Laporan Tugas Kecil 2

IF2211 Strategi Algoritma

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

oleh

Raden Rifqi Rahman (13520166)

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

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

Algoritma *divide and conquer* untuk menghitung *convex hull*

Untuk menghitung *convex hull* dari sebuah himpunan titik, dilakukan langkah-langkah sebagai berikut. Misalkan himpunan titik yang dimaksud adalah *P*. Jika *P* hanya terdiri atas 1 titik, maka titik tersebut adalah *convex hull* dari *P*. Jika tidak, lakukan langkah-langkah berikut.

Pada langkah pertama, cari titik yang berada di ujung kiri bawah '*bottom-leftmost*' dan ujung kanan atas '*top-rightmost*'. Hubungkan kedua titik tersebut sehingga membentuk sebuah garis *d*. Jika *d* adalah garis vertikal, maka *d* adalah *convex hull* dari *P*. Jika tidak, lanjutkan ke langkah berikut.

Buat sebuah himpunan titik, misalnya *C*, dan masukkan kedua titik sebelumnya ke dalam *C*. Bagi titik-titik pada *P* menjadi 2 bagian, yaitu titik-titik yang berada di atas *d*, misalkan *U*, dan di bawah *d*, misalkan *L*.

Untuk setiap titik pada *U*, cari titik dengan jarak terjauh dari *d*. Jika terdapat beberapa titik yang memiliki jarak yang sama, pilih titik yang membentuk sudut paling besar dengan kedua titik pada *d*. Secara programatik, hal ini dilakukan dengan memilih titik yang memiliki jarak terpendek ke titik tengah '*midpoint*' dari *d*. Hubungkan titik ini ke kedua titik pada *d* sehingga terbentuk 2 garis baru dan masukkan titik tersebut ke dalam *C*. Untuk setiap garis, jika masih terdapat titik di **atas** garis tersebut, ulangi langkah pada paragraf ini untuk garis tersebut dengan analogi *U* adalah himpunan titik-titik di atas garis tersebut, dan *d* adalah garis tersebut.

Untuk setiap titik pada *L*, lakukan hal yang sama dengan titik-titik pada *U*, hanya saja untuk setiap garis baru yang terbentuk, pengecekan yang dilakukan adalah apakah masih terdapat titik di **bawah** garis tersebut.

Setelah langkah-langkah untuk setiap titik pada *U* dan *L* dilakukan, maka *C* adalah himpunan titik yang merupakan *subset* dari *P* yang membentuk *convex hull* dari *P*.

Source code library

Berikut adalah *source code library* **myConvexHull** dalam bahasa Python yang telah diminifikasi untuk menghitung titik-titik pembentuk *convex hull* dari himpunan titik dengan algoritma *divide and conquer*.

(*Keterangan: penamaan fungsi yang berawalan '_' berarti hanya untuk digunakan secara internal oleh modul dan tidak untuk digunakan oleh pengguna modul, sesuai dengan pedoman pada PEP8.*)

`myConvexHull/__init__.py`

```
import numpy as np
from myConvexHull.point_utils import *
from myConvexHull.dtype import *
from myConvexHull.line_utils import *
from enum import Enum
```

```
def convex_hull(points, base_line=None, direction=None) -> Points:
    if base_line:
        if len(points) == 0:
            return np.ndarray([0, 2])
        if len(points) == 1:
            return points
        (farthest_point, index) = get_farthest_point(points, base_line)
        points = np.delete(points, [index], axis=0)
        new_base_line_a = (base_line[0], farthest_point)
        new_base_line_b = (farthest_point, base_line[1])
        func = get_upper_points if direction == Direction.UPWARDS else
get_lower_points
        if is_vertical(new_base_line_a):
            chl = np.ndarray([0, 2])
        else:
            new_points_a = func(points, new_base_line_a)
            chl = convex_hull(new_points_a, new_base_line_a, direction)
        if is_vertical(new_base_line_b):
            chr = np.ndarray([0, 2])
        else:
            new_points_b = func(points, new_base_line_b)
            chr = convex_hull(new_points_b, new_base_line_b, direction)

        hull = merge(chl, farthest_point, chr, direction)
    else:
        (leftmost_point, lmindex) = _get_leftmost_point(points)
        (rightmost_point, rmindex) = _get_rightmost_point(
            points
        )
        points = np.delete(points, [lmindex, rmindex], axis=0)
        line: Line = (leftmost_point, rightmost_point)
        if is_vertical(line):
            upper_points = np.ndarray([0, 2])
            lower_points = np.ndarray([0, 2])
        else:
            (upper_points, lower_points) = split(points, line)

        chu = convex_hull(upper_points, line, Direction.UPWARDS)
        chl = convex_hull(lower_points, line, Direction.DOWNWARDS)
        hull = first_merge(leftmost_point, chu, rightmost_point, chl)
    return hull
def split(points, line) -> tuple[Points, Points]:
    upper_points = get_upper_points(points, line)
    lower_points = get_lower_points(points, line)
    return (upper_points, lower_points)
def first_merge(left_vertex, upper_vertices, right_vertex, lower_vertices) ->
Points:
    hull = np.ndarray([0, 2])
```

```
    hull = np.append(hull, [left_vertex], axis=0)
    hull = np.append(hull, upper_vertices, axis=0)
    hull = np.append(hull, [right_vertex], axis=0)
    hull = np.append(hull, lower_vertices, axis=0)
    hull = np.append(hull, [left_vertex], axis=0)
    return hull
def merge(left_vertices, mid_vertex, right_vertices, direction) -> Points:
    hull = np.ndarray([0, 2])
    if direction == Direction.UPWARDS:
        hull = np.append(hull, left_vertices, axis=0)
        hull = np.append(hull, [mid_vertex], axis=0)
        hull = np.append(hull, right_vertices, axis=0)
    else:
        hull = np.append(hull, right_vertices, axis=0)
        hull = np.append(hull, [mid_vertex], axis=0)
        hull = np.append(hull, left_vertices, axis=0)
    return hull
def random_color() -> str:
    r = hex(np.random.randint(0, 256))[2:]
    g = hex(np.random.randint(0, 256))[2:]
    b = hex(np.random.randint(0, 256))[2:]
    if len(r) == 1:
        r = "0" + r
    if len(g) == 1:
        g = "0" + g
    if len(b) == 1:
        b = "0" + b
    return f"#{r}{g}{b}"
def _get_leftmost_point(points):
    if len(points) == 0:
        return None
    leftmost_point: Point = None
    index = 0
    for i in range(len(points)):
        point = points[i]
        if leftmost_point is None or less_than(point, leftmost_point):
            leftmost_point = point
            index = i
    return (leftmost_point, index)
def _get_rightmost_point(points):
    if len(points) == 0:
        return None
    rightmost_point: Point = None
    index = 0
    for i in range(len(points)):
        point = points[i]
        if rightmost_point is None or greater_than(point, rightmost_point):
            rightmost_point = point
```

```
            index = i
    return (rightmost_point, index)
class Direction(Enum):
    UPWARDS = 0
    DOWNWARDS = 1
```

myConvexHull/dtype.py

```
from typing import TypeAlias
import numpy.typing as npt

Point: TypeAlias = npt.NDArray
Points: TypeAlias = npt.NDArray
Line: TypeAlias = tuple[npt.NDArray, npt.NDArray]
NullableLine: TypeAlias = Line | None
```

myConvexHull/line_utils.py

```
from myConvexHull.dtype import Point, Points, Line
from myConvexHull.point_utils import X, Y, distance
import numpy as np
def get_upper_points(points, line) -> Points:
    upper_points = np.ndarray([0, 2])
    for point in points:
        if _is_above_line(point, line):
            upper_points = np.append(upper_points, [point], axis=0)
    return upper_points
def get_lower_points(points, line) -> Points:
    lower_points = np.ndarray([0, 2])
    for point in points:
        if _is_below_line(point, line):
            lower_points = np.append(lower_points, [point], axis=0)
    return lower_points
def get_farthest_point(points, line) -> Point:
    farthest_point = points[0]
    farthest_distance = _distance(farthest_point, line)
    index = 0
    for i in range(len(points)):
        point = points[i]
        dist = _distance(point, line)
        if dist > farthest_distance or (
            dist == farthest_distance and
            distance(point, _midpoint(line)) <
            distance(farthest_point, _midpoint(line))
        ):
            farthest_distance = dist
            farthest_point = point
            index = i
    return (farthest_point, index)
```

```
def is_vertical(line) -> bool:
    return line[0][X] == line[1][X]
def _is_above_line(point, line):
    slope = _slope(line[0], line[1])
    offset = _offset(line[0], slope)
    return point[Y] - slope * point[X] > offset
def _is_below_line(point, line):
    slope = _slope(line[0], line[1])
    offset = _offset(line[0], slope)
    return point[Y] - slope * point[X] < offset
def _slope(a: Point, b: Point) -> float:
    return (b[Y] - a[Y]) / (b[X] - a[X])
def _offset(a: Point, slope: float) -> float:
    return a[1] - slope * a[0]
def _distance(point, line):
    _point = np.array(point)
    a = np.array(line[0])
    b = np.array(line[1])
    return np.abs(np.cross(b - a, _point - a) / np.linalg.norm(b - a))
def _midpoint(line):
    return (line[0] + line[1]) / 2
```

myConvexHull/points_utils.py

```
import numpy as np
from myConvexHull.dtype import Point
Y = 1
X = 0
def less_than(a, b) -> bool:
    return a[X] < b[X] or (a[X] == b[X] and a[Y] < b[Y])
def greater_than(a, b) -> bool:
    return a[X] > b[X] or (a[X] == b[X] and a[Y] > b[Y])
def distance(a, b) -> float:
    return np.sqrt((a[X] - b[X]) ** 2 + (a[Y] - b[Y]) ** 2)
```

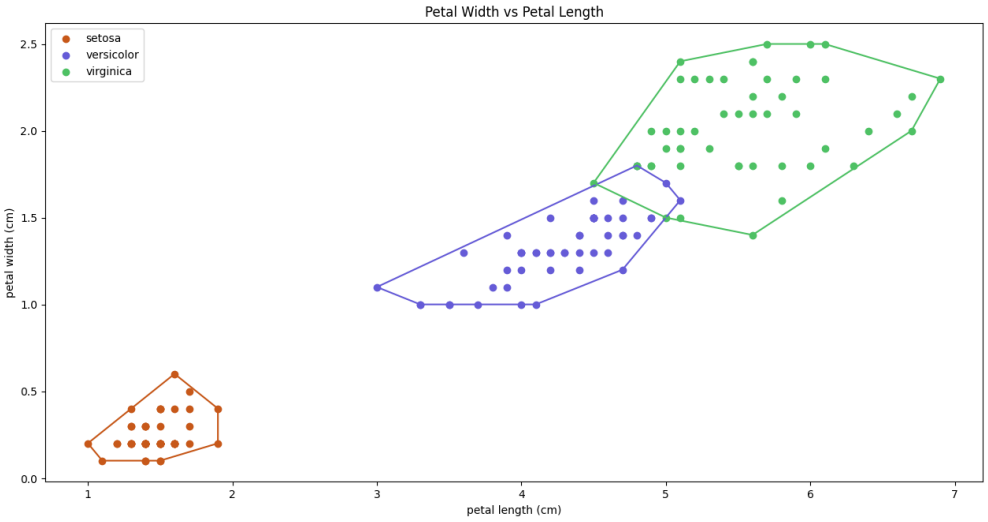
Contoh penggunaan *library*

Library tersebut telah diuji dengan 8 kasus uji yang berasal dari 4 *dataset* yang berbeda. Berikut adalah *script* dan *output* dalam pengujian *library*.

1. Atribut *Petal Width* vs *Petal Length* dari *dataset iris*

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import convex_hull, random_color
from myConvexHull.point_utils import X, Y

data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
plt.figure(figsize=(10, 6))
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 = convex_hull(bucket)
    color = random_color()
    plt.scatter(bucket[:, 0], bucket[:, 1],
                label=data.target_names[i], color=color)
    hull = np.transpose(hull)
    plt.plot(hull[X], hull[Y], color=color)

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

Gambar 1. *Script* uji dengan atribut *Petal Width* dan *Petal Length* dari *dataset iris*.

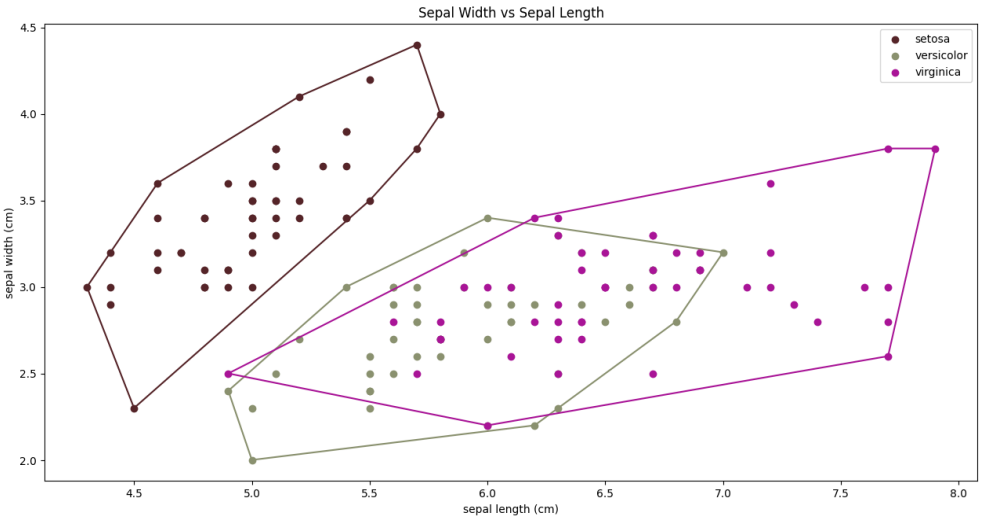


Gambar 2. *Output* pengujian dengan atribut *Petal Width* dan *Petal Length* dari *dataset iris*.

2. Atribut *Sepal Width* vs *Sepal Length* dari *dataset iris*

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import convex_hull, random_color
from myConvexHull.point_utils import X, Y

data = datasets.load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
plt.figure(figsize=(10, 6))
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 = convex_hull(bucket)
    color = random_color()
    plt.scatter(bucket[:, 0], bucket[:, 1],
                label=data.target_names[i], color=color)
    hull = np.transpose(hull)
    plt.plot(hull[X], hull[Y], color=color)

plt.legend()
plt.show()

```

Gambar 3. *Script* uji dengan atribut *Sepal Width* dan *Sepal Length* dari *dataset iris*.

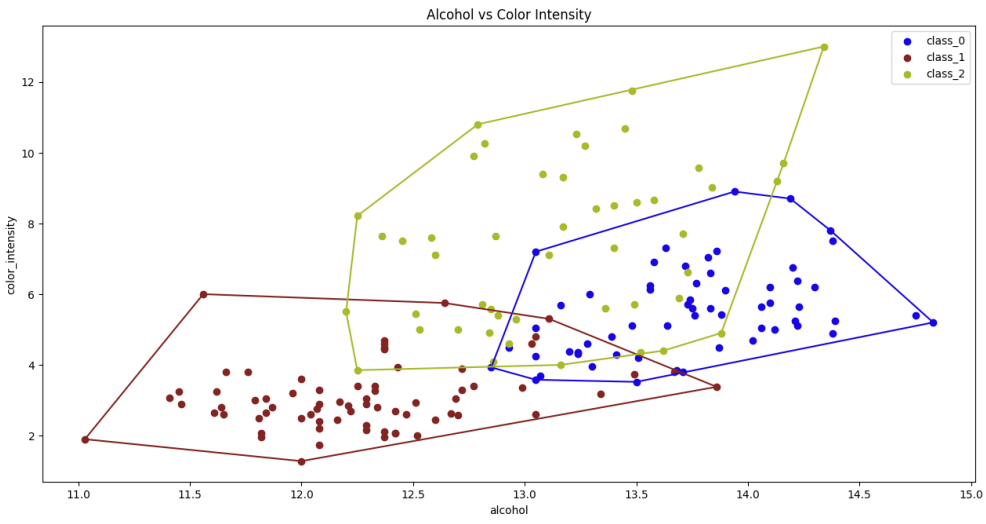


Gambar 4. *Output* pengujian dengan atribut *Sepal Width* dan *Sepal Length* dari *dataset iris*.

3. Atribut *Alcohol* vs *Color Intensity* dari *dataset wine*

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import convex_hull, random_color
from myConvexHull.point_utils import X, Y

data = datasets.load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
plt.figure(figsize=(10, 6))
plt.title('Alcohol vs Color Intensity')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[9])

for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:, [0, 9]].values
    hull = convex_hull(bucket)
    color = random_color()
    plt.scatter(bucket[:, 0], bucket[:, 1],
                label=data.target_names[i], color=color)
    hull = np.transpose(hull)
    plt.plot(hull[X], hull[Y], color=color)

plt.legend()
plt.show()

```

Gambar 5. *Script* uji dengan atribut *Alcohol* dan *Color Intensity* dari *dataset wine*.

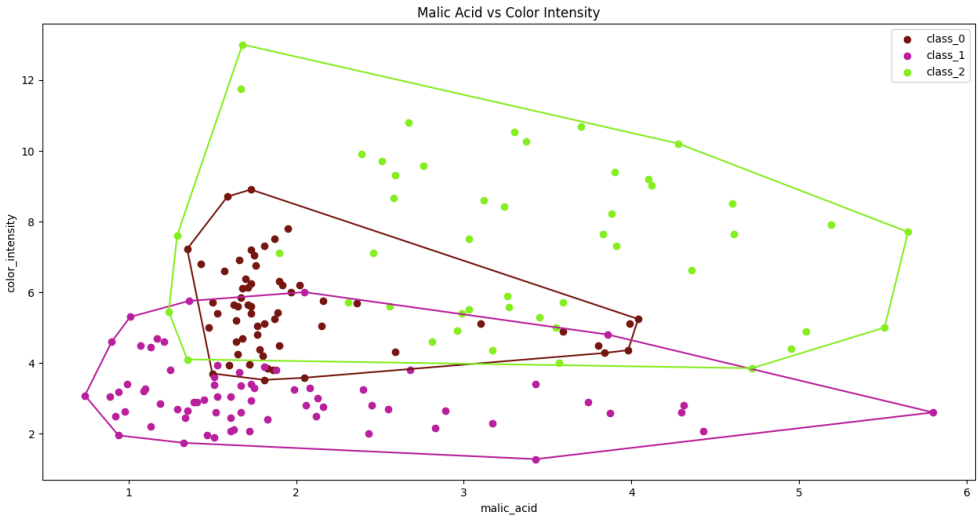


Gambar 6. *Output* pengujian dengan atribut *Alcohol* dan *Color Intensity* dari *dataset wine*.

4. Atribut *Malic Acid* vs *Color Intensity* dari *dataset wine*

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import convex_hull, random_color
from myConvexHull.point_utils import X, Y

data = datasets.load_wine()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
plt.figure(figsize=(10, 6))
plt.title('Malic Acid vs Color Intensity')
plt.xlabel(data.feature_names[1])
plt.ylabel(data.feature_names[9])

for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:, [1, 9]].values
    hull = convex_hull(bucket)
    color = random_color()
    plt.scatter(bucket[:, 0], bucket[:, 1],
                label=data.target_names[i], color=color)
    hull = np.transpose(hull)
    plt.plot(hull[X], hull[Y], color=color)

plt.legend()
plt.show()

```

Gambar 7. *Script* uji dengan atribut *Malic Acid* dan *Color Intensity* dari *dataset wine*.

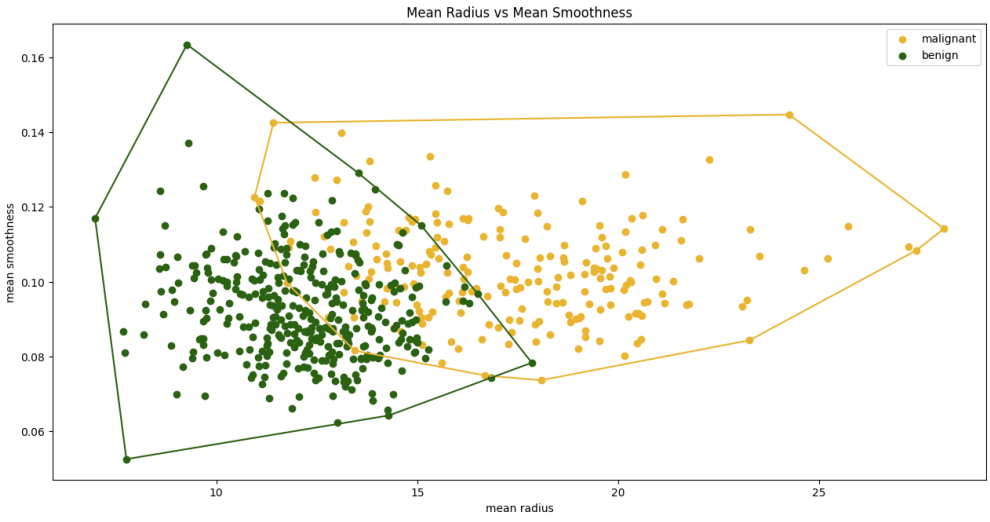


Gambar 8. *Output* pengujian dengan atribut *Malic Acid* dan *Color Intensity* dari *dataset wine*.

5. Atribut *Mean Radius* vs *Mean Smoothness* dari *dataset breast cancer*

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import convex_hull, random_color
from myConvexHull.point_utils import X, Y

data = datasets.load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
plt.figure(figsize=(10, 6))
plt.title('Mean Radius vs Mean Smoothness')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[4])

for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:, [0, 4]].values
    hull = convex_hull(bucket)
    color = random_color()
    plt.scatter(bucket[:, 0], bucket[:, 1],
                label=data.target_names[i], color=color)

    hull = np.transpose(hull)
    plt.plot(hull[X], hull[Y], color=color)

plt.legend()
plt.show()

```

Gambar 9. *Script* uji dengan atribut *Mean Radius* dan *Mean Smoothness* dari *dataset breast cancer*.

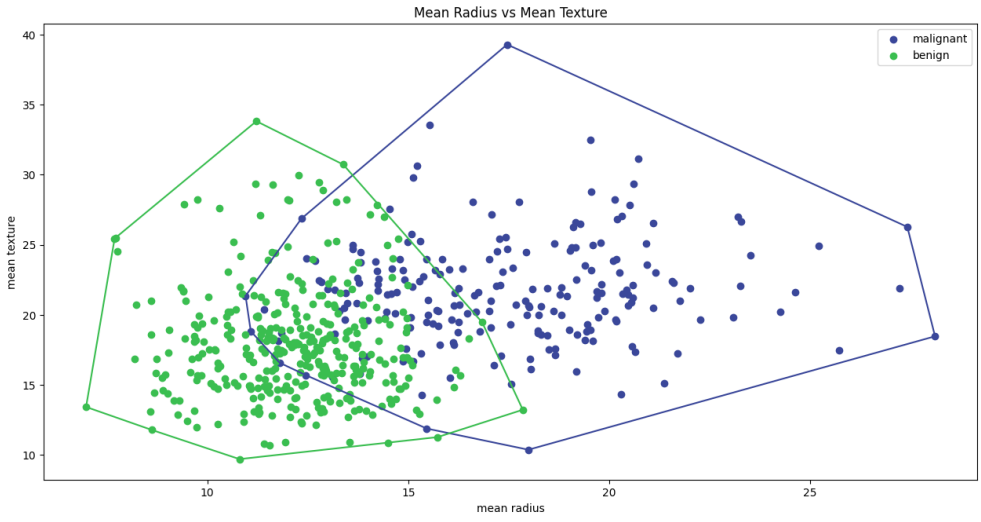


Gambar 10. *Output* pengujian dengan atribut *Mean Radius* dan *Mean Smoothness* dari *dataset breast cancer*.

6. Atribut *Mean Radius* vs *Mean Texture* dari *dataset breast cancer*

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import convex_hull, random_color
from myConvexHull.point_utils import X, Y

data = datasets.load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
plt.figure(figsize=(10, 6))
plt.title('Mean Radius vs Mean Texture')
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 = convex_hull(bucket)
    color = random_color()
    plt.scatter(bucket[:, 0], bucket[:, 1],
                label=data.target_names[i], color=color)
    hull = np.transpose(hull)
    plt.plot(hull[X], hull[Y], color=color)

plt.legend()
plt.show()

```

Gambar 11. *Script* uji dengan atribut *Mean Radius* dan *Mean Texture* dari *dataset breast cancer*.

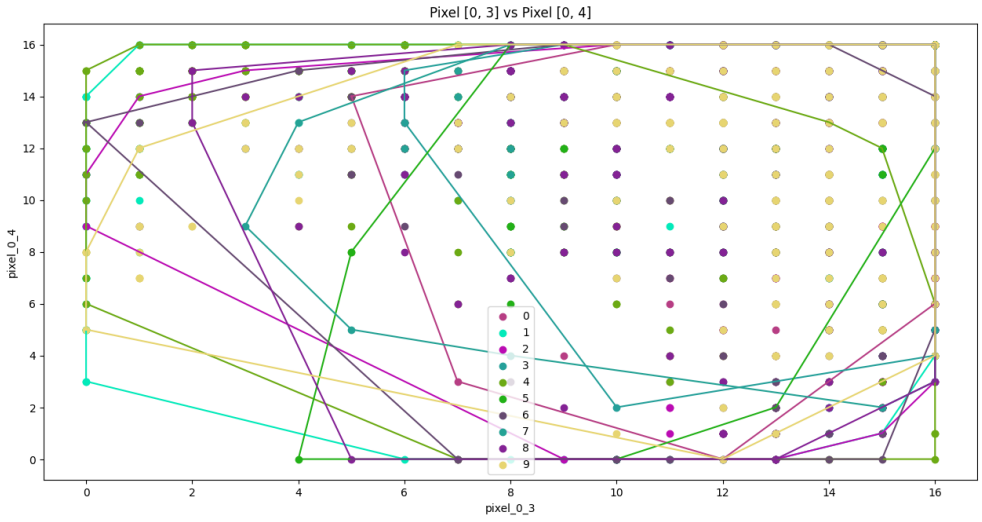


Gambar 12. *Output* pengujian dengan atribut *Mean Radius* dan *Mean Texture* dari *dataset breast cancer*.

7. Atribut *pixel_0_3* vs *pixel_0_4* dari *dataset digits*

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import convex_hull, random_color
from myConvexHull.point_utils import X, Y

data = datasets.load_digits()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
plt.figure(figsize=(10, 6))
plt.title('Pixel [0, 3] vs Pixel [0, 4]')
plt.xlabel(data.feature_names[3])
plt.ylabel(data.feature_names[4])

for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:, [3, 4]].values
    hull = convex_hull(bucket)
    color = random_color()
    plt.scatter(bucket[:, 0], bucket[:, 1],
                label=data.target_names[i], color=color)
    hull = np.transpose(hull)
    plt.plot(hull[X], hull[Y], color=color)

plt.legend()
plt.show()

```

Gambar 13. *Script* uji dengan atribut *pixel_0_3* dan *pixel_0_4* dari *dataset digits*.

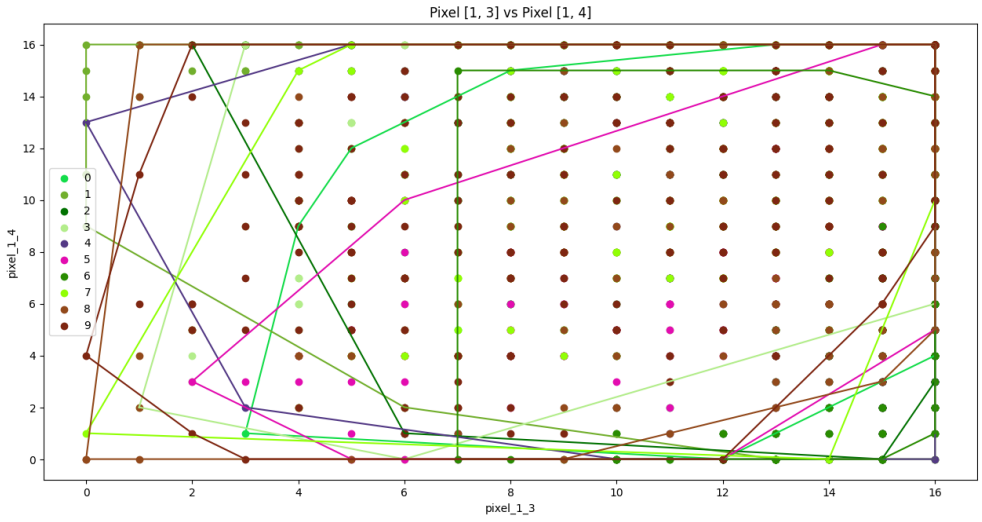


Gambar 14. *Output* pengujian dengan atribut *pixel_0_3* dan *pixel_0_4* dari *dataset digits*.

8. Atribut *pixel_1_3* vs *pixel_1_4* dari *dataset digits*

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import convex_hull, random_color
from myConvexHull.point_utils import X, Y

data = datasets.load_digits()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
plt.figure(figsize=(10, 6))
plt.title('Pixel [1, 3] vs Pixel [1, 4]')
plt.xlabel(data.feature_names[11])
plt.ylabel(data.feature_names[12])

for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:, [11, 12]].values
    hull = convex_hull(bucket)
    color = random_color()
    plt.scatter(bucket[:, 0], bucket[:, 1],
                label=data.target_names[i], color=color)
    hull = np.transpose(hull)
    plt.plot(hull[X], hull[Y], color=color)

plt.legend()
plt.show()

```

Gambar 15. *Script* uji dengan atribut *pixel_1_3* dan *pixel_1_4* dari *dataset digits*.



Gambar 16. *Output* pengujian dengan atribut *pixel_1_3* dan *pixel_1_4* dari *dataset digits*.

Selain 8 kasus uji di atas, *library* yang dibuat juga diuji secara acak '*random*' dengan konfigurasi yang dapat diatur. Berikut adalah *source code script* pengujian secara acak.

test/randomized.py

```python
import numpy as np
import matplotlib.pyplot as plt
import argparse
from myConvexHull import convex_hull, random_color
from myConvexHull.point_utils import X, Y
def throw():
    print(f"Incorrect argument format : {config}")
    print(f"Usage: [config [config [config [...]]]]")
    print()
    print(f"\tconfig\t\"[<x-min>, <x-max>, <y-min>, <y-max>, <count>]\"")
    print(f"\texample: \"[-50, 0, -50, -25, 15]\"")
    print()
    exit(1)
parser = argparse.ArgumentParser()
parser.add_argument("cfgs", nargs=argparse.REMAINDER)
args = parser.parse_args()
configs = []
cfgs = args.cfgs
if len(cfgs) == 0:
    configs.append([-50, 0, -50, -25, 15])
    configs.append([-25, 25, -10, 10, 15])
    configs.append([0, 50, 25, 50, 15])
for config in cfgs:
    try:
        config = eval(config)
    except:
        throw()
    if type(config) != list:
        throw()
    if len(config) != 5:
        throw()
    for i in range(5):
        if type(config[i]) != int:
            throw()
    configs.append(config)
plt.title('Randomized Testing')
plt.xlabel("x")
plt.ylabel("y")
for config in configs:
    x = np.random.randint(config[0], config[1], [config[4], 1])
    y = np.random.randint(config[2], config[3], [config[4], 1])
    points = x
    points = np.append(points, y, axis=1)
    hull = convex_hull(points)
    points = np.transpose(points)
    hull = np.transpose(hull)
    color = random_color()
```

```
    plt.scatter(points[X], points[Y], color=color)
    plt.plot(hull[X], hull[Y], color=color)
plt.show()
```

Untuk melakukan pengujian secara acak '*randomized testing*', siapkan satu atau beberapa konfigurasi data acak. Setiap konfigurasi mewakili satu kelompok data yang akan dicari *convex hull* dari plotnya. Konfigurasi data acak terdiri atas (1) nilai *x* minimum, (2) nilai *x* maksimum, (3) nilai *y* minimum, (4) nilai *y* maksimum, dan (5) banyak data; yang ditulis dalam format *list* sebagai berikut.

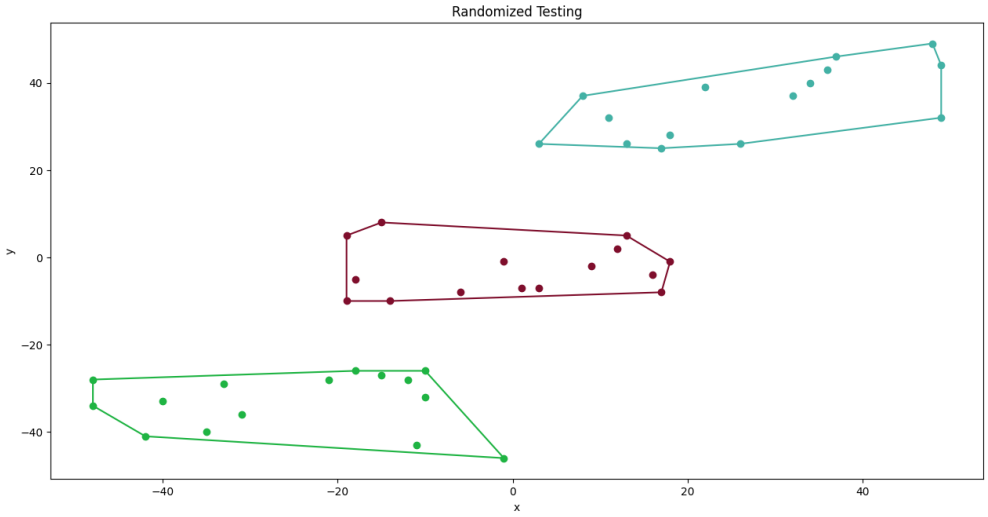
```
[<x-min>, <x-max>, <y-min>, <y-max>, <banyak-data>]
```

Selanjutnya, jalankan perintah berikut untuk melakukan pengujian.

```
py test/randomized.py [config [config [config [...]]]]
```

Sebagai contoh, berikut adalah hasil *randomized testing* yang dilakukan dengan konfigurasi default, yaitu 3 kelompok data yang masing-masing memiliki konfigurasi `[-50, 0, -50, -25, 15]`, `[-25, 25, -10, 10, 15]`, `[0, 50, 25, 50, 15]`; atau sama dengan menjalankan perintah di bawah ini.

```
py test/randomized.py "[-50,0,-50,-25,15]" "[-25,25,-10,10,15]"
"[0, 50,25,50,15]"
```



Gambar 17. *Output randomized testing* dengan konfigurasi *default*.

Alamat *repository source code library*

Source code library yang telah dibuat dapat diakses melalui tautan ini.

Check list

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.	✓	