LAPORAN TUGAS KECIL II

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

Laporan dibuat untuk memenuhi salah satu tugas mata kuliah

IF2211 Strategi Algoritma

Disusun oleh:

Dimas Shidqi Parikesit 13520087

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

DAFTAR ISI

Algoritma Divide and Conquer

Pada kasus ini digunakan algoritma quickhull yang berbasis divide and conquer. Algoritma ini terdiri dari beberapa bagian

1. Membagi kumpulan titik menjadi dua bagian

   Dalam melakukan pembagian ini, digunakan garis pembagi yang melewati titik yang berada di paling kiri dan titik yang berada di paling kanan.

2. Membagi himpunan titik berdasar posisi terhadap garis pembagi

   Himpunan titik kemudian dibagi berdasarkan posisi titik terhadap garis tersebut, apakah berada di kiri atau di kanan garis. Pengecekan ini dilakukan dengan menghitung determinan matriks antara titik kiri, titik kanan, dan titik yang dicek.

3. Mencari titik terjauh dari garis pembagi

   Apabila himpunan titik di kiri dan kanan garis pembagi tidak kosong, dicari titik yang letaknya terjauh dari garis pembagi. Penghitungan jarak menggunakan algoritma penghitungan jarak dari titik ke garis yang dibentuk oleh dua titik.

4. Mencari sisa titik yang berada di luar segitiga yang dibentuk garis pembagi dengan titik terjauh

   Dapat dibentuk segitiga antara titik terjauh dengan garis pembagi. Kemudian sisa titik pada himpunan dicek apakah berada di dalam atau diluar segitiga tersebut. Apabila titik di dalam segitiga, maka titik tersebut tidak perlu dicek karena pasti bukan merupakan bagian dari convex hull.

5. Mengulang nomor 3 sampai 4 untuk titik di luar segitiga secara rekursif

   Apabila himpunan titik di luar segitiga tidak kosong, maka dilakukan algoritma bagian 3 dan 4 dengan garis pembagi berupa garis antara titik garis pembagi sebelumnya dengan titik terjauh dan himpunan. Ini adalah implementasi divide and conquer pada algoritma ini.

6. Gabungkan titik yang menjadi convex hull

   Apabila himpunan titik di luar segitiga kosong, berarti sudah tidak ada titik untuk dicek apakah termasuk dalam convex hull. Oleh karena itu, semua titik yang sudah ditemukan perlu digabungkan untuk membentuk convex hull yang utuh. Ini adalah implementasi penggabungan pada algoritma divide and conquer

Kode Program

myConvexHull.py

```python
def scoreChecker(el1, el2, el3):
    # Penggunaan determinan matriks untuk mengecek posisi titik dari garis yang melewati dua titik
    x1 = el1[0]
    y1 = el1[1]
    x2 = el2[0]
    y2 = el2[1]
    x3 = el3[0]
    y3 = el3[1]
    return x1*y2 + x3*y1 + x2*y3 - x3*y2 - x2*y1 - x1*y3

def distance(l1, l2, p):
    # Algoritma penghitungan jarak titik terhadap garis
    return abs(((l2[0]-l1[0])*(l1[1]-p[1])) - ((l1[0]-p[0])*(l2[1]-l1[1]))) / (((l2[0]-l1[0])**2 + (l2[1]-l1[1])**2)**0.5)

def triangleArea(p1, p2, p3):
    # Rumus luas segitiga
    return abs((p1[0] * (p2[1] - p3[1]) + p2[0] * (p3[1] - p1[1]) + p3[0] * (p1[1] - p2[1])) / 2.0)

def inside(p1,p2,p3,p):
    # Mengecek apakah titik p berada di dalam segitiga yang dibentuk oleh titik p1, p2, p3
    a = triangleArea(p1,p2,p3)
    a1 = triangleArea(p,p2,p3)
    a2 = triangleArea(p1,p,p3)
    a3 = triangleArea(p1,p2,p)

    if(a == a1 + a2 + a3):
        return True
    else:
        return False

def subConvexHull(point, idxleft, idxright, array, subarr):
    # Perhitungan Sub Convex Hull
    idxMax = subarr[0]
    distleftMax = distance(array[idxleft], array[idxright], array[idxMax])

    for i in range(len(subarr)):
        if(distance(array[idxleft], array[idxright], array[subarr[i]]) > distleftMax):
            idxMax = subarr[i]
            distleftMax = distance(array[idxleft], array[idxright], array[idxMax])

    if [idxleft, idxright] in point:
        point.remove([idxleft, idxright])
    point.append([idxleft, idxMax])
    point.append([idxMax, idxright])

    subarr.remove(idxMax)

```

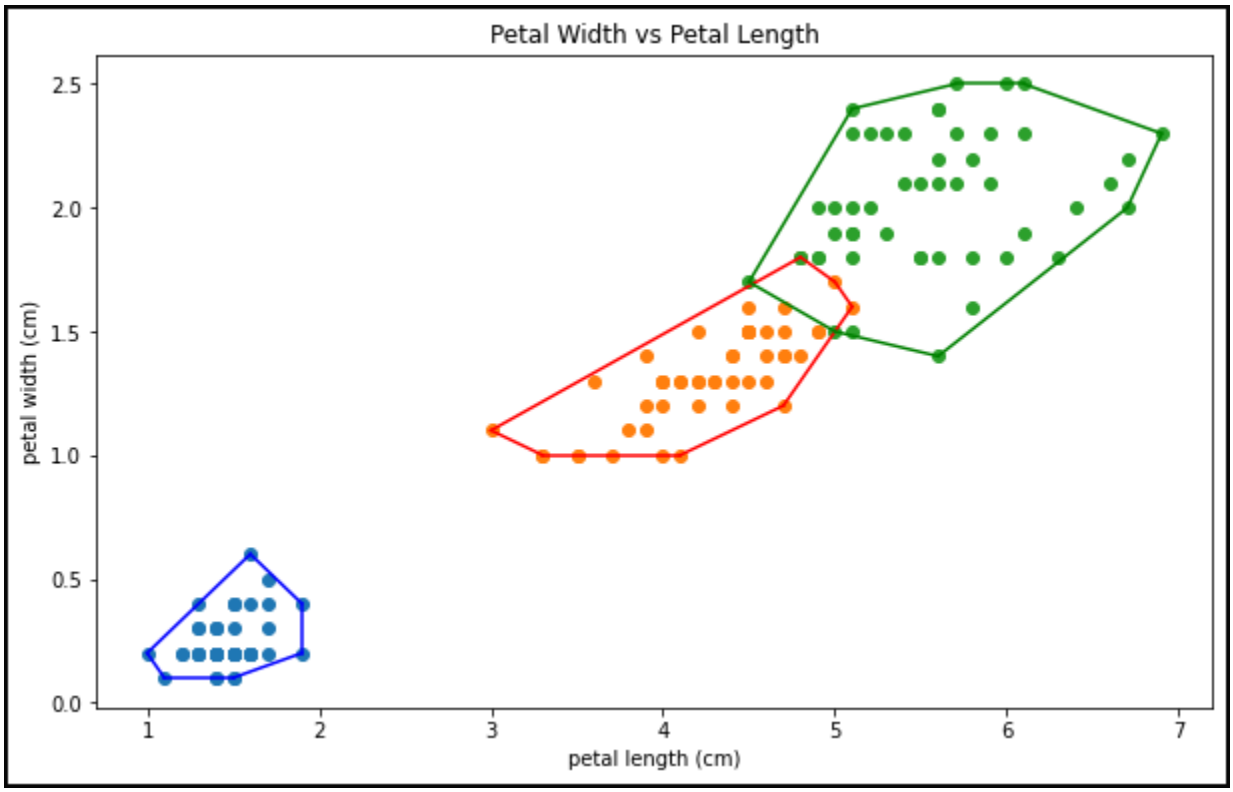
```
    for item in subarr:
        if(inside(array[idxleft], array[idxright], array[idxMax], array[item])):
            subarr.remove(item)

    outLeft = []
    outRight = []

    for i in range(len(subarr)):
        scoreLeft = scoreChecker(array[idxleft], array[idxMax], array[subarr[i]])
        scoreRight = scoreChecker(array[idxMax], array[idxright], array[subarr[i]])
        if(scoreLeft > 0):
            outLeft.append(subarr[i])

        if(scoreRight > 0):
            outRight.append(subarr[i])

    if len(outLeft)>0:
        subConvexHull(point, idxleft, idxMax, array, outLeft)

    if len(outRight)>0:
        subConvexHull(point, idxMax, idxright, array, outRight)

def myConvexHull(array):
    # Pembuatan array untuk menyimpan himpunan titik
    point = []

    # Cari titik paling kiri dan kanan sebagai garis pembagi
    idxleft = 0
    idxright = 0
    for i in range(len(array)):
        if(array[i][0] < array[idxleft][0]):
            idxleft = i
        if(array[i][0] > array[idxright][0]):
            idxright = i
    point.append([idxleft, idxright])

    # Penggolongan posisi titik terhadap garis pembagi
    arrLeft = []
    arrRight = []
    for i in range(len(array)):
        if(i != idxleft and i != idxright):
            score = scoreChecker(array[idxleft], array[idxright], array[i])
            if(score>0):
                arrLeft.append(i)
            elif(score<0):
                arrRight.append(i)
```

```
94
95      # Pencarian sub convex hull untuk sisi kiri
96      if len(arrLeft)>0:
97          subConvexHull(point, idxleft, idxright, array, arrLeft)
98
99      # Pencarian sub convex hull untuk sisi kanan
100     if len(arrRight)>0:
101         subConvexHull(point, idxright, idxleft, array, arrRight)
102
103     return point
```

Screenshot Input - Output Program

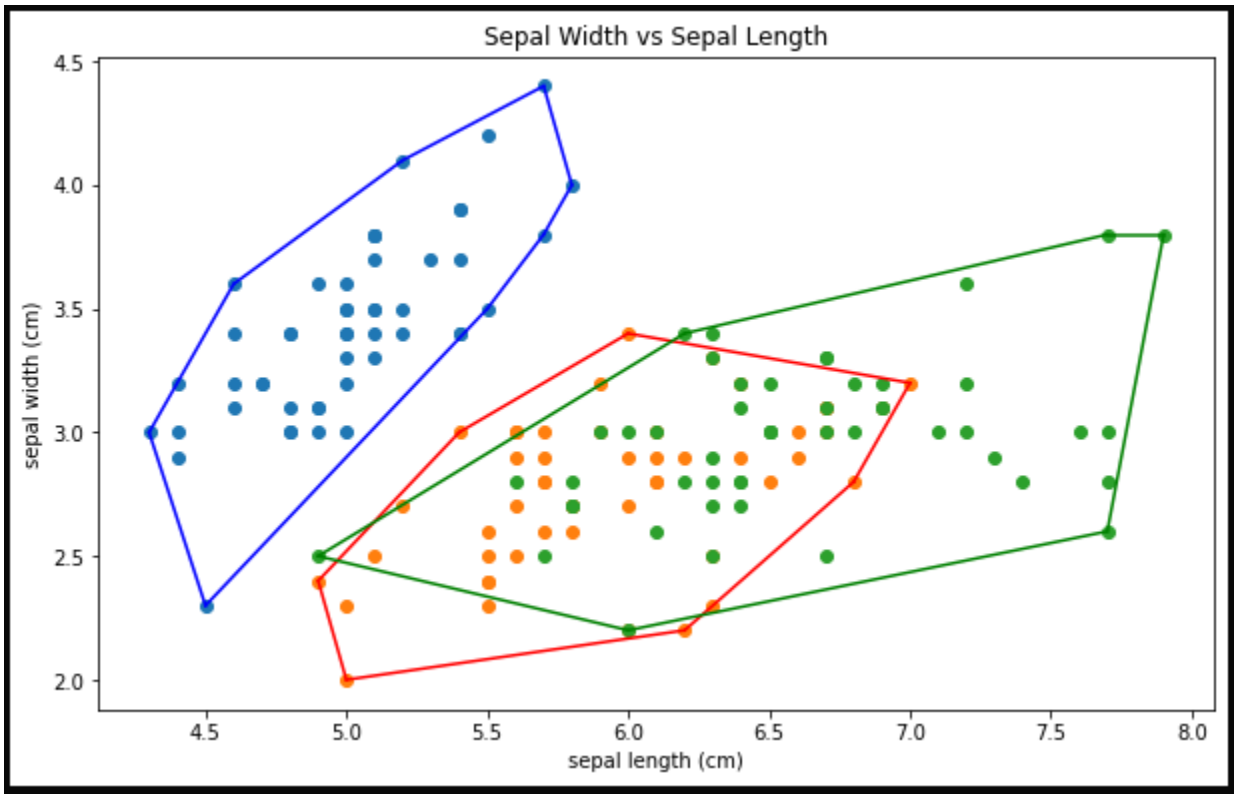
1. Dataset Iris, Petal Width vs Petal Length

```
data = datasets.load_iris()
#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('Petal Width vs Petal Length')
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])

for i in range(len(data.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:, [2, 3]].values

  newhull = myConvexHull(bucket)
  # print(newhull)

  plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
  for simplex in newhull:
    plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])

  # hull = ConvexHull(bucket)
  # for simplex in hull.simplices:
  #  plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.show()
```



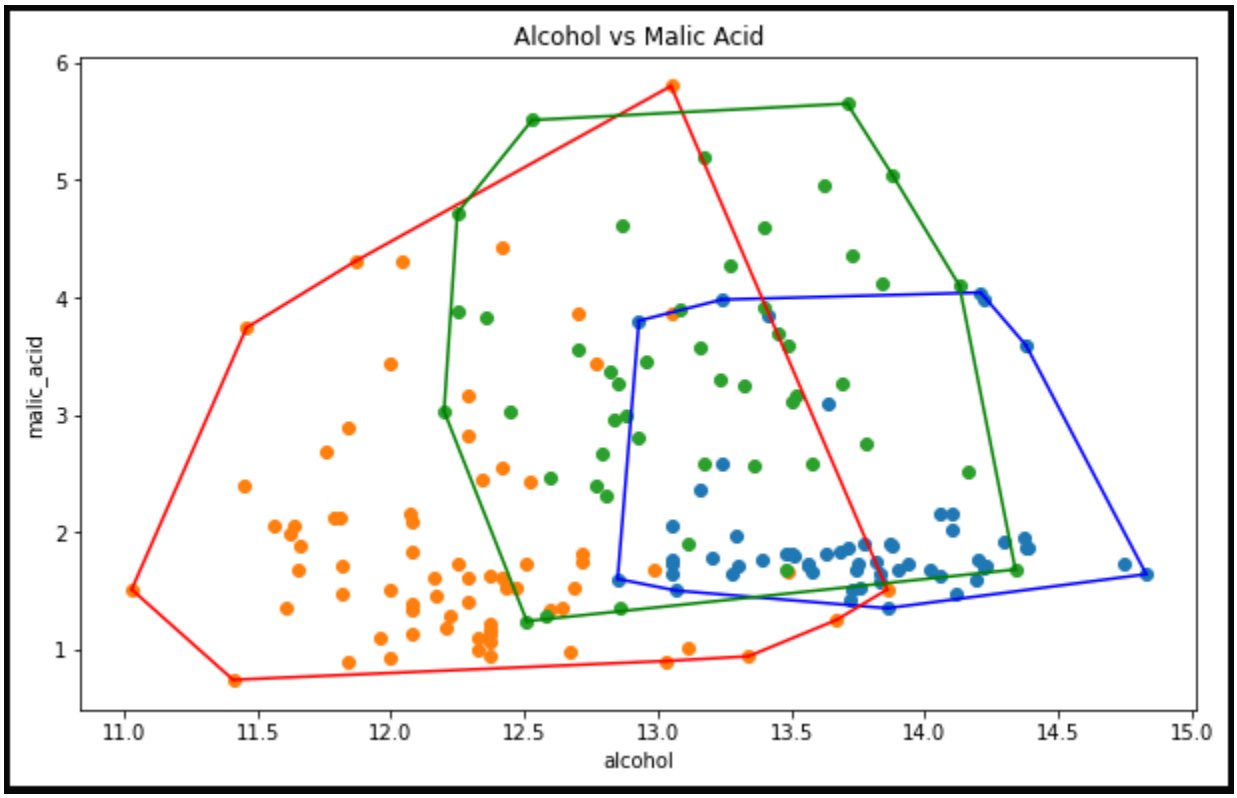
2. Dataset Iris, Sepal Width vs Sepal Length

```
data = datasets.load_iris()
#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('Sepal Width vs Sepal 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

  newhull = myConvexHull(bucket)
  # print(newhull)

  plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
  for simplex in newhull:
    plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])

  # hull = ConvexHull(bucket)
  # for simplex in hull.simplices:
  #  plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.show()
```



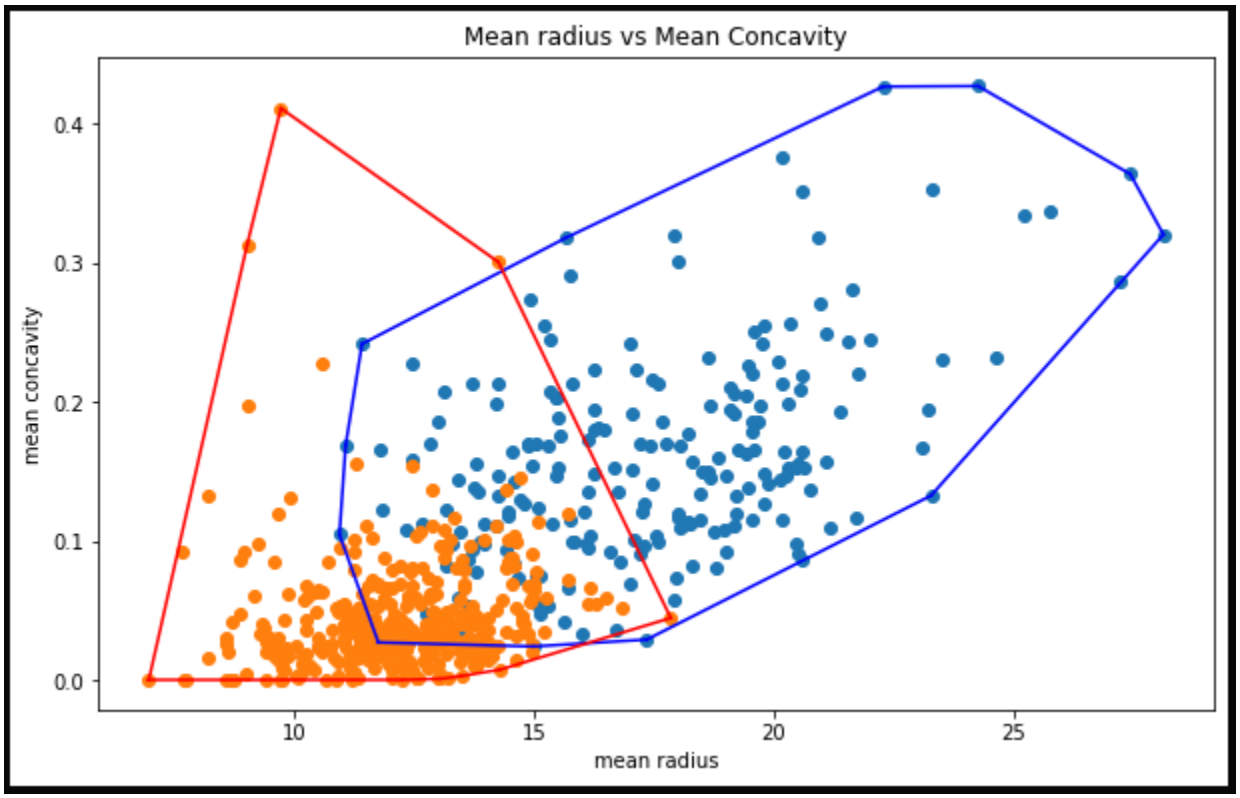
3. Dataset Wine, Alcohol vs Malic Acid

```
data = datasets.load_wine()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
df

# visualisasi hasil ConvexHull
plt.figure(figsize=(10, 6))
colors = ['b', 'r', 'g']
plt.title('Alcohol vs Malic Acid')
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

  newhull = myConvexHull(bucket)
  # print(newhull)

  plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
  for simplex in newhull:
    plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])

  # hull = ConvexHull(bucket)
  # for simplex in hull.simplices:
  #  plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.show()
```



4. Dataset Breast Cancer, Mean Radius vs Mean Concavity

```
data = datasets.load_breast_cancer()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
df

# visualisasi hasil ConvexHull
plt.figure(figsize=(10, 6))
colors = ['b', 'r', 'g']
plt.title('Mean radius vs Mean Concavity')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[6])

for i in range(len(data.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:, [0, 6]].values

  newhull = myConvexHull(bucket)
  # print(newhull)

  plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
  for simplex in newhull:
    plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])

  # hull = ConvexHull(bucket)
  # for simplex in hull.simplices:
  #  plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.show()
```


Mean radius vs Mean Concavity
mean concavity
mean radius
0.0
0.1
0.2
0.3
0.4
10
15
20
25

Link Kode Program

Repository Github : https://github.com/dParikesit/Tucil2_STIMA

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	✔	

Daftar Referensi

1. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf