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

Disusun Oleh:
Farnas Rozaan Iraqee (13520067)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021/2022

Algoritma *Divide and Conquer*

Berikut ini adalah deskripsi langkah-langkah algoritma *divide and conquer* yang digunakan dalam program *convex hull* ini. Pertama, masukan dari dataset akan terlebih dahulu dikonversi menjadi tipe Point (titik) dan kemudian kumpulan titik tersebut akan diurutkan berdasarkan koordinat axisnya. Apabila koordinat axisnya sama maka akan diurutkan berdasarkan koordinat ordinatnya. Setelah itu, akan diambil dua titik yang berada di paling kiri dan paling kanan untuk membentuk segmen garis yang membagi kumpulan titik menjadi dua bagian, yaitu bagian atas dan bawah segmen garis tersebut. Untuk bisa menentukan titik masuk bagian atas atau bawah akan dilakukan perhitungan determinan terlebih dahulu. Perhitungannya adalah sebagai berikut:

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

Titik (x_3, y_3) berada di atas garis ((x_1, y_1), (x_2, y_2)) jika hasil determinan positif, begitupun sebaliknya. Setelah kumpulan titik sudah terbagi ke dalam dua bagian, maka akan dilakukan proses pembentukan *convex hull* masing-masing pada bagian atas maupun bawah. Untuk bagian atas atau bawah, terdapat dua kemungkinan:

- Jika tidak ada titik di bagian tersebut, maka titik paling kiri dan paling kanan yang menjadi pembentuk *convex hull* bagian tersebut
- Jika ada titik di bagian tersebut, pilih sebuah titik yang memiliki jarak terjauh dari garis yang dibentuk titik paling kiri dan paling kanan.

Semua titik yang berada segitiga yang dibentuk oleh ketiga titik tersebut (titik paling kiri, paling kanan, dan titik terjauh dari segmen garis) akan diabaikan. Kemudian, proses di atas akan dilakukan juga oleh bagian yang yang dibentuk oleh segmen garis yang dibentuk titik paling kiri dan titik terjauh serta titik terjauh dan titik paling kanan. Ulangi proses tersebut sampai tidak ditemukan lagi titik yang berada di bagian luar dan kita akan mendapatkan hasil berupa kumpulan titik yang membentuk *convex hull*. Namun, untuk kode program yang diimplementasi pada tugas ini, hasil *convex hull* akan disimpan di dua *list*, yaitu *list convex hull* bagian atas dan bawah. Hal ini bertujuan untuk mempermudah proses visualisasi nanti.

Source Code Program dalam Bahasa Python

Program ini memiliki 1 *file*, yaitu myConvexHull.ipynb yang di dalamnya sudah berisi kode pustaka *convex hull* dan juga kode untuk visualisasi *dataset*. Namun, kode di bawah ini merupakan kode untuk memvisualisasikan *dataset* iris dengan pasangan atributk *sepal length* dan *sepal width*. Apabila hendak mengganti *dataset* atau pasangan atribut yang lain, yang harus dilakukan adalah mengganti `load_iris` dengan *dataset* lain atau mengubah indeks di `xlabel` dan `ylabel` serta di `iloc bucket` dengan indeks pasangan atribut yang diinginkan untuk divisualisasikan pada fungsi `visualization()`. Berikut ini adalah *source code program*-nya:

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

class Point:
    '''
    Kelas point
    '''
    def __init__(self, x, y):
        self.x = float(x)
        self.y = float(y)

    def __eq__(self, other):
        return self.x == other.x and self.y == other.y

    def __ne__(self, other):
        return not self == other

    def __gt__(self, other):
        if self.x > other.x:
            return True
        elif self.x == other.x:
            return self.y > other.y
        return False

    def __lt__(self, other):
        return not self > other

    def __ge__(self, other):
        if self.x > other.x:
            return True
        elif self.x == other.x:
            return self.y >= other.y
        return False

    def __le__(self, other):
        if self.x < other.x:
            return True
        elif self.x == other.x:
            return self.y <= other.y
```

```
        return False

    def __hash__(self):
        return hash(self.x)

def tuples_to_points(list_tuples):
    '''
    Konversi tipe menjadi point
    '''
    list_points = []
    for tuple in list_tuples:
        list_points.append(Point(tuple[0], tuple[1]))
    return list_points

def determinant(left, right, point):
    '''
    Menghitung determinan yang nantinya nilainya digunakan untuk
mengelompokkan titik point,
    terletak di sebelah atas atau bawah garis yang dibentuk titik left
dan right
    '''
    return (left.x * right.y + right.x * point.y + point.x * left.y) -
(left.y * right.x + right.y * point.x + point.y * left.x)

def construct_convex_hull(points, left, right, hull):
    '''
    Mengupdate hull apabila terdapat point yang memenuhi syarat untuk
dapat
    membentuk convex hull
    '''
    extreme_point = None
    extreme_distance = float("-inf")
    candidate_points = []
    for point in points:
        distance = determinant(left, right, point)
        isClose = np.isclose(distance, 0)
        if (not isClose):
            if (distance > 0):
                candidate_points.append(point)
                if (distance > extreme_distance):
                    extreme_distance = distance
                    extreme_point = point
    if extreme_point:
```

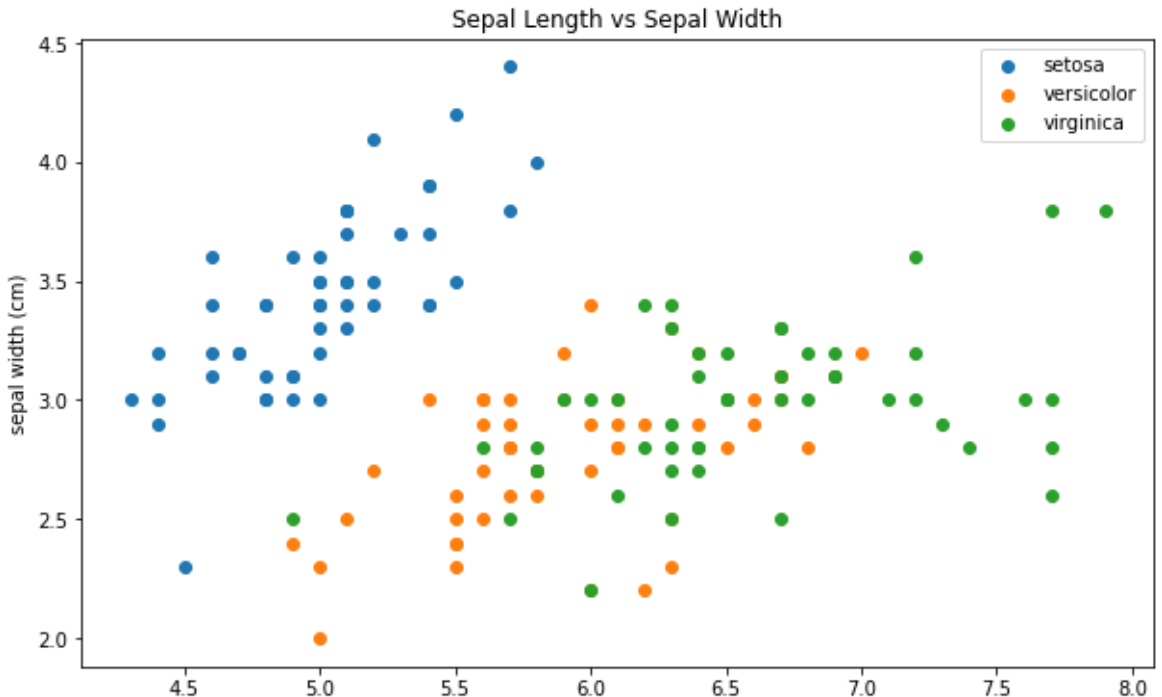
```
        construct_convex_hull(candidate_points, left, extreme_point,
hull)
        hull.append(extreme_point)
        construct_convex_hull(candidate_points, extreme_point, right,
hull)

def convex_hull(points):
    '''
    Menentukan convex hull
    '''
    sorted_points = sorted(tuples_to_points(points))
    total_points = len(points)

    # Mengambil titik yang terletak di absis paling kiri dan kanan
    leftmost_point = sorted_points[0]
    rightmost_point = sorted_points[total_points-1]

    # List convex hull
    upper_res = [leftmost_point, rightmost_point]
    lower_res = [leftmost_point, rightmost_point]
    # Menginisialisasi list untuk menyimpan titik-titik yang berada di
atas dan di bawah
    # garis yang dibentuk titik leftmost_point dan rightmost_point
    upper_area = []
    lower_area = []

    # Menyimpan titik yang memenuhi syarat ke list upper _area atau
lower_area
    for i in range(1, total_points-1):
        distance = determinant(leftmost_point, rightmost_point,
sorted_points[i])
        # Append point ke list upper_area apabila nilai distance > 0
        if (distance > 0):
            upper_area.append(sorted_points[i])
        # Append point ke list lower_area apabila nilai distance <= 0
        else:
            lower_area.append(sorted_points[i])

    # Divide & conquer (mengonstruksi convex hull untuk upper_area dan
lower_area)
    construct_convex_hull(upper_area, leftmost_point, rightmost_point,
upper_res)
```

```
    construct_convex_hull(lower_area, rightmost_point, leftmost_point,
lower_res)

    return sorted(upper_res), sorted(lower_res)

def construct_indices(points, points_convex_hull):
    '''
    Menyimpan indeks points, jika elemen point di points sama dengan
point di points_convex_hull
    '''
    points = tuples_to_points(points)
    indices = []

    for point in points_convex_hull:
        for i in range (len(points)):
            if point == points[i]:
                indices.append(i)

    return indices

def construct_simplices(upper, lower, points):
    '''
    Mengonstruksi simplices (pasangan indeks) dengan tujuan visualisasi
nanti
    '''
    upper_indices = construct_indices(points, upper)
    lower_indices = construct_indices(points, lower)
    simplices = []

    # upper
    for i in range (len(upper_indices)-1):
        simplices.append([upper_indices[i],upper_indices[i+1]])
    simplices.append([upper_indices[i],upper_indices[i+1]])

    #lower
    for i in range (len(lower_indices)-1):
        simplices.append([lower_indices[i],lower_indices[i+1]])
    simplices.append([lower_indices[i],lower_indices[i+1]])

    return simplices

def visualization():
    data = datasets.load_iris()
```

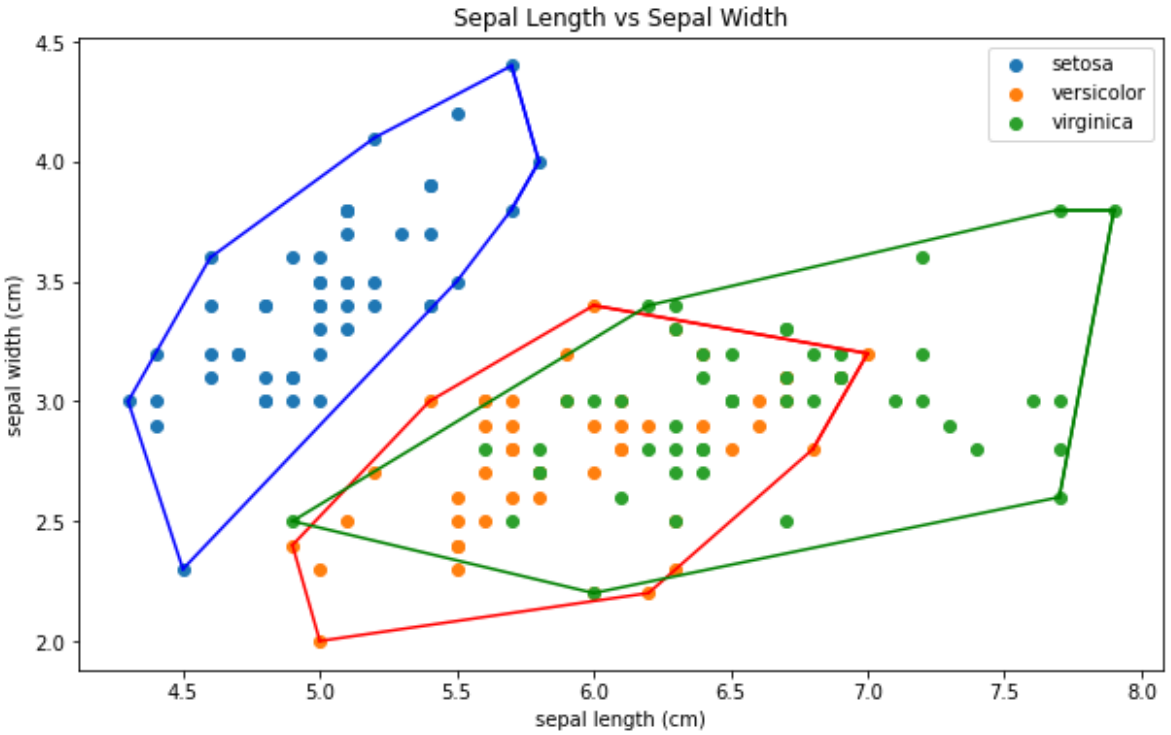
```
    #create a DataFrame
    df = pd.DataFrame(data.data, columns=data.feature_names)
    df['Target'] = pd.DataFrame(data.target)

    #visualisasi hasil myConvexHull
    plt.figure(figsize = (10, 6))
    colors = ['b', 'r', 'g']
    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)):
        bucket = df[df['Target'] == i]
        bucket = bucket.iloc[:,[0,1]].values
        upper, lower = convex_hull(bucket)
        simplices = construct_simplices(upper, lower, bucket)
        plt.scatter(bucket[:,0],bucket[:,1],
label=data.target_names[i])
        for simplex in simplices:
            plt.plot(bucket[simplex, 0], bucket[simplex,1], colors[i])
    plt.legend()

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

Screenshot Input dan *Output*

Berikut adalah visualisasi *dataset* iris dengan pasangan atribut *sepal length* dan *sepal width* sebelum diterapkan *convex hull*:

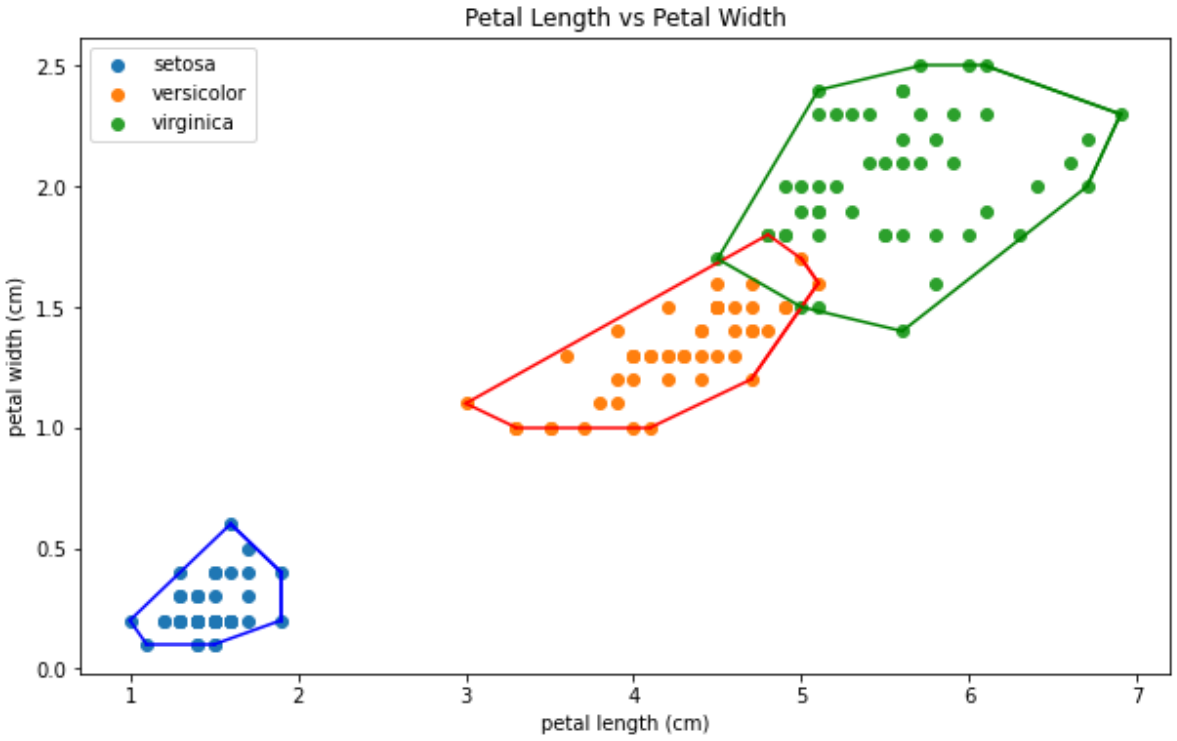


Berikut adalah visualisasi *dataset* iris dengan pasangan atribut *sepal length* dan *sepal width* sesudah diterapkan *convex hull*:

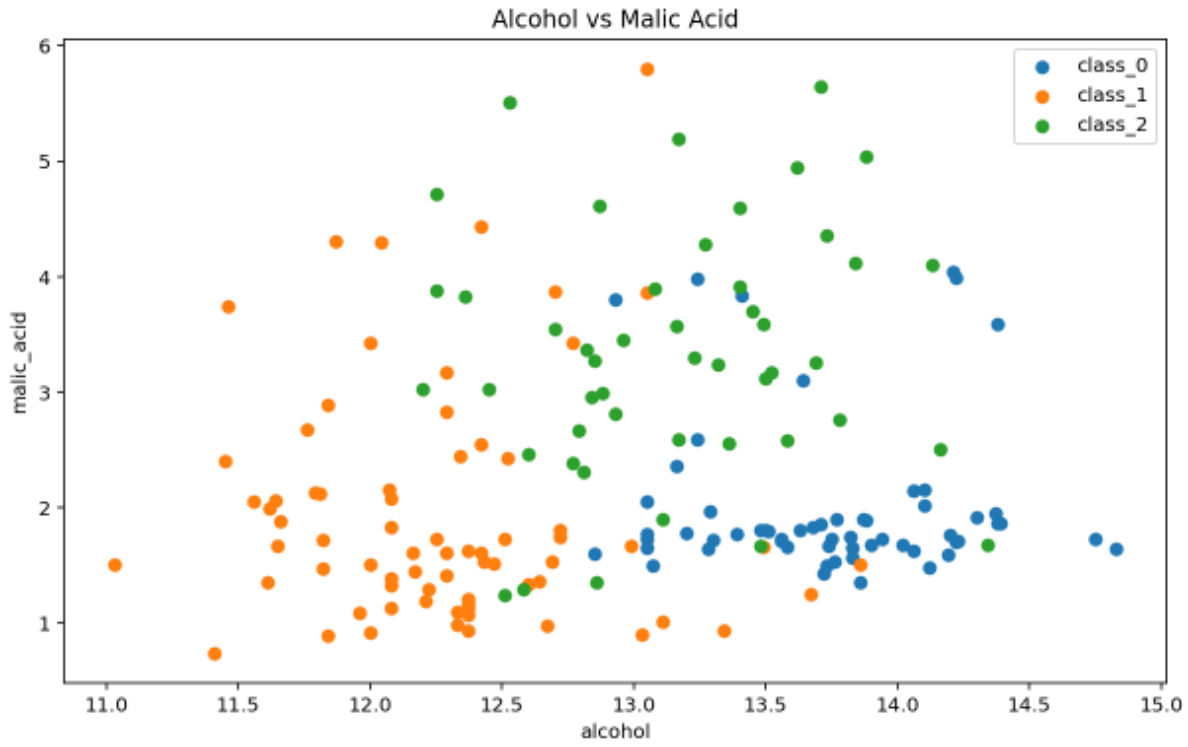


Berikut adalah visualisasi *dataset* iris dengan pasangan atribut *petal length* dan *petal width* sebelum diterapkan *convex hull*:

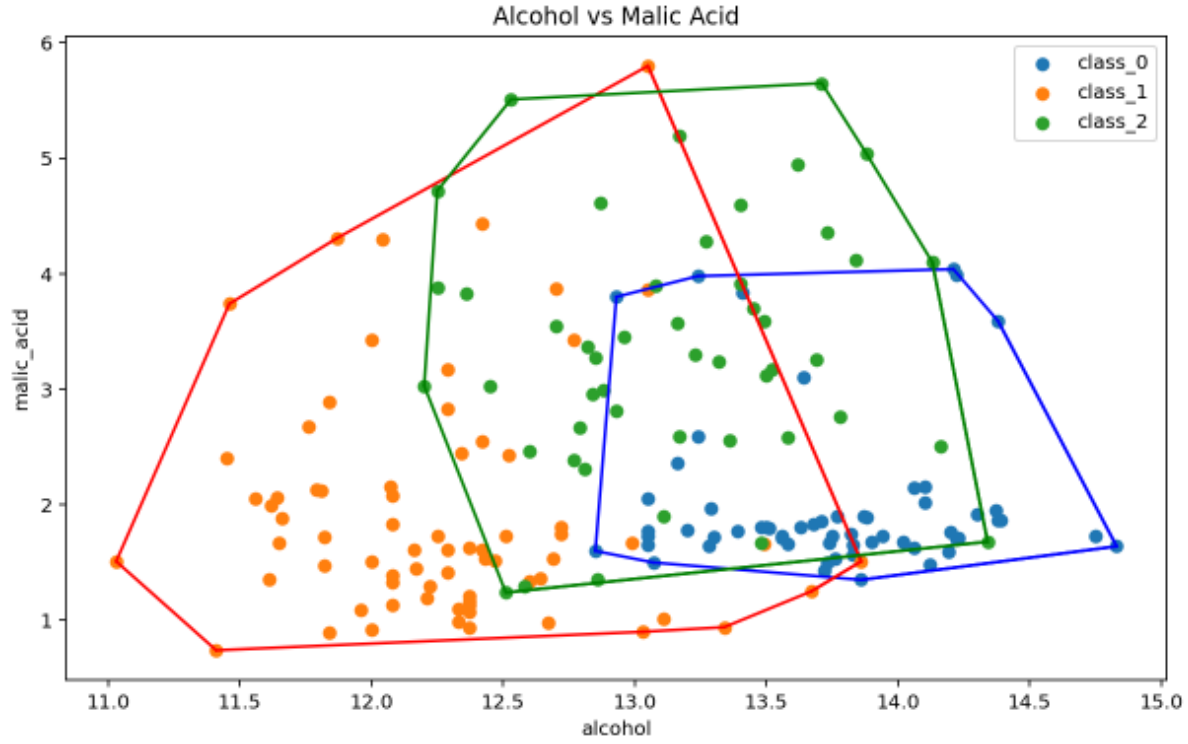
Berikut adalah visualisasi *dataset* iris dengan pasangan atribut *petal length* dan *petal width* sesudah diterapkan *convex hull*:



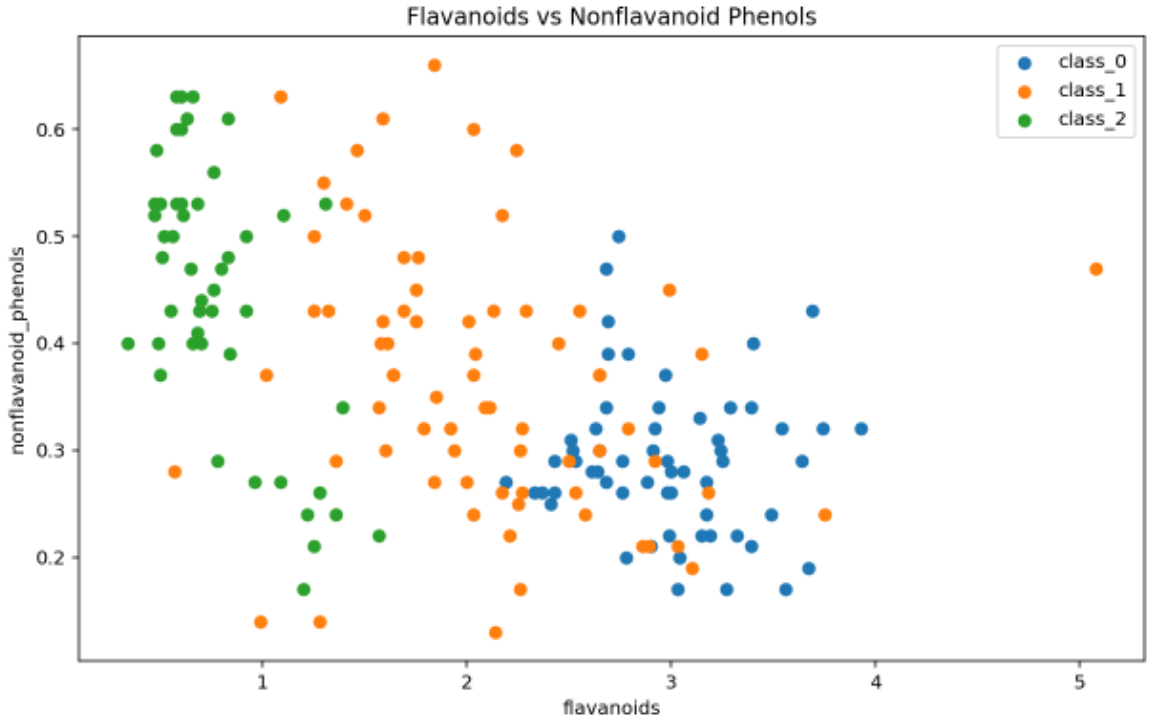
Berikut adalah visualisasi *dataset* wine dengan pasangan atribut *alcohol* dan *malic acid* sebelum diterapkan *convex hull*:



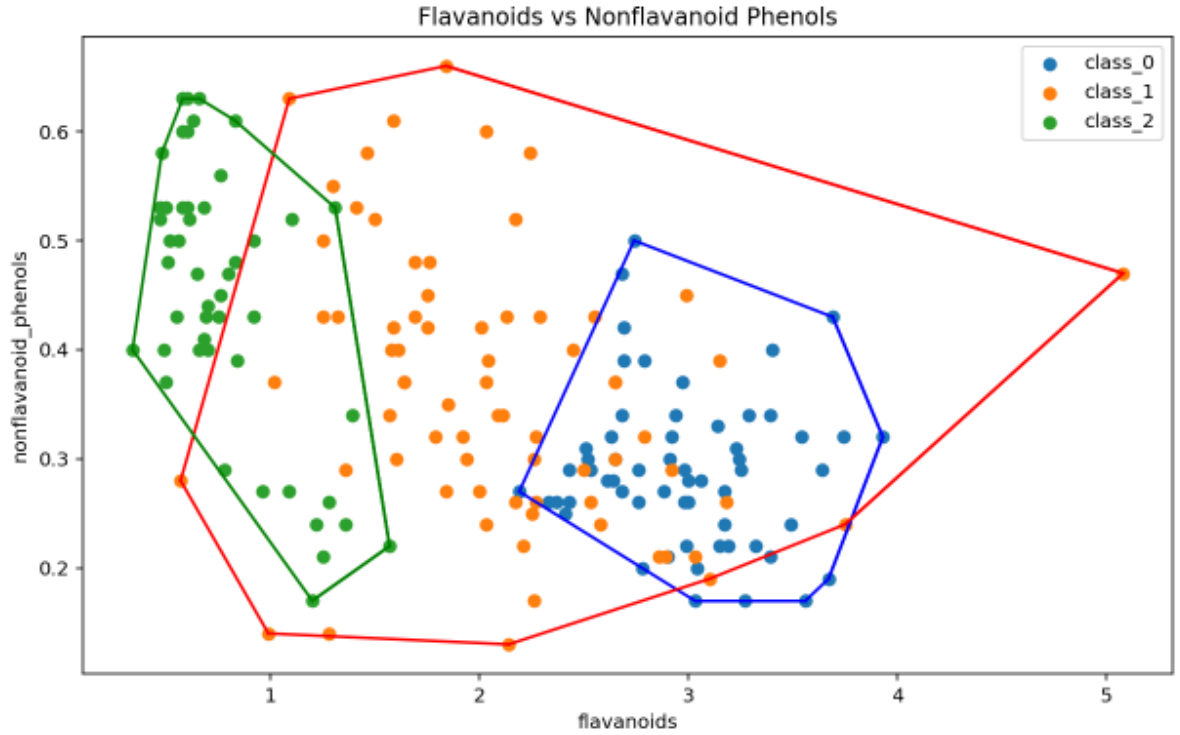
Berikut adalah visualisasi *dataset* wine dengan pasangan atribut *alcohol* dan *malic acid* setelah diterapkan *convex hull*:



Berikut adalah visualisasi *dataset* wine dengan pasangan atribut *flavanoids* dan *nonflavanoid phenols* sebelum diterapkan *convex hull*:



Berikut adalah visualisasi *dataset* wine dengan pasangan atribut *flavanoids* dan *nonflavanoid phenols* setelah diterapkan *convex hull*:



Repository Github

Berikut adalah link repository github yang akan diatur ke public setelah pengumpulan:
https://github.com/OjaanIr/TucilStima-ConvexHull

Status Program

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	✓	