Tugas Kecil Strategi Algoritma – IF2211

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

13520048 – Arik Rayi Arkananta

A. Algoritma *Divide and Conquer*

Pada program yang telah saya buat, digunakan algoritma *Divide and Conquer* dengan bahasa pemrograman python untuk mencari *Convex Hull*. Berikut adalah cara kerja algoritma *Divide and Conquer* dari sebuah array bucket yang berisi kumpulan koordinat titik-titik:

1. Pertama pencarian *Convex Hull* dibagi dua, yaitu bucket bagian atas dan bawah
2. lalu dicari titik dengan x terkecil leftIdx dan terbesar rightIdx sebagai batas bagian atas dan bawah
3. Dilakukan pengulangan yang mengiterasi seluruh elemen i dari bucket dan dicari jarak terjauh dari garis leftIdx-rightIdx dan juga yang sesuai dengan arah yang dicari
4. Perhitungan jarak titik i ke garis leftIdx-rightIdx dilakukan menggunakan rumus $\frac{|pa+qb+r|}{sqrt(p\text{^}2+q\text{^}2)}$, $pa + qb + r$ yang merupakan persamaan garis dicari menggunakan rumus $\frac{y-y1}{y2-y1} = \frac{x-x1}{x2-x1}$, lalu x dan y dimasukkan dengan x dan y dari titik i. Namun $sqrt(p\text{^}2 + q\text{^}2)$ tidak perlu dihitung karena perhitungan jarak ini hanya untuk perbandingan.
5. Jika terdapat jarak yang sama pada iterasi, dicari yang memiliki sudut i-leftIdx-rightIdx terbesar
6. Diulangi lagi algoritma 3-5 namun dibagi dua dengan leftIdx dan rightIdx diganti menjadi leftIdx & maxPoint dan juga maxPoint & rightIdx sampai tidak ada lagi titik diluar leftIdx dan rightIdx
7. Jika sudah tidak ada lagi titik diluar garis leftIdx-rightIdx, [leftIdx, rightIdx] dimasukkan ke dalam array hasil, karena garis leftIdx-rightIdx berarti merupakan bagian dari *Convex Hull*

B. Source code

Import dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
data = datasets.load_iris() # Ganti load_iris dengan yang lain jika ingin mengganti dataset
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
```

✓ 0.5s

Library pencarian Convex Hull

```python
from math import dist, acos

# Fungsi ini berfungsi untuk mencari sudut dari titik ABC
def getAngle(A, B, C):
    try:
        a = dist(B,C)
        b = dist(A,C)
        c = dist(A,B)
        return acos(((a**2)+(c**2)-(b**2))/(2*a*c))
    except:
        return 0;

# Fungsi ini merupakan algoritma divide and conquernya yang akan untuk mencari Convex Hull
def divideAndConquer(bucket, leftIdx, rightIdx, direction, result):
    maxPointDistAndIdx = [0,-1]

    # Jika leftIdx dan rightIdx belum ada maka dicari titik dengan x terkecil dan terbesar (pemanggilan awal fungsi)
    if leftIdx == -1 and rightIdx == -1:
        leftIdx = 0
        rightIdx = 0
        for i in range(len(bucket)):
            if bucket[i,0] < bucket[leftIdx,0]:
                leftIdx = i
            if bucket[i,0] > bucket[rightIdx,0]:
                rightIdx = i
```

```python
    # Pengulangan seluruh elemen pada bucket untuk mencari jarak terjauh dari garis leftIdx-rightIdx
    for i in range(len(bucket)):
        # Pencarian jarak titik i ke garis dari leftIdx dan rightIdx
        currentPoint = ((bucket[i,1] - bucket[leftIdx,1]) * (bucket[rightIdx,0] - bucket[leftIdx,0])
                        - (bucket[rightIdx,1] - bucket[leftIdx,1]) * (bucket[i,0] - bucket[leftIdx,0]))
        if currentPoint * direction ≥ 0:
            # Jika jarak lebih jauh maka maxPointDistAndIdx diganti
            if currentPoint * direction > maxPointDistAndIdx[0]:
                maxPointDistAndIdx[1] = i
                maxPointDistAndIdx[0] = currentPoint * direction
            # Jika jarak sama dengan maxPoint sebelumnya, maka dicari sudut terbesar menggunakan fungsi getAngle
            elif currentPoint * direction == maxPointDistAndIdx[0] and maxPointDistAndIdx[1] ≠ -1:
                if (getAngle(bucket[rightIdx],bucket[leftIdx],bucket[i])
                        > getAngle(bucket[rightIdx],bucket[leftIdx],bucket[maxPointDistAndIdx[1]])):
                    maxPointDistAndIdx[1] = i
                    maxPointDistAndIdx[0] = currentPoint * direction

    # Jika sudah tidak ada titik diluar garis leftIdx-rightIdx
    if maxPointDistAndIdx[1] == -1:
        result.append([leftIdx, rightIdx])
        return result

    result = divideAndConquer(bucket, leftIdx, maxPointDistAndIdx[1], direction, result)
    result = divideAndConquer(bucket, maxPointDistAndIdx[1], rightIdx, direction, result)
    return result
```

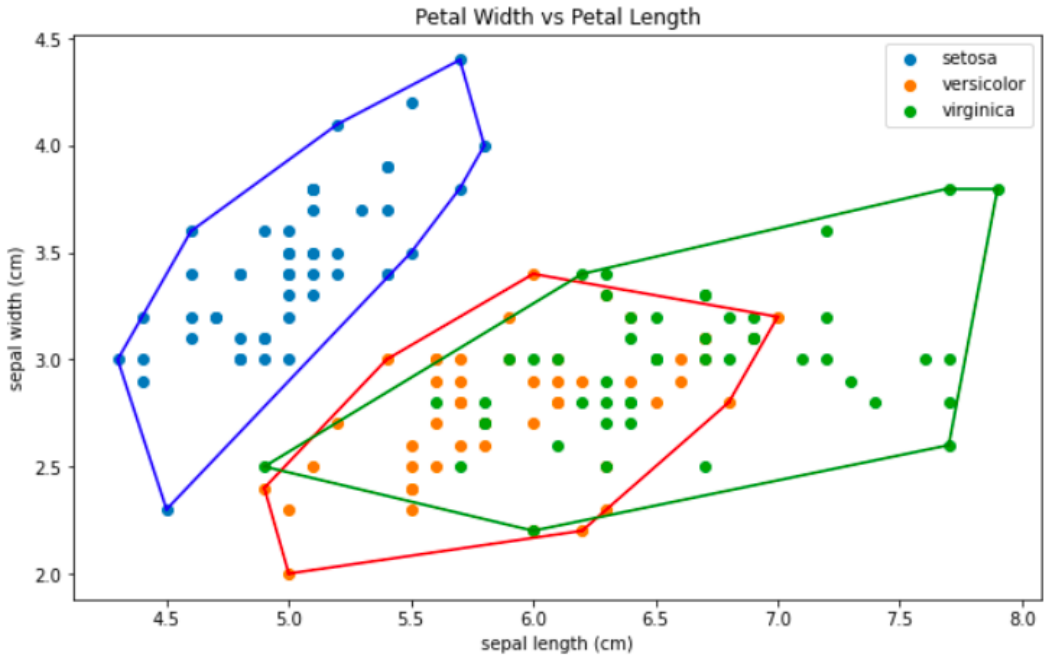
Visualisasi hasil Convex Hull

```
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)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    hull = ConvexHull(bucket) #bagian ini diganti dengan hasil implementasi
    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()
```

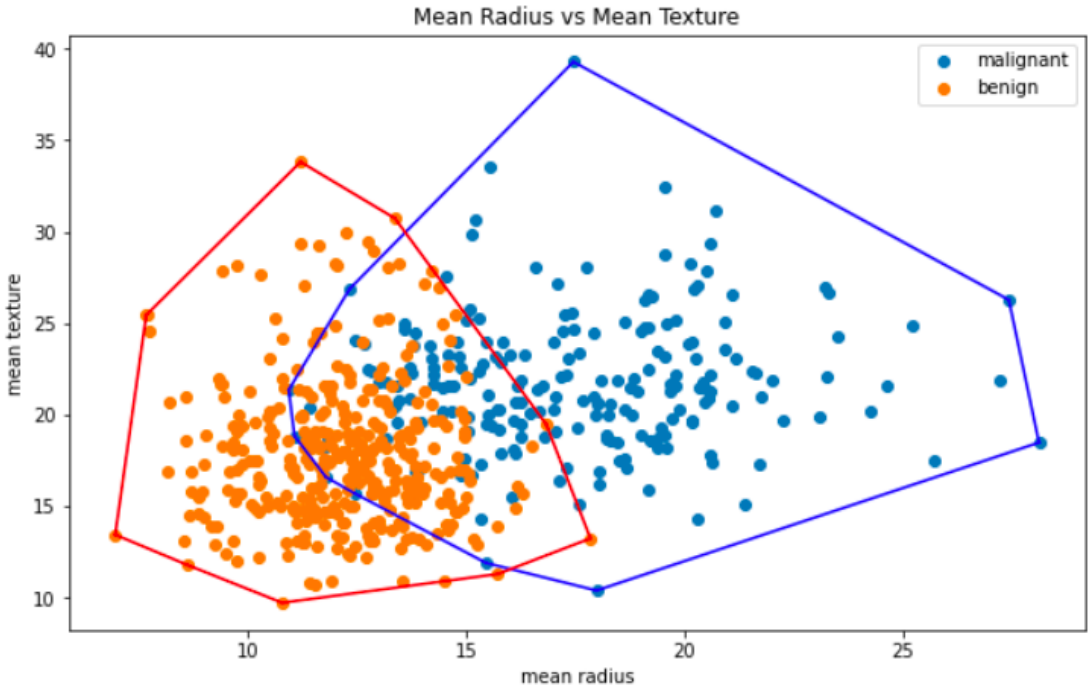
✓ 0.4s

C. Screenshot Input dan Output

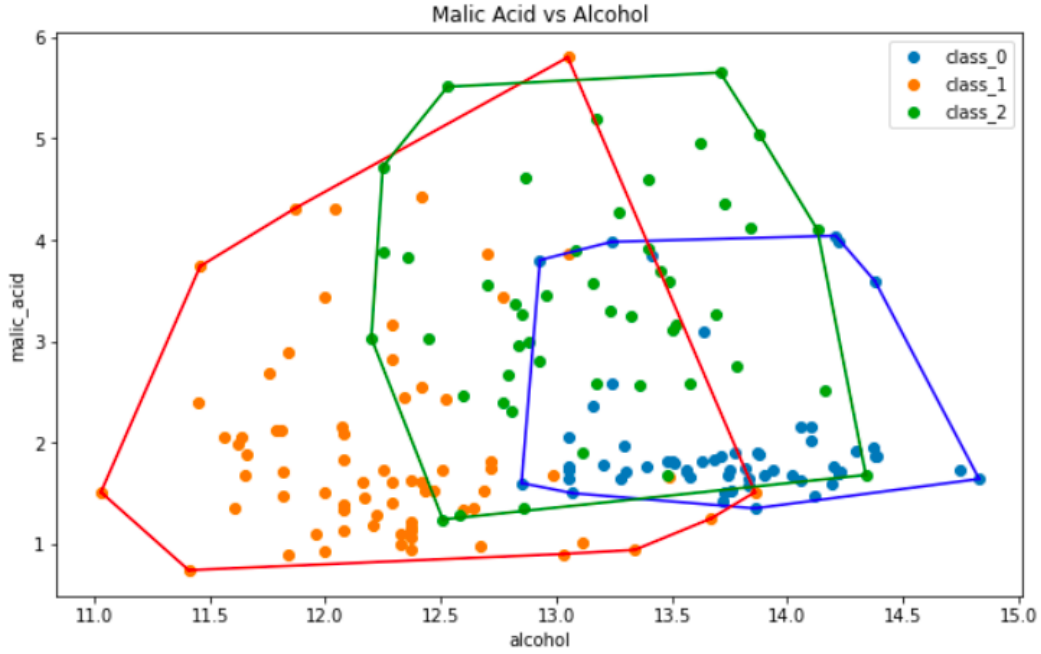
1. Dataset Iris



2. Dataset Breast_Cancer



3. Dataset Wine



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.	√	

Alamat repository:
https://github.com/arikrayi/Tucil-2-Stima-Convex-Hull