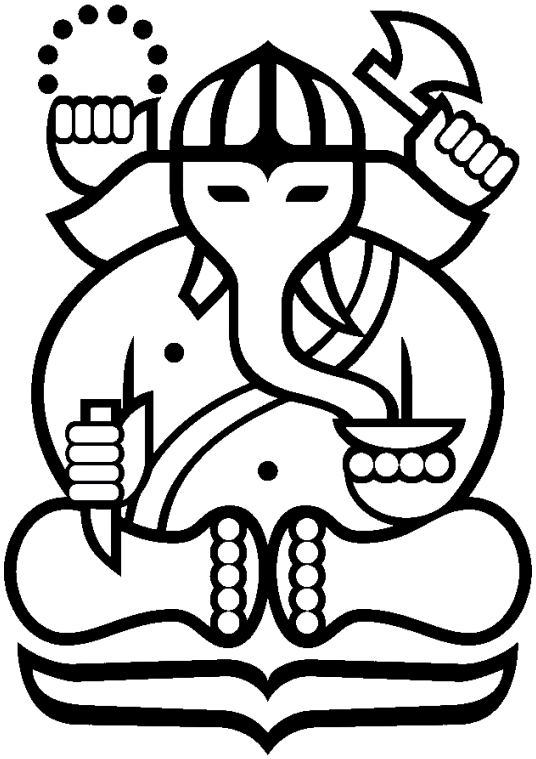
Laporan Tugas Kecil 2 IF2211 Strategi Algoritma
Implementasi *Convex Hull* untuk Visualisasi Tes *Linear Separability Dataset* dengan Algoritma *Divide and Conquer*



Disusun oleh:
13520110 - Farrel Ahmad

27 Februari 2022

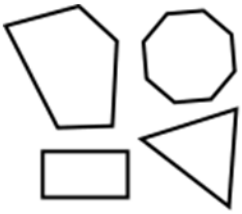
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2022

Daftar Isi

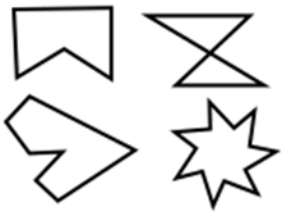
Algoritma *Divide and Conquer* pada *Convex Hull*

Algoritma *Divide and Conquer* adalah desain algoritma yang pendekatannya terdiri dari 3 tugas utama yang berurutan. Tugas tersebut adalah *divide*, *conquer, combine*. *Divide* adalah membagi persoalan utama menjadi upa-persoalan yang memiliki kemiripan dengan persoalan semula tetapi berukuran lebih kecil. *Conquer* adalah menyelesaikan masing-masing upa-persoalan yang lebih kecil, baik secara langsung saat basis, ataupun secara rekursif saat masih berukuran belum mencapai basis. *Combine* adalah menggabungkan masing-masing solusi upa-persoalan sehingga membentuk solusi dari persoalan utama.

Convex Hull adalah suatu bentuk yang terdiri dari himpunan bagian titik-titik dari seluruh titik yang ketika di-*plot* akan mengelilingi seluruh himpunan titik-titik lain di dalamnya. Berikut adalah contoh bentuk yang *convex* dan *non-convex*.



Gambar 1 : convex



Gambar 2 : non convex

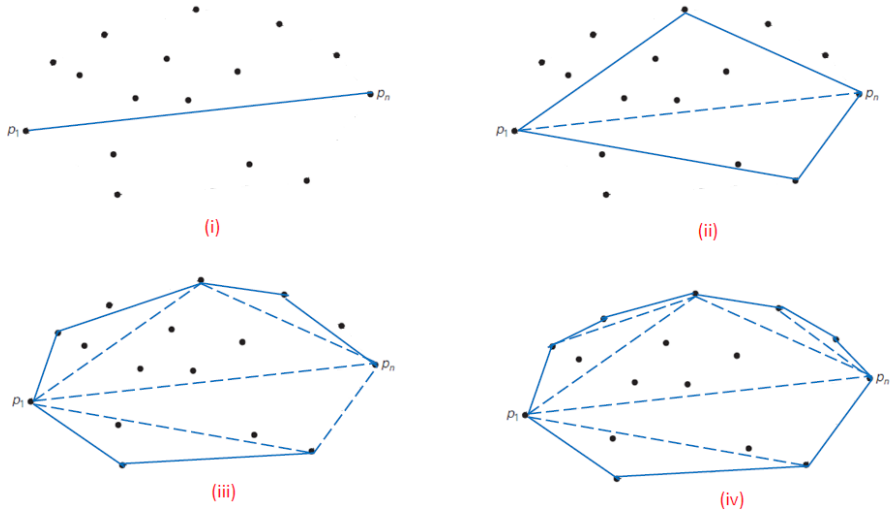
Convex hull dari suatu himpunan titik dapat dibuat dengan menggunakan algoritma yang memiliki pendekatan *divide and conquer*. Algoritma *convex hull* yang diimplementasikan pada tugas kecil ini memiliki langkah langkah adalah sebagai berikut,

1. Urutkan larik yang berisi titik-titik secara menaik berdasarkan x. Apabila ada x yang sama maka diurutkan menaik berdasarkan y.
2. Siapkan sebuah larik kosong yang akan menampung hasil titik *convex hull*.
3. Ambil sebuah titik paling kiri (x terkecil) dan titik paling kanan (x terbesar). Masukkan kedua titik ke dalam larik hasil.
4. Kedua titik tersebut dapat membentuk sebuah garis, bagi persoalan menjadi bagian atas dan bagian bawah garis kemudian selesaikan *convex hull* bagian atas dan bagian bawah secara berurutan.
5. Untuk menentukan apakah sebuah titik berada di atas atau di bawah garis dapat menggunakan rumus determinan berikut dengan x3 dan y3 adalah titik yang dicari, (x1,y1) dan (x2,y2) adalah kedua titik yang membentuk garis. Apabila hasilnya positif maka berada di atas garis, apabila hasilnya negatif maka berada di bawah garis.

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = x_1y_2 + x_3y_1 + x_2y_3 - x_3y_2 - x_2y_1 - x_1y_3$$

6. Selesaikan *convex hull* bagian atas dengan mengambil sub-himpunan titik yang berada di atas garis dengan menggunakan rumus determinan seperti pada langkah 5.
7. Cari titik terjauh pada bagian atas, kemudian sambungkan garisnya dan akan membentuk dua garis baru yang daerahnya harus diselesaikan, yaitu kiri atas dan kanan atas. Masukkan titik terjauh ke dalam larik hasil.
8. Selesaikan dengan mencari titik terjauh lainnya pada kiri atas dan kanan atas garis baru, hasil titik terjauh lainnya dimasukkan ke dalam larik hasil. Setelah itu titik terjauh baru akan menghasilkan garis baru lainnya.
9. Selesaikan bagian bawah sama seperti langkah 6 hingga langkah 8 hanya perbedaannya mengambil sub-himpunan titik yang berada di bawah garis dan mencari titik terjauh kemudian selesaikan titik terjauh lainnya pada kiri bawah dan kanan bawah hasil garis titik terjauh sebelumnya.
10. Larik hasil yang berisi titik-titik *convex hull* diurutkan secara *clockwise* atau berurutan arah jarum jam berdasarkan sudut yang dihasilkan dari titik tengah *convex hull*. Hal ini dilakukan agar dapat membentuk plot yang berurutan dan membentuk sirkuler sehingga *convex hull* dapat terbentuk.
11. Setelah diurutkan, larik hasil ditambahkan elemen pertama pada akhir larik agar dapat menyambungkan garis titik akhir dengan titik awal.

Berikut adalah ilustrasi dari algoritma di atas:



Gambar 3 : Ilustrasi Algoritma *Convex Hull*

Convex hull yang dibentuk dapat digunakan untuk visualisasi *linearly separable dataset test*. Jika ada dua data atau lebih yang masing-masing *convex hull*-nya saling lepas atau tidak beririsan, maka data tersebut *linearly separable* atau secara linear dapat dipisahkan. Apabila *convex hull*-nya ada yang beririsan, maka data tersebut *non linearly separable* atau tidak bisa dipisahkan secara linear.

Kode Program

Link Source Code :
https://github.com/farrel-a/tucil2-stima-ConvexHull
https://drive.google.com/drive/folders/1ktQuMQw26dsWjpnGP6tJPl0FDQ-4ag5K?usp=sharing

myConvexHull.py

```
#13520110 - Farrel Ahmad
from math import sqrt, atan2

def ConvexHull(arrXY, left=[], right=[], i=0, result = [], up=False):
    # Convex Hull Algorithm
    if (arrXY != [] and i==0):
        # Main Processs
        arrXY.sort()
        left = arrXY[0]
        right = arrXY[-1]
        result.append(left)
        result.append(right)
        LeftUpPoints = FindLeftUp(arrXY, left, right)
        RightDownPoints = FindRightDown(arrXY, left, right)

        # Recursive (go to i = 1) ; (i = 0 is for main process)
        ConvexHull(LeftUpPoints, left, right, 1, result, True)
        ConvexHull(RightDownPoints, left, right, 1, result, False)

        # Sort clockwise and add first element to plot circularly
        sort_cw(result)
        result.append(result[0])

    elif (arrXY != [] and i != 0 ):
        # Array of distance
        dist = []

        # Gradient not 0 nor infinity
        if (right[0] - left[0] != 0 and right[1] - left[1] != 0):
            gradient = (right[1]-left[1])/(right[0] - left[0])
            xleft = left[0]
            yleft = left[1]
            for i in range(len(arrXY)):
                point = arrXY[i]
                x = point[0]
                y = point[1]
                A = gradient
                B = -1
                C = gradient * -xleft + yleft
                distance = abs((A*x + B*y + C)) / (sqrt(A**2 + B**2))
                dist.append(distance)

        # Gradient infinity (Vertical Line)
```

```
        elif (right[0] - left[0] == 0):
            for i in range(len(arrXY)):
                point = arrXY[i]
                x = point[0]
                distance = abs(x-left[0])
                dist.append(distance)

        # Gradient zero (Horizontal line)
        elif (right[1] - left[1] == 0):
            for i in range(len(arrXY)):
                point = arrXY[i]
                y = point[1]
                distance = abs(y-left[1])
                dist.append(distance)

        # Find point with max distance to line
        dist_max = max(dist)
        idx_max = dist.index(dist_max)

        # If point not in result, add the point to the result
        if (arrXY[idx_max] not in result):
            result.append(arrXY[idx_max])

        # If checking points above line
        if (up):
            LeftUpPoints = FindLeftUp(arrXY, left, arrXY[idx_max])
            RightUpPoints = FindLeftUp(arrXY, arrXY[idx_max], right)
            # If there are points on left up
            if (len(LeftUpPoints) != 0):
                ConvexHull(LeftUpPoints, left, arrXY[idx_max], 1, result, up)
            # If there are points of right up
            if (len(RightUpPoints) != 0):
                ConvexHull(RightUpPoints, arrXY[idx_max], right, 1, result, up)

        # If checking points below line
        else:
            LeftDownPoints = FindRightDown(arrXY, left, arrXY[idx_max])
            RightDownPoints = FindRightDown(arrXY, arrXY[idx_max], right)
            # If there are points on left down
            if (len(LeftDownPoints) != 0):
                ConvexHull(LeftDownPoints, left, arrXY[idx_max], 1, result, up)
            # If there are points on right down
            if (len(RightDownPoints) != 0):
                ConvexHull(RightDownPoints, arrXY[idx_max], right, 1, result, up)

def PointDeterminant(p1, p2, p3):
    # Calculate determinant of points to determine point's position relative to line
    return p1[0]*p2[1] + p3[0]*p1[1] + p2[0]*p3[1] - p3[0]*p2[1] - p2[0]*p1[1] - p1[0]*p3[1]

def FindLeftUp(arrXY, Pleft, Pright):
    # Find points on left up based on PointDeterminant
    arr = []
```

```
    for i in range(len(arrXY)):
        if (PointDeterminant(Pleft, Pright, arrXY[i]) > 0  and arrXY[i] != Pleft and arrXY[i]
!= Pright):
            arr.append(arrXY[i])
    return arr

def FindRightDown(arrXY, Pleft, Pright):
    # Find points on right down based on PointDeterminant
    arr = []
    for i in range(len(arrXY)):
        if (PointDeterminant(Pleft, Pright, arrXY[i]) <= 0 and arrXY[i] != Pleft and arrXY[i]
!= Pright):
            arr.append(arrXY[i])
    return arr

def sort_cw(points):
    # Sort points in a clockwise sequence
    total_x = 0
    total_y = 0
    for x, y in points:
        total_x += x
        total_y += y
    centre_x = total_x/len(points)
    centre_y = total_y/len(points)

    angles = []
    for x, y in points :
        angle = atan2(y - centre_y, x - centre_x)
        angles.append(angle)

    angles_sorted = sorted(angles, reverse=True)

    idx = []
    for angle_sorted in angles_sorted:
        idx.append(angles.index(angle_sorted))
    cw_points = []

    for i in idx:
        cw_points.append(points[i])

    points.clear()
    for i in range(len(cw_points)):
        points.append(cw_points[i])
```

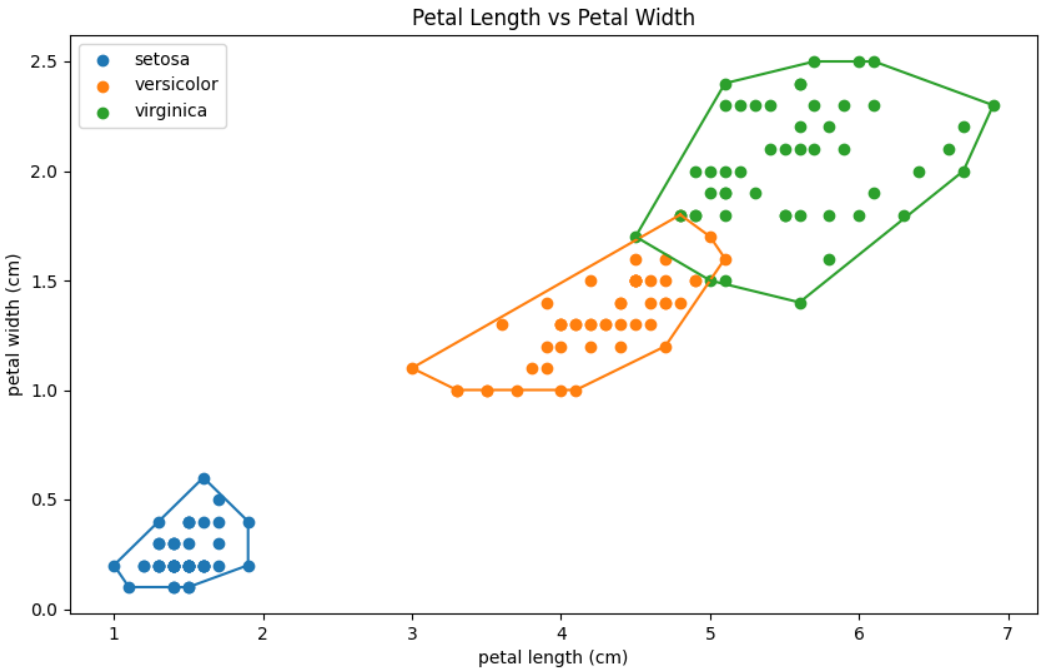
Penggunaan ConvexHull (Input)

```
    # Convex Hull Points
    hull = []
    ConvexHull(arrXY, result = hull)
```

main_iris_petal.py

```
#13520110 - Farrel Ahmad
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import *

""" Iris DataFrame  """
data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)

""" Iris DataFrame Convex Hull Visualization """
plt.figure(figsize = (10, 6))
plt.title('Petal Length vs Petal Width')
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])
for i in range(len(data.target_names)):
    # Find points (x,y) for each target
    bucket = df[df['Target'] == i]
    length = len(bucket)
    arrXY = []
    for j in range(length):
        try:
            x = bucket.loc[j+bucket.first_valid_index()][2]
            y = bucket.loc[j+bucket.first_valid_index()][3]
        except KeyError:
            continue
        else:
            arrXY.append([x,y])

    # Convex Hull Points
    hull = []
    ConvexHull(arrXY, result = hull)

    # Points Scatter Plot
    arrX, arrY = zip(*arrXY)
    plt.scatter(arrX, arrY, label = data.target_names[i])

    # Convex Hull Plot
    hullX, hullY = zip(*hull)
    plt.plot(hullX, hullY)

plt.legend()
plt.show()
```

Hasil main_iris_petal.py :

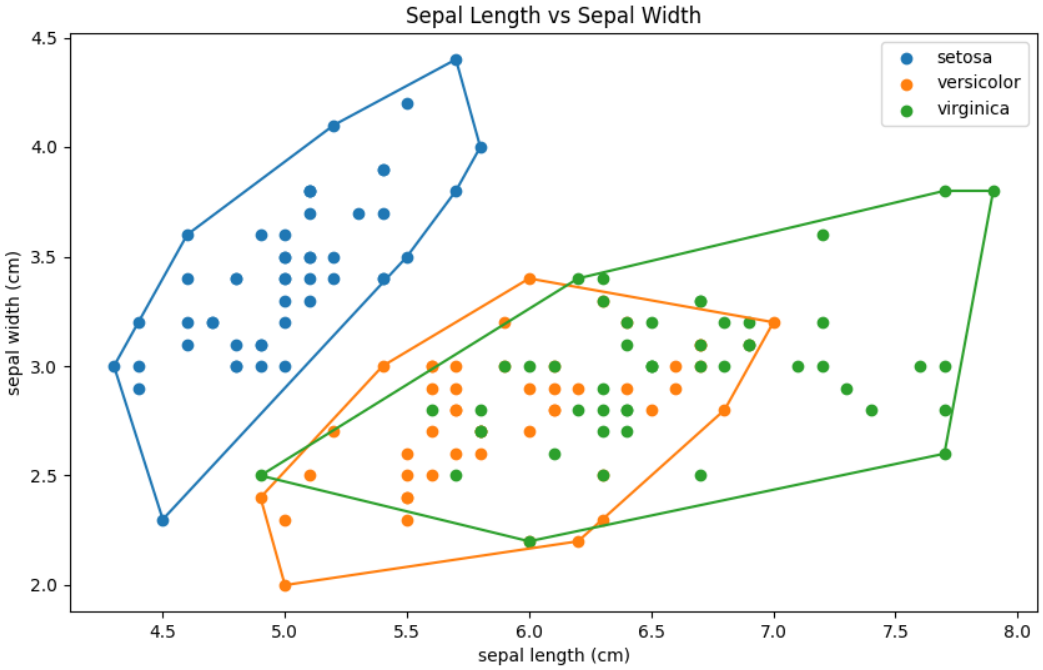


Setosa : *linearly separable* (saling lepas)
Versicolor : *non-linearly separable* (beririsan dengan Virginica)
Virginica : *non-linearly separable* (beririsan dengan Versicolor)

main_iris_sepal.py

```python
#13520110 - Farrel Ahmad
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import *

""" Iris DataFrame  """
data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)

""" Iris DataFrame Convex Hull Visualization """
plt.figure(figsize = (10, 6))
plt.title('Sepal Length vs Sepal Width')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])
for i in range(len(data.target_names)):
    # Find points (x,y) for each target
    bucket = df[df['Target'] == i]
    length = len(bucket)
    arrXY = []
    for j in range(length):
        try:
            x = bucket.loc[j+bucket.first_valid_index()][0]
            y = bucket.loc[j+bucket.first_valid_index()][1]
        except KeyError:
            continue
        else:
            arrXY.append([x,y])

    # Convex Hull Points
    hull = []
    ConvexHull(arrXY, result = hull)

    # Points Scatter Plot
    arrX, arrY = zip(*arrXY)
    plt.scatter(arrX, arrY, label = data.target_names[i])

    # Convex Hull Plot
    hullX, hullY = zip(*hull)
    plt.plot(hullX, hullY)

plt.legend()
plt.show()
```

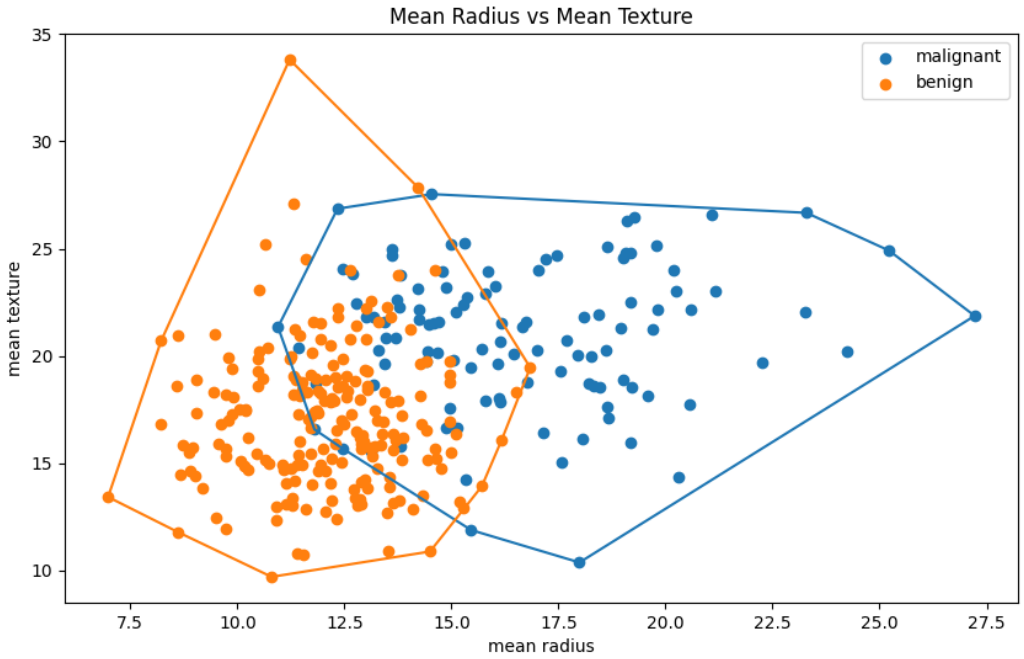
Hasil main_iris_sepal.py



Setosa : *linearly separable* (saling lepas)
Versicolor : *non-linearly separable* (beririsan dengan Virginica)
Virginica : *non-linearly separable* (beririsan dengan Versicolor)

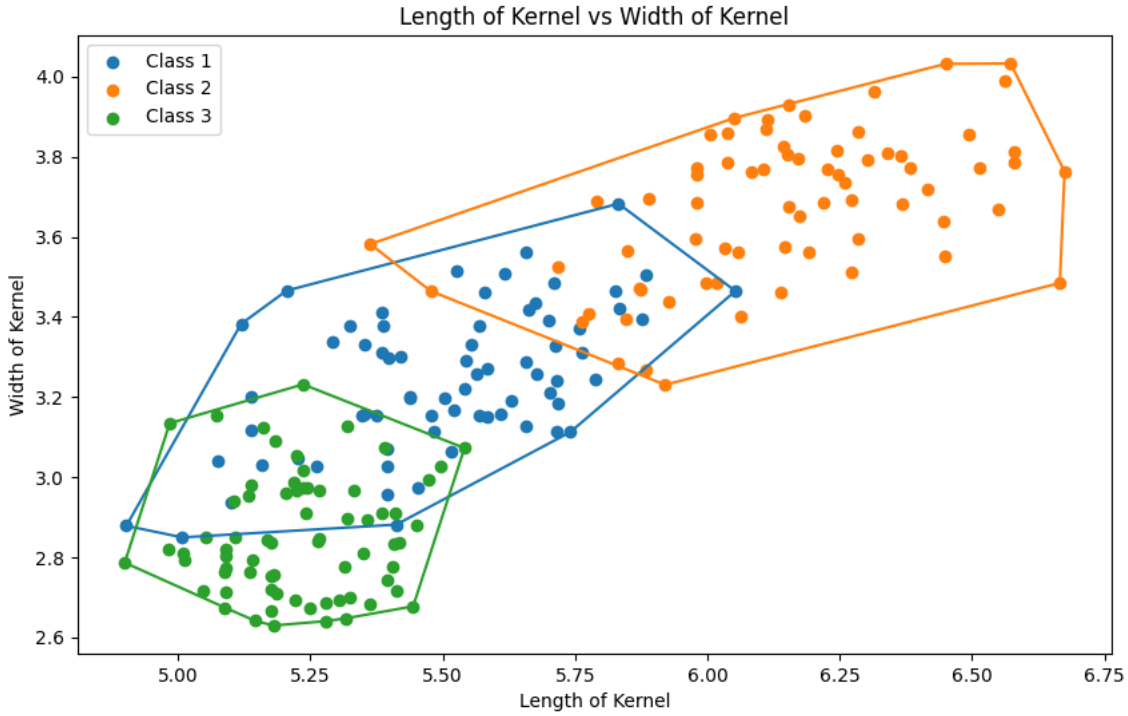
***Convex Hull* untuk dataset lainnya:**
– Kode lengkap dapat dilihat pada link yang terlampir sebelumnya –

Hasil main_breast_cancer.py



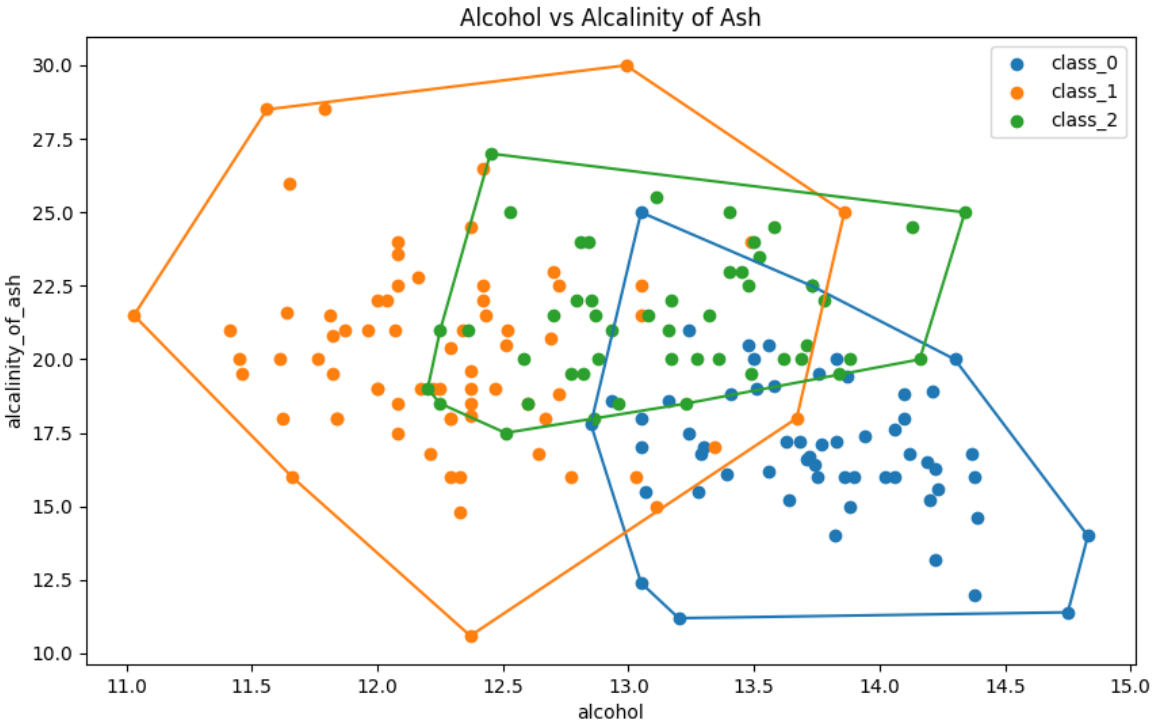
Malignant : *non-linearly separable* (beririsan dengan Benign)
Benign : *non-linearly separable* (beririsan dengan Malignant)

Hasil main_seeds.py



Class 1 : *non-linearly separable* (beririsan dengan Class 2 dan Class 3)
Class 2 : *non-linearly separable* (beririsan dengan Class 1)
Class 3 : *non-linearly separable* (beririsan dengan Class 1)

Hasil main_wine.py



class_0 : *non-linearly separable* (beririsan dengan class_1 dan class_2)
class_1 : *non-linearly separable* (beririsan dengan class_1 dan class_2)
class_2 : *non-linearly separable* (beririsan dengan class_1 dan class_2)

Daftar Pustaka

Brownlee, Jason. 2016. *10 Standard Datasets for Practicing Applied Machine Learning* https://machinelearningmastery.com/standard-machine-learning-datasets/. Diakses pada 27 Februari 2022.

Maulidevi, Nur Ulfa dan Rinaldi Munir. 2022. *Algoritma Divide and Conquer (Bagian 4)*. Bandung : Teknik Informatika ITB

Purdue University. 2020. *Machine Learning I Lecture 06 Linear Separability* https://engineering.purdue.edu/ChanGroup/ECE595/files/Lecture06_separable.pdf Diakses pada 27 Februari 2022.

Lampiran

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.	✓	