TUGAS KECIL

Implementasi Algoritma Divide and Conquer dalam Pencarian Convex Hull sebagai Pengujian Linear Separability

LAPORAN

Diajukan sebagai salah satu tugas mata kuliah IF2211 Strategi Algoritma pada

Semester II

Tahun Akademik 2021-2022

oleh

Owen Christian Wijaya 13520124

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

DAFTAR ISI

BAB I. ALGORITMA *DIVIDE AND CONQUER*

Dalam tugas kecil ini, algoritma *divide and conquer* digunakan untuk menemukan titik-titik dalam suatu kumpulan data dua-dimensi (x, y) yang membentuk sebuah *convex hull*. *Convex hull* adalah sebuah bangun datar cembung yang dibangun mengelilingi kumpulan data titik dalam suatu *scatter plot. Convex hull* umumnya digunakan untuk meninjau apakah dua buah set data dapat dipisahkan dengan suatu garis linear (*linear separability*). Dua set data disebut *linearly separable* (dapat dipisahkan secara linear) apabila *convex hull* yang dibentuk oleh kedua data tersebut tidak beririsan.

Algoritma yang digunakan diterapkan dalam *class* Convex, fungsi ConvexHull. Fungsi ConvexHull menerima suatu list berisi koordinat (x, y) dan mengubah list tersebut ke sebuah list berisi ADT Point. Fungsi akan mengurutkan list tersebut berdasarkan koordinat absis sehingga dapat ditemukan titik dengan absis minimum (`minAbs`) dan maksimum (`maxAbs`). Selanjutnya, fungsi akan membagi titik-titik yang tersisa ke bagian kiri dan kanan garis tergantung determinan titik tersebut dengan titik absis minimum dan maksimum menggunakan persamaan berikut:

$$\begin{vmatrix} minAbs.x & minAbs.y & 1 \\ maxAbs.x & maxAbs.y & 1 \\ p.x & p.y & 1 \end{vmatrix}$$

Titik p akan dikategorikan sebagai titik di sebelah kiri garis apabila hasil determinan positif. Sebaliknya, titik p akan dikategorikan berada di kanan garis apabila hasil determinan negatif. Apabila hasil determinan 0, maka titik tersebut berada di garis yang sama dengan garis yang dibentuk titik absis minimum dan maksimum, sehingga titik tersebut tidak dapat membentuk suatu *convex hull* dan tidak perlu dikategorikan.

Selanjutnya, fungsi akan mencari titik terjauh (`pMax`) dari bagian kiri dan bagian kanan. Proses pembagian di sisi kiri dilakukan pada fungsi `divideLeft`, sementara proses pembagian di si kanan dilakukan pada fungsi `divideRight`. Kedua fungsi ini bersifat rekursif, dengan basis apabila list kosong maka proses rekursi akan dihentikan. Sebaliknya, fungsi akan mencari titik terjauh dari garis tersebut menggunakan fungsi find`PMax`. Fungsi akan mencari jarak antara garis dengan titik menggunakan persamaan:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} \quad \rightarrow \quad (y_2 - y_1)(x - x_1) = (y - y_1)(x_2 - x_1)$$

$$\rightarrow (y_2 - y_1)x + (x_1 - x_2)y + (-x_1 y_2 + x_2 y_1) = 0$$

Ditemukan persamaan garis $Ax + By + C$ dengan $A = (y_2 - y_1), B = (x_1 - x_2), dan\ C = (-x_1y_2 + x_2y_1)$. Jarak antar garis dapat dihitung dengan persamaan $dist = |\frac{Ax+By+C}{\sqrt{A^2+B^2}}|$. Apabila ada dua titik dengan jarak yang sama, maka `pMax` ditentukan berdasarkan titik yang membuat sudut terbesar antara `pMax`, `minAbs` dan `maxAbs` menggunakan aturan cosinus ($a^2 = b^2 + c^2 - 2.b.c.\cos(\alpha)$). `pMax` akan ditambahkan ke dalam list titik, dan kemudian akan dilakukan proses rekursif (`divideLeft`/`divideRight` tergantung sisi) dengan membagi titik-titik yang tersisa ke sisi kiri / kanan dari garis yang dibentuk titik `minAbs` dan `pMax` serta garis `pMax` dan `maxAbs`. Proses ini akan terus berulang sampai tidak ada titik yang dapat dievaluasi lagi.

Setelah proses rekursi selesai, fungsi akan menggabungkan hasil titik-titik dengan mengurutkan titik pada bagian kiri secara terurut membesar berdasarkan absis, dan mengurutkan titik pada bagian kanan secara terurut mengecil berdasarkan absis, Fungsi akan mengembalikan dua buah list berisi daftar koordinat absis dan ordinat yang berkorespondensi. Kedua array dimasukkan sebagai parameter pada plt.plot untuk menggambarkan *convex hull*.

BAB II. *SOURCE PROGRAM* DALAM BAHASA PYTHON

2.1 process.py

(file utama berisi fungsi *divide and conquer*, direktori `src/lib/myConvexHull`)

```
from myConvexHull.utils import *

class Convex(object):
    def divideList(self, PointsList, minAbs, maxAbs, flag):
        '''
        FUNGSI DIVIDE:
        Membagi list of points menjadi dua bagian berdasarkan determinan
        relatif terhadap garis yang dibentuk titik absis minimum dan maksimum.
        Titik yang mempunyai determinan positif terhadap garis dimasukkan ke
        list leftSide, sementara titik dengan determinan negatif dimasukkan ke
        list rightSide.
        Argumen fungsi:
            - PointsList: list of points yang akan dibagi
            - minAbs, maxAbs: titik dengan absis minimum/maksimum
            - flag: integer untuk menentukan list mana yang akan dikembalikan
            (beberapa kasus membutuhkan hanya list kiri / kanan)
        '''
        leftSide = []
        rightSide = []
        for i in range(len(PointsList)):
            # kondisi untuk menambahkan titik adalah apabila:
            # - titik mempunyai nilai absis antara minAbs dan maxAbs
            # - titik mempunyai nilai absis sama dengan minAbs tapi ordinat
lebih tinggi
```

```
            # - titik mempunyai nilai absis sama dengan minAbs tapi ordinat
lebih rendah
            if (((PointsList[i].x > minAbs.x) or (PointsList[i].x == minAbs.x
and PointsList[i].y > minAbs.y))
                and ((PointsList[i].x < maxAbs.x) or (PointsList[i].x ==
maxAbs.x and PointsList[i].y < maxAbs.y))):

                # fungsi akan menambahkan point ke leftSide apabila determinan
positif,
                # sebaliknya akan menambahkan ke rightSide apabila determinan
negatif
                if (isDeterminantPositive(minAbs, maxAbs, PointsList[i])):
                    leftSide.append(PointsList[i])
                if (not isDeterminantPositive(minAbs, maxAbs, PointsList[i])):
                    rightSide.append(PointsList[i])


        # flag menunjukkan list mana yang akan dikembalikan
        if (flag > 0):
            return leftSide
        elif (flag < 0):
            return rightSide
        else:
            return leftSide, rightSide


    def divideLeft(self, PointsList, minAbs, maxAbs):
        '''
        Membagi list of points menjadi dua bagian berdasarkan determinan
        untuk bagian kiri dari garis awal. Fungsi ini akan melakukan
        pembagian secara rekursif sampai list kosong.
        Argumen fungsi:
            - PointsList: list of points yang akan dibagi
            - minAbs, maxAbs: titik dengan absis minimum/maksimum
        '''
        if len(PointsList) == 0:
            return []
        else:
            pMax = findPMax(PointsList, minAbs, maxAbs)
            PointsList.remove(pMax)
            leftTemp = self.divideLeft(self.divideList(PointsList, minAbs,
pMax, 1), minAbs, pMax)
            for point in leftTemp:
              PointsList.remove(point)
            rightTemp = self.divideLeft(self.divideList(PointsList, pMax,
maxAbs, 1), pMax, maxAbs)
            for point in rightTemp:
              PointsList.remove(point)
```

```
            return leftTemp + [pMax] + rightTemp

    def divideRight(self, PointsList, minAbs, maxAbs):
        '''
        Membagi list of points menjadi dua bagian berdasarkan determinan
        untuk bagian kanan dari garis awal. Fungsi ini akan melakukan
        pembagian secara rekursif sampai list kosong.
        Argumen fungsi:
            - PointsList: list of points yang akan dibagi
            - minAbs, maxAbs: titik dengan absis minimum/maksimum
        '''
        if len(PointsList) == 0: #basis apabila list kosong
            return []
        else:
            pMax = findPMax(PointsList, minAbs, maxAbs)
            PointsList.remove(pMax)
            leftTemp = self.divideRight(self.divideList(PointsList, minAbs,
pMax, -1), minAbs, pMax)
            for point in leftTemp:
              PointsList.remove(point)
            rightTemp = self.divideRight(self.divideList(PointsList, pMax,
maxAbs, -1), pMax, maxAbs)
            for point in rightTemp:
              PointsList.remove(point)
            return leftTemp + [pMax] + rightTemp

    def mergeList(self, leftRes, rightRes, minAbs, maxAbs):
        '''
        FUNGSI MERGE
        Menggabungkan titik yang berada di dalam list leftRes dan rightRes
        (hasil divide and conquer) dengan titik absis minimum/maksimum.
        Argumen fungsi:
            - leftRes, rightRes: hasil dari divideLeft dan divideRight
            - minAbs, maxAbs: titik dengan absis minimum/maksimum
        '''
        mergedList = []
        # lakukan pengurutan terhadap leftRes dan rightRes
        leftRes = quickSort(leftRes)
        rightRes = quickSort(rightRes)

        mergedList.append(minAbs)
        for i in range(len(leftRes)):
            mergedList.append(leftRes[i])

        mergedList.append(maxAbs)
        for i in range(len(rightRes) - 1, -1, -1):
            mergedList.append(rightRes[i])
```

```
        mergedList.append(mergedList[0])

        mergedX = [point.x for point in mergedList]
        mergedY = [point.y for point in mergedList]

        return [mergedX, mergedY]


    def ConvexHull(self, listOfPoints):
        '''
        Fungsi utama (saat pemanggilan)
        Menerima dataset dalam bentuk list dua dimensi (x, y)
        '''
        # membuat list yang berisi object Point
        PointsList = removeDupes(quickSort([Point(point[0], point[1]) for
point in listOfPoints]))

        # mencari nilai absis minimum dan maksimum untuk
        # mendapatkan garis yang membagi points menjadi 2 bagian
        minAbs = PointsList[0]
        maxAbs = PointsList[len(PointsList) - 1]

        # pembagian list ke bagian kiri dan kanan garis
        leftSide, rightSide = self.divideList(PointsList[1:-1], minAbs,
maxAbs, 0)

        # pemanggilan fungsi pembagian secara rekursif
        leftRes = self.divideLeft(leftSide, minAbs, maxAbs)
        rightRes = self.divideRight(rightSide, minAbs, maxAbs)

        # pemanggilan fungsi merging
        return self.mergeList(leftRes, rightRes, minAbs, maxAbs)
```

2.2 utils.py

(file berisi fungsi utilitas untuk membantu fungsi utama, direktori `src/lib/myConvexHull`)

```
import math

class Point:
    '''
    Class Point untuk merepresentasikan sebuah titik
    dalam bidang dua-dimensi.
    '''
    def __init__(self, x, y):
        self.x = x
        self.y = y
```

```
    def sameAs(self, p):
        return self.x == p.x and self.y == p.y

def isDeterminantPositive(p1, p2, p3):
    '''
    Menghitung determinan antara dua buah titik (p1 dan p2)
    yang membentuk sebuah garis dan titik p3 untuk menentukan
    letak sebuah titik relatif terhadap garis.
    - Determinan positif: titik berada pada sisi kiri relatif dari garis
    - Determinan negatif: titik berada pada sisi kanan relatif dari garis
    - Determinan nol: titik berada pada garis dan dapat diabaikan
    Argumen fungsi:
        - p1, p2: titik yang membentuk garis (absis minimum dan maksimum)
        - p3: titik yang dibandingkan
    '''
    det = ((p1.x * p2.y) + (p3.x * p1.y) + (p2.x * p3.y)
           - (p3.x * p2.y) - (p1.x * p3.y) - (p2.x * p1.y))
    if (det > 0):
        return True
    if (det < 0):
        return False

def distance(p1, p2, p3):
    '''
    Menghitung jarak antara garis yang dibentuk p1 dan p2
    dengan titik p3.
    Argumen fungsi:
        p1, p2: titik yang membentuk garis (absis minimum dan maksimum)
        p3: titik yang dibandingkan
    '''
    A = p1.y - p2.y
    B = p2.x - p1.x
    C = p1.x * p2.y - p2.x * p1.y
    dist = abs((A * p3.x + B * p3.y + C) / ((A ** 2 + B ** 2) ** 0.5))
    return dist

def findAngle(minAbs, maxAbs, pMax):
    '''
    Fungsi untuk mencari sudut pMax, minAbs, maxAbs
    menggunakan aturan cosinus
    '''
    aSide = (pMax.x - maxAbs.x) ** 2 + (pMax.y - maxAbs.y) ** 2
    bSide = (pMax.x - minAbs.x) ** 2 + (pMax.y - minAbs.y) ** 2
    cSide = (maxAbs.x - minAbs.x) ** 2 + (maxAbs.y - minAbs.y) ** 2
    cos = (aSide - bSide - cSide) / (-2 * (bSide ** 0.5) * (cSide ** 0.5))
    return math.acos(round(cos, 10)) * 180 / math.pi
```

```
def findPMax(PointsList, minAbs, maxAbs):
    '''
    Fungsi untuk menentukan pMax, titik berjarak terjauh dari garis
    yang dibentuk minAbs dan maxAbs.
    Argumen fungsi:
        - PointsList: list of points yang akan dibagi
        - minAbs, maxAbs: titik dengan absis minimum/maksimum
    '''
    currentDistance = 0
    maxDistance = 0
    maxIndex = 0
    for i in range(0, len(PointsList)):

        currentDistance = distance(minAbs, maxAbs, PointsList[i])
        currentAngle = findAngle(minAbs, maxAbs, PointsList[i])
        maxAngle = findAngle(minAbs, maxAbs, PointsList[maxIndex])
        # syarat pembaruan pMax adalah apabila
        # - jarak terjauh lebih besar dari jarak sebelumnya
        # - atau jarak terjauh sama dengan jarak sebelumnya, tetapi
        #   sudut pMax-minAbs-maxAbs lebih besar dibandingkan sudut sebelumnya
        if (currentDistance > maxDistance or (currentDistance == maxDistance
and
            currentAngle > maxAngle)):
            maxDistance = currentDistance
            maxAngle = currentAngle
            maxIndex = i
    return PointsList[maxIndex]

def quickSort(PointsList):
    '''
    Fungsi untuk melakukan quicksort menggunakan list comprehension
    dalam Python
    Fungsi akan membagi PointsList menjadi pivot, bagian yang lebih kecil
    dari pivot, dan bagian yang lebih besar dari pivot, dan
    melakukan pemrosesan list secara rekursif hingga kosong
    '''

    if len(PointsList) == 0:
        return []
    else:
        pivot = PointsList[0]
        smaller = quickSort([point for point in PointsList[1:] if (point.x <
pivot.x) or
                    (point.x == pivot.x and point.y < pivot.y)])
        bigger = quickSort([point for point in PointsList[1:] if (point.x >
pivot.x) or
                   (point.x == pivot.x and point.y > pivot.y)])
```

```
        return smaller + [pivot] + bigger

def removeDupes(PointsList):
    '''
    Fungsi untuk menghapus points duplikat dari PointsList
    Argumen:
        - PointsList: list yang akan disaring
    '''
    PList = []
    for point in PointsList:
        i = 0
        found = False
        while (i < len(PointsList) and not found):
            if point.sameAs(PointsList[i]):
                break
            else:
                i += 1
        if not found:
            PList.append(point)
    return PList
```

2.3 main.py

```
import matplotlib.pyplot as plt
import pandas as pd
import random
from sklearn import datasets
from myConvexHull.process import Convex
from scipy.spatial import ConvexHull

print("\nDriver for the myConvexHull Library")
print("using datasets from the sklearn library")
ConvexObj = Convex()

while True:
    #input opsi dataset dari library sklearn
    print("\n")
    print("Choose your dataset: ")
    print("1. Iris dataset")
    print("2. Wine dataset")
    print("3. Digits dataset")
    print("4. Breast cancer dataset")


    option = int(input("Input your option (0 to exit)\n>>> "))
    while (option not in range(0, 5)):
        print("Invalid input! Try again.\n")
```

```
        option = int(input("Input your option >>> "))

    if (option == 1):
        data = datasets.load_iris()
    elif (option == 2):
        data = datasets.load_wine()
    elif option == 3:
        data = datasets.load_digits()
    elif option == 4:
        data = datasets.load_breast_cancer()
    elif option == 0:
        print("Exiting program...\n")
        break

    # pemanggilan ddataset yang diinginkan
    df = pd.DataFrame(data.data, columns = data.feature_names)
    df['Target'] = pd.DataFrame(data.target)

    # output kolom yang tersedia
    print("\nAvailable columns:")
    for i in range(len(df.columns) - 1):
        print("{}. {}".format(i, df.columns[i]))

    print("Note: the y-column taken would be the column right next to the
inputted column!")

    # input kolom menandakan kolom x, kolom y adalah input kolom + 1
    cols = int(input("Input your column (max: {}): ".format(len(df.columns) -
3)))
    while (cols + 1 >= len(df.columns) - 1):
        print("Invalid input! Try again.\n")
        cols = int(input("Input your column (max: {}):
".format(len(df.columns) - 3)))

    title = data.feature_names[cols] + " vs " + data.feature_names[cols + 1]
    print("Generating convex hull for {}...".format(title))

    plt.figure(figsize = (10, 6))
    plt.title(title + ' (myConvexHull)')
    plt.xlabel(data.feature_names[cols])
    plt.ylabel(data.feature_names[cols + 1])

    colors = ['b', 'r', 'g']

    # Penggunaan library myConvexHull
    for i in range(len(data.target_names)):
        # pengambilan data dari kolom cols dengan kolom yang bersebelahan
```

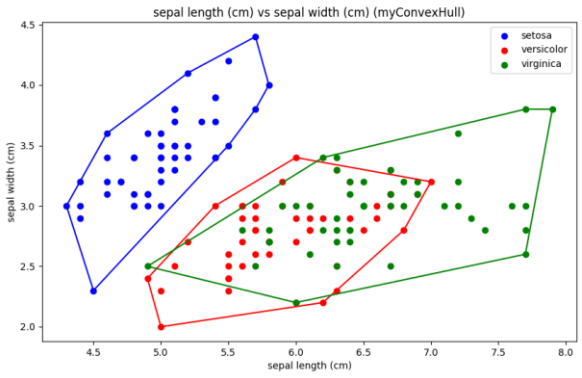
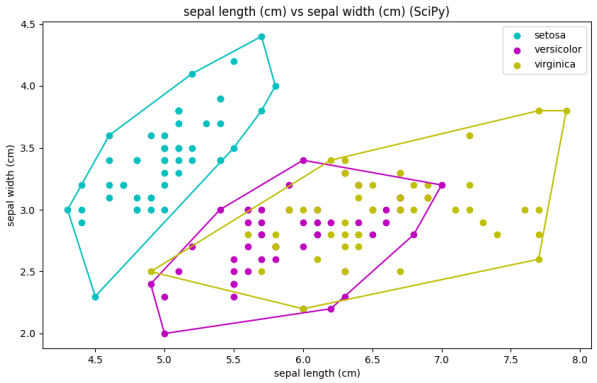
```
        bucket = df[df['Target'] == i].iloc[:, [cols, cols + 1]].values

        # pemanggilan fungsi ConvexHull
        hull = ConvexObj.ConvexHull(bucket)

        # generator warna random untuk plot dengan kategori di atas 3
        if (i >= len(colors)):
            tempColor = random.uniform(0, 1), random.uniform(0, 1),
random.uniform(0, 1)
        else:
            tempColor = colors[i]

        # generasi plot
        plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i],
color = tempColor)

        # memasukkan data dari hasil eksekusi fungsi ConvexHull
        # - hull[0] = array berisi titik x
        # - hull[1] = array berisi titik y
        # untuk setiap i, hull[0][i], hull[1][i] adalah pasangan titik (x, y)
        plt.plot(hull[0], hull[1], color = tempColor)
    plt.legend()
    plt.show()


    # Perbandingan menggunakan library SciPy
    plt.figure(figsize = (10, 6))
    plt.title(title + ' (SciPy)')
    plt.xlabel(data.feature_names[cols])
    plt.ylabel(data.feature_names[cols + 1])

    scolors = ["c", "m", "y"]
    for i in range(len(data.target_names)):

        # pemanggilan data dengan metode yang sama seperti library
myConvexHull
        # inisialisasi fungsi berbeda
        bucket = df[df['Target'] == i].iloc[:, [cols, cols + 1]].values
        hull = ConvexHull(bucket)

        # generator warna random untuk plot dengan kategori di atas 3
        if (i >= len(scolors)):
            tempColor = random.uniform(0, 1), random.uniform(0, 1),
random.uniform(0, 1)
        else:
            tempColor = scolors[i]
        # generasi plot
```

```
        plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i],
color = tempColor)
        # memasukkan data hasil eksekusi fungsi ConvexHull
        for simplex in hull.simplices:
            plt.plot(bucket[simplex, 0], bucket[simplex, 1], color =
tempColor)
    plt.legend()
    plt.show()
```

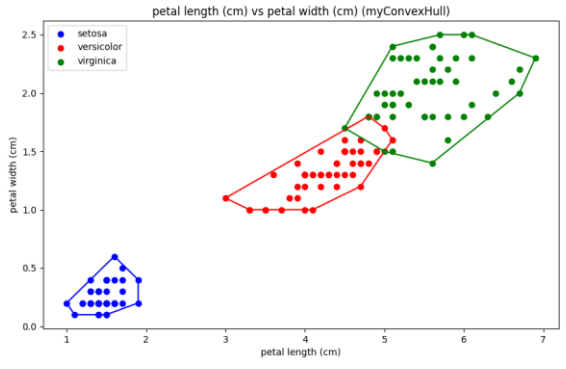
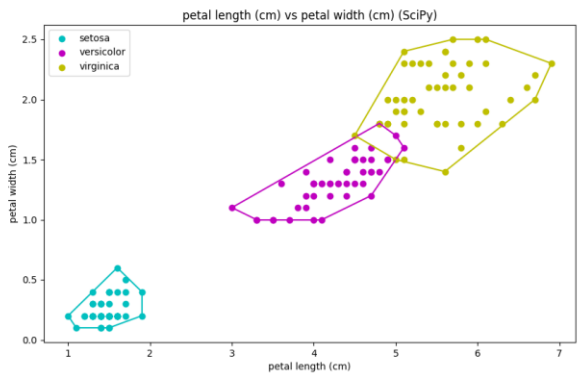
BAB III. PENGUJIAN

Pengujian dilakukan menggunakan dataset pada *library* `sklearn` dan dijalankan dengan file `main.py`. File `main.py` dijalankan di IDE *`Visual Studio Code / PyCharm Community Edition`*. Hasil yang dilampirkan pada pengujian adalah scatter plot yang ditunjukkan oleh fungsi `plt.show()` *(*pastikan pustaka `matplotlib` sudah terpasang sebelum pengujian, informasi lebih lanjut pada file `README.md`*)*.

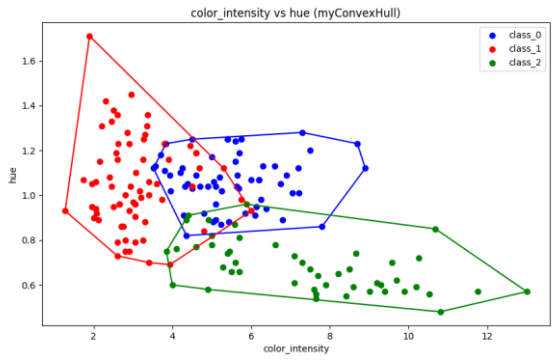
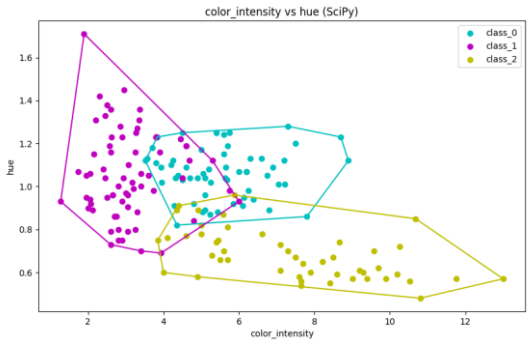
3.1 Sepal Length vs Sepal Width

Opsi dataset pada main = 1 (iris) , opsi kolom = 0	
	
Gambar 3.1.1 Pengujian menggunakan Pustaka myConvexHull	Gambar 3.1.2 Pengujian menggunakan Pustaka SciPy

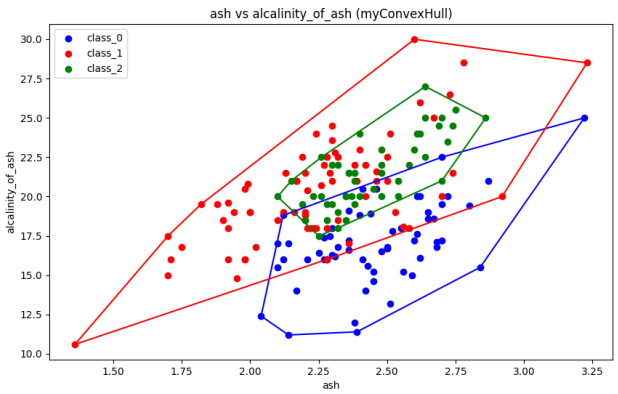
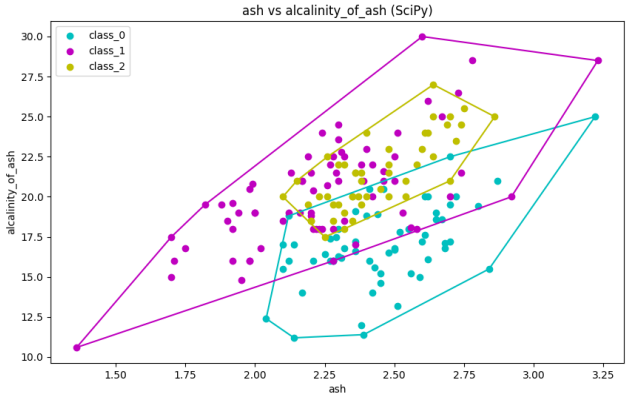
3.2 Petal Length vs Petal Width

Opsi dataset pada main = 1 (iris), opsi kolom = 2	
	
Gambar 3.2.1 Pengujian menggunakan Pustaka myConvexHull	Gambar 3.2.2 Pengujian menggunakan Pustaka SciPy

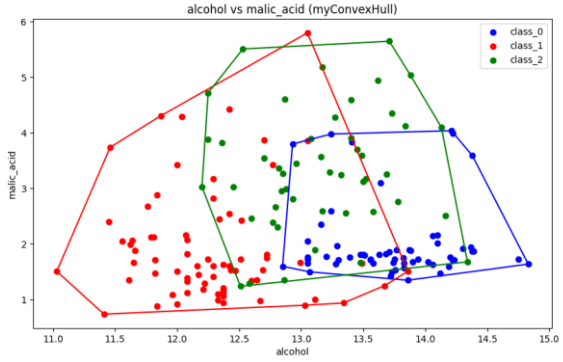
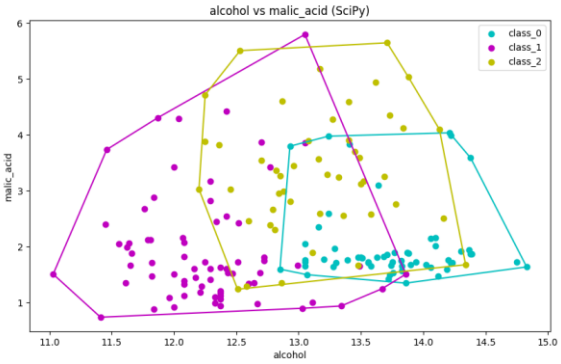
3.3 Color Intensity vs Hue

Opsi dataset pada main = 2 (wine), opsi kolom = 9	
	
Gambar 3.3.1 Pengujian menggunakan Pustaka myConvexHull	Gambar 3.3.2 Pengujian menggunakan Pustaka SciPy

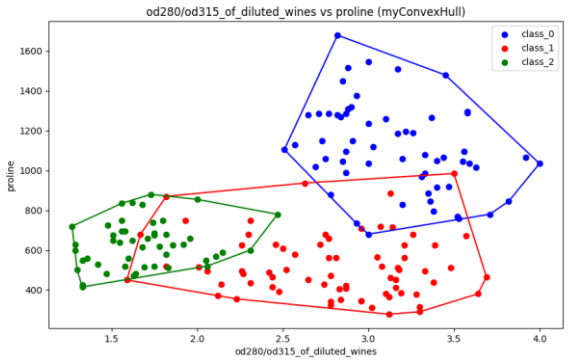
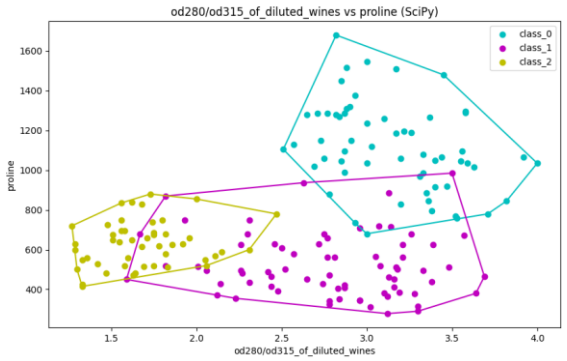
3.4 Ash vs Alcalinity of Ash

Opsi dataset pada main = 2 (wine), opsi kolom = 2	
	
Gambar 3.4.1 Pengujian menggunakan Pustaka myConvexHull	Gambar 3.4.2 Pengujian menggunakan Pustaka SciPy

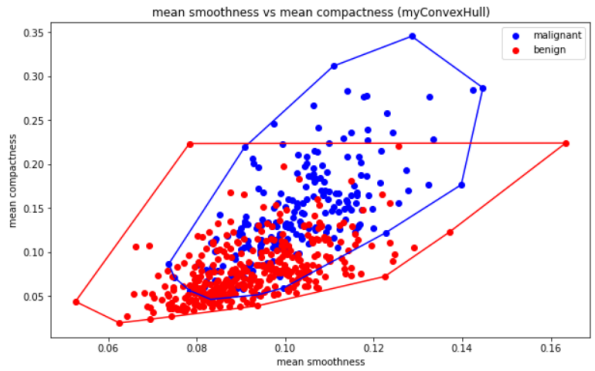
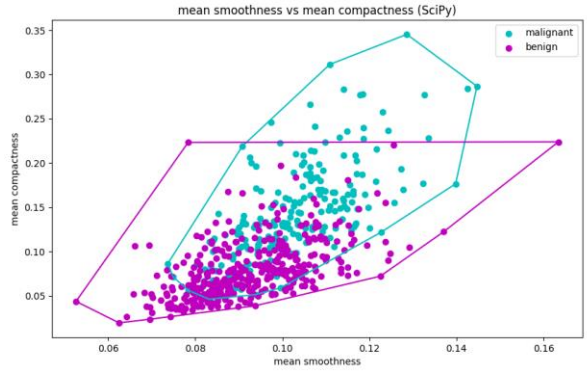
3.5 Alcohol vs Malic Acid

Opsi dataset pada main = 2 (wine), opsi kolom = 0	
	
Gambar 3.5.1 Pengujian menggunakan Pustaka myConvexHull	Gambar 3.5.2 Pengujian menggunakan Pustaka SciPy

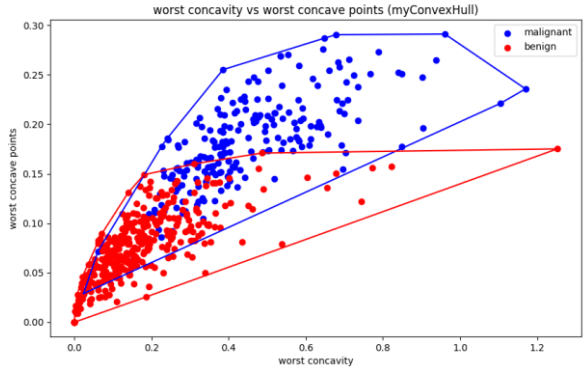
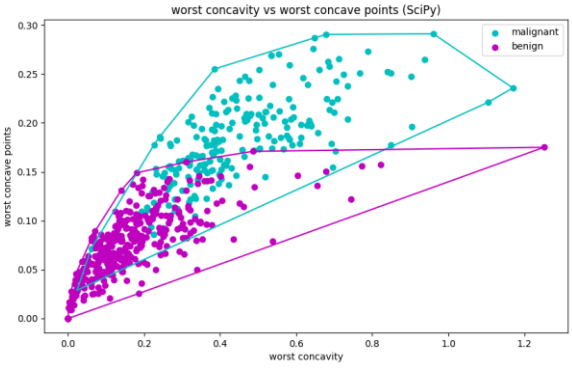
3.6 OD280/OD315 of Diluted Wines vs Proline

Opsi dataset pada main = 2 (breast_cancer), opsi kolom = 11	
	
Gambar 3.6.1 Pengujian menggunakan Pustaka myConvexHull	Gambar 3.6.2 Pengujian menggunakan Pustaka SciPy

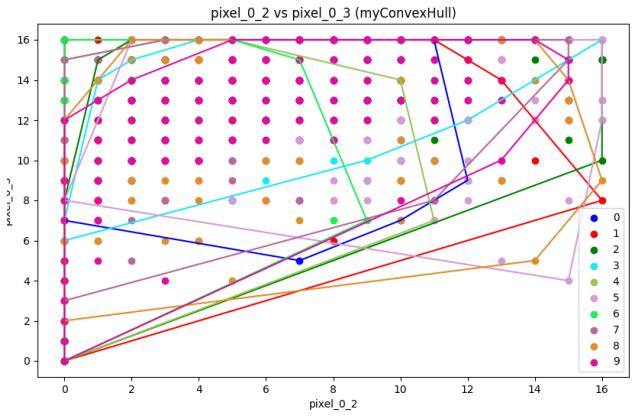
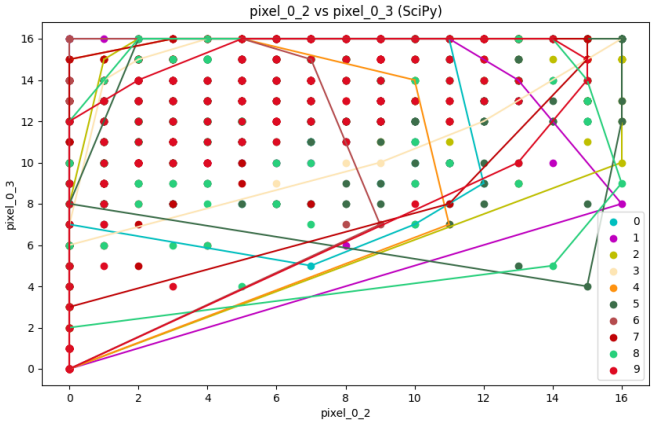
3.7 Mean Smoothness vs Mean Compactness

Opsi dataset pada main = 4 (breast_cancer), opsi kolom = 4	
	
Gambar 3.7.1 Pengujian menggunakan Pustaka myConvexHull	Gambar 3.7.2 Pengujian menggunakan Pustaka SciPy

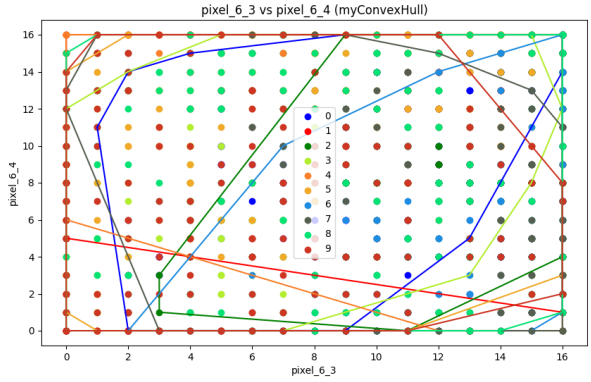
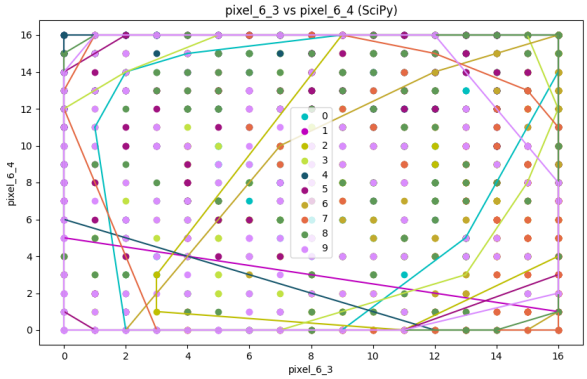
3.8 Worst Concavity vs Worst Concave Points

Opsi dataset pada main = 4 (breast_cancer), opsi kolom = 26	
	
Gambar 3.8.1 Pengujian menggunakan Pustaka myConvexHull	Gambar 3.8.2 Pengujian menggunakan Pustaka SciPy

3.9 pixel_0_2 vs pixel_0_3

Opsi dataset pada main = 3 (digits), opsi kolom = 2	
	
Gambar 3.9.1 Pengujian menggunakan Pustaka myConvexHull	Gambar 3.9.2 Pengujian menggunakan Pustaka SciPy

3.10 pixel_6_3 vs pixel_6_4

Opsi dataset pada main = 3 (digits), opsi kolom = 51	
	
Gambar 3.10.1 Pengujian menggunakan Pustaka myConvexHull	Gambar 3.10.2 Pengujian menggunakan Pustaka SciPy

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. Program dapat menerima input dan menuliskan output untuk dataset lainnya.	✓	

BAB IV. *REPOSITORY*

Repository dapat diakses via `https://github.com/clumsyyyy/TucilStima2`