Laporan Tugas Kecil 2 IF2211 Strategi Algoritma

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


Disusun oleh:

Muhammad Akyas David Al Aleey 13520011

**PROGRAM STUDI TEKNIK INFORMATIKA**
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**
**INSTITUT TEKNOLOGI BANDUNG**
**BANDUNG**
**2022**

## A. Algoritma *Divide and Conquer* untuk Menyelesaikan *Convex Hull*

Untuk menyelesaikan persoalan pada tugas kecil ini, digunakan algoritma *divide and conquer* yang akan mencari himpunan titik-titik penyusun sebuah *convex hull* dari suatu senarai yang berisi sekumpulan koordinat dua dimensi. Himpunan titik pada bidang planar dapat dikatakan *convex* jika untuk sembarang dua titik pada bidang tersebut, seluruh segmen garis yang berakhir di kedua titik tersebut berada pada himpunan tersebut. Dengan kata lain, *convex hull* adalah poligon terkecil yang disusun dari subset titik sedemikian sehingga tidak ada titik dari himpunan awal yang berada di luar poligon tersebut.

Implementasi algoritma *divide and conquer* diawali dengan mengurutkan himpunan seluruh koordinat secara terurut membesar dengan memprioritaskan nilai absis terlebih dahulu daripada nilai ordinatnya. Proses pengurutan dilakukan dengan memanfaatkan algoritma QuickSort yang mengurutkan data melalui pendekatan rekursif. Selanjutnya, pilih titik paling kiri ($p_1$) dan titik paling kanan ($p_n$) yang berlaku sebagai titik ekstrim yang akan membentuk *convex hull* untuk kumpulan titik tersebut. Karena garis yang menghubungkan $p_1$ dan $p_n$ membagi himpunan titik menjadi dua, maka dilakukan segmentasi bagian himpunan titik menjadi dua bagian yaitu bagian atas dan bawah. Partisi ditentukan oleh besarnya nilai determinan atau hasil *cross product* dari himpunan titik terhadap garis $p_1p_n$. Suatu titik berada di sebelah atas garis jika hasil determinan bernilai positif, berada di bawah jika bernilai negatif, dan berada tepat di garis tersebut jika hasilnya nol. Semua titik yang berada pada sepanjang garis $p_1p_n$ dapat diabaikan karena titik tersebut tidak mungkin membentuk *convex hull*.

Dari hasil partisi di atas, kedepannya kumpulan titik di bagian atas garis akan membentuk *convex hull* bagian atas, dan kumpulan titik di bagian bawah garis akan membentuk *convex hull* bagian bawah.

Terdapat dua kemungkinan yang mungkin untuk himpunan titik di bagian atas dan bawah garis p1pn yaitu:

1. Jika himpunan titik tersebut kosong atau tidak ada titik lain selain p1 dan pn, maka kedua titik tersebut akan membentuk convex hull pada bagian tersebut.
2. Jika himpunan titik tidak kosong, pilih sebuah titik yang memiliki jarak terjauh dari garis $p_1p_n$ (misal $p_{max}$). Jika terdapat beberapa titik dengan jarak yang sama, pilih sebuah titik yang memaksimalkan sudut $p_{max}p_1p_n$.

Dari kemungkinan kedua, setelah dipilih titik dengan jarak dan sudut yang maksimal, akan terbentuk sebuah segitiga dengan titik sudut $p_1$, $p_n$, $p_{max}$. Setelah itu, dilakukan lagi partisi untuk bagian atas garis $p_1p_{max}$ dan garis $p_{max}p_n$ yang nantinya akan membentuk *convex hull*. Semua titik yang berada di dalam daerah segitiga $p_{max}$, $p_1$, $p_n$ diabaikan untuk pemeriksaan lebih lanjut karena tidak mungkin membentuk *convex hull*. Proses partisi tersebut akan terus dilakukan hingga seluruh bagian memenuhi kemungkinan pertama yang sudah dijelaskan di atas. Setelah semua titik telah diproses, akan didapat himpunan titik-titik yang akan membentuk *convex hull* yang diinginkan.

## B. *Source Code* Program

Program pencarian *convex hull* ini ditulis dengan menggunakan bahasa Python yang terbagi menjadi dua file yaitu *myConvexHull.py* sebagai pustaka berisi algoritma *divide and conquer* yang akan mengembalikan pasangan indeks dari kumpulan titik dua dimensi yang membentuk *convex hull* dengan urutan searah jarum jam dan *main.py* yang berisi program utama untuk pengolahan dataset.

### 1. myConvexHull.py

Terdapat 7 method yang ada di dalam kode ini, di antaranya yaitu:

a. Fungsi **partition(arr_of_points, low, high)**

Fungsi ini akan mengambil elemen terakhir sebagai pivot kemudian menempatkan semua elemen yang lebih kecil dari pivot di sebelah kiri dan semua elemen yang lebih besar di sebelah kanan pivot. Fungsi ini akan mengembalikan indeks pivot setelah proses penukaran elemen.

```python
import math
import numpy as np

# Fungsi partition mengambil elemen terakhir sebagai pivot
# kemudian menempatkan semua elemen yang lebih kecil dari
# pivot di sebelah kiri dan semua elemen yang lebih besar
# di sebelah kanan pivot. Fungsi ini akan mengembalikan
# indeks pivot

def partition(arr_of_points, low, high):

    i = (low-1)
    pivot = arr_of_points[high]

    for j in range(low, high):

        # Mengecek apakah elemen sekarang lebih kecil atau
        # sama dengan pivot berdasarkan nilai absis terlebih
        # dahulu kemudian nilai ordinat
        if (arr_of_points[j][0] < pivot[0]) or
           (arr_of_points[j][0] <= pivot[0] and arr_of_points[j][1] <= pivot[1]):

            # Menukar elemen dengan pivot
            i = i+1
            temp = arr_of_points[i]
            arr_of_points[i] = arr_of_points[j]
            arr_of_points[j] = temp

    temp = arr_of_points[i+1]
    arr_of_points[i+1] = arr_of_points[high]
    arr_of_points[high] = temp

    return i+1
```

b. Prosedur **quickSort(arr_of_points, low, high)**

Prosedur quickSort akan mengurutan array of points dengan memanfaatkan algoritma quicksort.

```python
# Prosedur pengurutan array of points dengan algoritma quicksort
def quickSort(arr_of_points, low, high):

    if len(arr_of_points) == 1:
        return arr_of_points

    if low < high:

        pi = partition(arr_of_points, low, high)

        # Memproses pengurutan array sebelah
        # kiri dan kanan indeks partisi
        quickSort(arr_of_points, low, pi-1)
        quickSort(arr_of_points, pi+1, high)
```

c. Fungsi **determinant(a, b, c)**

Fungsi determinant akan menghitung determinan atau hasil cross product dari titik c terhadap garis yang menghubungkan titik a dan b.

```
# Menghitung determinan dari titik c terhadap titik a dan b
def determinant(a, b, c):

    return (a[0] * b[1]) + (c[0] * a[1]) + (b[0] * c[1]) -
           (c[0] * b[1]) - (b[0] * a[1]) - (a[0] * c[1])
```

d. Fungsi **point_to_line_distance(p1, p2, p3)**

Fungsi point_to_line_distance akan menghitung jarak dari titik p3 ke garis yang menghubungkan titik p1 dan p2.

```
# Menghitung jarak dari titik p3 ke garis yang dihubungkan oleh titik p1 dan p2
def point_to_line_distance(p1, p2, p3):

    p1=np.array(p1)
    p2=np.array(p2)
    p3=np.array(p3)

    return np.cross(p2-p1,p3-p1)/np.linalg.norm(p2-p1)
```

e. Fungsi **getAngle(p1, p2, p3)**

Fungsi getAngle akan mengembalikan besar sudut yang dibentuk oleh titik p1p3 dan p2p3.

```
# Menghitung besar sudut yang dibentuk oleh titik p3 terhadap titik p1 dan p2
def getAngle(p1, p2, p3):

    angle = math.degrees(math.atan2(p3[1]-p2[1], p3[0]-p2[0]) - math.atan2(p1[1]-p2[1], p1[0]-p2[0]))

    if angle < 0:
        return angle + 360

    else:
        return angle
```

f. Prosedur **find_hull(sorted_points, hull_set, leftmost_idx, rightmost_idx, convex_set)**

Prosedur find_hull akan mencari titik-titik terluar yang dapat menyusun *convex hull* dengan algoritma *divide and conquer.*

```
# Mencari titik-titik ekstrim penyusun convex hull dengan algoritma divide and conquer.
def find_hull(sorted_points, hull_set, leftmost_idx, rightmost_idx, convex_set):

    if len(hull_set):

        extreme_dis = -1
        extreme_angle = 0
        candidate_points1 = []
        candidate_points2 = []

        # Mencari titik ekstrim (Pmax) yang memiliki jarah terjauh dari garis P1Pn
        # Jika terdapat beberapa titik dengan jarak yang sama, akan dipilih
        # titik dengan sudut terhadap P1Pn yang terbesar.
        for i in hull_set:
            curr_dis = point_to_line_distance(sorted_points[leftmost_idx],
                                              sorted_points[rightmost_idx], sorted_points[i])

            if (curr_dis > extreme_dis):
                extreme_dis = curr_dis
                extreme_idx = i
                extreme_angle = getAngle(sorted_points[leftmost_idx],
                                         sorted_points[rightmost_idx], sorted_points[i])

            elif (curr_dis == extreme_dis):
                angle_point = getAngle(sorted_points[leftmost_idx],
                                       sorted_points[rightmost_idx], sorted_points[i])

                if (angle_point > extreme_angle):
                    extreme_idx = i
                    extreme_angle = angle_point

        # Mengklasifikasikan kumpulan titik di sebelah kiri garis P1Pmax
        # dan di sebelah kanan garis PmaxPn
        for i in hull_set:

            if (extreme_idx != i):

                det1 = determinant(sorted_points[leftmost_idx],
                                   sorted_points[extreme_idx], sorted_points[i])
                det2 = determinant(sorted_points[extreme_idx],
                                   sorted_points[rightmost_idx], sorted_points[i])

                if det1 > 0 and det2 < 0:
                    candidate_points1.append(i)

                elif det2 > 0 and det1 < 0:
                    candidate_points2.append(i)

        find_hull(sorted_points, candidate_points1, leftmost_idx, extreme_idx, convex_set)
        find_hull(sorted_points, candidate_points2, extreme_idx, rightmost_idx, convex_set)

    # Jika tidak ada titik lain selain P1 dan Pn, maka titik tersebut menjadi
    # pasangan titik pembentuk convex hull
    else:
        convex_set.append([leftmost_idx, rightmost_idx])
```

g. Fungsi **convex_hull(sorted_points)**

Fungsi convex_hull akan mengembalikan array berisi pasangan indeks dari koordinat yang mewakili titik-titik penyusun *convex hull*. Pasangan indeks ini nantinya akan dihubungkan oleh suatu garis lurus.

```python
# Algoritma utama pencarian convex hull dari array of points yang sudah terurut.
# Fungsi ini akan mengembalikan array berisi pasangan indeks dari koordinat yang
# mewakili titik-titik penyusun convex hull. Pasangan indeks ini nantinya akan
# dihubungkan oleh suatu garis lurus.
def convex_hull(sorted_points):

    arr_len = len(sorted_points)
    upper_hull = []
    lower_hull = []
    convex_set = []

    # Mengklasifikasi titik-titik yang membentuk convex hull bagian atas
    # atau bawah dari garis yang menghubungkan titik pada indeks pertama (P1)
    # dengan titik pada indeks terakhir (Pn)
    for i in range(1, arr_len - 1):
        det = determinant(sorted_points[0], sorted_points[arr_len - 1], sorted_points[i])

        if det > 0:
            upper_hull.append(i)

        elif det < 0:
            lower_hull.append(i)

    find_hull(sorted_points, upper_hull, 0, arr_len - 1, convex_set)
    find_hull(sorted_points, lower_hull, arr_len - 1, 0, convex_set)

    return convex_set
```

**2. main.py**

Terdapat 2 method yang ada di dalam kode ini, di antaranya yaitu:

a. Prosedur getInput()

Prosedur ini akan menampilkan daftar dataset yang dapat diolah kemudian meminta masukan berupa nomor dataset yang akan digunakan. Setelah dataset dipilih, akan ditampilkan daftar atribut yang ada di dalam dataset tersebut dan *user* akan memasukkan 2 buah atribut yang akan digunakan.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import myConvexHull
import random
from sklearn import datasets

def getInput():

    print("============================================================")
    print("| Berikut daftar dataset yang dapat kalian coba:           |")
    print("| 1. Iris                                                  |")
    print("| 2. Digits                                                |")
    print("| 3. Wine                                                  |")
    print("| 4. Breast cancer                                         |")
    print("============================================================")
    opsi = int(input("Masukkan nomor dataset yang ingin dicoba: "))
    print("+------------------------------------------------------------+")

    # Memuat dataset sesuai pilihan user
    if (opsi == 1):
        data = datasets.load_iris()
    elif (opsi == 2):
        data = datasets.load_digits()
    elif (opsi == 3):
        data = datasets.load_wine()
    else:
        data = datasets.load_breast_cancer()

    doProcess(data)
```

b. Prosedur doProcess()

Prosedur ini akan mengolah data yang sudah dipilih oleh *user* sekaligus memanggil fungsi convex_hull yang ada di file myConvexHull.py

```python
def doProcess(data):

    # Create a DataFrame
    df = pd.DataFrame(data.data, columns=data.feature_names)
    df['Target'] = pd.DataFrame(data.target)
    print("Dimensi dataset yang dipilih sebesar:", df.shape)
    print("Dengan spesifikasi ukuran yaitu", df.shape[0], "baris dan", df.shape[1], "kolom.")
    print("+------------------------------------------------------------+")
    print("Berikut daftar kolom yang dapat digunakan:")

    i = 1
    for col in df.columns:
        if (col != "Target"):
            print(str(i) + ". " + str(col))
            i += 1
    print("+------------------------------------------------------------+")
    column_1 = int(input("Masukkan nomor kolom pertama yang akan digunakan: "))
    column_2 = int(input("Masukkan nomor kolom kedua yang akan digunakan: "))
    print("==============================================================")

    # Visualisasi hasil ConvexHull
    plt.figure(figsize = (10, 6))
    plt.title(str(df.columns[column_1 - 1]) + " vs " + str(df.columns[column_2 - 1]))
    plt.xlabel(data.feature_names[column_1 - 1])
    plt.ylabel(data.feature_names[column_2 - 1])

    for i in range(0, len(data.target_names)):
        bucket = df[df['Target'] == i]
        bucket = bucket.iloc[:, [column_1 - 1, column_2 - 1]].values.tolist()
        myConvexHull.quickSort(bucket, 0, len(bucket) - 1)
        hull = myConvexHull.convex_hull(bucket) #bagian ini diganti dengan hasil
                                                #implementasi ConvexHull Divide & Conquer

        colors2 = random.uniform(0, 1), random.uniform(0, 1), random.uniform(0, 1)

        bucket = np.array(bucket)
        hull = np.array(hull)
        plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i], color = colors2)
        for simplex in hull:
            plt.plot(bucket[simplex, 0], bucket[simplex, 1], color = colors2)

    plt.legend()
    plt.show()

if __name__ == "__main__":
    getInput()
```

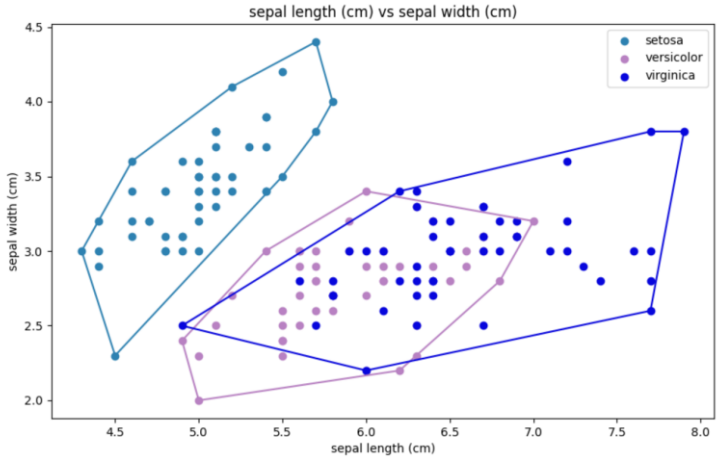
## 3. *Screenshot* Input dan Output

### 1. Dataset Iris: sepal length (cm) vs sepal width (cm)

Input:

```
==============================================================
| Berikut daftar dataset yang dapat kalian coba:             |
| 1. Iris                                                    |
| 2. Digits                                                  |
| 3. Wine                                                    |
| 4. Breast cancer                                           |
==============================================================
Masukkan nomor dataset yang ingin dicoba: 1
+------------------------------------------------------------+
Dimensi dataset yang dipilih sebesar: (150, 5)
Dengan spesifikasi ukuran yaitu 150 baris dan 5 kolom.
+------------------------------------------------------------+
Berikut daftar kolom yang dapat digunakan:
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)
+------------------------------------------------------------+
Masukkan nomor kolom pertama yang akan digunakan: 1
Masukkan nomor kolom kedua yang akan digunakan: 2
==============================================================
```

Output:

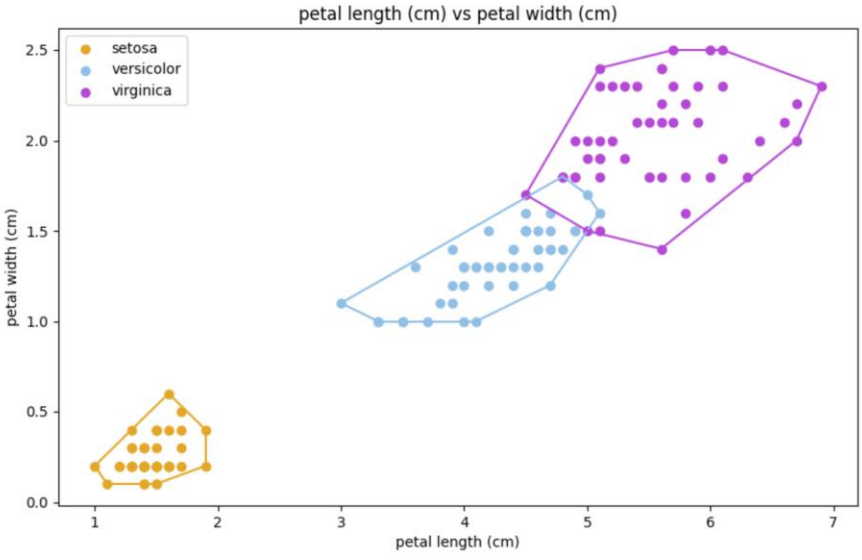



## 2. Dataset Iris: petal length(cm) vs petal length(cm)

Input:

```
============================================================
| Berikut daftar dataset yang dapat kalian coba:           |
| 1. Iris                                                  |
| 2. Digits                                                |
| 3. Wine                                                  |
| 4. Breast cancer                                         |
============================================================
Masukkan nomor dataset yang ingin dicoba: 1
+----------------------------------------------------------+
Dimensi dataset yang dipilih sebesar: (150, 5)
Dengan spesifikasi ukuran yaitu 150 baris dan 5 kolom.
+----------------------------------------------------------+
Berikut daftar kolom yang dapat digunakan:
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)
+----------------------------------------------------------+
Masukkan nomor kolom pertama yang akan digunakan: 3
Masukkan nomor kolom kedua yang akan digunakan: 4
============================================================
```

Output:

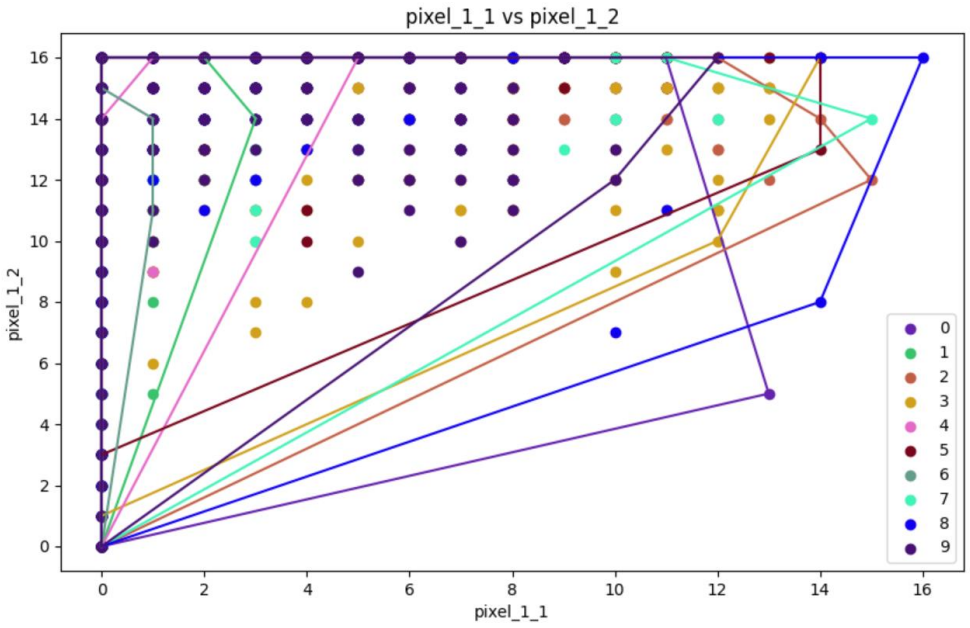


### 3. Dataset Digits: pixel_1_1 vs pixel_1_2

Input:

```
============================================================
| Berikut daftar dataset yang dapat kalian coba:           |
| 1. Iris                                                  |
| 2. Digits                                                |
| 3. Wine                                                  |
| 4. Breast cancer                                         |
============================================================
Masukkan nomor dataset yang ingin dicoba: 2
+----------------------------------------------------------+
Dimensi dataset yang dipilih sebesar: (1797, 65)
Dengan spesifikasi ukuran yaitu 1797 baris dan 65 kolom.
+----------------------------------------------------------+
Berikut daftar kolom yang dapat digunakan:
1. pixel_0_0
2. pixel_0_1
3. pixel_0_2
4. pixel_0_3
5. pixel_0_4
6. pixel_0_5
7. pixel_0_6
8. pixel_0_7
9. pixel_1_0
10. pixel_1_1
11. pixel_1_2
12. pixel_1_3
13. pixel_1_4
14. pixel_1_5
15. pixel_1_6
16. pixel_1_7
17. pixel_2_0
18. pixel_2_1
19. pixel_2_2
20. pixel_2_3
21. pixel_2_4
22. pixel_2_5
23. pixel_2_6
24. pixel_2_7
25. pixel_3_0
26. pixel_3_1
27. pixel_3_2
28. pixel_3_3
29. pixel_3_4
30. pixel_3_5
31. pixel_3_6
32. pixel_3_7
33. pixel_4_0
34. pixel_4_1
35. pixel_4_2
36. pixel_4_3
37. pixel_4_4
38. pixel_4_5
39. pixel_4_6
40. pixel_4_7
41. pixel_5_0
42. pixel_5_1
43. pixel_5_2
44. pixel_5_3
45. pixel_5_4
46. pixel_5_5
47. pixel_5_6
48. pixel_5_7
49. pixel_6_0
50. pixel_6_1
51. pixel_6_2
52. pixel_6_3
53. pixel_6_4
54. pixel_6_5
55. pixel_6_6
56. pixel_6_7
57. pixel_7_0
58. pixel_7_1
59. pixel_7_2
60. pixel_7_3
61. pixel_7_4
62. pixel_7_5
63. pixel_7_6
64. pixel_7_7
+----------------------------------------------------------+
Masukkan nomor kolom pertama yang akan digunakan: 10
Masukkan nomor kolom kedua yang akan digunakan: 11
============================================================
```

Output:

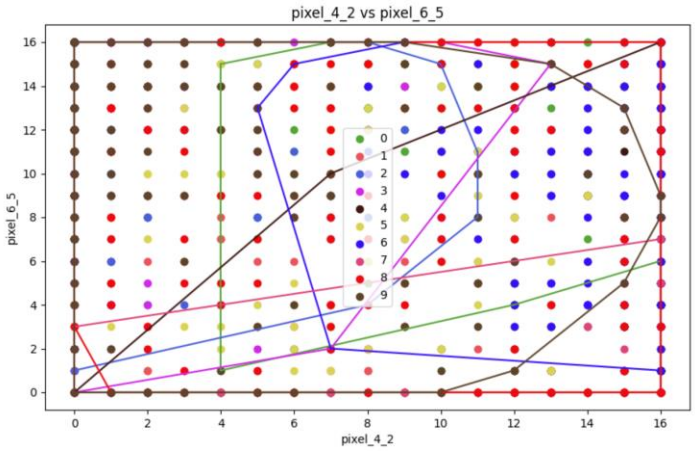


## 4. Dataset Digits: pixel_4_2 vs pixel_6_5

Input:

```
============================================================
| Berikut daftar dataset yang dapat kalian coba:           |
| 1. Iris                                                  |
| 2. Digits                                                |
| 3. Wine                                                  |
| 4. Breast cancer                                         |
============================================================
Masukkan nomor dataset yang ingin dicoba: 2
+----------------------------------------------------------+
Dimensi dataset yang dipilih sebesar: (1797, 65)
Dengan spesifikasi ukuran yaitu 1797 baris dan 65 kolom.
+----------------------------------------------------------+
Berikut daftar kolom yang dapat digunakan:
1. pixel_0_0
2. pixel_0_1
3. pixel_0_2
4. pixel_0_3
5. pixel_0_4
6. pixel_0_5
7. pixel_0_6
8. pixel_0_7
9. pixel_1_0
10. pixel_1_1
11. pixel_1_2
12. pixel_1_3
13. pixel_1_4
14. pixel_1_5
15. pixel_1_6
16. pixel_1_7
17. pixel_2_0
18. pixel_2_1
19. pixel_2_2
20. pixel_2_3
21. pixel_2_4
22. pixel_2_5
23. pixel_2_6
24. pixel_2_7
25. pixel_3_0
26. pixel_3_1
27. pixel_3_2
28. pixel_3_3
29. pixel_3_4
30. pixel_3_5
31. pixel_3_6
32. pixel_3_7
33. pixel_4_0
34. pixel_4_1
35. pixel_4_2
36. pixel_4_3
37. pixel_4_4
38. pixel_4_5
39. pixel_4_6
40. pixel_4_7
41. pixel_5_0
42. pixel_5_1
43. pixel_5_2
44. pixel_5_3
45. pixel_5_4
46. pixel_5_5
47. pixel_5_6
48. pixel_5_7
49. pixel_6_0
50. pixel_6_1
51. pixel_6_2
52. pixel_6_3
53. pixel_6_4
54. pixel_6_5
55. pixel_6_6
56. pixel_6_7
57. pixel_7_0
58. pixel_7_1
59. pixel_7_2
60. pixel_7_3
61. pixel_7_4
62. pixel_7_5
63. pixel_7_6
64. pixel_7_7
+----------------------------------------------------------+
Masukkan nomor kolom pertama yang akan digunakan: 35
Masukkan nomor kolom kedua yang akan digunakan: 54
============================================================
```

Output:

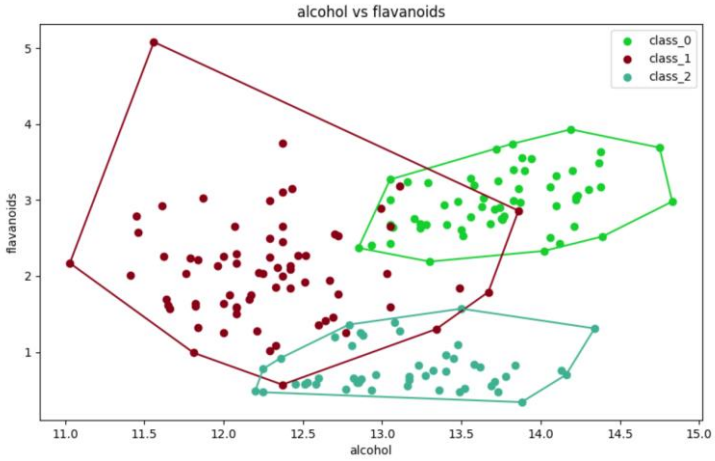


## 5. Dataset Wine: Alcohol vs Flavanoids

Input:

```
============================================================
| Berikut daftar dataset yang dapat kalian coba:           |
| 1. Iris                                                  |
| 2. Digits                                                |
| 3. Wine                                                  |
| 4. Breast cancer                                         |
============================================================
Masukkan nomor dataset yang ingin dicoba: 3
+----------------------------------------------------------+
Dimensi dataset yang dipilih sebesar: (178, 14)
Dengan spesifikasi ukuran yaitu 178 baris dan 14 kolom.
+----------------------------------------------------------+
Berikut daftar kolom yang dapat digunakan:
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
+----------------------------------------------------------+
Masukkan nomor kolom pertama yang akan digunakan: 1
Masukkan nomor kolom kedua yang akan digunakan: 7
============================================================
```

Output:

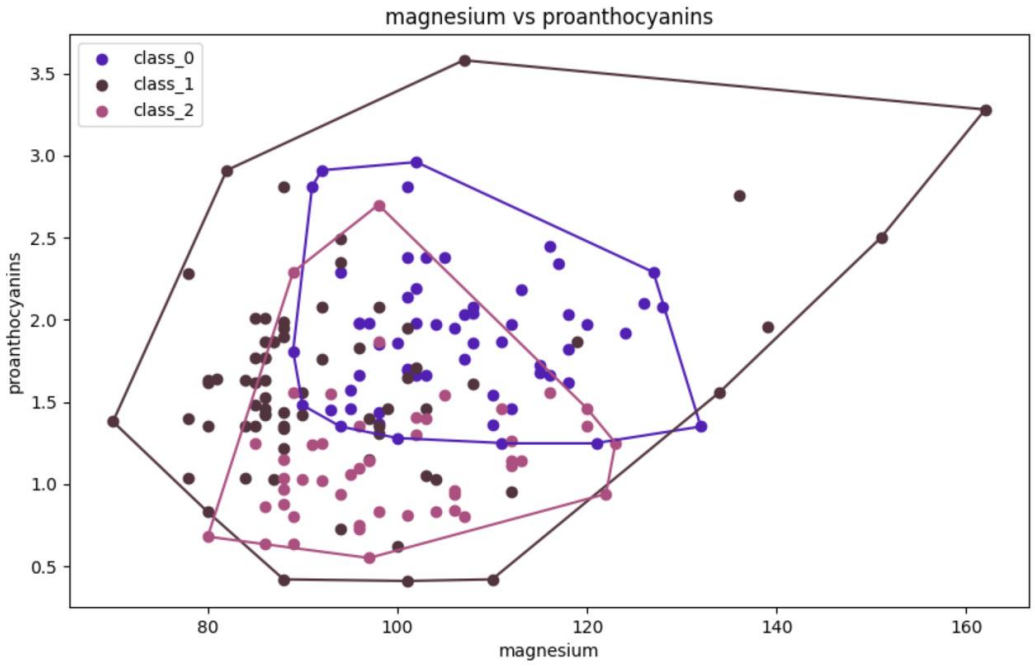


## 6. Dataset Wine: Magnesium vs Proanthocyanins

Input:

```
============================================================
| Berikut daftar dataset yang dapat kalian coba:           |
| 1. Iris                                                  |
| 2. Digits                                                |
| 3. Wine                                                  |
| 4. Breast cancer                                         |
============================================================
Masukkan nomor dataset yang ingin dicoba: 3
+----------------------------------------------------------+
Dimensi dataset yang dipilih sebesar: (178, 14)
Dengan spesifikasi ukuran yaitu 178 baris dan 14 kolom.
+----------------------------------------------------------+
Berikut daftar kolom yang dapat digunakan:
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
+----------------------------------------------------------+
Masukkan nomor kolom pertama yang akan digunakan: 5
Masukkan nomor kolom kedua yang akan digunakan: 9
============================================================
```

Output:

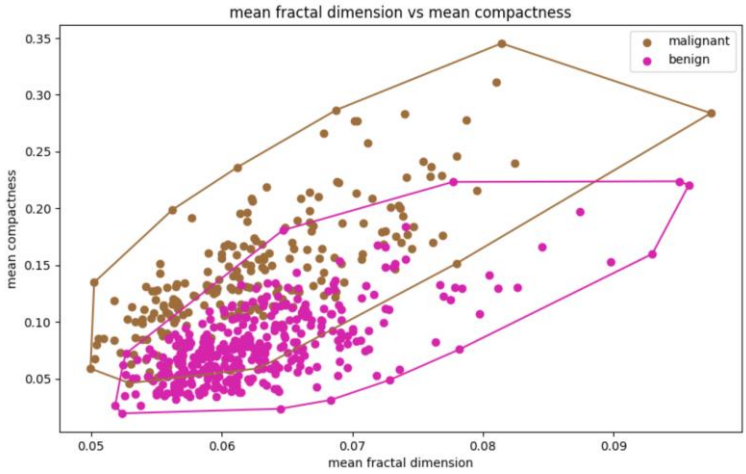


## 7. Dataset Breast Cancer: mean fractal dimension vs mean compactness

Input:

```
============================================================
| Berikut daftar dataset yang dapat kalian coba:           |
| 1. Iris                                                  |
| 2. Digits                                                |
| 3. Wine                                                  |
| 4. Breast cancer                                         |
============================================================
Masukkan nomor dataset yang ingin dicoba: 4
+----------------------------------------------------------+
Dimensi dataset yang dipilih sebesar: (569, 31)
Dengan spesifikasi ukuran yaitu 569 baris dan 31 kolom.
+----------------------------------------------------------+
Berikut daftar kolom yang dapat digunakan:
1. mean radius
2. mean texture
3. mean perimeter
4. mean area
5. mean smoothness
6. mean compactness
7. mean concavity
8. mean concave points
9. mean symmetry
10. mean fractal dimension
11. radius error
12. texture error
13. perimeter error
14. area error
15. smoothness error
16. compactness error
17. concavity error
18. concave points error
19. symmetry error
20. fractal dimension error
21. worst radius
22. worst texture
23. worst perimeter
24. worst area
25. worst smoothness
26. worst compactness
27. worst concavity
28. worst concave points
29. worst symmetry
30. worst fractal dimension
+----------------------------------------------------------+
Masukkan nomor kolom pertama yang akan digunakan: 10
Masukkan nomor kolom kedua yang akan digunakan: 6
============================================================
```

Output:

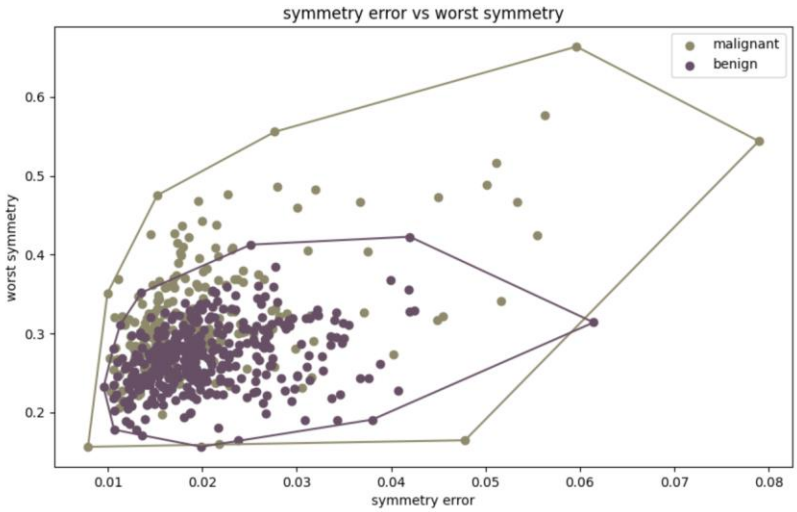



## 8. Dataset Breast Cancer: symmetry error vs worst symmetry

Input:

```
============================================================
| Berikut daftar dataset yang dapat kalian coba:           |
| 1. Iris                                                  |
| 2. Digits                                                |
| 3. Wine                                                  |
| 4. Breast cancer                                         |
============================================================
Masukkan nomor dataset yang ingin dicoba: 4
+----------------------------------------------------------+
Dimensi dataset yang dipilih sebesar: (569, 31)
Dengan spesifikasi ukuran yaitu 569 baris dan 31 kolom.
+----------------------------------------------------------+
Berikut daftar kolom yang dapat digunakan:
1. mean radius
2. mean texture
3. mean perimeter
4. mean area
5. mean smoothness
6. mean compactness
7. mean concavity
8. mean concave points
9. mean symmetry
10. mean fractal dimension
11. radius error
12. texture error
13. perimeter error
14. area error
15. smoothness error
16. compactness error
17. concavity error
18. concave points error
19. symmetry error
20. fractal dimension error
21. worst radius
22. worst texture
23. worst perimeter
24. worst area
25. worst smoothness
26. worst compactness
27. worst concavity
28. worst concave points
29. worst symmetry
30. worst fractal dimension
+----------------------------------------------------------+
Masukkan nomor kolom pertama yang akan digunakan: 19
Masukkan nomor kolom kedua yang akan digunakan: 29
============================================================
```

Output:




## 4. Alamat Drive Kode Program

Alamat github:

https://github.com/Kyasaaa/Tucil-2-IF2211-Strategi-Algoritma

## 5. Tabel Penilaian

| Poin | Ya | Tidak |
|---|---|---|
| 1. Pustaka myConvexHull berhasil dibuat dan tidak ada kesalahan | ✓ | |
| 2. Convex hull yang dihasilkan sudah benar | ✓ | |
| 3. Pustakan 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 | ✓ | |