

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 02

Dosen : Dr. Nur Ulfa Maulidevi, S.T., M.Sc.



DISUSUN OLEH:

Yakobus Iryanto Prasethio (13520104)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER II TAHUN 2021-2022

Daftar Isi

Daftar Isi	1
BAB I	2
BAB II	3
BAB III	6
BAB IV	10
BAB V	10
BAB VI	10

BAB I

Langkah Algoritma *Brute Force*

Deskripsi Langkah Algoritma

Algoritma *Divide and Conquer* adalah algoritma yang membagi persoalan menjadi beberapa persoalan lebih kecil yang mirip dengan persoalan semula. Algoritma ini kemudian akan menyelesaikan masing – masing persoalan yang telah dipecah tersebut, baik secara langsung apabila ukuran persoalan sudah kecil ataupun menggunakan rekursif apabila ukuran persoalan masih cukup besar.

Langkah – langkah algoritma untuk program *Convex hull*:

1. Program akan menerima array of points dan mencari titik yang memiliki nilai x paling kecil dan besar, misalnya titik A dan B. Pasangan titik inilah yang akan membentuk garis tengah sebagai awal dari algoritma *divide and conquer*.
2. Program kemudian akan mencari titik terjauh dari garis tengah di bagian atas garis, misalnya titik C, kemudian program akan membentuk garis dari titik ekstrem yang baru dengan titik awal yang pertama (garis dari titik C ke titik A). Program akan melakukan rekursi untuk mengecek semua titik yang tidak memiliki titik B (titik yang dicek berada di sisi lain garis).
3. Rekursi akan berakhir ketika tidak ada lagi titik terjauh yang ditemukan, dalam kata lain semua titik sudah dilingkupi oleh garis *convex hull* di sektor itu.
4. Setelah rekursi pertama selesai, program akan membentuk garis dari titik ekstrem yang baru dengan titik awal yang kedua (garis dari titik C ke titik B). Kemudian, program akan melakukan rekursi untuk mengecek semua titik yang tidak memiliki titik A (titik yang dicek berada di sisi lain garis).
5. Program kemudian akan melakukan langkah 2, 3, dan 4 untuk sisi bawah garis.
6. Fungsi *myConvexHull* akan mengembalikan array of indexes yang akan digunakan untuk melakukan *plotting* grafik *convex hull*.

BAB II

Source Program dalam Bahasa Python

Kode dapat terlihat lebih jelas di dalam file **main.ipynb**

```
# Function to find the extreme points (only leftmost and rightmost) within the dataset
def findExtremePoints(ptsMatrix):
    idxMin = 0
    idxMax = 0
    minX = ptsMatrix[0, 0]
    maxX = ptsMatrix[0, 0]
    for i in range(len(ptsMatrix)):
        if (ptsMatrix[i, 0] < minX):
            idxMin = i
            minX = ptsMatrix[i, 0]
        if (ptsMatrix[i, 0] > maxX):
            idxMax = i
            maxX = ptsMatrix[i, 0]
    return idxMin, idxMax
```

```
# Function to check if point c is above or below the line made by point a and b
# The line direction is important:
# If the line is made from left to right, then above is the counter-clockwise direction
# If the line is made from right to left, then above is the clockwise direction
def isPointAbove(a, b, c):
    x1 = a[0]
    x2 = b[0]
    x3 = c[0]
    y1 = a[1]
    y2 = b[1]
    y3 = c[1]

    # Calculate the cross product
    cross = (x2-x1)*(y3-y2) - (y2-y1)*(x3-x2)
    # Positive cross means the point is on the counter-clockwise direction of the line
    if (cross > 0):
        return 1
    # Negative cross means the point is on the clockwise direction of the line
    elif (cross < 0):
        return -1
    # If cross is 0, then the point is on the line
    else:
        return 0
```

```
# Function to find the distance from point c to a line from point a to b
def distancePointtoLine(a, b, c):
    deltaX = b[0] - a[0]
    deltaY = b[1] - a[1]
    return ((abs(deltaY * c[0] - deltaX * c[1] + b[0]*a[1] - b[1]*a[0]))/math.sqrt(deltaY**2 + deltaX**2))
```

```

# Function to group all elements inside the array into sets of two
def turnArraytoMatrix(hull):
    hullMatrix = []
    # Iterate through the array, skipping 1 element along the way
    for i in range(0, len(hull), 2):
        hullElmt = []
        hullElmt.append(hull[i])
        hullElmt.append(hull[i+1])

        hullMatrix.append(hullElmt)
    return hullMatrix

```

```

# Function to find the convex hull using divide and conquer algorithm
# Function will create a line from the starting point to the furthest point and find points outside the segment,
# setting that point as the new furthest point
def findConvexHull(points, idx1, idx2, pos, hull):
    idxMaxDist = -1
    maxDist = 0

    # Iterate through all the points available within the dataset
    for i in range(len(points)):
        # Find the distance of each points to the center line
        distance = distancePointtoLine(points[idx1], points[idx2], points[i])

        # If the point we are currently scanning is on the correct position (either above or below based on pos),
        # then set the index and max distance
        if ((isPointAbove(points[idx1], points[idx2], points[i]) == pos) and distance > maxDist):
            idxMaxDist = i
            maxDist = distance

    # If no more points are found (meaning all the points in that sector has been covered), append both edges of the line,
    # then return the finished convex hull array
    if (idxMaxDist == -1):
        hull.append(idx1)
        hull.append(idx2)
        return hull

    # Obtain points outside of the current hull using recursion
    # Recurse points above (outside the triangle) of the line created from the furthest point to idx1
    # Pos flag within the function parameters is multiplied by (-1) to find points on the other side of the line,
    # not the side that has the other main points
    findConvexHull(points, idxMaxDist, idx1, (-1)*(isPointAbove(points[idxMaxDist], points[idx1], points[idx2])), hull)
    # Recurse points above (outside the triangle) of the line created from the furthest point to idx2
    findConvexHull(points, idxMaxDist, idx2, (-1)*(isPointAbove(points[idxMaxDist], points[idx2], points[idx1])), hull)

```

```

# Function to find the convex hull of a given array
def myConvexHull(points):
    # Find the leftmost and rightmost points
    idxMin, idxMax = findExtremePoints(points)

    # Initialize a hull array to store convex hull's points
    hull = []

    # Call findConvexHull to recursively check points above the center line
    findConvexHull(points, idxMin, idxMax, 1, hull)

    # Call findConvexHull to recursively check points below the center line
    findConvexHull(points, idxMin, idxMax, -1, hull)

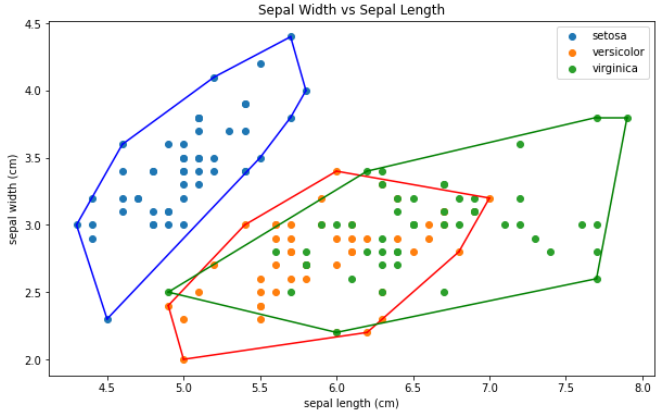
    # Group resulting array into sets of two
    hullMatrix = turnArraytoMatrix(hull)
    return hullMatrix

```

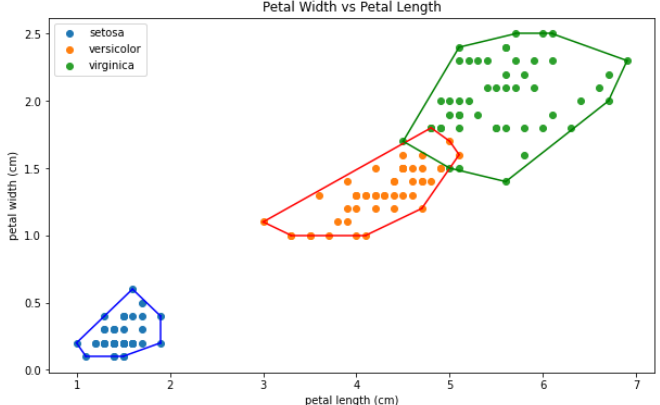
BAB III

Screenshot Input dan Output

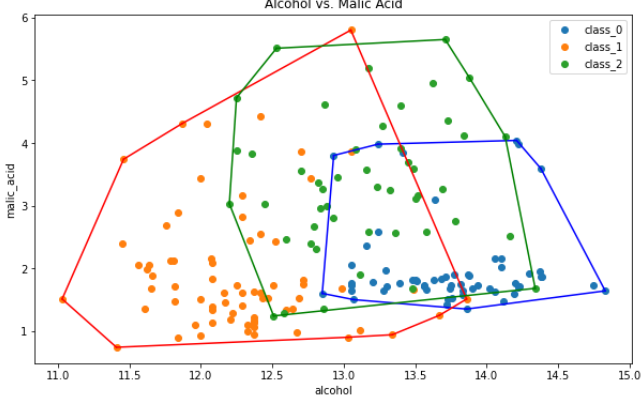
3.1. Sepal width vs. Sepal length dari dataset Iris

Gambar	Keterangan
<pre># Iris dataset data = datasets.load_iris() df = pd.DataFrame(data.data, columns=data.feature_names) df['Target'] = pd.DataFrame(data.target) print(df.shape) df</pre>	<p>Import dataset iris ke dalam dataFrame df</p>
<pre>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 hull = myConvexHull(bucket) 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()</pre>	<p>Program untuk menampilkan visualisasi <i>convex hull</i> pasangan atribut sepal width vs. sepal length</p>
	<p>Hasil <i>convex hull</i> dari dataset iris dengan pasangan atribut sepal width vs. sepal length</p>

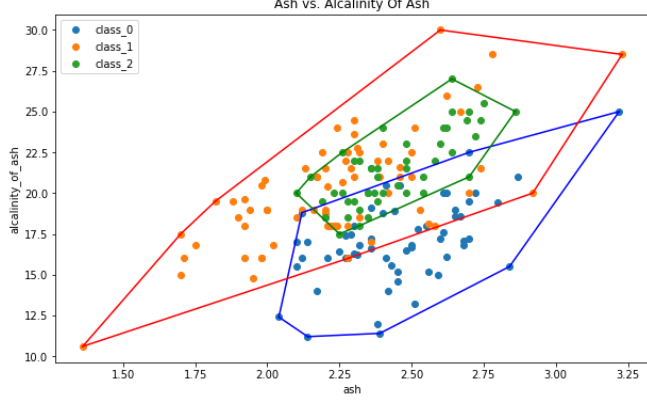
3.2. Petal width vs. Petal length dari dataset Iris

Gambar	Keterangan
<pre># Iris dataset data = datasets.load_iris() df = pd.DataFrame(data.data, columns=data.feature_names) df['Target'] = pd.DataFrame(data.target) print(df.shape) df</pre>	<p>Import dataset iris ke dalam dataframe df</p>
<pre>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 hull = myConvexHull(bucket) 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()</pre>	<p>Program untuk menampilkan visualisasi <i>convex hull</i> pasangan atribut petal width vs. petal length</p>
	<p>Hasil <i>convex hull</i> dari dataset iris dengan pasangan atribut petal width vs. petal length</p>

3.3. Alcohol vs. Malic Acid dari dataset Wine

Gambar	Keterangan
<pre># Wine dataset data2 = datasets.load_wine() df2 = pd.DataFrame(data2.data, columns=data2.feature_names) df2['Target'] = pd.DataFrame(data2.target) print(df2.shape) df2</pre>	<p>Import dataset wine ke dalam dataFrame df2.</p>
<pre>plt.figure(figsize=(10,6)) colors = ['b','r','g'] plt.title('Alcohol vs. Malic Acid') plt.xlabel(data2.feature_names[0]) plt.ylabel(data2.feature_names[1]) for i in range(len(data2.target_names)): bucket = df2[df2['Target'] == i] bucket = bucket.iloc[:,[0,1]].values hull = myConvexHull(bucket) plt.scatter(bucket[:, 0], bucket[:, 1], label=data2.target_names[i]) for simplex in hull: plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i]) plt.legend()</pre>	<p>Program untuk menampilkan visualisasi <i>convex hull</i> pasangan atribut alcohol vs. malic acid</p>
	<p>Hasil <i>convex hull</i> dari dataset wine dengan pasangan atribut alcohol vs. malic acid</p>

3.4. Alcohol vs. Malic Acid dari dataset Wine

Gambar	Keterangan
<pre># Wine dataset data2 = datasets.load_wine() df2 = pd.DataFrame(data2.data, columns=data2.feature_names) df2['Target'] = pd.DataFrame(data2.target) print(df2.shape) df2</pre>	<p>Import dataset wine ke dalam dataFrame df2.</p>
<pre>plt.figure(figsize=(10,6)) colors = ['b','r','g'] plt.title('Ash vs. Alcalinity Of Ash') plt.xlabel(data2.feature_names[2]) plt.ylabel(data2.feature_names[3]) for i in range(len(data.target_names)): bucket = df2[df2['Target'] == i] bucket = bucket.iloc[:,[2,3]].values hull = myConvexHull(bucket) plt.scatter(bucket[:, 0], bucket[:, 1], label=data2.target_names[i]) for simplex in hull: plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i]) plt.legend()</pre>	<p>Program untuk menampilkan visualisasi <i>convex hull</i> pasangan atribut ash vs. alcalinity of ash</p>
	<p>Hasil <i>convex hull</i> dari dataset wine dengan pasangan atribut ash vs. alcalinity of ash</p>

BAB IV

Checklist Program

Poin	Ya	Tidak
1. Pustaka <i>myConvexHull</i> berhasil dibuat dan tidak ada kesalahan	✓	
2. <i>Convex hull</i> yang dihasilkan sudah benar	✓	
3. Pustaka <i>myConvexHull</i> dapat digunakan untuk menampilkan <i>convex hull</i> setiap label dengan warna yang berbeda.	✓	
4. Bonus: program dapat menerima input dan menuliskan output untuk dataset lainnya.	✓	

BAB V

Link Kode Program

Kode program dapat dilihat pada halaman github berikut

<https://github.com/YakobusIP/Tugas-Kecil-Stima-2-Convex-Hull-Algorithm.git>

BAB VI

Daftar Referensi

Munir, Rinaldi. 2022. Algoritma *Divide and Conquer* (Bagian 1). Diakses pada 26 Februari 2022, dari [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-\(2021\)-Bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf)

Munir, Rinaldi. 2022. Algoritma *Divide and Conquer* (Bagian 4). Diakses pada 26 Februari 2022, dari [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-\(2022\)-Bagian4.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf)