LAPORAN TUGAS KECIL II

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

Laporan Ini Dibuat untuk Memenuhi Tugas Perkuliahan

Mata Kuliah Strategi Algoritma (IF2211)

KELAS 01

Dosen : Dr. Masayu Leylia Khodra, S.T., M.T.

Rio Alexander Audino / 13520088

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER II TAHUN 2021/2022

I. Algoritma Divide and Conquer

Program yang kali ini saya buat adalah implementasi convex hull untuk visualisasi tes linear separability dataset dengan algoritma divide and conquer. Pencarian Convex Hull dilakukan menggunakan strategi Divide and Conquer. Secara umum, terdapat tiga langkah utama dalam penerapan strategi divide and conquer, tiga langkah tersebut yaitu:

a. Mencari dua titik ekstrem yang membagi dua dataset
b. Melakukan pembagian dataset berdasarkan garis yg terbentuk
c. Melakukan pembagian sampai semua daerah berada di dalam convex hull

Dengan melakukan pembagian dataset menjadi dua atau beberapa bagian, program tidak perlu mengecek semua titik satu-persatu (tidak dengan brute force). Tiap titik-titik terujung yang terpilih akan mencipatkan suatu convex hull yang melingkupi semua titik dataset. Berikut adalah langkah-langkah dari implementasi Divide and Conquer,

d. Mengambil dua titik paling ujung kanan dan ujung kiri
e. Membuat sebuah garis pemisah yang menghubungkan titik ujung kanan dan titik ujung kiri
f. Membagi titik-titik pada dataset menjadi dua bagian yang dipisahkan oleh garis sebelumnya
g. Dari setiap bagian terbentuk, mencari titik terjauh dari garis pembatas. Titik terjauh ini akan menjadi titik convex hull baru.
h. Menghubungkan garis ujung kanan dan ujung kiri dengan titik terjauh (Titik ujung kanan dengan titik terjauh serta titik ujung kiri dengan titik terjauh), sehingga terbentuk sebuah segitiga dari titik ujung kanan, titik ujung kiri, dan titik terjauh.
i. Jika terdapat dua titik terjauh dengan jarak yang sama, dipilih titik yang membentuk sudut terbesar
j. Kembali melakukan langkah c pada titik-titik di luar segitiga
k. Iterasi terus dilakukan sampai tidak ada titik di luar segitiga
l. Setiap titik terujung/titik terjauh yang didapat akan menjadi titik convex

Berikut adalah visualisasi dari implementasi strategi Divide and Conquer pada program pencarian Convex Hull,

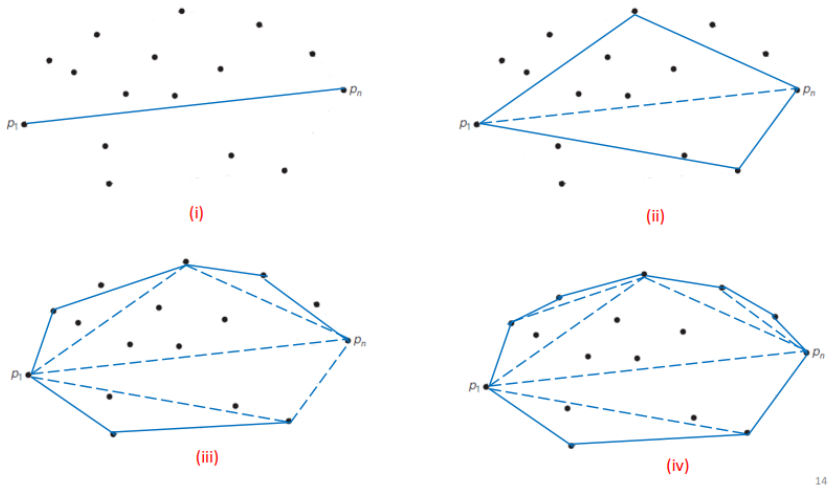


Figure 1 Visualisasi Implementasi Divide and Conquer pada Algoritma Pencarian Convex Hull

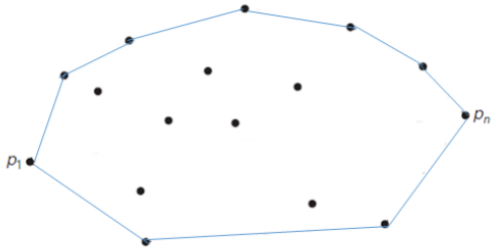
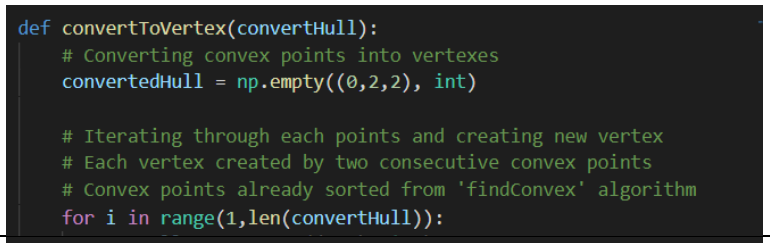
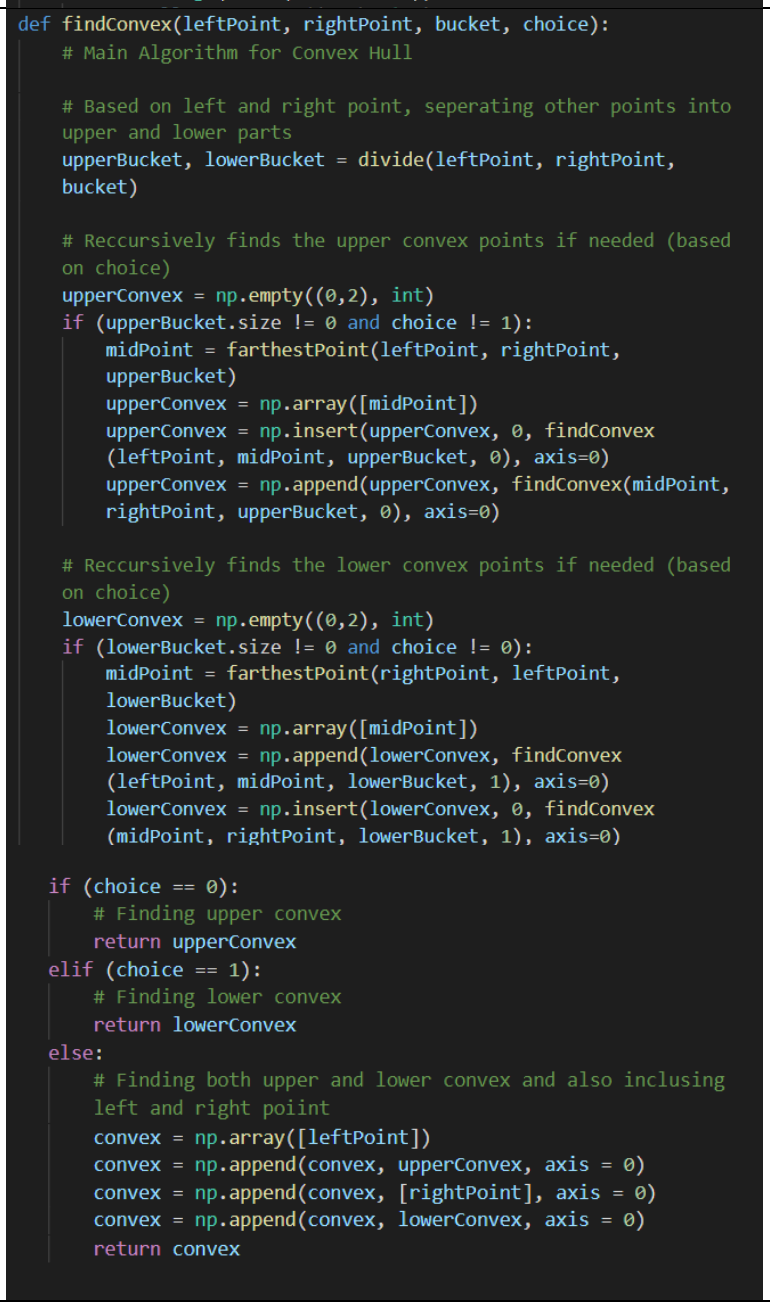
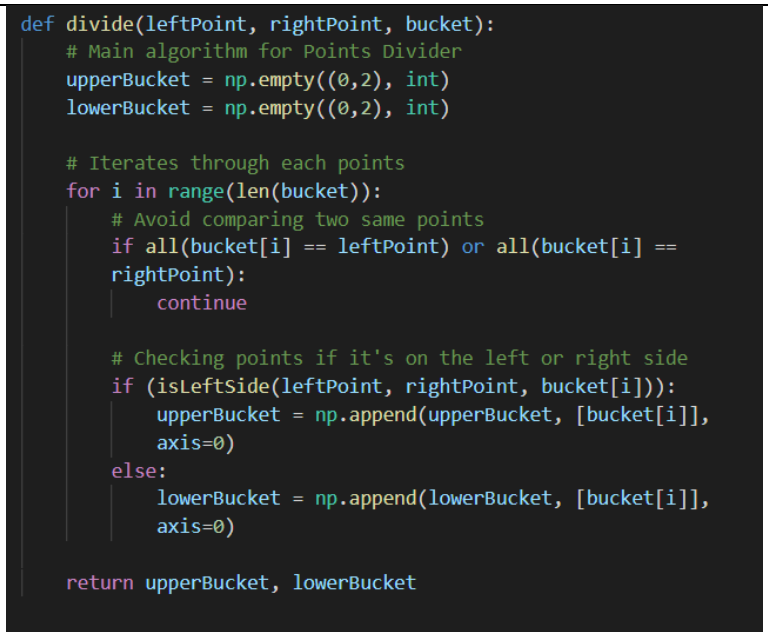
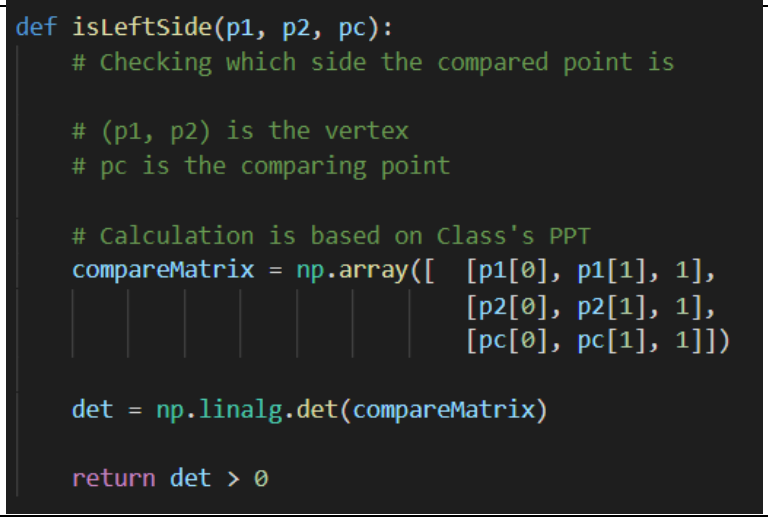


Figure 2 Visualisasi hasil akhir algoritma pencarian Convex Hull

II. Kode Program

file	Skrinshut	Deskripsi Kode
ConvexHull.py	`def main():` `  # CHOOSE DATASET` `  data = datasets.load_wine()` `  # Converting to pandas dataframe` `  df = pd.DataFrame(data.data, columns=data.feature_names)` `  df['Target'] = pd.DataFrame(data.target)` `  # Choosing features & Visualization` `  choosed_feature = choose_feature(data)`	main adalah fungsi yang menjadi badan utama program. Di dalam fungsi main terdapat pemanggilan fungsi lainnya seperti, meng-load dataset, pemilihan atribut, serta visualisasi hasil.
	`def choose_feature(data):` `  choosed_feature = [0,0]` `  while True:` `  print("Choose two feature: ")` `  for i, name in enumerate(data.feature_names):`	choose_feature adalah fungsi pemilihan atribut dari sebuah dataset yang akan dicari convex hull-nya.
	`def visualize(df, data, features):` `  #visualisasi hasil ConvexHull` `  plt.figure(figsize = (10, 6))` `  plt.title(f'{data.feature_names[features[0]]} vs {data.feature_names [features[1]]}')` `  plt.xlabel(data.feature_names[features[0]])` `  plt.ylabel(data.feature_names[features[1]])` `  for i in range(len(data.target_names)):` `  bucket = df[df['Target'] == i]`	visualize adalah fungsi yang melakukan pencarian convex hull dari sebuah dataset dan memvisualisasikannya. Proses pencarian convex hull di-handle dengan fungsi tersendiri.
	`def convex_hull(bucket):` `  return ConvexHullRio.main(bucket)`	convex_hull adalah fungsi yang meng-handle proses pencarian convex hull. Fungsi convex_hull memanggil fungsi dari library ConvexHullRio.
ConvexHullRio.py	`def main(bucket):` `  # Main Algorithm` `  print("\nFinding ConveExes...")` `  bucket = bucket[bucket[:, 0].argsort()]` `  convex = findConvex(bucket[0], bucket[-1], bucket, 2)` `  print("Finished")` `  return convertToVertex(convex)`	main adalah badan utama dalam program pencarian titik convex hull. Di dalam main terdapat fungsi lain seperti, fungsi pencarian titik convex dan fungsi konversi titik konvex menjadi grais-garis convex.

		convertToVertex adalah fungsi yang mengkonversi titik-titik convex yang telah dicari sebelumnya, menjadi garis-garis convex agar dapat divisualisasikan.
ConvexFinder.py		findConvex adalah fungsi utama dalam pencarian titik-tititk convex. Di dalam fungsi findConvex terjadi penerapan strategi Divide n Conquer. Proses rekursifitas pada strategi Divide n Conquer juga terjadi pada fungsi ini.

DivideAndConquer.py		divide adalah fungsi yang membagi dataset ke dalam dua buah array. Pembagian dataset didasarkan garis pemisah yang dibentuk oleh lelftPoint dan rightPoint. Di dalam fungsi divide terjadi pemanggilan isLeftSide yang menentukan posisi sebuah titik terhadap suatu garis.
		isLeftSide adalah fungsi yang mengecek posisi sebuah titik terhadap sebuah garis. Fungsi ini memanfaatkan perkalian matriks yang telah dijelaskan di kelas.

	`` def farthestPoint(leftPoint, rightPoint, bucket): # Finding the farthest point from vertex (leftPoint, rightPoint) # Initiating Values maxDist = 0 maxAngle = 0 iConvexPoints = 0 # Iterting through each points for i in range(len(bucket)): # Calculating distance using cross product dist = np.linalg.norm(np.cross(rightPoint-leftPoint, leftPoint-bucket[i]))/np.linalg.norm(rightPoint-leftPoint) # Comparing the distances if (maxDist < dist): # Calculating angle using cosine rules cosineAngle = 0.5 * np.dot(bucket[i] - leftPoint, rightPoint-bucket[i]) / (np.linalg.norm(bucket[i] - leftPoint) * np.linalg.norm(rightPoint-bucket[i])) maxAngle = np.arccos(cosineAngle) maxDist = dist iConvexPoints = i # Comparing angles elif (maxDist == dist): # Calculating angle using cosine rule cosineAngle = 0.5 * np.dot(bucket[i] - leftPoint, rightPoint-bucket[i]) / (np.linalg.norm(bucket[i] - leftPoint) * np.linalg.norm(rightPoint-bucket[i])) angle = np.arccos(cosineAngle) if (maxAngle < angle): iConvexPoints = i maxAngle = angle return bucket[iConvexPoints] ```	farthestPoint adalah fungsi yang mencari titik terjauh dari sebuah garis. Perhitungan jarak dilakukan menggunakan cross product. Jika terdapat jarak terjauh yang sama, diambil titik yang membentuk sudut terbesar.

III. Skrinshut Input-Output Program

a. Pasangan Atribut (petal-length, petal-width)

Input:

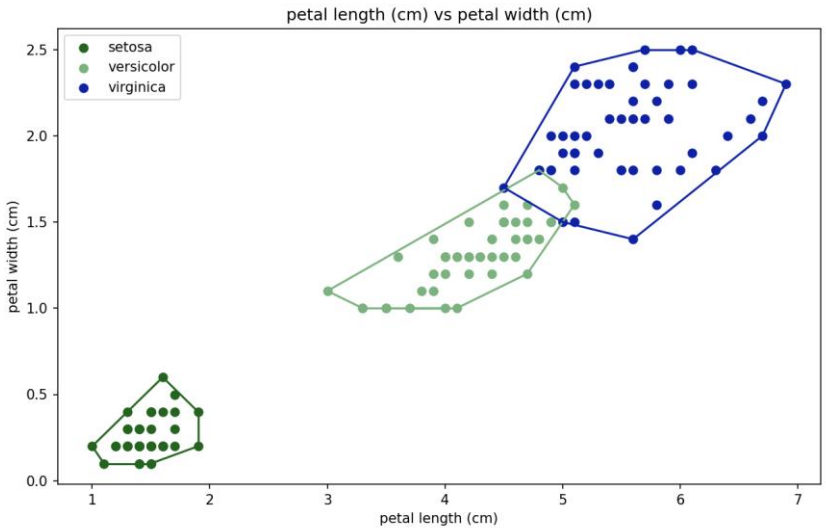
```
Choose two feature:
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)
Feature-x : 3
Feature-y : 4

Finding ConveExes...
Finished

Finding ConveExes...
Finished

Finding ConveExes...
Finished
PS C:\Users\rioau\Documents\ITB\2Tingkat 2\Sem2\Tugas\STIMA\Tucil 2>
```

Output:



b. Pasangan Atribut (sepal-length, sepal-width)

Input:

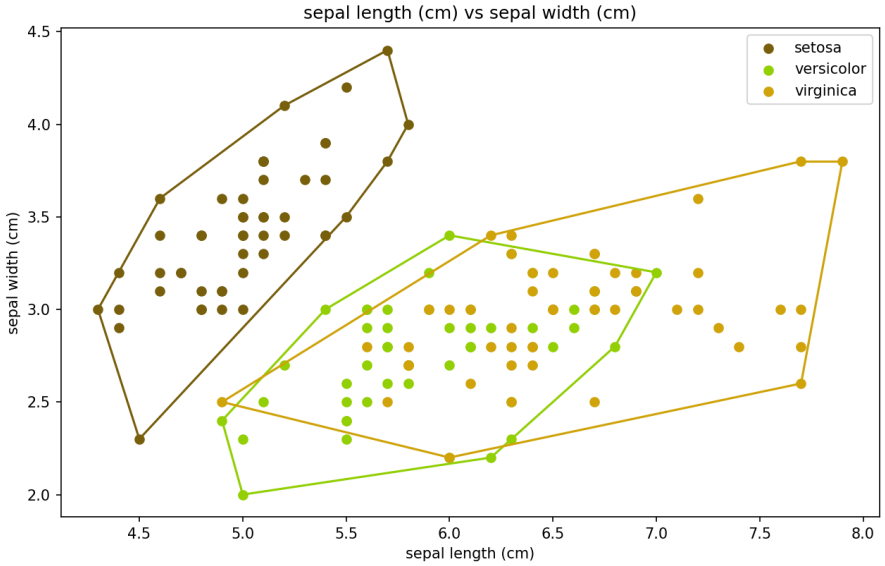
```
Choose two feature:
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)
Feature-x : 1
Feature-y : 2

Finding ConveExes...
Finished

Finding ConveExes...
Finished

Finding ConveExes...
Finished
PS C:\Users\rioau\Documents\ITB\2Tingkat 2\Sem2\Tugas\STIMA\Tucil 2>
```

Output:



c. Pasangan Atribut (alchohol vs color_intensity)

Input:

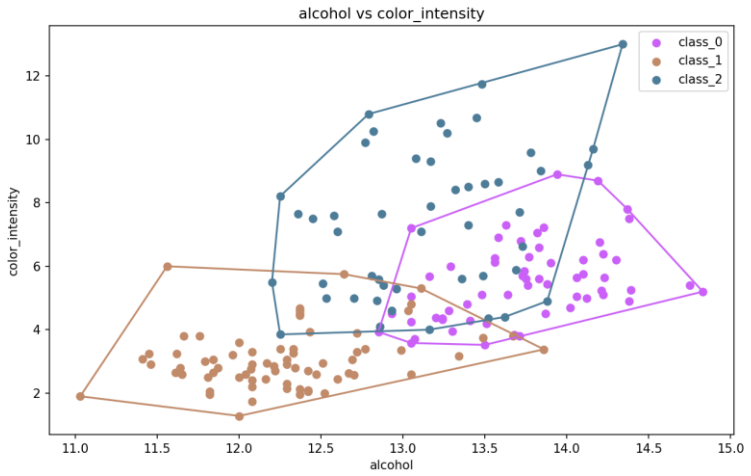
```
Choose two feature:
1. alcohol
2. malic_acid
3. ash
4. alcalinity_of_ash
5. magnesium
6. total_phenols
7. flavanoids
8. nonflavanoid_phenols
9. proanthocyanins
10. color_intensity
11. hue
12. od280/od315_of_diluted_wines
13. proline
Feature-x : 1
Feature-y : 10

Finding ConveExes...
Finished

Finding ConveExes...
Finished

Finding ConveExes...
Finished
PS C:\Users\rioau\Documents\ITB\2Tingkat 2\Sem2\Tugas\STIMA\Tucil2_13520088>
```

Output:



d. Pasangan Atribut (malic_acid vs total_phenols)

Input:

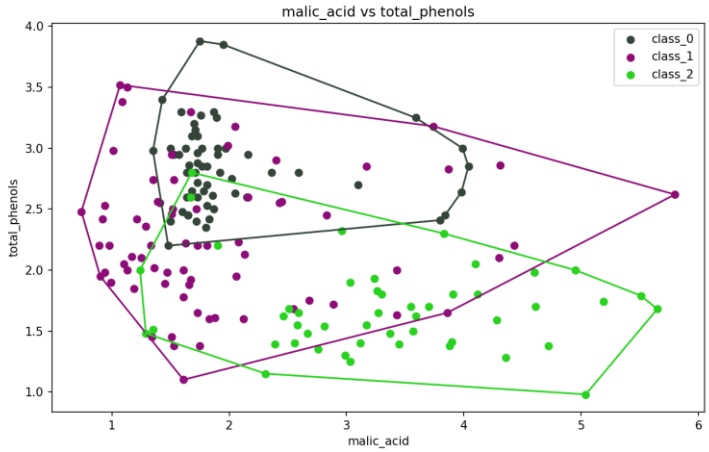
```
Choose two feature:
1. alcohol
2. malic_acid
3. ash
4. alcalinity_of_ash
5. magnesium
6. total_phenols
7. flavanoids
8. nonflavanoid_phenols
9. proanthocyanins
10. color_intensity
11. hue
12. od280/od315_of_diluted_wines
13. proline
Feature-x : 2
Feature-y : 6

Finding ConveExes...
Finished

Finding ConveExes...
Finished

Finding ConveExes...
Finished
PS C:\Users\rioau\Documents\ITB\2Tingkat 2\Sem2\Tugas\STIMA\Tucil2_13520088>
```

Output:



IV. Source Code Program

Link Github : https://github.com/Audino723/Tucil2_13520088