Laporan Tugas Kecil 2 Strategi Algoritma IF2211
Penerapan Algoritma *Divide and Conquer* untuk Membuat *Convex Hull*

Vincent Prasetiya Atmadja
K03/13520099

A. Algoritma Divide and Conquer

Algoritma Divide and Conquer merupakan salah satu algoritma yang populer digunakan di dunia Informatika. Algoritma ini membagi persoalan menjadi persoalan yang lebih kecil, kemudian menyelesaikannya, lalu menggabungkan solusinya kembali. Secara umum, algoritma ini terdiri dari 3 bagian yaitu

1. Divide : membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya berukuran hampir sama)
2. Conquer (solve): menyelesaikan masing-masing upa-persoalan (secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar).
3. Combine: mengabungkan solusi masing-masing upa-persoalan sehingga membentuk solusi persoalan semula

Algoritma ini dapat diterapkan pada berbagai persoalan, salah satunya adalah pembuatan convex hull. Convex Hull dari sebuah himpunan titik S didefinisikan sebagai himpunan convex terkecil yang mengandung S. Himpunan titik pada bidang planar disebut convex 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.

Desain algoritma yang penulis gunakan untuk mendesain convex hull akan penulis jelaskan di bawah ini. Hasil akhir dari algoritma ini adalah kumpulan pasangan titik yang jika digambarkan akan terhubung dengan garis dan membentuk Convex Hull dan input yang digunakan adalah kumpulan data.

1. Lakukan *preprocessing* pada data masukan dengan cara menghabus data yang kembar dan urutkan data secara menaik berdasarkan absis, diikuti ordinatnya.
2. Ambil dua titik ekstrem dari kumpulan data ini, yaitu kedua titik ini adalaha elemen pertama dan terakhir pada data yang telah diurutkan. Sebut kedua titik ini sebagai P_1 dan P_2.
3. Bagi bidang menjadi 2 bagian dengan garis yang menghubungkan P_1P_2 sebagai pembagi. Pengecekan posisi titik lainnya dapat dilakukan dengan pengecekan determinan. Hasil akhir dari tahap ini

adalah dua buah bagian data, data pertama berada di bagian kiri atau atas dari garis dan sisanya berada di bagian kanan atau bawah dari garis.

4. Olah kedua bagian data masing-masing dengan melakukan tahap sebagai berikut.
    a. Apabila kumpulan data sudah kosong, tambahkan titik P_1 dan P_2 pada kumpulan pasangan titik hasil algoritma, kemudian abaikan bagian b dan c.
    b. Pilih titik P_3 yang terdapat pada kumpulan data sehingga P_3 memiliki jarak terjauh dengan garis yang dibuat oleh P_1P_2. Apabila ada titik yang berjarak sama, ambil titik yang memaksimalkan sudut $P_3P_1P_2$. Perhitungan jarak dapat dilakukan dengan menganggap $P_1P_2P_3$ sebagai sebuah segitiga, mencari luasnya dengan determinan, kemudian mencari tinggi segitiga dengan alas P_1P_2. Pencarian sudut akan memanfaatkan aturan cosinus.
    c. Buat kumpulan data terbaru yang berisi titik yang berada di luar seigitiga $P_1P_2P_3$. Titik ini dikelompokkan berdasarkan posisinya apakah diluar P_1P_3 atau P_2P_3.
    d. Olah setiap bagian lainnya dengan mengulangi langkah a, b, c, dan d ini.

B. Source Code Program

Program yang penulis buat, terdiri dari 2 file python.. File tersebut adalah myConvexHull.py yang berisi library Convex Hull dan main.py yang berisi main program. Kode selengkapnya dapat dilihat pada repository penulis, yaitu (https://github.com/TheOne28/Convex_Hull).

myConvexHull.py

```python
from copy import deepcopy
from math import isclose, acos

def determinant(p1, p2, p3):
    #menghitung determinan dari p1, p2, p3 -> p1 dan p2 membentuk garis dan
    #p1, p2, dan p3 adalah pasangan titik
    x1, y1 = p1[0], p1[1]
    x2, y2 = p2[0], p2[1]
    x3, y3 = p3[0], p3[1]

    return (x1*y2) + (x3 * y1) + (x2 * y3) - (x3 * y2) - (x2 * y1) - (x1 * y3)
```

```python
def findLength(p1, p2):
    #Menghitung panjang garis yang dibentuk oleh titik p1 dan p2
    difx = p1[0] - p2[0]
    dify = p1[1] - p2[1]

    return ((difx ** 2) + (dify ** 2)) ** 0.5

def findCosine(find, param1, param2):
    #Menghitung cosine yang dibentuk oleh garis find param1, dan param2
    #Sudut yang dibentuk adalah sudut antara param1 dan param2, dengan find adalah sisi di depan sudut
    #Masing-masing find, param1, dan param2 berisi panjang dari masing" garis

    denom = (param1 ** 2) + (param2 ** 2) - (find ** 2)
    denom /= (2 * param2 * param1)
    return denom

def checkexist(array, toCheck):
    #Mengecek kemuncukan titik toCheck di arrays
    for each in array:
        if(each[0] == toCheck[0] and each[1] == toCheck[1]):
            return True
    return False

def createDict(data):
    #Membuat dictionary dengan key berupa pasangan point dan value adalah indexnya di dataset awal
    dataInDict = {}

    for i in range(len(data)):
        dataInDict[tuple(data[i])] = i

    return dataInDict
```

```python
def preprocessing(data):
    dataInDict = createDict(data)

    cleanData = []
    [cleanData.append(x) for x in data if not checkexist(cleanData, x)]

    sortedData = sorted(cleanData, key = lambda x: (x[0], x[1]))

    return dataInDict, sortedData


def convexhull(data):
    #Membuat deepcopy dari data awal untuk mengurangi risiko kerusakan data awal
    copyData = deepcopy(data)

    dataInDict, sortedData = preprocessing(copyData)

    p1 = sortedData[0]
    p2 = sortedData[len(sortedData) - 1]

    left_part = []
    right_part = []

    for i in range(1, len(sortedData) - 1):
        if(determinant(p1, p2, sortedData[i]) > 0):
            left_part.append(sortedData[i])
        elif(determinant(p1, p2, sortedData[i]) < 0):
            right_part.append(sortedData[i])

    hullPair = []
    DCStep(left_part, p1, p2, hullPair, dataInDict)
    DCStep(right_part, p2, p1, hullPair, dataInDict)

    return hullPair
```

```python
def DCStep(array, p1, p2, final, dataInDict):
    if(len(array) == 0):
        newPair = []
        newPair.append(dataInDict[tuple(p1)])
        newPair.append(dataInDict[tuple(p2)])

        if(not checkexist(final, newPair)):
            final.append(newPair)
    else:
        length = findLength(p1, p2)
        #length -> p1 p2

        maxLength = 0
        maxAngle = 0.0
        p3 = []

        for i in range(len(array)):
            length2 = findLength(p1, array[i])
            length3 = findLength(p2, array[i])

            cosine = findCosine(length3, length2, length)

            #Ini kasus cosine > 1, dalam beberapa uji coba terjadi karena ketidakdepatan float, saya belum menemukan solusinya
            if(cosine > 1 or cosine < -1):      Vincent Atmadja, 32 minutes ago • Update readme and add library used
                cosine = 1
            area = determinant(p1, p2, array[i])
            distance = abs(area) / length

            if(distance > maxLength):
                maxLength = distance
                p3 = array[i]
                maxAngle = acos(cosine)
            elif(isclose(distance, maxLength)):
                angle = acos(cosine)
                if(angle > maxAngle):
                    maxLength = distance
```

```python
    first_part = []
    second_part = []

    if(len(p3) != 0):
        for i in range(len(array)):
            if(array[i][0] != p3[0] or array[i][1] != p3[1]):
                #Sebelah kiri garis p1pmax
                if(determinant(p1, p3, array[i]) > 0):
                    first_part.append(array[i])
                #Sebelah kanan garis p2pmax
                if(determinant(p2, p3, array[i]) < 0):
                    second_part.append(array[i])
        DCStep(first_part, p1, p3, final, dataInDict)
        DCStep(second_part, p3, p2, final, dataInDict)
```

main.py

```python
from sklearn import datasets        Vincent Atmadja, an hour ago • Clean library and make main program
import pandas as pd
import matplotlib.pyplot as plt
from myConvexHull import convexhull

color = {
    "blue" : "b",
    "green" : "g",
    "red" : "r",
    "cyan" : "c",
    "magenta" : "m",
    "yellow" : "y",
    "black" : "k",
    "white" : "w"
}

color_list = ["blue", "green", "red", "cyan", "magenta", "yellow", "black", "white"]

def getDatabase():
    print("Welcome to ConvexHull")
    print("Currenlt, we only works for 3 databases: ")
    print("1. Iris")
    print("2. Wine")
    print("3. Breast_Cancer")

    choice = 0
    while(True):
        choice = int(input("Which databases do you want to use (input the number)? "))

        if(choice >= 1 and choice <= 3):
            break
        else:
            print("That number is not available, please input another number")
```

```python
    if(choice == 1):
        return datasets.load_iris()
    elif(choice == 2):
        return datasets.load_wine()
    else:
        return datasets.load_breast_cancer()

def getFeatureName(data):
    i = 1
    featureNames = {}

    for each in data.feature_names:
        featureNames[i] = each
        i += 1

    print("Available feature to be plotted: ")

    for each in featureNames:
        print("{}.{}".format(each, featureNames[each]))

    id1 = 0
    id2 = 0

    while(True):
        id1 = int(input("Which features do you want to use for x axis (input the number)? "))

        if(id1 >= 1 and id1 <= len(featureNames)):
            break
        else:
            print("That number is not available, please input another number")

    while(True):
        id2 = int(input("Which features do you want to use for y axis (input the number)? "))
```

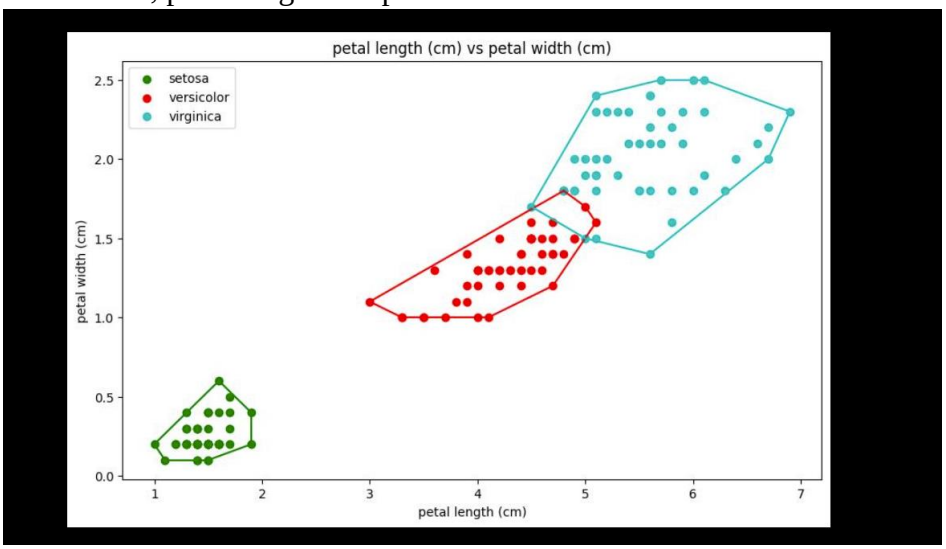
```
        if(id2 >= 1 and id2 <= len(featureNames) and id2 != id1 ):
            break
        elif(id1 == id2):
            print("That number is already selected for feature in x axis, please input another number")
        else:
            print("That number is not available, please input another number")

    return featureNames, id1 - 1, id2 - 1

def getColor(count):
    selectedColor = []
    print("Now choose, {} colors for your line".format(count))
    print("Available color: ")

    for each in color_list:
        print("  {}".format(each))

    while(count > 0):
        select = ""
        while(True):
            select = input("Which color do you want to use? ")

            if(select.lower() in color_list):
                selectedColor.append(color[select])
                break
            else:
                print("Those color is not available, please input another color")


        count -= 1

        if(count != 0):
            print("Next Color !")
```

```
    return selectedColor

def main():
    data = getDatabase()
    df = pd.DataFrame(data.data, columns=data.feature_names)

    feature_names, id1, id2 = getFeatureName(data)
    df['Target'] = pd.DataFrame(data.target)

    plt.figure(figsize = (10, 6))
    plt.title('{} vs {}'.format(feature_names[id1 + 1], feature_names[id2 + 1]))
    plt.xlabel(data.feature_names[id1])
    plt.ylabel(data.feature_names[id2])

    selected_color = getColor(len(data.target_names))

    for i in range(len(data.target_names)):
        bucket = df[df['Target'] == i]
        bucket = bucket.iloc[:,[id1, id2]].values

        hull = convexhull(bucket)
        plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i], color=selected_color[i]
        for simplex in hull:
            plt.plot(bucket[simplex, 0], bucket[simplex, 1], selected_color[i])

    plt.legend()

    output = input("Please choose name for your output file: ")
    plt.savefig("../test/output/{}.jpg".format(output), bbox_inches="tight")

if (__name__ == "__main__"):
    main()
```

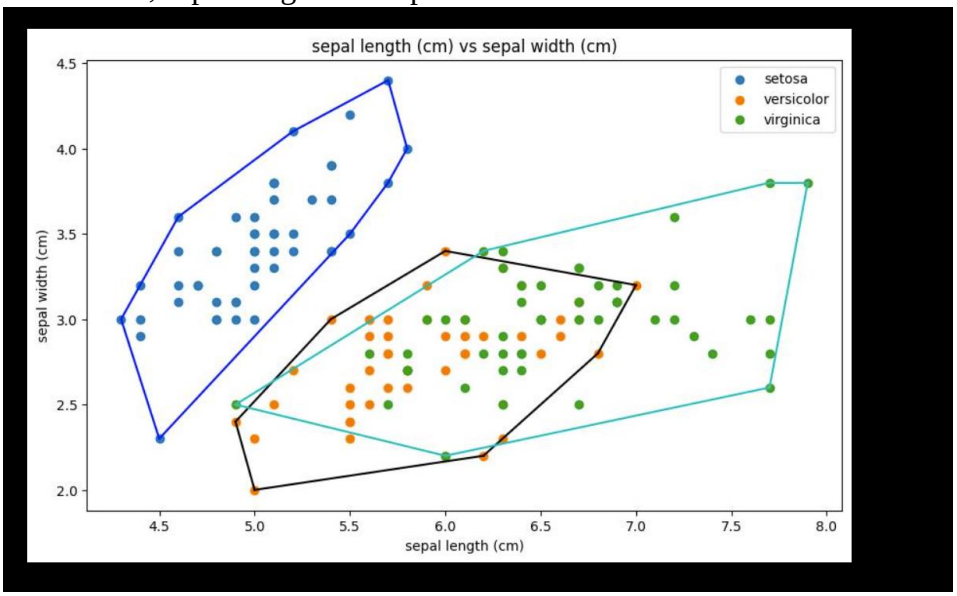
C. Input dan Output

Berikut ini akan penulis lampirkan beberapa input dan output dari beberapa dataset dan pasangan parameter yang penulis uji. Untuk selengkapnya dapat dilihat pada folder test/output.

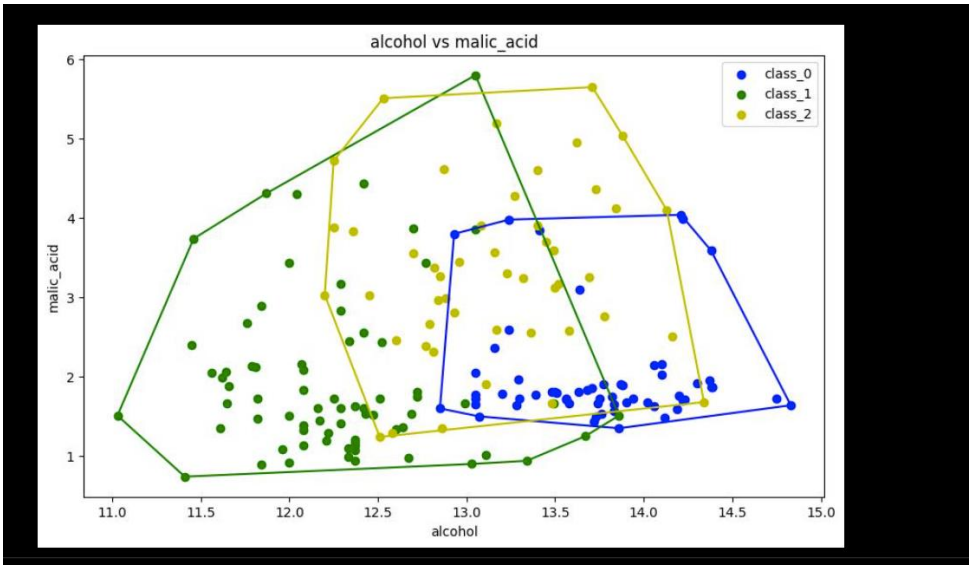
1. Dataset Iris, petal-length dan petal-width



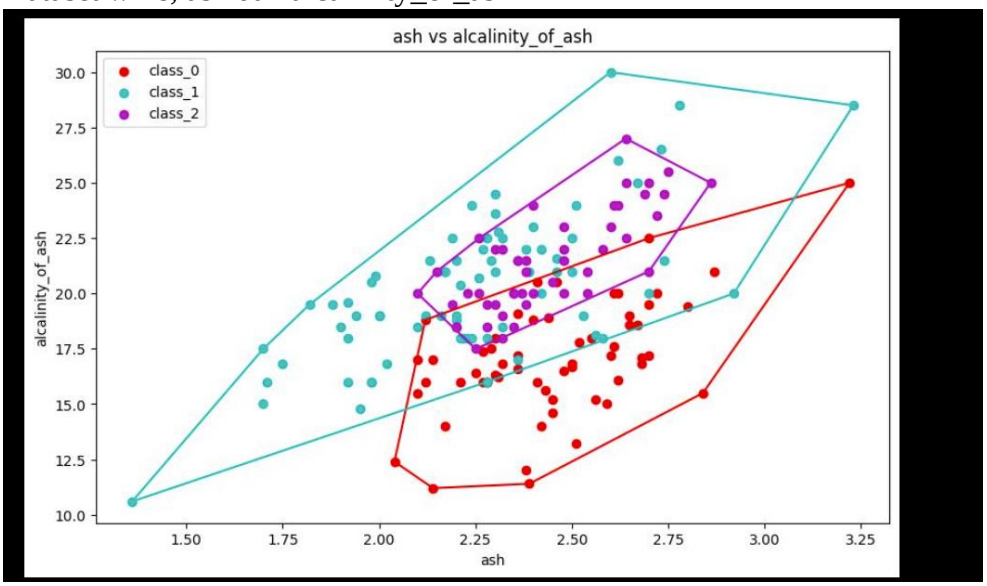
2. Dataset iris, sepal-length dan sepal-width



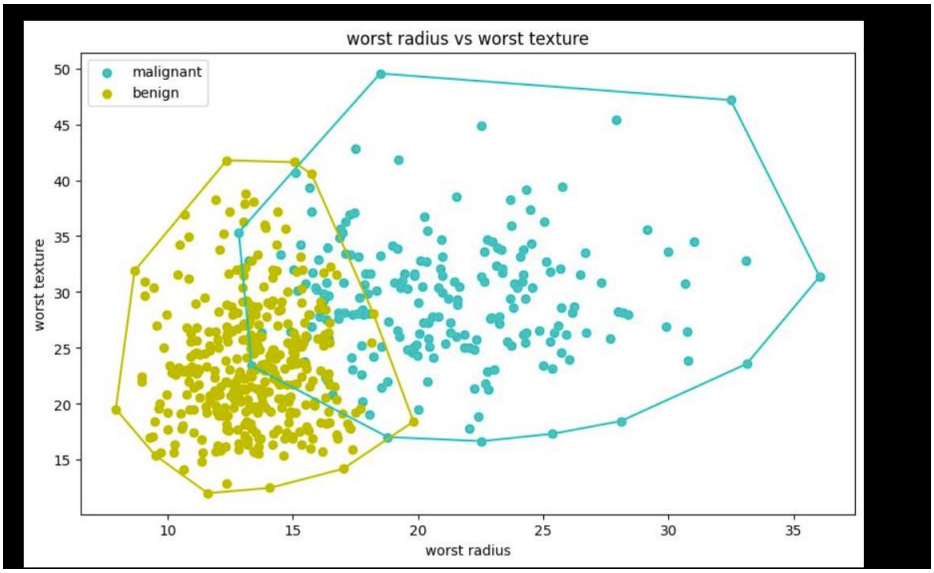
3. Dataset wine, alcohol dan malic_acid



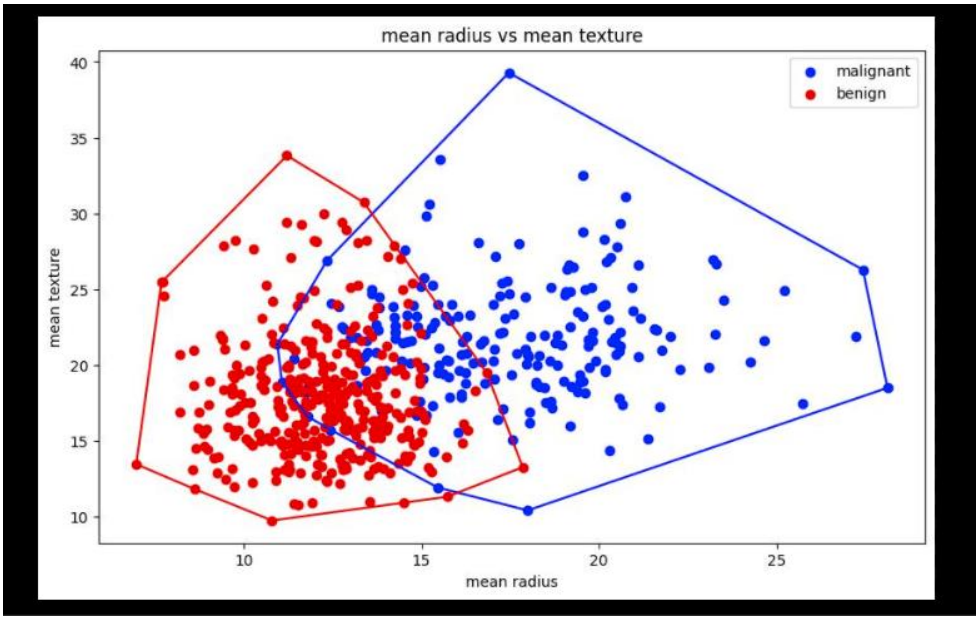
4. Dataset wine, ash dan alcanility_of_ash



5. Dataset Breact_cancer, worst radiusd an worst texture



6. Dataset Breast_cancer, mean radius dan mean texture



D. Extra

Github : https://github.com/TheOne28/Convex_Hull

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. Bonus: program dapat menerima input dan menuliskan output untuk dataset lainnya.		