LAPORAN TUGAS KECIL II

IF2211 STRATEGI ALGORITMA

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

Gede Prasidha Bhawarnawa

13520004

K01

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

I. Algoritma *Divide and Conquer* yang Digunakan Untuk Membentuk *Convex Hull* secara Garis Besar

Convex Hull adalah sebuah bentuk convex terkecil yang memuat seluruh data di dalamnya. Bentuk *convex* sendiri dapat diartikan sebagai sebuah bentuk yang untuk setiap sembarang dua titik di dalamnya, dapat ditarik garis lurus antara dua titik tersebut dan setiap segmen garis tersebut berada di dalam bentuk tersebut.

Terdapat banyak manfaat dari penggunaan *convex hull* ini, khususnya di bidang *data science*. Salah satu penggunaannya adalah dalam pengujian *linear separability dataset*. *Linear separability dataset* adalah pengujian untuk mengetahui apakah dua *dataset* terpisah atau independen terhadap satu sama lain. Pengujian ini dilakukan dengan cara memetakan kedua *dataset* ke dalam suatu bidang planar berdimensi dua dan melihat apakah bisa ditarik sebuah garis linear yang memisahkan kedua *dataset* tersebut. Bila bisa dipisahkan, maka kedua *dataset* terpisah secara linear. Dengan menggunakan *convex hull*, maka cukup dilihat apakah sisi dari kedua *convex hull dataset* saling tumpang-tindih (*overlapping*) atau tidak. Bila iya, maka kedua *dataset* tidak linear terpisah.

Convex Hull pada program yang didesain penulis dibuat dengan mengimplementasikan algoritma *divide and conquer*. Algoritma *divide and conquer* adalah algoritma yang membagi sebuah persoalan menjadi beberapa sub-persoalan yang lebih kecil sampai mencapai sebuah ukuran tertentu yang mudah diselesaikan dan menggabungkan hasil dari semua upa-persoalan yang sudah digabungkan.

Berikut adalah alur logika dan cara kerja dari program pembuat *convex hull* yang digunakan oleh penulis:

1. Diterima input *dataset* berisikan data-data yang dapat dipetakan ke dalam sebuah bidang planar berdimensi dua (absis dan ordinat). Data-data tersebut disimpan sebagai sebuah *list of Coordinates* dengan *Coordinate* adalah sebuah *abstract data type* berisikan absis dan ordinat.
2. Mengurutkan semua data (titik) pada *list of Coordinates* berdasarkan kenaikan absisnya. Bila ditemukan dua atau lebih titik dengan nilai absis yang sama, titik-titik diurutkan berdasarkan kenaikan ordinatnya.
3. Menyimpan titik pada *list of Coordinates* dengan indeks terkecil (0) dan indeks terbesar (-1). Kedua titik ini, titik minimum dan maksimum, akan menjadi pembangun utama dari *convex hull*.

4. Membagi isi dari *list of Coordinates* ke dalam dua bagian, atas dan bawah. Pembagian menggunakan nilai determinan antara titik minimum dan maksimum dan titik uji dari *list of Coordinates*. Bila nilai determinannya lebih besar dari nol, titik uji akan dimasukkan ke dalam golongan atas. Bila lebih kecil dari nol, titik uji akan dimasukkan ke dalam golongan bawah. Bila sama dengan nol, diabaikan karena berada tepat pada garis.
5. Untuk bagian atas, cari titik pada golongan atas *list of Coordinates* yang merupakan titik terjauh dari titik minimum dan maksimum, simpan sebagai titik *temp*. Setelah ditemukan, cari tau apabila ada titik terjauh antara titik maksimum dengan titik *temp,* dengan mencari titik determinan dengan nilai positif. Bila ada, maka ulangi langkah 5 dengan titik *temp* sebagai titik maksimum terbaru. Bila tidak ada, maka gabungkan list titik minimum dengan titik *temp* dan rekursi kembali dengan gabungan list sebagai list awal dan titik maksimum global yang didapatkan pada langkah 3 sebagai titik maksimum.
6. Ulangi langkah 5 sampai dengan tidak ditemukan lagi titik terjauh antara titik *temp* dengan titik maksimum.
7. Ulangi langkah 5 dan 6 untuk bagian bawah. Perbedaannya adalah titik terjauh dicari dengan menggunakan nilai determinan negatif.
8. Gabungkan list bagian atas dan bagian bawah dan di-*plot* sebagai sebuah grafik. Grafik yang ditampilkan adalah grafik *convex hull* untuk *dataset* yang bersangkutan.

II. Kode Program (*Source Code*)

Fungsi Dasar untuk membuat Convex Hull

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

Gede Prasidha Bhawarnawa - 13520004 - K01

```
### Library yang Digunakan untuk Membuat Convex Hull
**Jalankan dua balok kode di bawah sebelum menjalankan balok kode lainnya!**
```

Markdown

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

[73] ✓ 0.9s Python

```
# Pembuatan class Coordinates untuk menyimpan nilai X dan Y suatu titik
class Coordinates:
    def __init__(self, x, y):
        self.x = x
        self.y = y
```

```python
# Pembuatan class Coordinates untuk menyimpan nilai X dan Y suatu titik
class Coordinates:
    def __init__(self, x, y):
        self.x = x
        self.y = y

# Mengurutkan semua koordinat berdasarkan kenaikan absisnya. Bila ada dua titik dengan absis yang sama, diurutkan berdasarkan kenaikan absisnya.
# sorting dilakukan dengan menggunakan selection sort (diimplementasikan dengan dasar algoritma Bubble Sort)
def sortList(CoordinatesList):
    # snippet code ini berfungsi untuk mengurutkan semua CoordinatesList berdasarkan kenaikan absisnya
    # terdapat kemungkinan urutan ordinat belum tepat
    for i in range(0, len(CoordinatesList)):
        toBeSwitchedIndex = i
        for j in range(i+1, len(CoordinatesList)):
            if CoordinatesList[j].x < CoordinatesList[toBeSwitchedIndex].x:
                toBeSwitchedIndex = j
        CoordinatesList[i], CoordinatesList[toBeSwitchedIndex] = CoordinatesList[toBeSwitchedIndex], CoordinatesList[i]
    # snippet ccode ini berfungsi untuk mengurutkan CoordinatesList yang sudah terurut berdasarkan kenaikan nilai absisnya
    #   sesuai dengan kenaikan nilai ordinatnya bila ditemukan dua titik dengan absis yang sama
    #   loop diulangi sampai tidak ada proses swap
    isNotSorted = True
    while isNotSorted:
        swapCount = 0
        for i in range(0, len(CoordinatesList) - 1):
            if ((CoordinatesList[i].x == CoordinatesList[i+1].x) and (CoordinatesList[i].y > CoordinatesList[i + 1].y)):
                swapCount += 1
                CoordinatesList[i], CoordinatesList[i + 1] = CoordinatesList[i + 1], CoordinatesList[i]
        if swapCount == 0:
            isNotSorted = False
    return CoordinatesList
```

```python
# Membagi semua titik yang tersisa menjadi dua bagian, LeftCoordinatesList dan RightCoordinatesList, berdasarkan nilai determinannya
# Determinan > 0 : Titik berada pada atas/kiri pembatas, masuk ke dalam list LeftCoordinatesList
# Determinan = 0 : Titik berada pada pembatas, diabaikan
# Determinan < 0 : Titik berada pada bawah/kanan pembatas, masuk ke dalam list RightCoordinatesList
def splitByDeterminant(start, end, CoordinatesList, LeftCoordinatesList, RightCoordinatesList):
    while (len(CoordinatesList) > 0):
        var = CoordinatesList[0]
        value = determinant(start, var, end)
        if value < 0:
            CoordinatesList.pop(0)
            RightCoordinatesList.append(var)
        elif value > 0:
            CoordinatesList.pop(0)
            LeftCoordinatesList.append(var)
        else:
            CoordinatesList.pop(0)
    return LeftCoordinatesList, RightCoordinatesList

# Mengembalikan nilai determinan untuk menentukan posisi sebuah titik di atas, di bawah, atau pada garis start-end
# Terdapat kasus error handling bila menerima input yang None pada start, var, atau end
def determinant(start, var, end):
    if start is None or var is None or end is None:
        return 0
    else:
        value = start.x * end.y + var.x * start.y + end.x * var.y - var.x * end.y - end.x * start.y - start.x * var.y
        return value

# Menghitung sudut yang dibentuk oleh 3 titik dengan var sebagai titik acuan perhitungan
# Terdapat error handling bila ditemukan hasil dotProduct di luar range [-1, 1], yaitu 90 derajat karena cos(90) = Nan
def measureAngle(start, var, end):
    startToVar = [var.x - start.x, var.y - start.y]
    varToEnd = [end.x - var.x, end.y - var.y]
    dotProduct = np.dot(startToVar, varToEnd)
    if not(abs(dotProduct) > 1):
        tempAngle = np.arccos(dotProduct)
    else:
        tempAngle = 90
    return tempAngle

```

```python
# Menghitung jumlah titik yang berada di luar garis antara titik start dan end (mengembalikan integer).
# Digunakan sebagai uji syarat rekursif
def isThereOuterConvexEdge(start, end, CoordinatesList, upperConvexHull):
    count = 0
    for var in CoordinatesList:
        if var is not None:
            value = determinant(start, var, end)
            if (upperConvexHull):
                if (value > 0):
                    count += 1
            else:
                if (value < 0):
                    count += 1
    return count

# Untuk menguji sebuah titik berada di dalam atau di luar garis (mengembalikan boolean)
def isOutsideInsideLine(start, var, end, isOutside):
    if isOutside:
        return (determinant(start, var, end) > 0)
    else:
        return (determinant(start, var, end) < 0)
```

```python
# Mencari list Coordinates yang membangun Convex Hull bagian atas dan bawah
# Dasar Rekursi (Base Case) : Tidak ada titik terjauh dan terluar (ujung Convex Hull) diantara titik awal dan akhir. Mengembalikan set list Coordinates (awal + akhir)
# Fungsi Rekursi : Mencari titik terjauh dan terluar dan menjadikannya sebagai titik akhir baru saat memanggil fungsi rekursi lagi
def recursiveConvexHull(origin, startList, tempList, endList, isUpper):
    countFinish = isThereOuterConvexEdge(startList[-1], endList[0], origin, isUpper)
    # Fungsi rekursi
    if countFinish != 0:
        maxDistanceFromStartEnd = -1
        maxAngleFromStartEnd = -1
        maxPoint = None
        # s = start, e = end, m = gradien, c = constant
        # membuat garis dengan rumus y = mx + c
        s = startList[-1]
        if (len(tempList) == 0):
            e = endList[0]
        else:
            e = tempList[0]
        m = (e.y - s.y) / (e.x - s.x)
        c = s.y - m * s.x
        for var in origin:
            if (var not in startList) and (var not in endList) and (var not in tempList) and (isOutsideInsideLine(s, var, e, isUpper)):
                tempDistanceFromStartEnd = abs((m * var.x - var.y + c) / math.sqrt(m ** 2 + 1))
                if (tempDistanceFromStartEnd > maxDistanceFromStartEnd):
                    tempAngle = measureAngle(s,var,e)
                    maxDistanceFromStartEnd = tempDistanceFromStartEnd
                    maxAngleFromStartEnd = tempAngle
                    maxPoint = var
                elif (tempDistanceFromStartEnd == maxDistanceFromStartEnd):
                    tempAngle = measureAngle(s,var,e)
                    if (tempAngle > maxAngleFromStartEnd):
                        maxAngleFromStartEnd = tempAngle
                        maxPoint = var
        tempList.insert(0,maxPoint)
```

```
        # untuk mengecek apakah ada titik di antara titik maks sebelumnya dengan titik awal yang merupakan convex edge
        count = 999
        if maxPoint is not None:
            count = isThereOuterConvexEdge(s, maxPoint, origin, isUpper)
        # Error Handling untuk kasus elemen maxPoint adalah None
        if count == 0 or maxPoint is None:
            newStartList = startList + tempList
            return recursiveConvexHull(origin, newStartList, [], endList, isUpper)
        else:
            return recursiveConvexHull(origin, startList, [maxPoint], endList, isUpper)
    # Basis rekursi
    else:
        return startList + endList
```

```
# Mengubah CoordinatesList menjadi sebuah 2D list untuk diplot oleh matplotlib
def createListForPlot(CoordinatesList):
    listX = []
    listY = []
    for i in range(len(CoordinatesList)):
        if CoordinatesList[i] != None:
            listX.append(CoordinatesList[i].x)
            listY.append(CoordinatesList[i].y)
    listCombine = [listX, listY]
    return listCombine

# Fungsi utama (main)
def myConvexHull(bucket):
    CoordinatesList = []
    LeftCoordinatesList = []
    RightCoordinatesList = []
    ConvexEdgesList = []
    for i in range(0, len(bucket)):
        CoordinatesList.append(Coordinates(bucket[i,0], bucket[i,1]))
    CoordinatesList = sortList(CoordinatesList)
    # Menyimpan titik p1(x1,y1)
    startEdge = CoordinatesList[0]
    # Menyimpan titik pn(xn,yn)
    endEdge = CoordinatesList[-1]
    # Bagi sisa titik berdasarkan nilai determinan
    copyCoordinatesList = CoordinatesList[:]
    LeftCoordinatesList, RightCoordinatesList = splitByDeterminant(startEdge, endEdge, copyCoordinatesList[1:-1], LeftCoordinatesList, RightCoordinatesList)
    LeftCoordinatesList = sortList(LeftCoordinatesList)
    RightCoordinatesList = sortList(RightCoordinatesList)
    upLeftConvexEdges = recursiveConvexHull(LeftCoordinatesList, [startEdge], [], [endEdge], True)
    downRightConvexEdges = recursiveConvexHull(RightCoordinatesList, [startEdge], [], [endEdge], False)
    return upLeftConvexEdges, downRightConvexEdges, CoordinatesList

```

Kode untuk menampilkan sklearn.datasets.load_iris()

```
title1 = "1. Sepal Length vs Sepal Width"
title2 = "2. Petal Length vs Petal Width"
chosenTitle = ""
column1 = 0
column2 = 1
print("Jenis Convex Hull yang ingin dihasilkan dari sklearn.datasets.load_iris() menggunakan myConvexHull: ")
print(title1)
print(title2)
isValidated = False
while not isValidated:
    query = int(input("Masukkan angka sesuai jenis Convex Hull yang ingin dihasilkan : "))
    if query == 1:
        chosenTitle = title1
        column1 = 0
        column2 = 1
        isValidated = True
    elif query == 2:
        chosenTitle = title2
        column1 = 2
        column2 = 3
        isValidated = True

fig, ax = plt.subplots(figsize = (10, 6))
colors = ['r', 'g', 'b']

ax.set_title(chosenTitle)

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

ax.set_xlabel(data.feature_names[column1])
ax.set_ylabel(data.feature_names[column2])

for i in range(len(data.target_names)):
    legend = "target = " + str(i)
    bucket = df[df['Target'] == i].iloc[:, [column1, column2]].values
    up, down, all = myConvexHull(bucket)
    hullDown = createListForPlot(down)
    ax.scatter(hullDown[0], hullDown[1], color = colors[i])
    ax.plot(hullDown[0], hullDown[1], color = colors[i])
    hullUp = createListForPlot(up)
    ax.scatter(hullUp[0], hullUp[1], color = colors[i])
    ax.plot(hullUp[0], hullUp[1], color = colors[i])
    listAll = createListForPlot(all)
    ax.scatter(listAll[0], listAll[1], color = colors[i], label = legend)
    ax.legend()
plt.show()
```

Kode untuk menampilkan sklearn.datasets.load_wine()

```
title1 = "1. Alcohol vs Malic Acid"
title2 = "2. Ash vs Alcalinity of Ash"
title3 = "3. Flavanoids vs Nonflavanoids Phenols"
title4 = "4. Color Intensity vs Hue"
chosenTitle = ""
column1 = 0
column2 = 1
print("Jenis Convex Hull yang ingin dihasilkan dari sklearn.datasets.load_iris() menggunakan myConvexHull: ")
print(title1)
print(title2)
print(title3)
print(title4)
isValidated = False
while not isValidated:
    query = int(input("Masukkan angka sesuai jenis Convex Hull yang ingin dihasilkan : "))
    if query == 1:
        chosenTitle = title1
        column1 = 0
        column2 = 1
        isValidated = True
    elif query == 2:
        chosenTitle = title2
        column1 = 2
        column2 = 3
        isValidated = True
    elif query == 3:
        chosenTitle = title3
        column1 = 6
        column2 = 7
        isValidated = True
    elif query == 4:
        chosenTitle = title4
        column1 = 9
        column2 = 10
        isValidated = True
```

```
fig, ax = plt.subplots(figsize = (10, 6))
colors = ['r', 'g', 'b', 'c', 'm', 'y', 'k']

ax.set_title(chosenTitle)

data = datasets.load_wine(as_frame=True)
df = pd.DataFrame(data.data, columns = data.feature_names)
df['Target'] = pd.DataFrame(data.target)

ax.set_xlabel(data.feature_names[column1])
ax.set_ylabel(data.feature_names[column2])

for i in range(len(data.target_names)):
    legend = "target = " + str(i)
    bucket = df[df['Target'] == i].iloc[:, [column1, column2]].values
    up, down, all = myConvexHull(bucket)
    hullDown = createListForPlot(down)
    ax.scatter(hullDown[0], hullDown[1], color = colors[i])
    ax.plot(hullDown[0], hullDown[1], color = colors[i])
    hullUp = createListForPlot(up)
    ax.scatter(hullUp[0], hullUp[1], color = colors[i])
    ax.plot(hullUp[0], hullUp[1], color = colors[i])
    listAll = createListForPlot(all)
    ax.scatter(listAll[0], listAll[1], color = colors[i], label = legend)
    ax.legend()
plt.show()
```

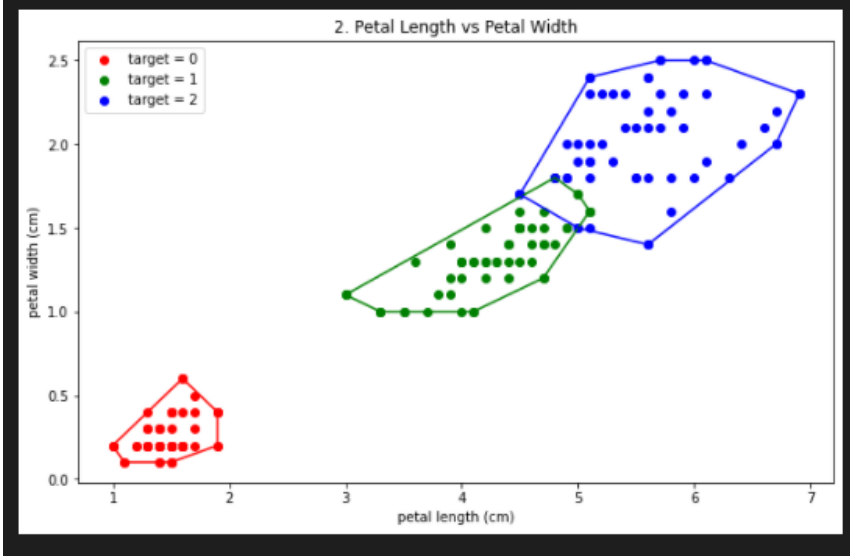
III. *Screenshot Input/Output* Program untuk Setiap Dataset yang Diujikan

Dataset sklearn.datasets.load_iris()

Input/Output Petal length vs Petal width

```
2
Masukkan angka sesuai jenis Convex Hull yang ingin dihasilkan : (Press 'Enter' to confirm or 'Escape' to cancel)
```

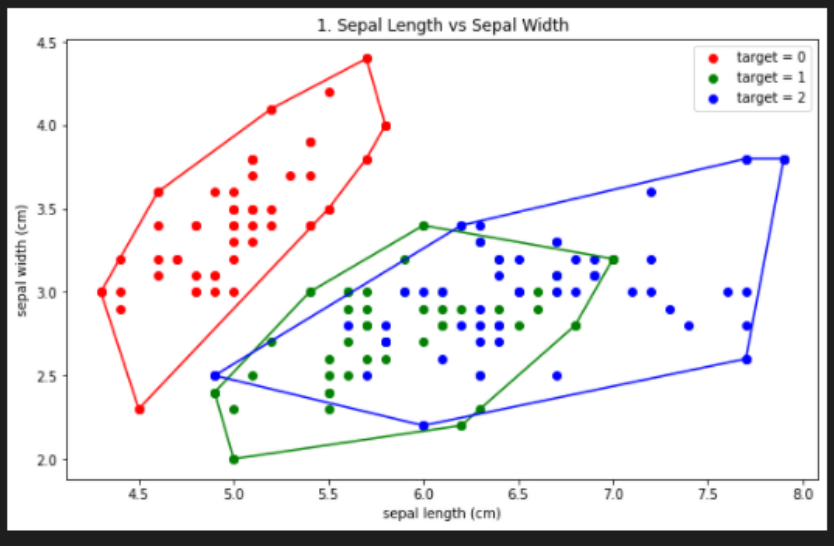
```
Jenis Convex Hull yang ingin dihasilkan dari sklearn.datasets.load_iris() menggunakan myConvexHull:
1. Sepal Length vs Sepal Width
2. Petal Length vs Petal Width
```



Input/Output Sepal length vs Sepal width

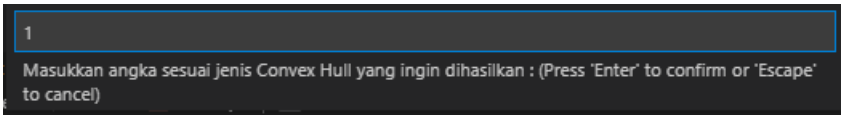
```
1
Masukkan angka sesuai jenis Convex Hull yang ingin dihasilkan : (Press 'Enter' to confirm or 'Escape' to cancel)
```

```
Jenis Convex Hull yang ingin dihasilkan dari sklearn.datasets.load_iris() menggunakan myConvexHull:
1. Sepal Length vs Sepal Width
2. Petal Length vs Petal Width
```

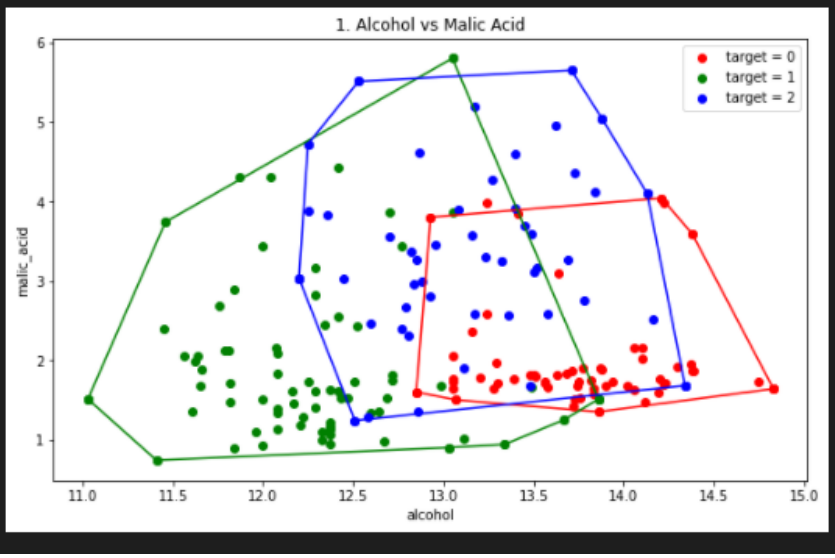


Dataset sklearn.datasets.load_wine()

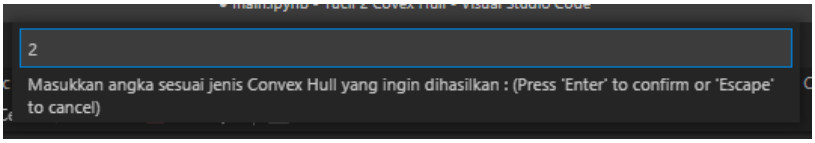
Input/Output Alcohol vs Malic Acid



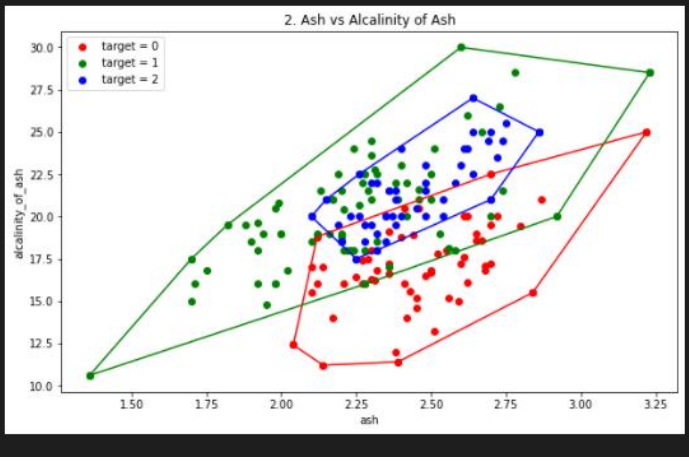

```
Jenis Convex Hull yang ingin dihasilkan dari sklearn.datasets.load_iris() menggunakan myConvexHull:
1. Alcohol vs Malic Acid
2. Ash vs Alcalinity of Ash
3. Flavanoids vs Nonflavanoids Phenols
4. Color Intensity vs Hue
```



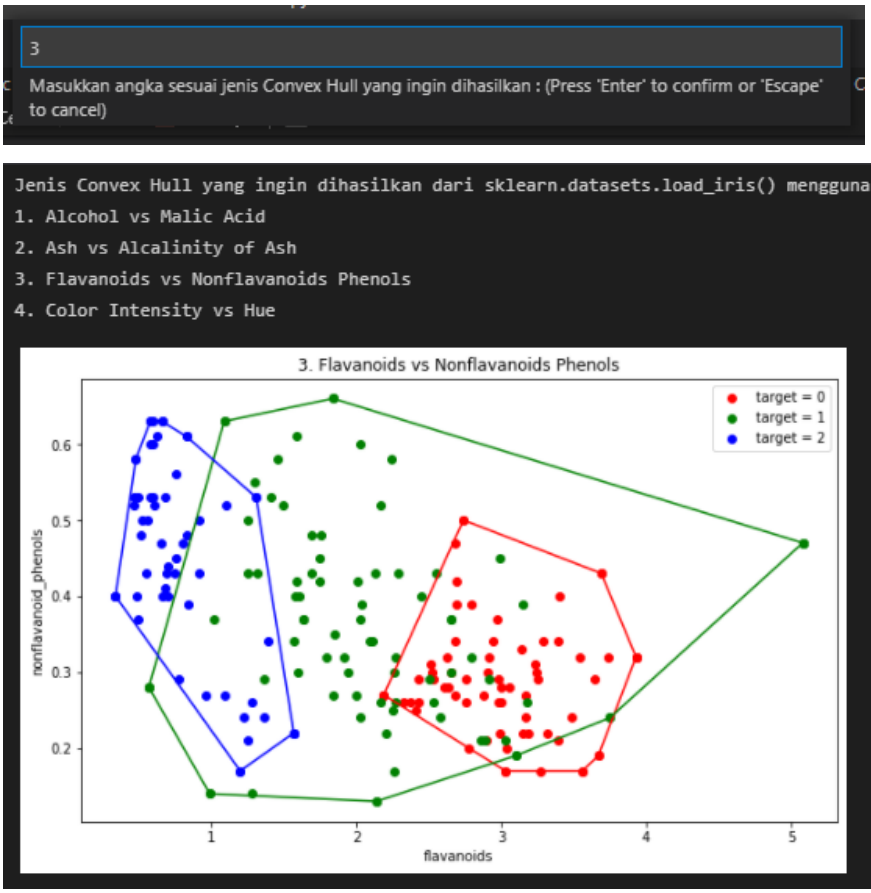
Input/Output Ash vs Alcalinity of Ash



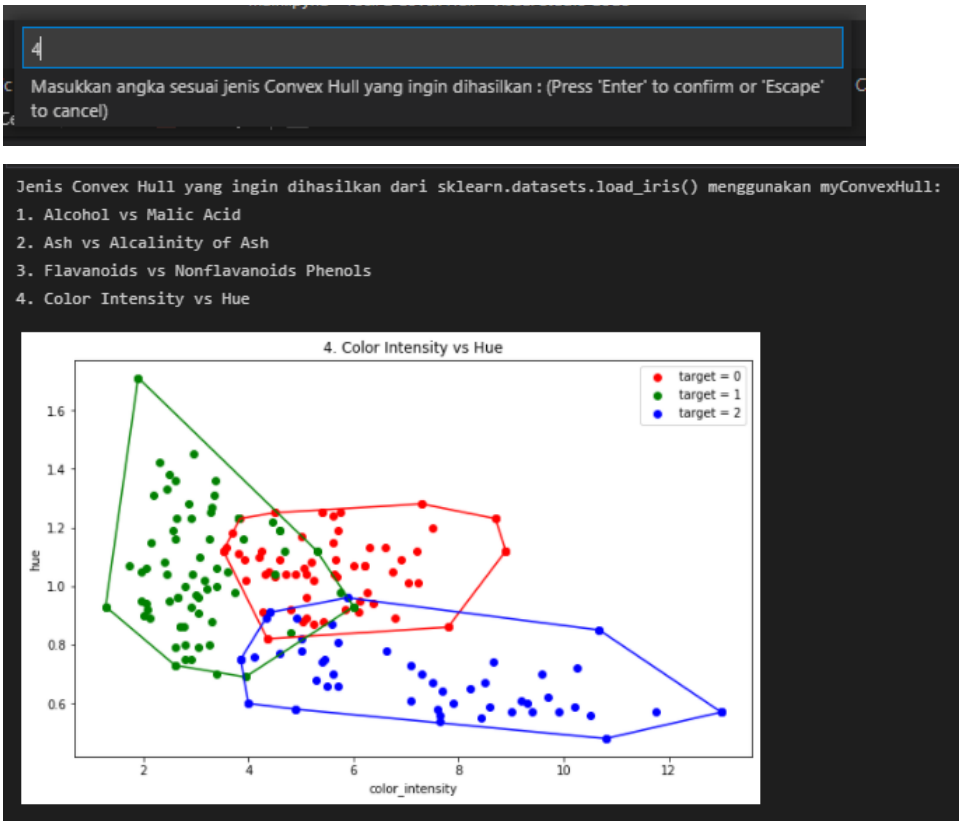

```
Jenis Convex Hull yang ingin dihasilkan dari sklearn.datasets.load_iris() menggunakan myConvexHull:
1. Alcohol vs Malic Acid
2. Ash vs Alcalinity of Ash
3. Flavanoids vs Nonflavanoids Phenols
4. Color Intensity vs Hue
```



Input/Output Flavanoids vs Nonflavanoids Phenols



Input/Output Color Intensity vs Hue



IV. Alamat (*Link*) GitHub Source Code

https://github.com/LordGedelicious/Convex-Hull-with-Python

V. *Checklist* Pencapaian dan Progress Program

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 deengan warna yang berbeda	√	
4. **Bonus :** Program dapat menerima *input* dan menuliskan *output* untuk *dataset* lainnya.	√	