LAPORAN TUGAS KECIL
IF2211 STRATEGI ALGORITMA

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

Disusun oleh :

Vieri Mansyl 13520092

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

DAFTAR ISI

BAB I
Algoritma dalam implementasi *Convex Hull* dalam tes *Linearly Separability Dataset*

Convex Hull adalah merupakan sebutan untuk himpunan *convex* terkecil yang ‘membungkus’ seluruh data di dalam himpunan tersebut. Suatu himpunan titik pada bidang planar disebut sebagai *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. Salah satu penggunaannya ialah pada ujicoba keterpisahan linier pada dataset. Pada tugas besar ini, digunakan algoritma *Divide and Conquer* untuk penyelesaian permasalahan *Convex Hull* ini.

Implementasi algoritma *Divide and Conquer* pada program diawali dengan pengurutan seluruh koordinat secara terurut membesar berdasarkan nilai absisnya, lalu menghubungkan dua titik (misalkan p1 dan p2) , yang mana p1 merupakan titik dengan absis terkecil dan p2 merupakan titik dengan absis terbesar. Adanya garis p1p2 membagi himpunan titik – titik menjadi 2 daerah atas dan bawah (misalkan S1 dan S2). Pada masing – masing daerah, dicari titik terjauh daripada garis p1p2 , lalu titik yang didapat dihubungkan pada titik p1 dan p2 sehingga membentuk suatu segitiga. Selanjutnya, dilakukan hal yang sama terhadap titik – titik yang berada di luar segitiga. Seluruh titik – titik yang tercatat ‘paling jauh’ selanjutnya akan dihubungkan sehingga garis hubungan antar titik akan ‘membungkus’ seluruh titik yang ada pada himpunan.

BAB II

Kode Program

Kode Program myConvexHull.py berisikan fungsionalitas – fungsionalitas yang menunjang implementasi *convex hull* dan dapat diakses melalui link berikut.

https://github.com/VieriMansyl/Vieri-Tucil2-STIMA

Kode program dapat dilihat di bawah berikut.

1. **ndarray_to_list(l) → list of point**
   mengubah n-dimensi dari numPy array menjadi bentuk list lalu mengembalikan list tersebut.

```
def ndarray_to_list(l):
    newL = l.tolist()
    newL.sort()
    return newL
```

2. **det(p1,p2,p3) → float**
   menghitung determinan dari pada 3 titik p1 , p2 , p3.

```
def det(p1,p2,p3):
    return (p1[0]*p2[1]) + (p3[0]*p1[1]) + (p2[0]*p3[1]) - (p1[0]*p3[1]) - (p2[0]*p1[1]) - (p3[0]*p2[1])
```

3. **whichArea(p1,p2,p3) → integer**
   menentukan daerah keberadaan p3 terhadap garis yang dibentuk oleh titik p1 dan p2.

```
def whichArea(p1,p2,p3):
    determinan = det(p1,p2,p3)
    if(determinan == 0):
        return 0
    elif (determinan > 0):
        return 1
    elif (determinan < 0):
        return -1
```

4. **divideArea(p1,arr,p2) → list of point**
   membagi titik – titik pada himpunan titik arr terhadap gari p1p2.

```python
def divideArea(p1,arr,p2):
#mengembalikan daerah s1 dan s2 terhadap garis p1p2
    s1 = [] ; s2 = []
    dontcare = 0
    for idx in range(len(arr)):
        area = whichArea(p1,p2,arr[idx])
        if(area == 1):
            s1.append(arr[idx])
        if(area == 0):
            dontcare += 1
        if(area == -1):
            s2.append(arr[idx])

    return s1,s2
```

5. **eliminateInsideTriangle(p1,p2,p3,points,area) → list of point**

   mengeliminasi titik – titik pada himpunan titik points yang berada di dalam segitiga p1p2p3 berdasarkan daerah areanya.

```python
def eliminateInsideTriangle(p1,p2,p3,points,area):
    newPoint = []
    for point in points:
        a1 = whichArea(p1,p3,point)
        a2 = whichArea(p2,p3,point)
        if(area == 1):
            if(a1 == 1 or a2 == -1):
                newPoint.append(point)
        elif(area == 2):
            if(a1 == -1 or a2 == 1):
                newPoint.append(point)

    return newPoint
```

6. **farthestPoint(p1.arr.p2) → point**

   mencari titik terjauh yang terdapat pada himpunan titik arr terhadap garis p1p2.

   N.B. : untuk menghitung jarak suatu titik p terhadap garis p1p2 , digunakan modul dari pustaka numPy.linalg.norm , sebuah fungsi panggilan untuk mempermudah perhitungan jarak dengan mengimplementasikan perhitungan vektor.

```python
def farthestPoint(p1,arr,p2):
    farthest = [0,0]
    d = 0
    for idx in range(len(arr)):
        point1   = np.array(p1)
        point2   = np.array(p2)
        point    = np.array(arr[idx])
        distance = np.linalg.norm(np.cross(point2-point1, point1-point))/np.linalg.norm(point2-point1)

        if(d < distance):
            d = distance
            farthest = point.tolist()

    return farthest
```

7. findConvex(p1,arr,p2,area) → list of point

mencari titik-titik pembentuk convexHull.

```
def findConvex(p1,arr,p2,area):
    if(len(arr) == 0):
        return []
    if(len(arr) == 1):      #bersisa titik terluar atau tidak memiliki titik di luar hull
        return arr
    else:
        points = []
        if(area == 0):
            s1 , s2 = divideArea(p1 , arr , p2)

            #DAERAH S1
            newP1 = farthestPoint(p1,s1,p2)     #titik terjauh daerah S1
            points.append(newP1)
            newS1 = eliminateInsideTriangle(p1,p2,newP1,s1,1)

            s1_fromS1_left , ignorePart = divideArea(p1,newS1,newP1)       #daerah kiri-atas antara p1-newP1 (s1_fromS1_left)
            s1_fromS1_right , ignorePart = divideArea(newP1,newS1,p2)      #daerah kanan-atas antara newP1-p2 (s1_fromS1_right)
            newPoint1 = findConvex(p1 , s1_fromS1_left , newP1 , 1)
            newPoint2 = findConvex(newP1 , s1_fromS1_right , p2 , 1)

            #DAERAH S2
            newP2 = farthestPoint(p1,s2,p2)     #titik terjauh daerah S2
            points.append(newP2)
            newS2 = eliminateInsideTriangle(p1,p2,newP2,s2,2)

            ignorePart , s2_fromS2_left = divideArea(p1,newS2,newP2)       #daerah kiri-bawah antara p1-newP2 (s2_fromS2_left)
            ignorePart , s2_fromS2_right = divideArea(newP2,newS2,p2)      #daerah kanan-bawah antara newP2-p2 (s2_fromS2_right)
            newPoint3 = findConvex(p1 , s2_fromS2_left , newP2 , 2)
            newPoint4 = findConvex(newP2 , s2_fromS2_right , p2 , 2)

            return points + newPoint1 + newPoint2 + newPoint3 + newPoint4
```

```
        elif (area == 1):
            newP = farthestPoint(p1,arr,p2)
            points.append(newP)
            newS1 = eliminateInsideTriangle(p1,p2,newP,arr,1)

            s1_left , ignorePart = divideArea(p1,newS1,newP)
            s1_right , ignorePart = divideArea(newP,newS1,p2)
            newPoint1 = findConvex(p1, s1_left , newP , 1)
            newPoint2 = findConvex(newP, s1_right , p2 , 1)

            return points + newPoint1 + newPoint2

        elif (area == 2):
            newP = farthestPoint(p1,arr,p2)
            points.append(newP)
            newS2 = eliminateInsideTriangle(p1,p2,newP,arr,2)

            ignorePart , s2_left = divideArea(p1,newS2,newP)
            ignorePart , s2_right = divideArea(newP,newS2,p2)
            newPoint1 = findConvex(p1, s2_left , newP , 2)
            newPoint2 = findConvex(newP, s2_right , p2 , 2)

            return points + newPoint1 + newPoint2
```

8. convexHull(bucket) → list of point

Mengembalikan kumpulan titik pembentuk *convex hull*.

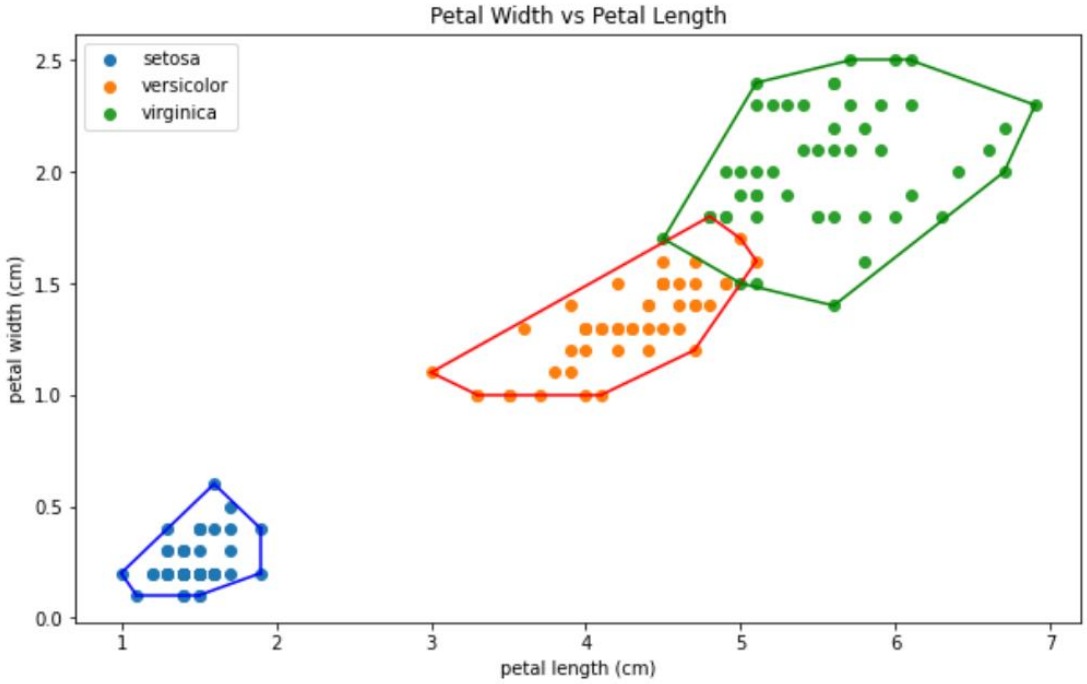
```
def convexHull(bucket):
    newBucket = ndarray_to_list(bucket)
    p1 = newBucket[0]
    p2 = newBucket[len(newBucket)-1]
    Points = findConvex(p1 , newBucket , p2 , 0)
    Points.sort()
    return [p1] + Points + [p2]
```

BAB III
Hasil Eksekusi Program

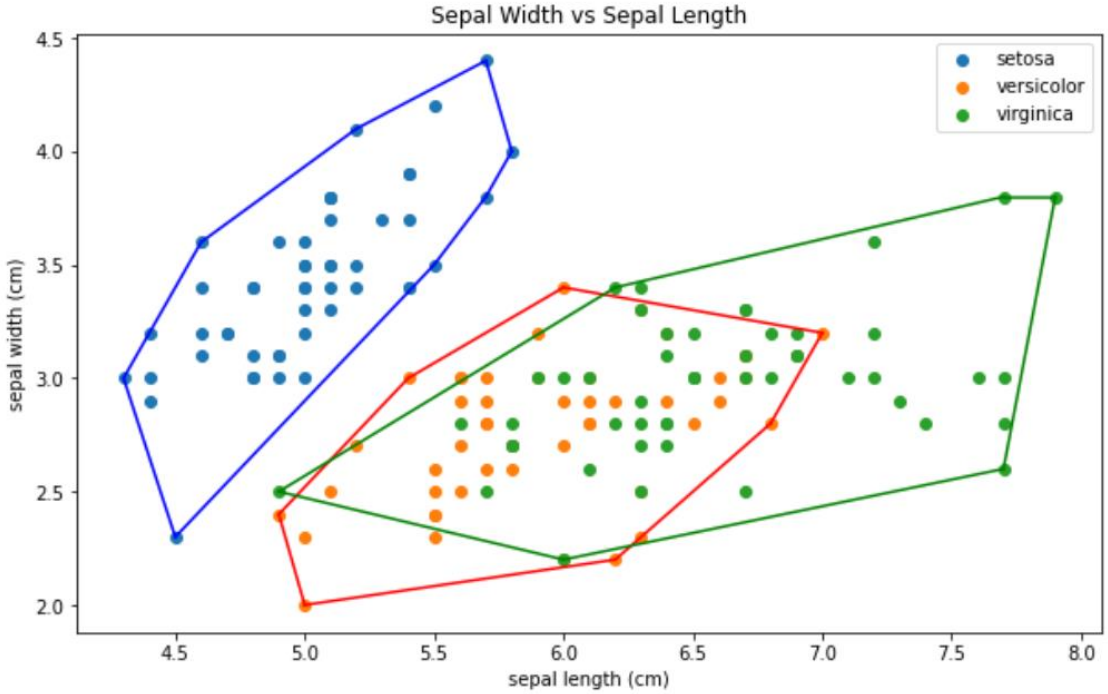
Berikut merupakan hasil eksekusi program terhadap beberapa dataset yang tersedia dari pustaka *scikit-learn*.

a. Dataset Iris

1. Petal Width vs Petal Length

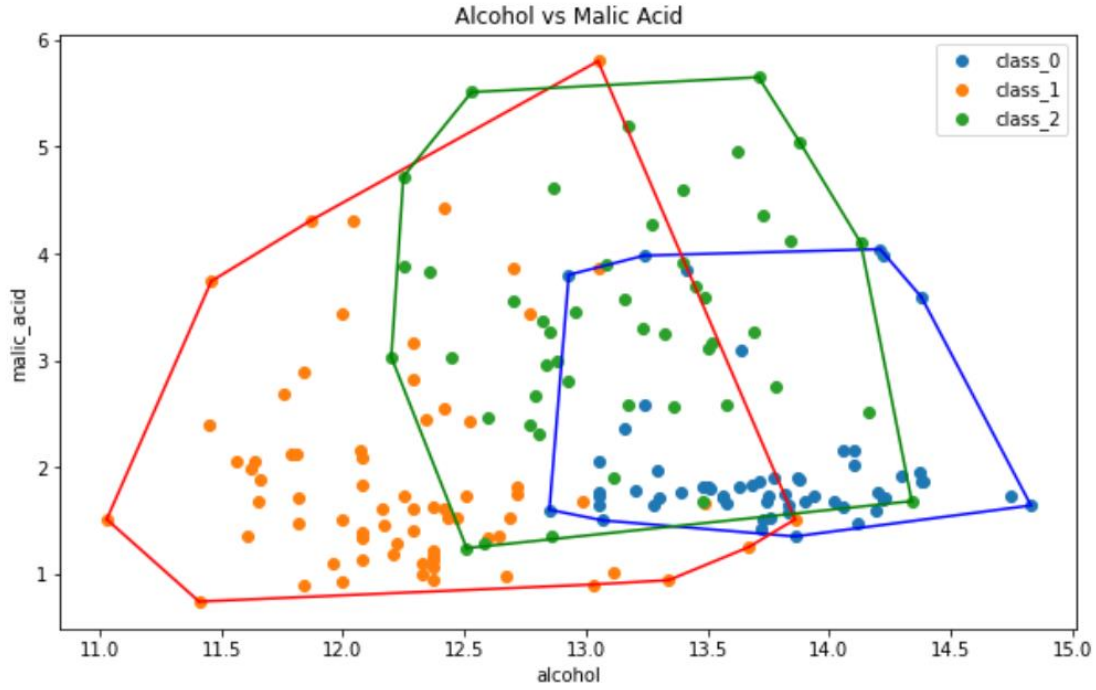


2. Sepal Width vs Sepal Length

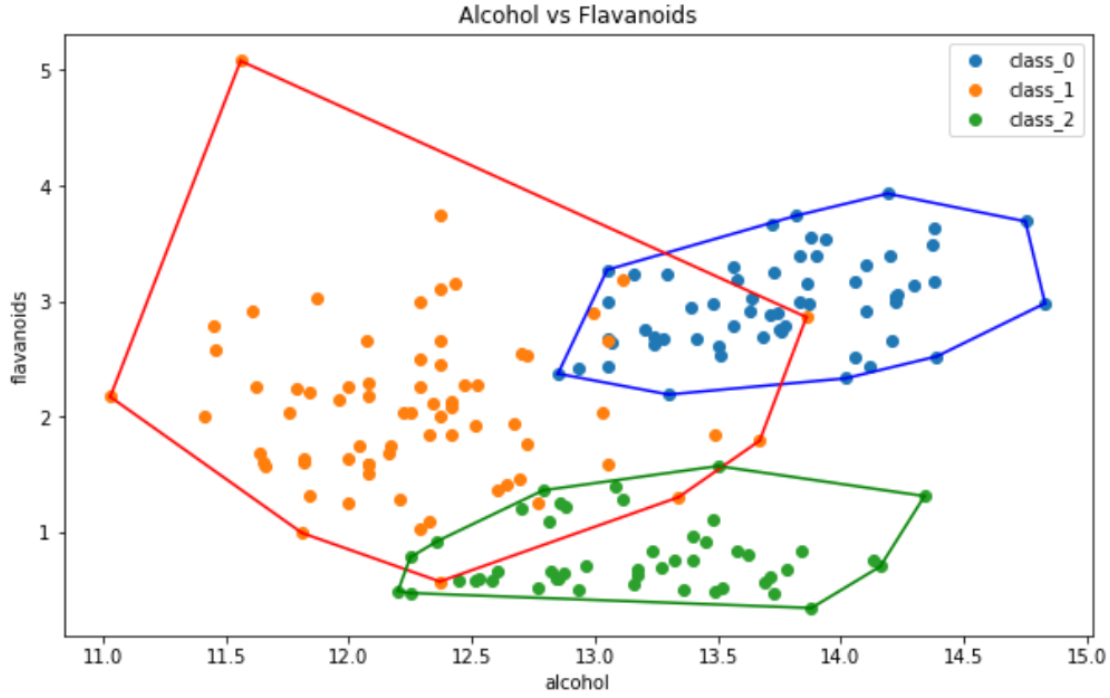


b. Dataset Wine

1. Alcohol vs Malic Acid

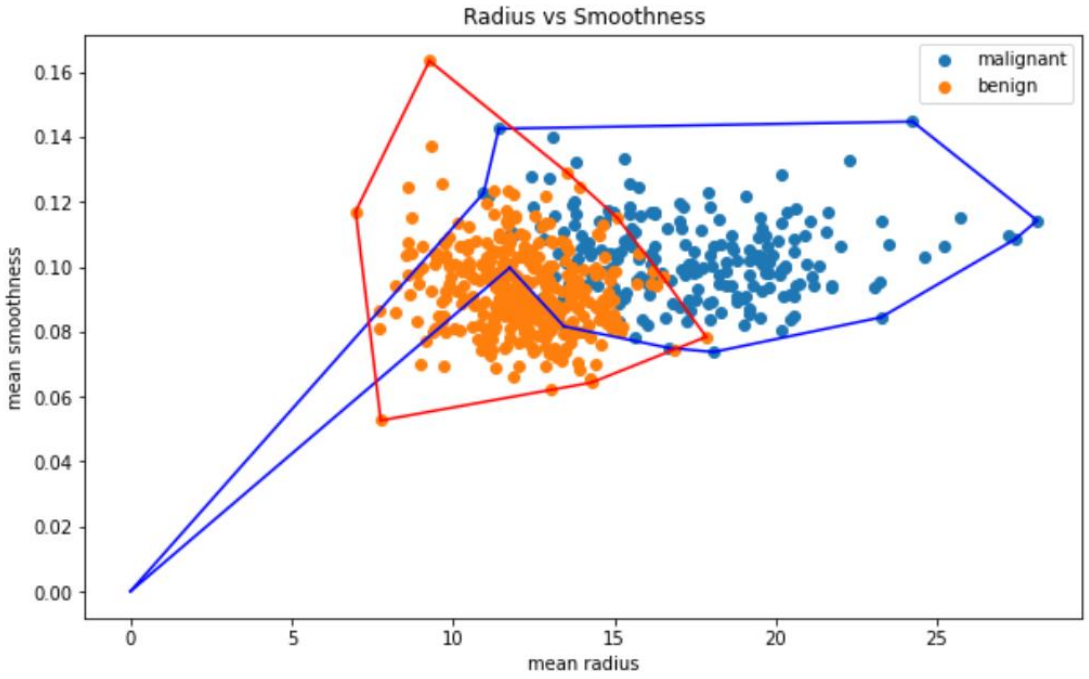


2. Alcohol vs Flavanoids

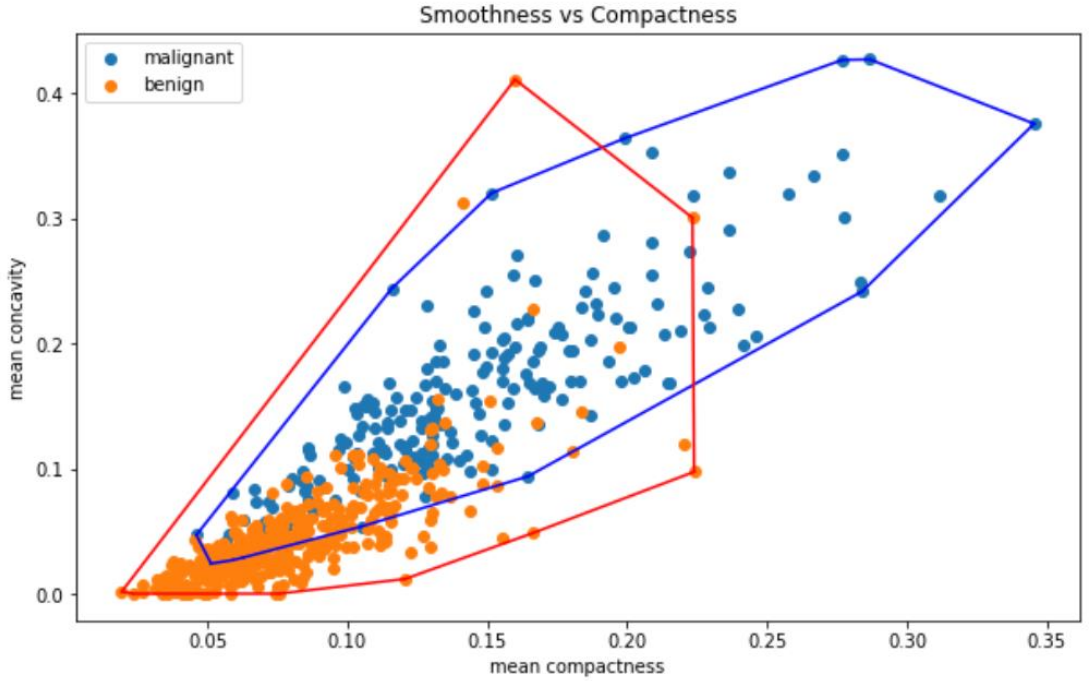


c. Dataset Breast_Cancer

3. Radius vs Smoothness



4. Smoothness vs Compactness



Tabel Penilaian

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.		✓