Tugas Kecil 2 Strategi Algoritma Semester II/2022:
Implementasi Convex Hull untuk Visualisasi Tes Linear Separability Dataset dengan Algoritma Divide and Conquer

Oleh: Nathanael Santoso / 13520129

Bab I
Algoritma

1.1. Pendahuluan

Algoritma Divide and Conquer merupakan jenis algoritma yang membagi sebuah masalah menjadi bagian yang lebih kecil kemudian menyelesaikan bagian kecil tersebut sampai mendapatkan hasil dari keseluruhan bagian kecil tersebut. Untuk menyelesaikan permasalahan menghitung titik pada convex hull suatu dataset, dapat diterapkan algoritma divide and conquer. Convex hull adalah titik terluar sekumpulan titik sehingga sembarang titik p pada dataset berada di dalam convex hull dataset tersebut. Dengan mengambil dua titik ekstrim dataset sebagai titik ekstrim convex hull, dapat dibagi convex hull menjadi bagian kiri dan kanan. Bagian-bagian tersebut kemudian diambil titik terjauh dan dibagi kembali menjadi bagian luar dan bagian dalam dan seterusnya hingga semua titik berada di dalam convex hull. Algoritma ini termasuk sub-kategori Divide and Conquer hard split, easy merge.

1.2. Alur Umum

Algoritma menemukan solusi dengan langkah sebagai berikut:

1. Membuat Objek Bernama ConvexHull dengan atribut array dari titik bernama points dan hull, kemudian menambahkan titik ke dalam points dengan method obj.add(titik).
2. Points kemudian disortir menaik berdasarkan x, dan jika x sama, berdasarkan y.
3. Titik pertama dan terakhir diambil sebagai titik ekstrim convex hull, dan membagi points menjadi bagian kiri dan kanan.
4. Dari Bagian tersebut, diambil titik terjauh dari garis yang dibentuk titik ekstrim kemudian dibagi lagi menjadi titik luar dan titik dalam.
5. Mengulangi langkah 4 untuk titik luar memakai titik yang membagi titik luar dan titik dalam sebagai titik ekstrim sampai tidak ada lagi titik luar.
6. Jika sudah tidak ada titik luar, titik yang sudah ditemukan dimasukkan ke dalam array hull dengan searah jarum jam agar mudah divisualisasikan.

1.3. Penjelasan Metode

Untuk menjalankan algoritma di atas, diperlukan implementasi metode sebagai berikut:

- **add(self, point)** : menambahkan point ke array points
- **sortPoints(self)** : menyortir point secara menaik menurut bagian x point, kemudian bagian y.
- **getPartitions(p1, p2, p3)** : mendapatkan apakah p3 berada di sebelah kiri p1p2 atau sebelah kanan dengan metode determinan.
- **getLength(x, y)** : mendapatkan panjang sebuah vektor.
- **getDistance(p1, p2, p3)** : mendapatkan jarak titik p3 dari garis p1p2.
- **getAngle(p1, p2, p3)** : mendapatkan sudut dari p2p1p3.
- **getDivided(points, direction=0, p1=None, pn=None)** : mendapatkan titik di kiri atau kanan sebuah garis p1pn pada himpunan titik points. Direction menentukan kiri atau kanan.
- **pickFurthest(points)** : memilih titik terjauh pada points dari garis p1pn.

- **getHull(self, points=None)** : mendapatkan titik pada convex hull dan menambahkannya pada array hull secara searah jarum jam.

Bab II
Implementasi

2.1. Implementasi Library Convex Hull "myConvexHull.py"

```
# Import Library For Data Visualization
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from math import sqrt

# Import Test Datasets
from sklearn import datasets

class ConvexHull():
    points = []
    hull = []

    def __init__(self):
        self.points = []
        self.hull = []

    def add(self, point):
        self.points.append(point)

    def sortPoints(self):
        self.points = sorted(self.points, key=lambda x: [x[0], x[1]])

    def getPartition(p1, p2, p3):
        #       x1    * y2    + x3    * y1    + x2    * y3      -   x3    *
y2    - x2    * y1    - x1    * y3
        return (p1[0] * p2[1] + p3[0] * p1[1] + p2[0] * p3[1]) - (p3[0] *
p2[1] + p2[0] * p1[1] + p1[0] * p3[1])

    def getLength(x, y):
        return sqrt(x ** 2 + y ** 2)

    def getDistance(p1, p2, p3):
        num = abs((p2[0] - p1[0]) * (p1[1] - p3[1]) -
                  (p1[0] - p3[0]) * (p2[1] - p1[1]))
        den = ConvexHull.getLength(p2[0] - p1[0], p2[1] - p1[1])
        return num / den

    def getAngle(p1, p2, p3):
        v1 = [p2[0] - p1[0], p2[1] - p1[1]]
        v2 = [p3[0] - p1[0], p3[1] - p1[1]]
        num = (v1[0] * v2[0] + v1[1] * v2[1])
        den = (ConvexHull.getLength(v1[0], v1[1]) *
ConvexHull.getLength(v2[0], v2[1]))
        return np.arccos( num / den )
```

```
    def getDivided(points, direction=0, p1=None, pn=None):
        partition = []
        if p1 is None and pn is None:
            p1 = points[0]
            pn = points[-1]
        partition.append(p1)
        for point in points[1:-1]:
            determinant = ConvexHull.getPartition(p1, pn, point)
            if not direction and determinant > 0:
                partition.append(point)
            elif direction and determinant < 0:
                partition.append(point)
        partition.append(pn)
        return partition

    def pickFurthest(points):
        candidate = points[1]
        p1 = points[0]
        pn = points[-1]
        for point in points[2:-1]:
            d1 = ConvexHull.getDistance(p1, pn, candidate)
            d2 = ConvexHull.getDistance(p1, pn, point)
            if d1 < d2 or (d1 == d2 and ConvexHull.getAngle(p1, pn,
candidate) < ConvexHull.getAngle(p1, pn, point)):
                candidate = point
        return candidate

    def getHull(self, points=None):
        if points is None:
            self.sortPoints()
            # Append Hullpoints in a clockwise manner
            self.hull.append(self.points[0])  # p1
            self.getHull(points=ConvexHull.getDivided(self.points,
direction=0))  # left points
            self.hull.append(self.points[-1])  # pn
            self.getHull(points=ConvexHull.getDivided(self.points,
direction=1))  # right points
        elif len(points) > 2:
            if len(points) == 3:
                self.hull.append(points[1])
            else:
                pmax = ConvexHull.pickFurthest(points)
                if ConvexHull.getPartition(points[0], points[-1], pmax) > 0:
                    self.getHull(points=ConvexHull.getDivided(points,
direction=0, p1=points[0], pn=pmax)) # points left of p1-pmax
                    self.hull.append(pmax) # pmax
```

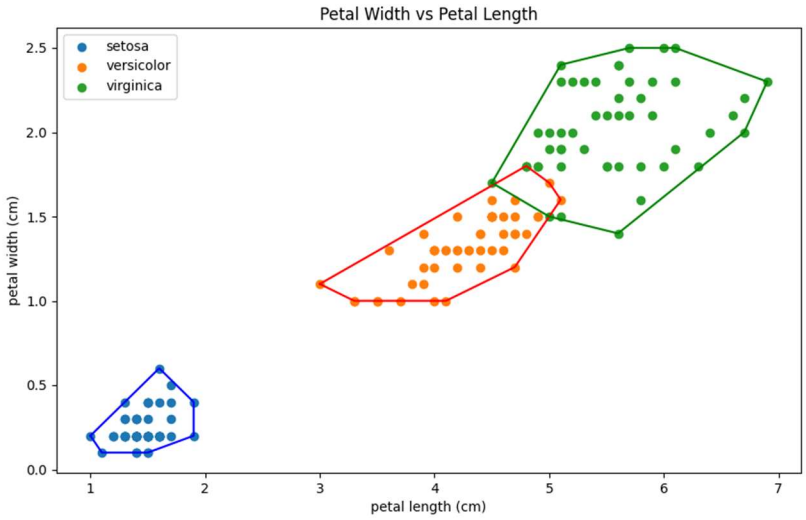
```
                    self.getHull(points=ConvexHull.getDivided(points,
direction=1, p1=points[-1], pn=pmax)) # points right of pn-pmax
                else:
                    self.getHull(points=ConvexHull.getDivided(points,
direction=0, p1=points[-1], pn=pmax)) #points left of pn-pmax
                    self.hull.append(pmax) #pmax
                    self.getHull(points=ConvexHull.getDivided(points,
direction=1, p1=points[0], pn=pmax)) #points right of p1-pmax
```

2.2. Implementasi Uji Coba "myConvexHull.py"

```
if __name__ == "__main__":
    data = datasets.load_%dataset%() # diganti dengan dataset yang
diperlukan (cth: wine, breast_cancer, dsb.)
    # create a DataFrame
    df = pd.DataFrame(data.data, columns=data.feature_names)
    df['Target'] = pd.DataFrame(data.target)
    # visualisasi hasil ConvexHull
    plt.figure(figsize=(10, 6))
    colors = ['b', 'r', 'g']
    plt.title('%Table_Title%') # Diganti dengan judul tabel sesuai
    plt.xlabel(data.feature_names[0]) # index xlabel dan ylabel sesuai
parameter yang dibandingkan
    plt.ylabel(data.feature_names[1])
    for i in range(len(data.target_names)):
        bucket = df[df['Target'] == i] # hilangkan jika tidak ada
        bucket = bucket.iloc[:, [0, 1]].values
        hull = ConvexHull()
        for point in bucket:
            hull.add(point)
        hull.getHull()
        plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
        hx = [point[0] for point in hull.hull]
        hy = [point[1] for point in hull.hull]
        hx.append(hull.hull[0][0]) # Menambah point pertama untuk menutup
Hull pada plot
        hy.append(hull.hull[0][1])
        plt.plot(hx, hy, colors[i])
    plt.legend()
    plt.savefig('%PATH%') # Diganti dengan path yang dituju
    plt.show()
```

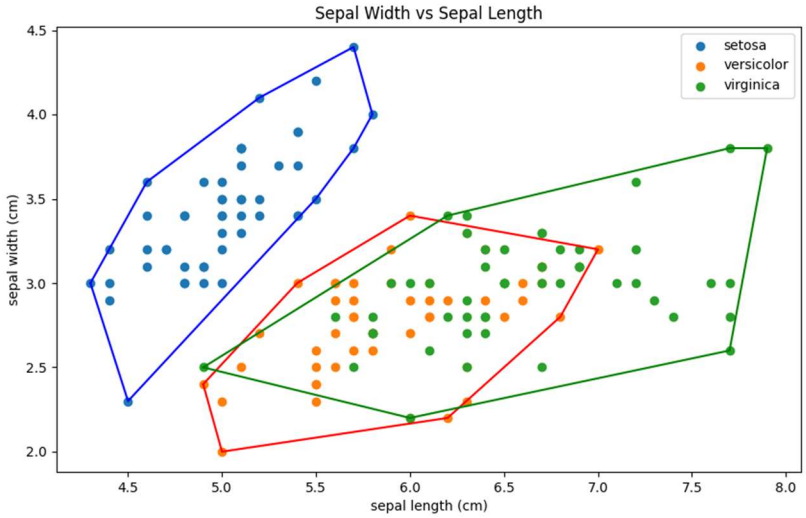
Bab III
Hasil Pengujian

3.1. Petal.png
Diambil dari dataset iris dengan parameter petal width terhadap petal length



Gambar 3.1. Petal Width vs Petal Length

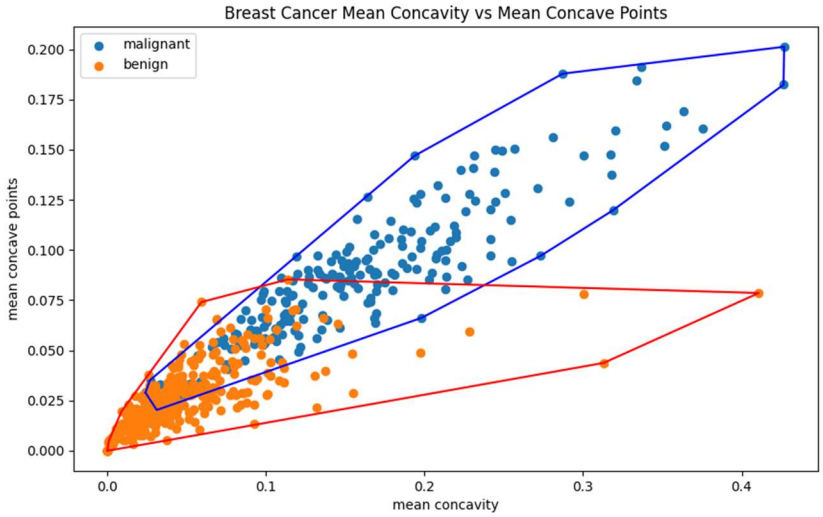
3.2. Sepal.png
Diambil dari dataset iris dengan parameter sepal width terhadap sepal length



Gambar 3.2. Sepal Width vs Sepal Length

3.3. Concavity.png

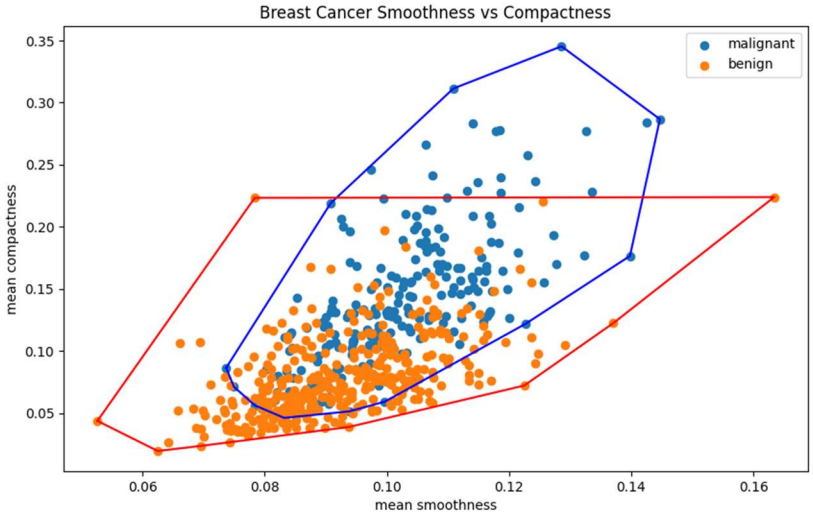
Diambil dari dataset breast_cancer dengan parameter mean concavity terhadap mean concave points



Gambar 3.3. Mean Concavity vs Mean Concave Points

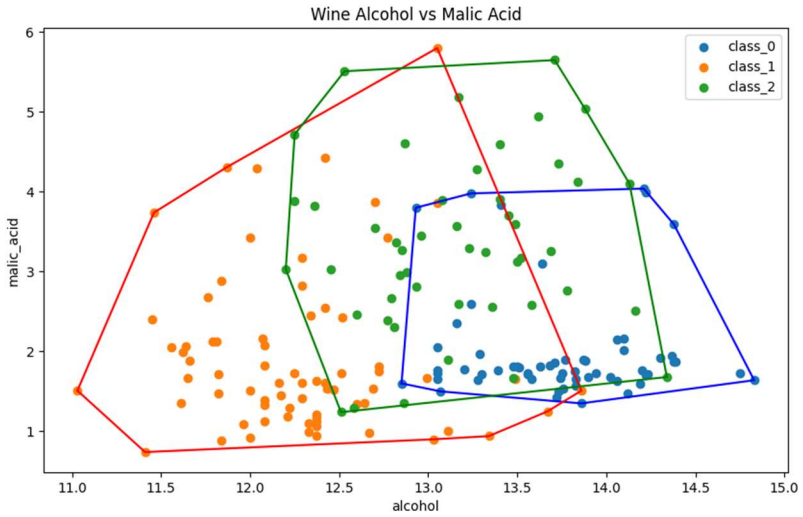
3.4. Smooth.png

Diambil dari dataset breast_cancer dengan parameter mean smoothness terhadap mean compactness



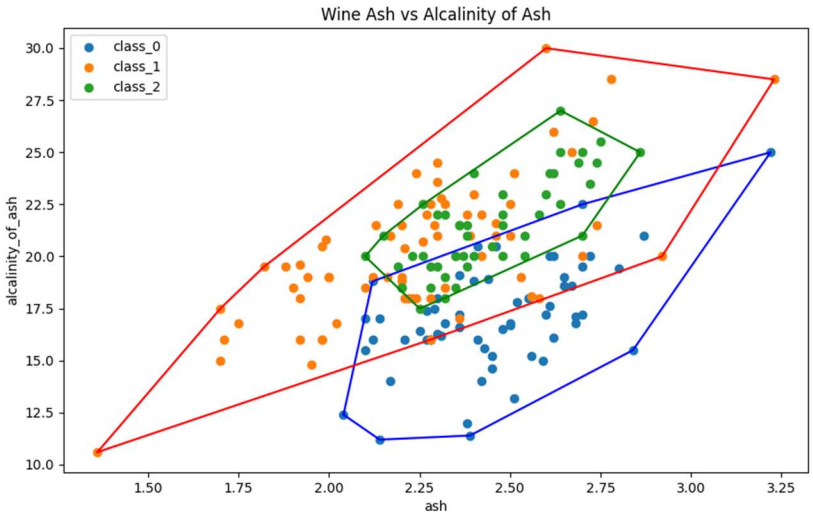
Gambar 3.4. Smoothness vs Compactness

3.5. Alcohol.png
Diambil dari dataset alcohol dengan parameter konsentrasi Alcohol terhadap Malic Acid



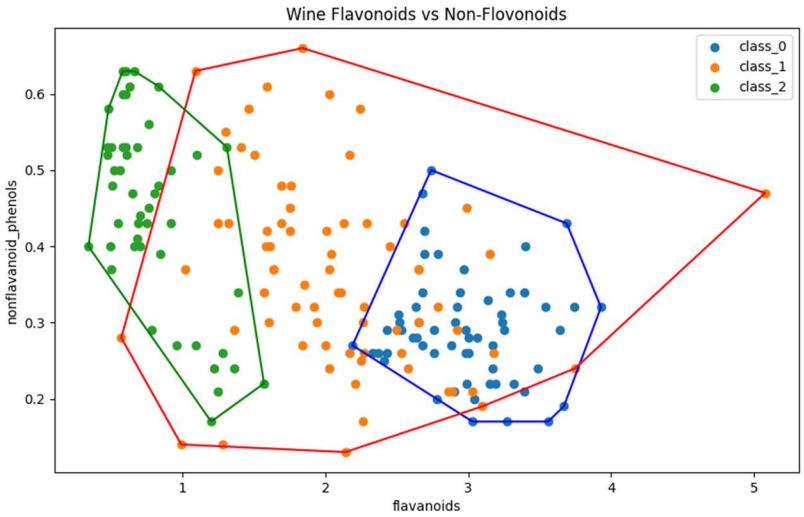
Gambar 3.5. Alcohol vs Malic Acid

3.6. Ash.png
Diambil dari dataset alcohol dengan parameter Ash terhadap Alcalinity dari Ash



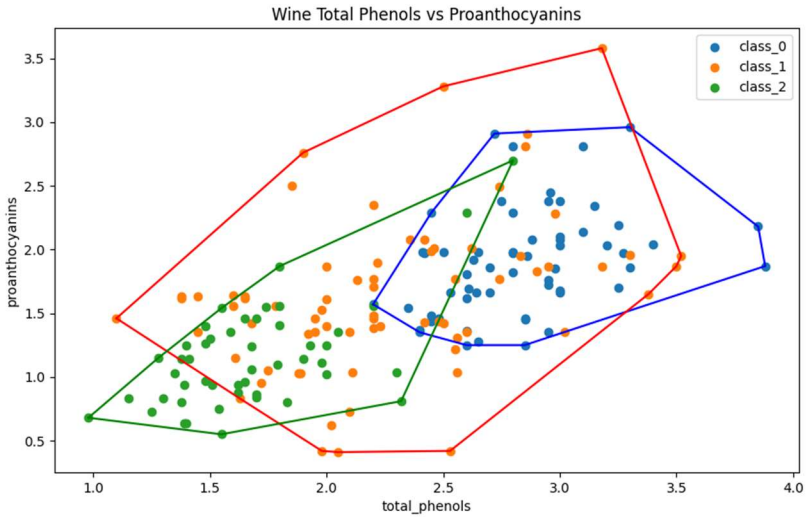
Gambar 3.6. Ash vs Alcalinity of Ash

3.7. Flavonoids.png
Diambil dari dataset alcohol dengan parameter Flavonoid terhadap Non-Flavonoid Phenols



Gambar 3.7. Flavonoids vs Non-Flavonoids

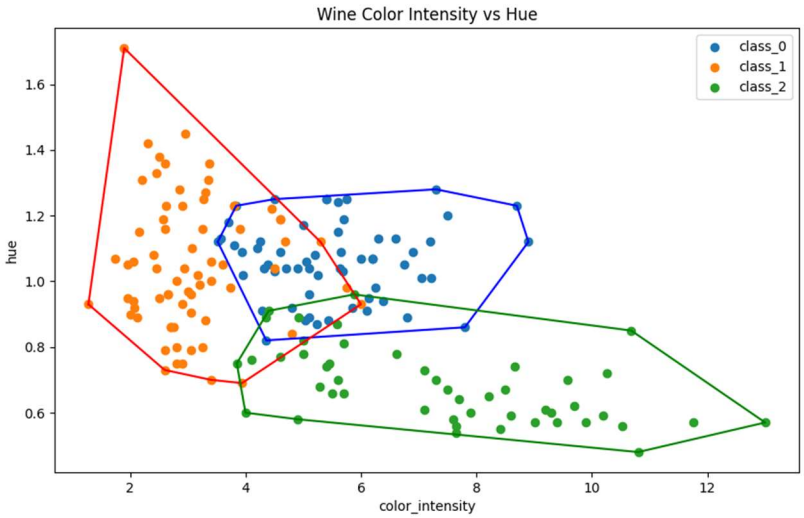
3.8. Phenols.png
Diambil dari dataset alcohol dengan parameter Total Phenols terhadap Proanthocyanins



Gambar 3.8. Total Phenols vs Proanthocyanins

3.9. Color.png

Diambil dari dataset alcohol dengan parameter Color Intensity terhadap Hue



Gambar 3.9. Color Intensity vs Hue

Bab IV
Kesimpulan

4.1. Kesimpulan

Menggunakan stategi algoritma divide and conquer dapat mencari convex hull dari sekumpulan titik pada dataset.

4.2. Tabel Kelengkapan

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.	✓	

Tabel 4.1. Tabel Kelengkapan Tugas Kecil

4.3. Link Penting

Link Repo Github:
https://github.com/nart4hire/Tucil2_STIMA.git

Link Google Drive:
https://drive.google.com/drive/folders/1gd0IcdqUk6yXHYlKrJtYWvzjAFhYR9Om?usp=sharing