut yang dibentuk 3 titik
def getAngle(a, b, c):
    ang = math.degrees(math.atan2(c[1]-b[1], c[0]-b[0]) - math.atan2(a[1]-b[1], a[0]-b[0]))
    return ang + 360 if ang < 0 else ang

# Fungsi isEqual digunakan untuk membandingkan apakah dua buah koordinat memiliki nilai yang sama
def isEqual(koor1, koor2):
    return koor1[0] == koor2[0] and koor1[1] == koor2[1]

# Fungsi isExist digunakan untuk mengetahui apakah koordinat terdapat di dalam array
def isExist(koordinat, array):
    for value in array:
        if(koordinat[0] == value[0] and koordinat[1] == value[1]):
            return True
    return False

# Fungsi determinan digunakan untuk mengetahui apakah titik pada koor 3 terleta di atas garis yang dibentuk koor 1 dan koor 2 atau tidak
def determinan(koor1, koor2, koor3):
    return koor1[0]*koor2[1] + koor3[0] * koor1[1] + koor2[0] * koor3[1] - koor3[0] * koor2[1] - koor2[0] * koor1[1] - koor1[0] * koor3[1]

# Fungsi merge
def mergeHasil(result, array):
    for koordinat in array:
        if(not isExist(koordinat, result)):
```

```

        result = np.concatenate((result,[koordinat]), axis=0)
    return result

# Fungsi pisah array
def pisahArray(getMin, getMax, bucket):
    aboveKoor = np.empty((0,3), float)
    belowKoor = np.empty((0,3), float)
    for koordinat in bucket:
        if(determinan(getMin, getMax, koordinat) > 0.0 and not isEqual(getMin,
koordinat) and not isEqual(getMax, koordinat)):
            aboveKoor = np.concatenate((aboveKoor,[koordinat]), axis=0)
        elif (determinan(getMin, getMax, koordinat) < 0.0 and not isEqual(getMin,
koordinat) and not isEqual(getMax, koordinat)):
            belowKoor = np.concatenate((belowKoor,[koordinat]), axis=0)
    return aboveKoor, belowKoor

# Fungsi untuk mempush hasil ke dalam result
def pushResult(getMin, getMax, result, array):
    indexmin = int(getMin[2])
    indexmax = int(getMax[2])
    indextemp = int(array[0][2])
    result = np.concatenate((result, [[indexmin, indextemp]]), axis=0)
    result = np.concatenate((result, [[indextemp, indexmax]]), axis=0)
    return result

# Fungsi Rekursif

# Fungsi convexHullScratch digunakan sebagai fungsi rekursif
def convexHullScratch(bucket, getMin, getMax, mark):
    result = np.empty((0,2), int)
    aboveKoor, belowKoor = pisahArray(getMin, getMax, bucket)

    if((len(aboveKoor) == 0 and (mark == 2 or mark == 0)) or (len(belowKoor) == 0 and
(mark == 1 or mark == 0))):
        result = np.concatenate((result, [[int(getMin[2]), int(getMax[2])]]), axis=0)
    if(len(aboveKoor) == 1 and (mark == 2 or mark == 0)):
        result = pushResult(getMin, getMax, result, aboveKoor)
    if(len(belowKoor) == 1 and (mark == 1 or mark == 0)):
        result = pushResult(getMin, getMax, result, belowKoor)

    if(len(belowKoor) > 0 and (mark == 0 or mark == 1)):
        temp = belowKoor[0]
        hasil = 0
        for koordinat in belowKoor:
            if(hasil < getAngle(koordinat, getMin, getMax)):

```

```

        hasil = getAngle(koordinat, getMin, getMax)
        temp = koordinat
    result1 = convexHullScratch(belowKoor, getMin, temp, 1)
    result2 = convexHullScratch(belowKoor, temp, getMax, 1)
    result = mergeHasil(result, result1)
    result = mergeHasil(result, result2)

if(len(aboveKoor) > 0 and (mark == 0 or mark == 2)):
    temp = aboveKoor[0]
    hasil = 0
    for koordinat in aboveKoor:
        if(hasil < getAngle(getMax, getMin, koordinat)):
            hasil = getAngle(getMax, getMin, koordinat)
            temp = koordinat
    result1 = convexHullScratch(aboveKoor, getMin, temp, 2)
    result2 = convexHullScratch(aboveKoor, temp, getMax, 2)
    result = mergeHasil(result, result1)
    result = mergeHasil(result, result2)

return result

# Fungsi Utama

# Fungsi convexHull digunakan sebagai fungsi inisialisasi awal
def myConvexHull(data):
    bucket = np.array(data)
    index = np.array([[i] for i in range(len(bucket))])
    bucket = np.append(bucket, index, axis=1)
    getMax = bucket[0]
    getMin = bucket[0]
    for koordinat in bucket:
        if(getMax[0] < koordinat[0]):
            getMax = koordinat
        if(getMin[0] > koordinat[0]):
            getMin = koordinat
        if(getMin[0] == koordinat[0] and getMin[1] > koordinat[1]):
            getMin = koordinat
        if(getMax[0] == koordinat[0] and getMax[1] < koordinat[1]):
            getMax = koordinat
    hull = convexHullScratch(bucket, getMin, getMax, 0)
    return hull

```

```

# 1. Data set Iris Sepal Width Vs Sepal Length

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

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[:,[0,1]].values
    hull = 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()

# 2. Data set Iris Petal Width Vs Petal Length

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

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(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. Data set Wine Alcohol Vs Malic Acid

data = datasets.load_wine()

```

```

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

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 = 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()

# 4. Data set Wine Magnesium Vs Total Phenols

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

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Magnesium vs Total Phenols')
plt.xlabel(data.feature_names[4])
plt.ylabel(data.feature_names[5])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[4,5]].values
    hull = 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()

```

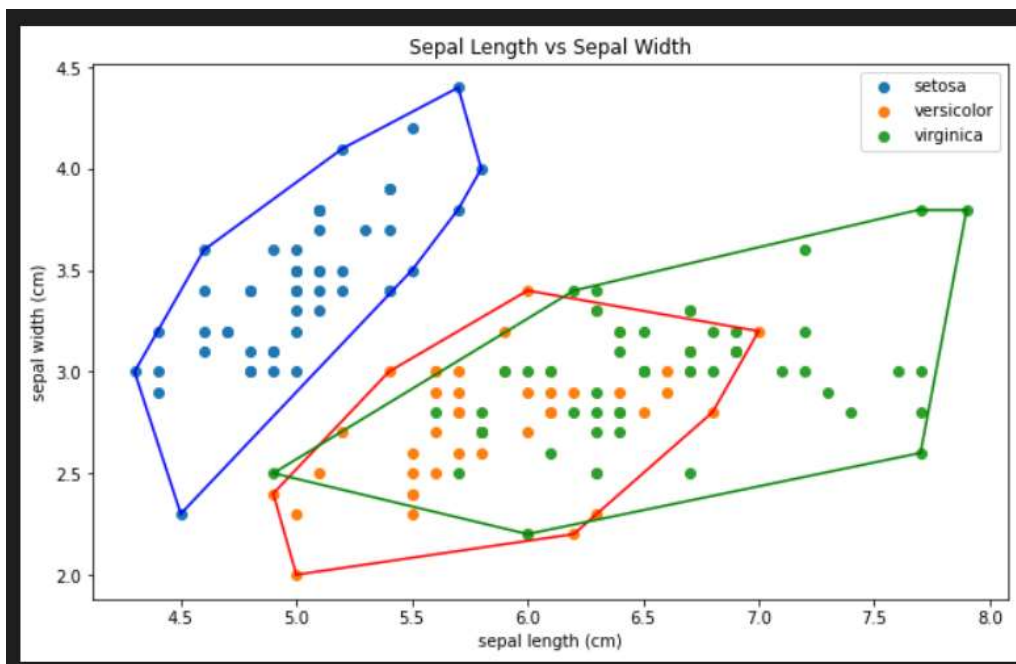
### C. Screenshots *input* dan *output*

#### 1. Iris Sepal Length Vs Sepal Width

Input

sepal length (cm)	sepal width (cm)
5.1	3.5
4.9	3.0
4.7	3.2
4.6	3.1
5.0	3.6

Output



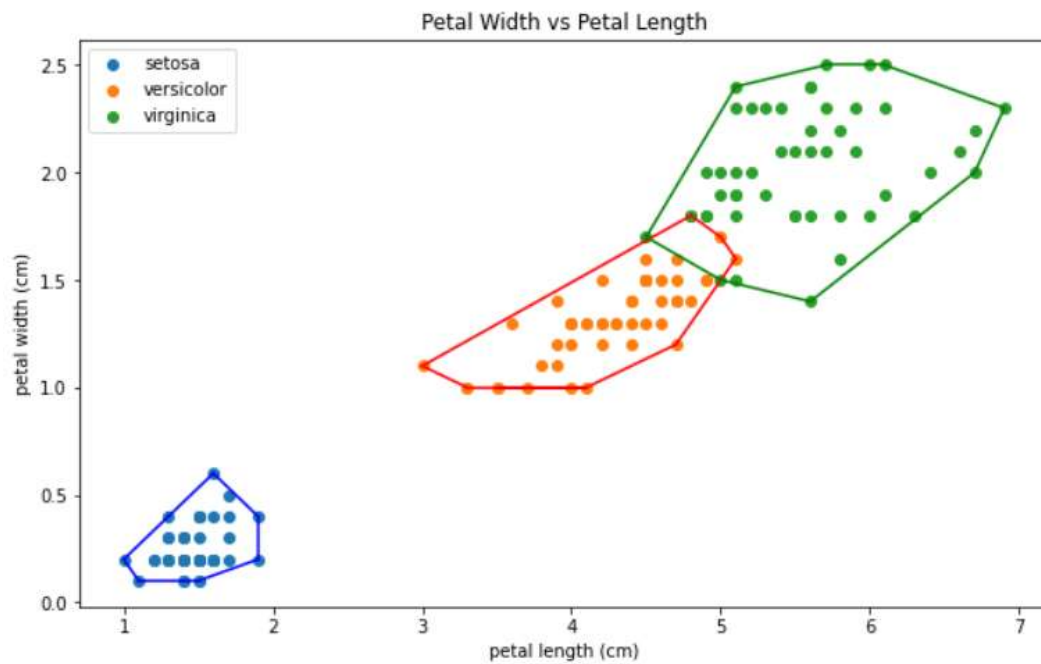


## 2. Iris Petal Length Vs Petal Width

Input

petal length (cm)	petal width (cm)
1.4	0.2
1.4	0.2
1.3	0.2
1.5	0.2
1.4	0.2

Output

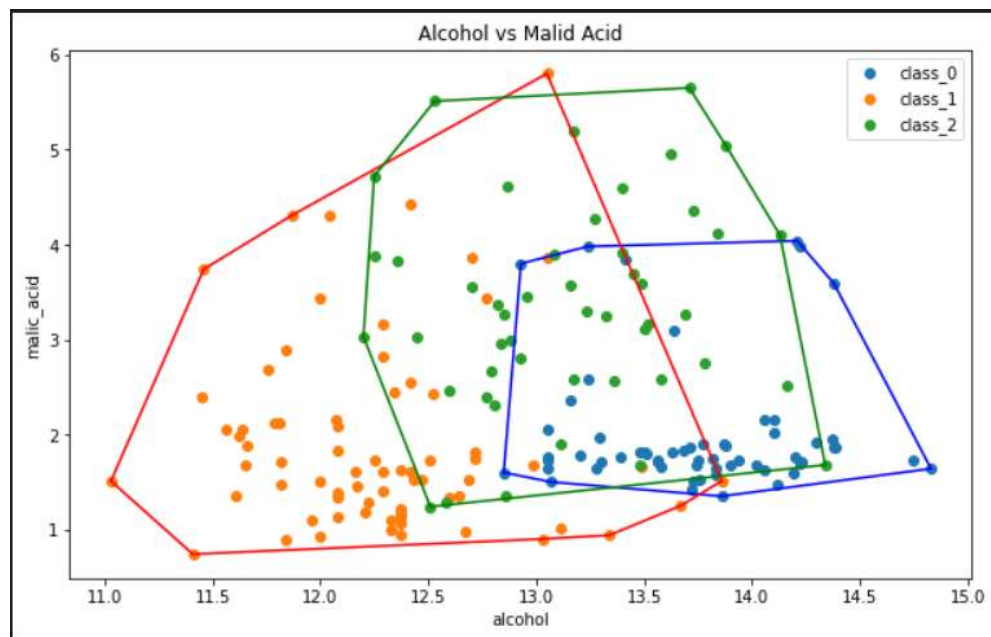


### 3. Wine Alcohol Vs Malid Acid

Input

alcohol	malic_acid
14.23	1.71
13.20	1.78
13.16	2.36
14.37	1.95
13.24	2.59

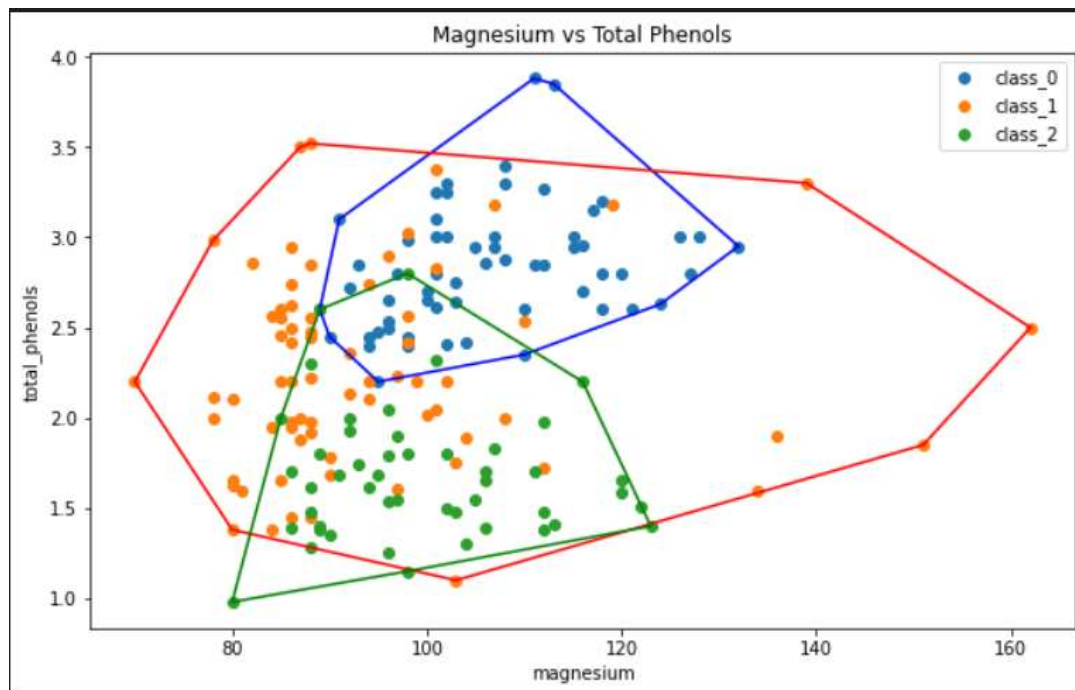
Output



#### 4. Wine Magnesium Vs Total Phenols Input

magnesium	total_phenols
127.0	2.80
100.0	2.65
101.0	2.80
113.0	3.85
118.0	2.80

#### Output



#### D. Alamat Drive

<https://drive.google.com/drive/folders/1vU3RZqWG7wck6gr0oSUXSAWbE87ys1ZR?usp=sharing>

#### E. Alamat Github

<https://github.com/pandora-1/ConvexHull-Python.git>

#### F. Tabel Checklist

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. <b>Bonus:</b> program dapat menerima input dan menuliskan output untuk dataset lainnya.	√	