

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:

- Mencari dua titik ekstrem yang membagi dua dataset
- Melakukan pembagian dataset berdasarkan garis yg terbentuk
- 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 menciptakan suatu convex hull yang melingkupi semua titik dataset. Berikut adalah langkah-langkah dari implementasi Divide and Conquer,

- Mengambil dua titik paling ujung kanan dan ujung kiri
- Membuat sebuah garis pemisah yang menghubungkan titik ujung kanan dan titik ujung kiri
- Membagi titik-titik pada dataset menjadi dua bagian yang dipisahkan oleh garis sebelumnya
- Dari setiap bagian terbentuk, mencari titik terjauh dari garis pembatas. Titik terjauh ini akan menjadi titik convex hull baru.
- 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.
- Jika terdapat dua titik terjauh dengan jarak yang sama, dipilih titik yang membentuk sudut terbesar
- Kembali melakukan langkah c pada titik-titik di luar segitiga
- Iterasi terus dilakukan sampai tidak ada titik di luar segitiga
- 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,

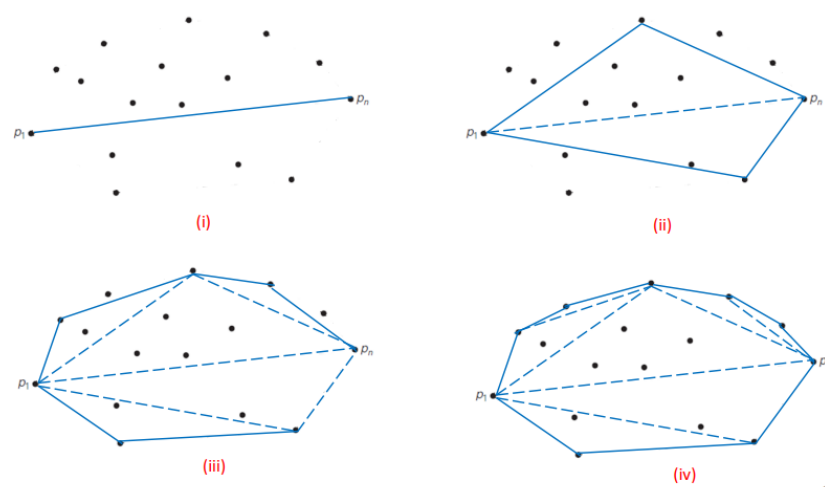


Figure 1 Visualisasi Implementasi Divide and Conquer pada Algoritma Pencarian Convex Hull

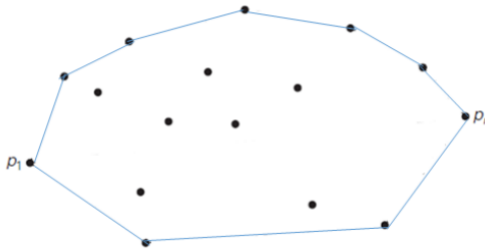


Figure 2 Visualisasi hasil akhir algoritma pencarian Convex Hull

## II. Kode Program

file	Skrinshut	Deskripsi Kode
ConvexHull.py	<pre>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 &amp; Visualization     choosed_feature = choose_feature(data)</pre>	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.
	<pre>def choose_feature(data):     choosed_feature = [0,0]     while True:         print("Choose two feature: ")         for i, name in enumerate(data.feature_names):             print(i, name, "\n")</pre>	choose_feature adalah fungsi pemilihan atribut dari sebuah dataset yang akan dicari convex hull-nya.
	<pre>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]         bucket = bucket.iloc[:, [features[0], features[1]]].values</pre>	visualize adalah fungsi yang melakukan pencarian convex hull dari sebuah dataset dan memvisualisasikannya. Proses pencarian convex hull di-handle dengan fungsi tersendiri.
	<pre>def convex_hull(bucket):     return ConvexHullRio.main(bucket)</pre>	convex_hull adalah fungsi yang meng-handle proses pencarian convex hull. Fungsi convex_hull memanggil fungsi dari library ConvexHullRio.
ConvexHullRio.py	<pre>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)</pre>	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.

	<pre> def convertToVertex(convertHull):     # Converting convex points into vertexes     convertedHull = np.empty((0,2,2), int)      # Iterating through each points and creating new vertex     # Each vertex created by two consecutive convex points     # Convex points already sorted from 'findConvex' algorithm     for i in range(1,len(convertHull)): </pre>	<p>convertToVertex adalah fungsi yang mengkonversi titik-titik convex yang telah dicari sebelumnya, menjadi garis-garis convex agar dapat divisualisasikan.</p>
ConvexFinder.py	<pre> def findConvex(leftPoint, rightPoint, bucket, choice):     # Main Algorithm for Convex Hull      # Based on left and right point, seperating other points into     upper and lower parts     upperBucket, lowerBucket = divide(leftPoint, rightPoint,     bucket)      # Reccursively finds the upper convex points if needed (based     on choice)     upperConvex = np.empty((0,2), int)     if (upperBucket.size != 0 and choice != 1):         midPoint = farthestPoint(leftPoint, rightPoint,         upperBucket)         upperConvex = np.array([midPoint])         upperConvex = np.insert(upperConvex, 0, findConvex         (leftPoint, midPoint, upperBucket, 0), axis=0)         upperConvex = np.append(upperConvex, findConvex(midPoint,         rightPoint, upperBucket, 0), axis=0)      # Reccursively finds the lower convex points if needed (based     on choice)     lowerConvex = np.empty((0,2), int)     if (lowerBucket.size != 0 and choice != 0):         midPoint = farthestPoint(rightPoint, leftPoint,         lowerBucket)         lowerConvex = np.array([midPoint])         lowerConvex = np.append(lowerConvex, findConvex         (leftPoint, midPoint, lowerBucket, 1), axis=0)         lowerConvex = np.insert(lowerConvex, 0, findConvex         (midPoint, rightPoint, lowerBucket, 1), axis=0)      if (choice == 0):         # Finding upper convex         return upperConvex     elif (choice == 1):         # Finding lower convex         return lowerConvex     else:         # Finding both upper and lower convex and also including         left and right point         convex = np.array([leftPoint])         convex = np.append(convex, upperConvex, axis = 0)         convex = np.append(convex, [rightPoint], axis = 0)         convex = np.append(convex, lowerConvex, axis = 0)         return convex </pre>	<p>findConvex adalah fungsi utama dalam pencarian titik-titik convex. Di dalam fungsi findConvex terjadi penerapan strategi Divide n Conquer. Proses rekursifitas pada strategi Divide n Conquer juga terjadi pada fungsi ini.</p>

DivideAndConquer.py	<pre> def divide(leftPoint, rightPoint, bucket):     # Main algorithm for Points Divider     upperBucket = np.empty((0,2), int)     lowerBucket = np.empty((0,2), int)      # Iterates through each points     for i in range(len(bucket)):         # Avoid comparing two same points         if all(bucket[i] == leftPoint) or all(bucket[i] ==             rightPoint):             continue          # Checking points if it's on the left or right side         if (isLeftSide(leftPoint, rightPoint, bucket[i])):             upperBucket = np.append(upperBucket, [bucket[i]],                 axis=0)         else:             lowerBucket = np.append(lowerBucket, [bucket[i]],                 axis=0)      return upperBucket, lowerBucket </pre>	<p>divide adalah fungsi yang membagi dataset ke dalam dua buah array. Pembagian dataset didasarkan garis pemisah yang dibentuk oleh leftPoint dan rightPoint. Di dalam fungsi divide terjadi pemanggilan isLeftSide yang menentukan posisi sebuah titik terhadap suatu garis.</p>
	<pre> def isLeftSide(p1, p2, pc):     # Checking which side the compared point is      # (p1, p2) is the vertex     # pc is the comparing point      # Calculation is based on Class's PPT     compareMatrix = np.array([ [p1[0], p1[1], 1],                                 [p2[0], p2[1], 1],                                 [pc[0], pc[1], 1]])      det = np.linalg.det(compareMatrix)      return det &gt; 0 </pre>	<p>isLeftSide adalah fungsi yang mengecek posisi sebuah titik terhadap sebuah garis. Fungsi ini memanfaatkan perkalian matriks yang telah dijelaskan di kelas.</p>

```
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. Screenshot Input-Output Program

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

Input:

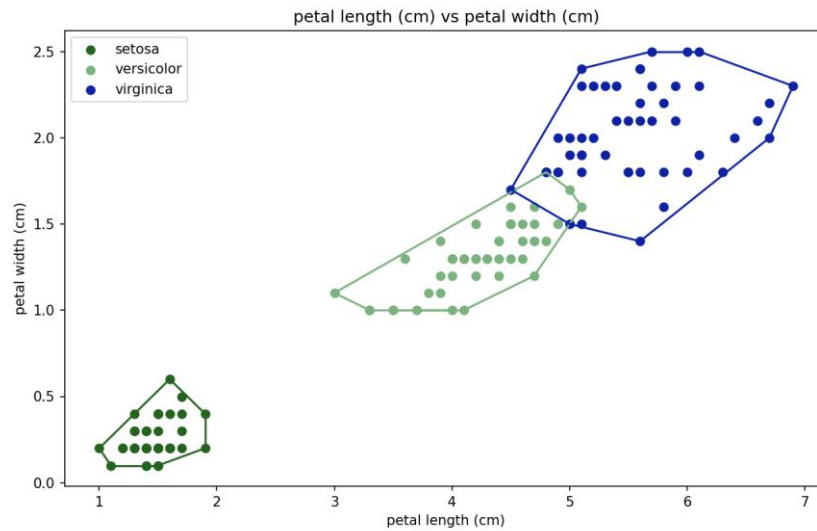
```
Choose two feature:
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)
Feature-x : 3
Feature-y : 4

Finding ConveExes...
Finished

Finding ConveExes...
Finished

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

Output:



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

Input:

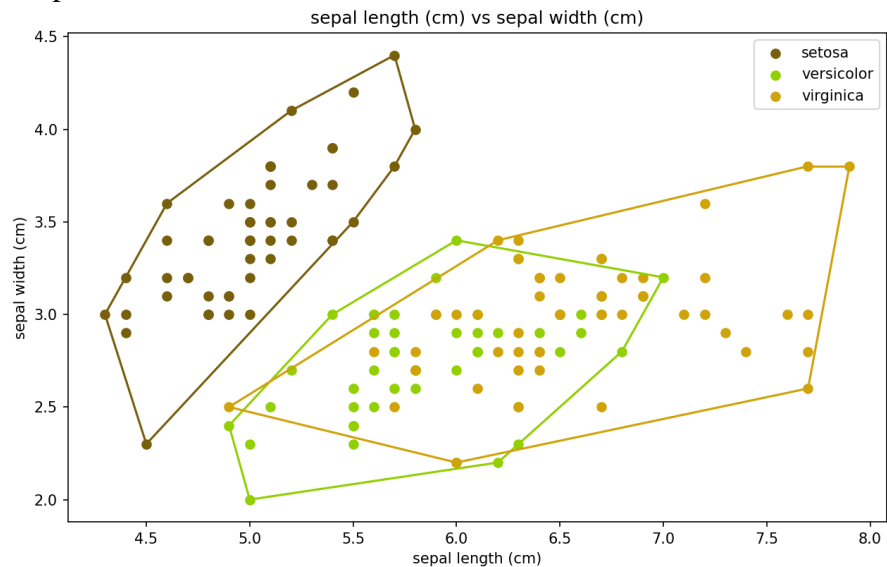
```
Choose two feature:
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)
Feature-x : 1
Feature-y : 2

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 (alcohol vs color\_intensity)

Input:

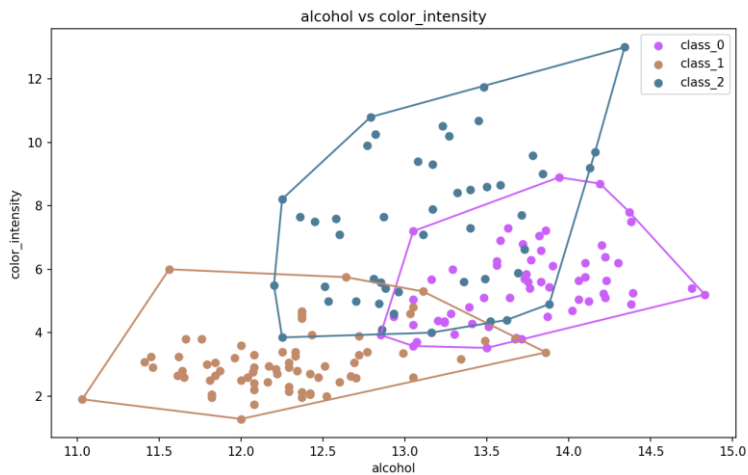
```
Choose two feature:
1. alcohol
2. malic_acid
3. ash
4. alcalinity_of_ash
5. magnesium
6. total_phenols
7. flavanoids
8. nonflavanoid_phenols
9. proanthocyanins
10. color_intensity
11. hue
12. od280/od315_of_diluted_wines
13. proline
Feature-x : 1
Feature-y : 10

Finding ConveExes...
Finished

Finding ConveExes...
Finished

Finding ConveExes...
Finished
PS C:\Users\rloau\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

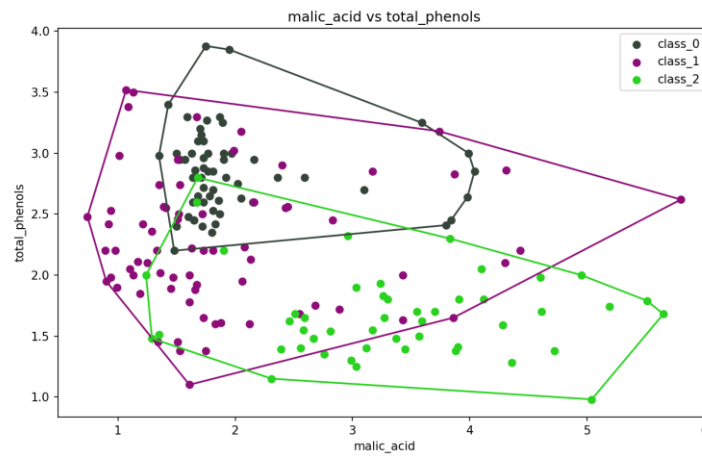
Finding ConveExes...
Finished

Finding ConveExes...
Finished

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

Output:





#### IV. Source Code Program

Link Github : [https://github.com/Audino723/Tucil2\\_13520088](https://github.com/Audino723/Tucil2_13520088)