Laporan Tugas Kecil 2

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

IF2211 Strategi Algoritma

Fawwaz Anugrah Wiradhika Dharmasatya – 13520086 – K02

I. Algoritma

Implementasi penyelesaian Convex Hull dengan algoritma Divide and Conquer ini diimplementasikan dalam bahasa Python. Program dimulai dengan memilih dan memuat dataset, baik dari dataset yang disediakan sklearn maupun dari file .csv di folder test.

Setelah data dimuat dan disimpan sebagai dataframe pandas, program akan membuat objek dari kelas MyConvexHull dengan parameter data tersebut. Kelas tersebut memiliki atribut, data yang dimasukkan, warna point dan garis bila diplotkan, dan list pasangan titik yang membentuk garis yang membentuk Convex Hull. Program lalu membentuk scatter plot dari data tersebut dan membentuk Convex Hull menggunakan list pasangan titik dari kelas tersebut.

Saat kelas MyConvexHull membuat list pasangan titik, kelas ini akan menggunakan metode dalam kelas tersebut untuk membentuk Convex Hull menggunakan algoritma Divide and Conquer.

Cara kerja algoritma pembentukan Convex Hull menggunakan algoritma Divide and Conquer adalah sebagai berikut:

1. Mengurutkan semua data pasangan titik berdasarkan sumbu-x. Jika ada yang sumbu-x nya sama, maka diurutkan menaik berdasarkan sumbu-y.
2. Memeriksa semua titik kecuali titik paling awal dan titik paling akhir menggunakan persamaan:

$$\begin{bmatrix} x1 & y1 & 1 \\ x2 & y2 & 1 \\ x3 & y3 & 1 \end{bmatrix} = x1y2 + x3y1 + x2y3 - x3y2 - x2y1 - x1y3.. (1)$$

   Dengan x1 dan y1 merupakan koordinat x dan y titik awal, x2 dan y2 merupakan koordinat x dan y titik akhir, serta x3 dan y3 merupakan koordinat x dan y titik yang diuji. Jika hasil persamaan bernilai positif, maka titik yang diuji berada di atas garis uji dan dimasukkan ke list titik yang berada di atas garis uji. Jika hasil persamaan bernilai negatif, maka titik yang diuji berada di bawah garis uji dan dimasukkan ke list titik yang berada di bawah garis uji.
3. Masukkan titik awal ke daftar titik yang membentuk Convex Hull
4. Periksa semua titik di upalarik bagian atas.
5. Cari titik terjauh di upalarik. Titik terjauh adalah titik dengan jarak terjauh dari garis yang dibentuk titik awal dan titik akhir. Jika ada yang jaraknya sama, pilih titik yang menyebabkan sudut antara titik awal, titik yang diperiksa, dan titik akhir yang lebih besar.
6. Periksa semua titik uji terhadap garis yang dibentuk titik awal dengan titik terjauh menggunakan persamaan nomor (1). Jika nilainya positif, maka titik tersebut berada di sebelah kiri garis dan dimasukkan ke sebuah upalarik. Jika nilainya negatif atau nol maka diabaikan.
7. Periksa semua titik uji terhadap garis yang dibentuk titik terjauh dengan titik akhir menggunakan persamaan nomor (1). Jika nilainya positif, maka titik tersebut berada di sebelah kanan garis dan dimasukkan ke sebuah upalarik. Jika nilainya negatif atau nol maka diabaikan.
8. Urut kedua upalarik, baik yang sebelah kiri atau kanan , secara menaik terbesar berdasarkan sumbu-x. Jika ada yang nilai-x nya sama, maka urutkan menaik berdasarkan sumbu-y.
9. Lakukan pemanggilan rekursif untuk memproses upalarik, baik yang kiri dan kanan. Langkah diulangi ke langkah nomor (5)

10. Masukkan titik terjauh, titik-titik pembentuk Convex Hull sebelah kiri, serta titik-tiitk pembentuk Convex Hull sebelah kanan ke daftar titik pembentuk Convex Hull sementara.
11. Setelah semua Convex Hull di bagian atas garis utama sudah diperoleh, urutkan menaik secara sumbu-x dan bila ada yang x nya sama maka urutkan berdasarkan sumbu-y secara menaik, lalu masukkan ke daftar titik pembentuk ConvexHull utama.
12. Tambahkan titik akhir garis utama ke daftar titik pembentuk ConvexHull.
13. Periksa semua titik di upalarik bagian bawah.
14. Cari titik terjauh di upalarik. Titik terjauh adalah titik dengan jarak terjauh dari garis yang dibentuk titik awal dan titik akhir. Jika ada yang jaraknya sama, pilih titik yang menyebabkan sudut antara titik awal, titik yang diperiksa, dan titik akhir yang lebih besar.
15. Periksa semua titik uji terhadap garis yang dibentuk titik awal dengan titik terjauh menggunakan persamaan nomor (1). Jika nilainya negatif, maka titik tersebut berada di sebelah kiri garis dan dimasukkan ke sebuah upalarik. Jika positif atau nol maka diabaikan.
16. Periksa semua titik uji terhadap garis yang dibentuk titik terjauh dengan titik akhir menggunakan persamaan nomor (1). Jika nilainya negatif, maka titik tersebut berada di sebelah kanan garis dan dimasukkan ke sebuah upalarik. Jika nilainya positif atau nol maka diabaikan.
17. Urut kedua upalarik, baik yang sebelah kiri atau kanan , secara menaik terbesar berdasarkan sumbu-x. Jika ada yang nilai-x nya sama, maka urutkan menaik berdasarkan sumbu-y.
18. Lakukan pemanggilan rekursif untuk memproses upalarik, baik yang kiri dan kanan. Langkah diulangi ke langkah nomor (14)
19. Masukkan titik terjauh, titik-titik pembentuk Convex Hull sebelah kiri, serta titik-tiitk pembentuk Convex Hull sebelah kanan ke daftar titik pembentuk Convex Hull sementara.
20. Setelah semua Convex Hull di bagian bawah garis utama sudah diperoleh, urutkan menurun secara sumbu-x dan bila ada yang x nya sama maka urutkan berdasarkan sumbu-y secara menurun, lalu masukkan ke daftar titik pembentuk ConvexHull utama.
21. Konversikan daftar titik tersebut menjadi sebuah list berisi pasangan titik yang bersebelahan di dalam ConvexHull.

II. Kode Program

Tabel 2.1. *Source Code* file main.py (Kode Penggunaan Library)

```
#PROGRAM UTAMA
if __name__ == '__main__':
    def getAxis(data):
        #Mendapatkan index sumbu yang akan diproses
        print("Daftar Kolom di Dataset")
        for i in range(len(df.columns)):
            print(i+1,".",df.columns[i])
        while(True):
            sumbu_x = int(input("Pilih nomor kolom yang ingin menjadi
sumbu-x: "))
            if(sumbu_x>=1 and sumbu_x<=len(df.columns)):
                break
            else:
                print("Pilihan tidak valid!")
        while(True):
            sumbu_y = int(input("Pilih nomor kolom yang ingin menjadi
sumbu-y: "))
            if(sumbu_y>=1 and sumbu_y<=len(df.columns)and
sumbu_x!=sumbu_y):
                break
```

```
            else:
                print("Pilihan tidak valid!")
        return (sumbu_x-1,sumbu_y-1)

    import pandas as pd
    import matplotlib.pyplot as plt
    import myConvexHull as my
    from sklearn import datasets
    valid = False
    data = None
    print("Pilih sumber dari data yang ingin Anda masukkan(dalam angka): ")
    print("1.Dataset sklearn")
    print("2.CSV")
    sumber_data = int(input("Masukkan pilihan Anda: "))
    if(sumber_data==1):
        while(not valid):
            print("Pilih dataset yang diinginkan: ")
            print("1.iris")
            print("2.digits")
            print("3.wine")
            print("4.breast cancer")
            pilihan = int(input("Masukkan pilihan Anda(dalam angka): "))
            if(pilihan==1):
                data = datasets.load_iris()
                valid = True
            elif(pilihan==2):
                data = datasets.load_digits()
                valid = True
            elif(pilihan==3):
                data = datasets.load_wine()
                valid = True
            elif(pilihan==4):
                data = datasets.load_breast_cancer()
                valid = True
            else:
                print("Pilihan Tidak Valid!")

    elif(sumber_data==2):
        nama = input("Masukkan nama file(tanpa perlu menuliskan .csv) yang
berada di folder test: ")
        data = pd.read_csv("../test/"+nama+".csv")
        valid = True
    else:
        print("Pilihan tidak valid!\nKeluar program...")
    if(valid):
        df = pd.DataFrame(data.data, columns=data.feature_names)
        df['Target'] = pd.DataFrame(data.target)
        print("Dimensi data: ",df.shape)
        df.head()

        #visualisasi hasil ConvexHull
        plt.figure(figsize = (10, 6))
        sumbu_x,sumbu_y = getAxis(df)
        columns_list = df.columns.values.tolist()
        plt.title(columns_list[sumbu_y]+" vs "+columns_list[sumbu_x])
        plt.xlabel(columns_list[sumbu_x])
        plt.ylabel(columns_list[sumbu_y])
        #Asumsi ada atribut target_names
```

```
        for i in range(len(data.target_names)):
            bucket = df[df['Target'] == i]
            bucket = bucket.iloc[:,[sumbu_x,sumbu_y]].values
            hull = my.MyConvexHull(bucket)
            plt.scatter(bucket[:, 0], bucket[:, 1],
label=data.target_names[i],c=hull.color)
            for simplex in hull.simplices:
                plt.plot(bucket[simplex, 0], bucket[simplex, 1],
hull.color)
        plt.legend()
        plt.show()
```

Tabel 2.2. *Source Code* file myConvexHull.py(Bagian utama library)

```
import util as ut
import numpy as np
class MyConvexHull:
    color_idx = 0
    colors = ['b','r','g','c','m','y','k','#7b3f00']##7b3f00->chocolate
    def __init__(self,coordinates):
        self.coordinates = coordinates
        self.simplices = self.getConvexHull(coordinates)
        self.color =
MyConvexHull.colors[(MyConvexHull.color_idx)%len(MyConvexHull.colors)]
        MyConvexHull.color_idx += 1

    def sort(self,coordinates,axis=0):#0:x,1:y
        #pake quicksort
        if len(coordinates)>1:
                k,coordinates = self.partisi(coordinates)
                left = self.sort(coordinates[0:k+1])
                right = self.sort(coordinates[k+1:len(coordinates)])
                coordinates = np.concatenate((left,right))
        return coordinates
    def sorty(self,coordinates):
        #sort berdasarkan y agar terurut membesar
        if(len(coordinates)>1):
            final_array= []
            subarr =[coordinates[0]]
            for i in range(1,len(coordinates)):
                if(coordinates[i][0]==subarr[0][0]):
                    #x nya sama
                    subarr.append(coordinates[i])
                else:
                    final_array = final_array +  self.sub_sorty(subarr)
                    subarr = [coordinates[i]]
            if(subarr):
                    final_array = final_array + self.sub_sorty(subarr)
            return final_array
        elif len(coordinates)==1:
            return coordinates
        else:
            return []
    def sub_sorty(self,coordinates):
        #sub fungsi untuk solve sort sumbu y
        if len(coordinates)>1:
                k,coordinates = self.partisi(np.array(coordinates),1)
                coordinates = list(coordinates)
```

```
                left = self.sub_sorty(coordinates[0:k+1])
                right = self.sub_sorty(coordinates[k+1:len(coordinates)])
                coordinates = left +right
        return coordinates

    def partisi(self,coordinates,axis=0):#0->x,1->Y
        #buat partisi larik
        #sort dari x nya dulu
        pivot_idx = len(coordinates)//2
        pivot = coordinates[pivot_idx]
        p = 0
        q = len(coordinates)-1
        while p<=q:
            while p<len(coordinates):
                if(coordinates[p][axis] >= pivot[axis]):
                    break
                p += 1
            while q>=0:
                if coordinates[q][axis] <= pivot[axis]:
                    break
                q -=1
            if p<=q:
                coordinates[[p,q]] = coordinates[[q,p]]
                p+=1
                q-=1
        return (q,coordinates)

    def getConvexHull(self,coordinates):
        #dapetin list convexhull
        temp = np.array([[coordinates[i][0],coordinates[i][1],i] for i in
range(len(coordinates))])
        temp1 = self.sort(temp)
        sorted_coordinates = list(self.sorty(temp1))
        start = sorted_coordinates[0]
        end = sorted_coordinates[-1]
        left = []
        right = []
        for i in range(1,len(sorted_coordinates)-1):
            check_position =
ut.checkPosition(start,end,sorted_coordinates[i])
            if(check_position>0):
                #kalo positif berarti di kiri
                left.append(sorted_coordinates[i])
            elif(check_position<0):
                #kalo negatif berarti di kanan
                right.append(sorted_coordinates[i])
            #kalo misal 0 berarti bisa diabaikan(karena segaris)
        simplices  = [start]
        simplices = simplices +
list(self.sorty(self.sort(np.array(self.searchConvexHullUpper(left,start,en
d)))))
        simplices.append(end)
        convex_kanan = self.searchConvexHullLower(right,start,end)
        for i in range(len(convex_kanan)-1,-1,-1):
            simplices.append(convex_kanan[i])
        #simplices = simplices + convex_kanan
        simplices_map = []
        simplices = [list(simplice) for simplice in simplices]
```

```
        for i in range(len(simplices)):
            n = i
            n1 = (i+1) % len(simplices)
            map1  = [int(simplices[n][2]),int(simplices[n1][2])]
            simplices_map.append(map1)
        simplices_map = np.array(simplices_map)
        return simplices_map

    def searchConvexHullUpper(self,coordinates,start,end):
        if(len(coordinates)>1):
            #buat nyari di bagian atas
            start_point = start
            end_point = end
            convex_simplices = []
            left = []
            right = []
            #cari titik terjauh
            if(len(coordinates)>1 and not(start[0]==end[0] and
start[1]!=end[1])):
                farthet_point,idx =
ut.getFarthestPoint(coordinates,start,end)
                convex_simplices.append(farthet_point)
                coordinates.pop(idx)
                for j in range(len(coordinates)):
                    if(ut.checkPosition(start_point,farthet_point,coordinat
es[j])>0):
                        left.append(coordinates[j])
                    elif(ut.checkPosition(farthet_point,end_point,coordinat
es[j])>0):
                        right.append(coordinates[j])
                left = list(self.sorty(self.sort(np.array(left))))
                right = list(self.sorty(self.sort(np.array(right))))
                left_simplices = []
                right_simplices = []
                if(len(left)>0):
                    left_simplices =
self.searchConvexHullUpper(left,start,farthet_point)
                if(len(right)>0):
                    right_simplices =
self.searchConvexHullUpper(right,farthet_point,end)
                convex_simplices = convex_simplices + left_simplices +
right_simplices
            else:
                convex_simplices = convex_simplices + coordinates
            return convex_simplices
        elif(len(coordinates)==1):
            return coordinates
        else:
            return []

    def searchConvexHullLower(self,coordinates,start,end):
        if(len(coordinates)>1):
            #buat nyari di bagian bawah
            start_point = start
            end_point = end
            convex_simplices = []
            left = []
            right = []
```

```
            #cari titik terjauh
            if(len(coordinates)>1 and not(start[0]==end[0] and
start[1]==end[1])):
                farthet_point,idx =
ut.getFarthestPoint(coordinates,start,end)
                convex_simplices.append(farthet_point)
                coordinates.pop(idx)
                for j in range(len(coordinates)):
                    if(ut.checkPosition(start_point,farthet_point,coordinat
es[j])<0):
                        left.append(coordinates[j])
                    elif(ut.checkPosition(farthet_point,end_point,coordinat
es[j])<0):
                        right.append(coordinates[j])
                left = list(self.sorty(self.sort(np.array(left))))
                right = list(self.sorty(self.sort(np.array(right))))
                left_simplices = []
                right_simplices = []
                if(len(left)>0):
                    left_simplices =
self.searchConvexHullLower(left,start,farthet_point)
                if(len(right)>0):
                    right_simplices =
self.searchConvexHullLower(right,farthet_point,end)
                convex_simplices = convex_simplices + left_simplices +
right_simplices
            else:
                convex_simplices = convex_simplices + coordinates
            return list(self.sort(np.array(convex_simplices)))
        elif(len(coordinates)==1):
            return coordinates
        else:
            return []
```

Tabel 2.3. *Source Code* util.py(Bagian utilitas library)

```
from cmath import sqrt
import numpy as np
def checkPosition(p1,p2,p3):
    #p1->titik paling kiri
    #p2->titik paling kanan
    #p3->titik yang ingin diuji
    #[0]->x
    #[1]->y
    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 getDistance(p1,p2):
    #mengembalikan jarak antara 2 titik
    return sqrt((p1[0]-p2[0])**2+(p1[1]-p2[1])**2).real

def getDegree(p1,p2,pmain):
    #mendapatkan sudut p1pmainp2 menggunakan aturan cosinus
    c = getDistance(p1,p2)
    b = getDistance(p1,pmain)
    a = getDistance(p2,pmain)
    cosx = (a**2-b**2-c**2)/(-2*b*c)
    return np.arccos(cosx)
```

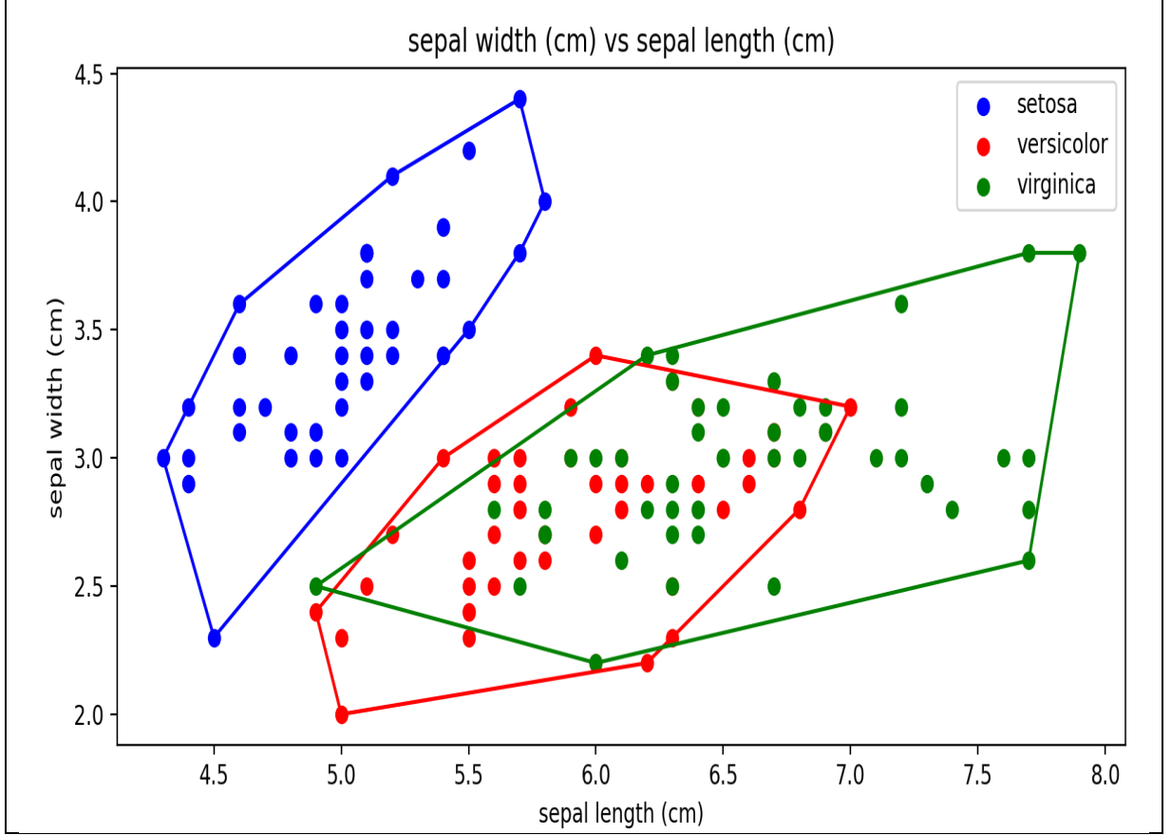
```
def distanceFromLine(start_point,end_point,check_point):
    #menghitung jarak suatu titik dari garis yg dibentuk oleh start_point
dan end_point
    A = end_point[1]-start_point[1]
    B = -(end_point[0]-start_point[0])
    C = start_point[1]*(end_point[0]-start_point[0])-
start_point[0]*(end_point[1]-start_point[1])
    return abs(A*check_point[0]+B*check_point[1]+C)/(sqrt(A**2+B**2))

def getFarthestPoint(coordinates,start,end):
        #mendapatkan titik terjauh
        idx = 0
        farthest_point = coordinates[0]
        farthest_distance = distanceFromLine(start,end,coordinates[0])
        biggest_degree = getDegree(start,end,coordinates[0])
        for i in range(len(coordinates)):
            distance = distanceFromLine(start,end,coordinates[i])
            degree=getDegree(start,end,coordinates[i])
            if(distance>farthest_distance):
                farthest_point = coordinates[i]
                farthest_distance = distance
                biggest_degree = degree
                idx = i
            elif(distance==farthest_distance and degree>biggest_degree):
                farthest_point = coordinates[i]
                farthest_distance = distance
                biggest_degree = degree
                idx = i
        return (farthest_point,idx)
```

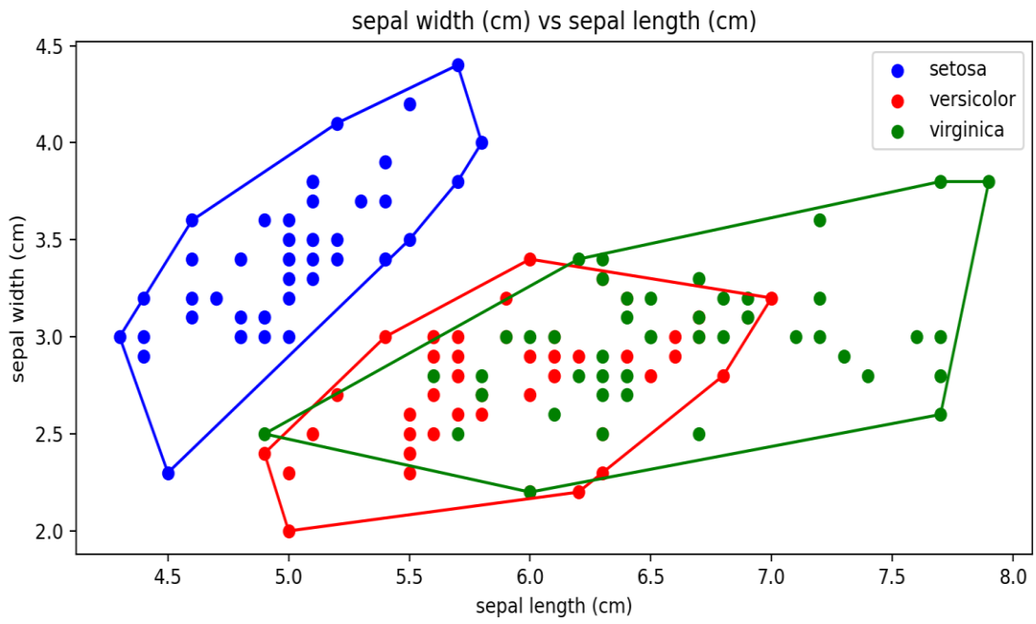
III. *Screenshot Input-Output*

1) Tabel 3.1 Dataset Iris (sepal width (cm) vs sepal length (cm))

Menggunakan *library* myConvexHull

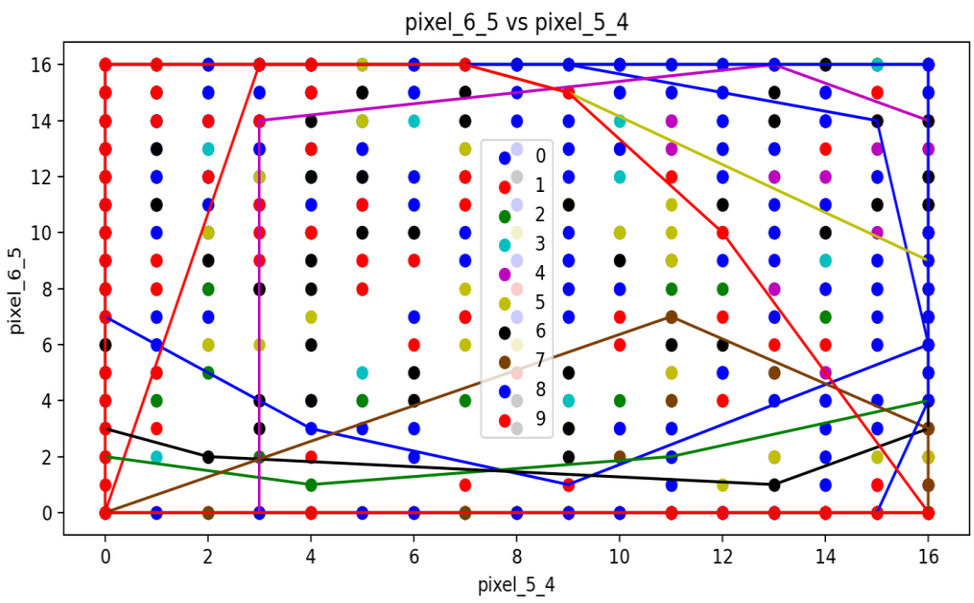


Menggunakan *library* bawaan scipy.spatial

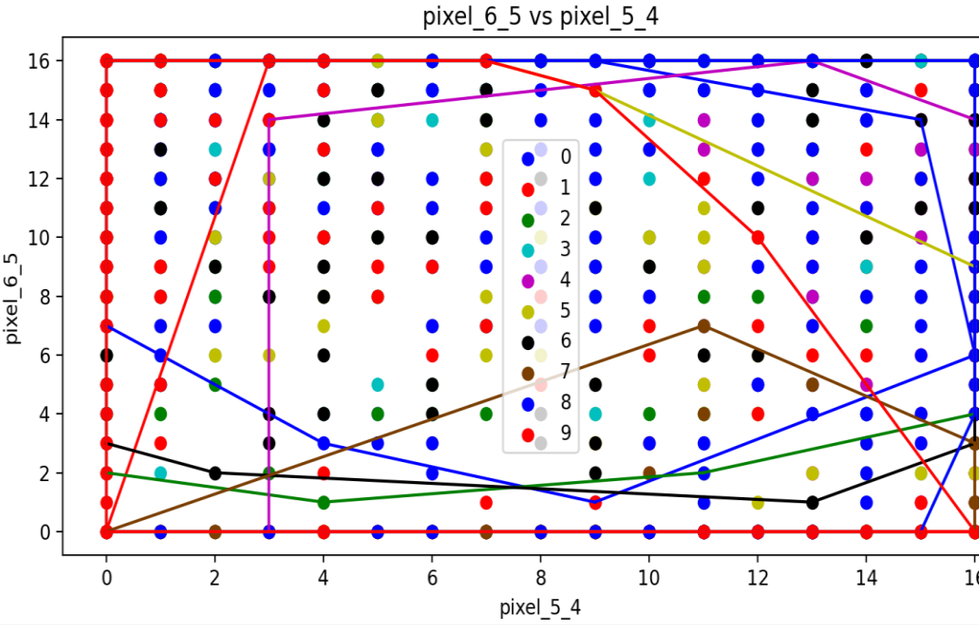


2) Tabel 3.2 Dataset digits (pixel_6_5 vs pixel_5_4)

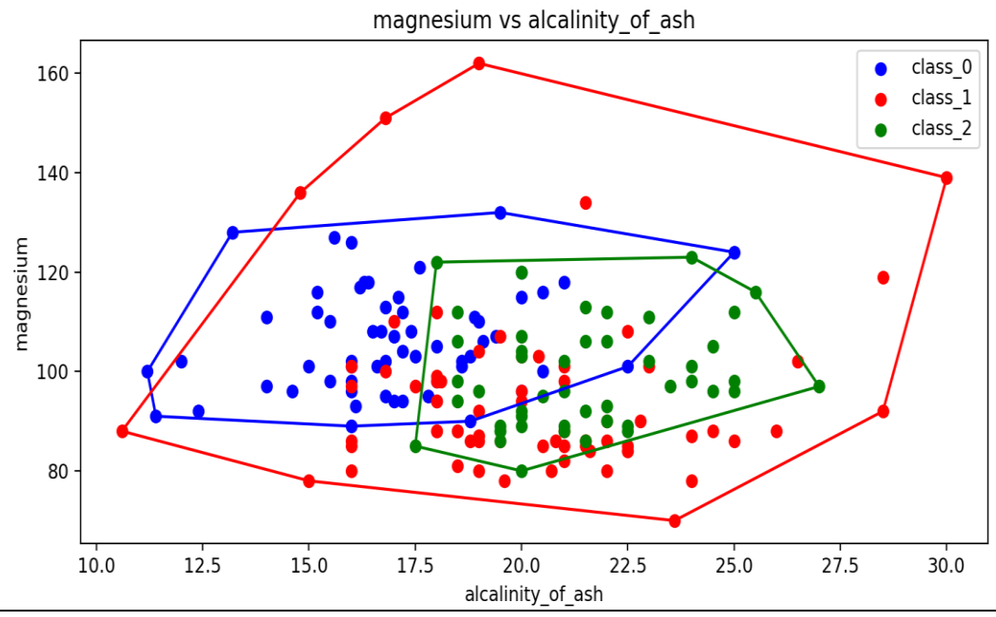
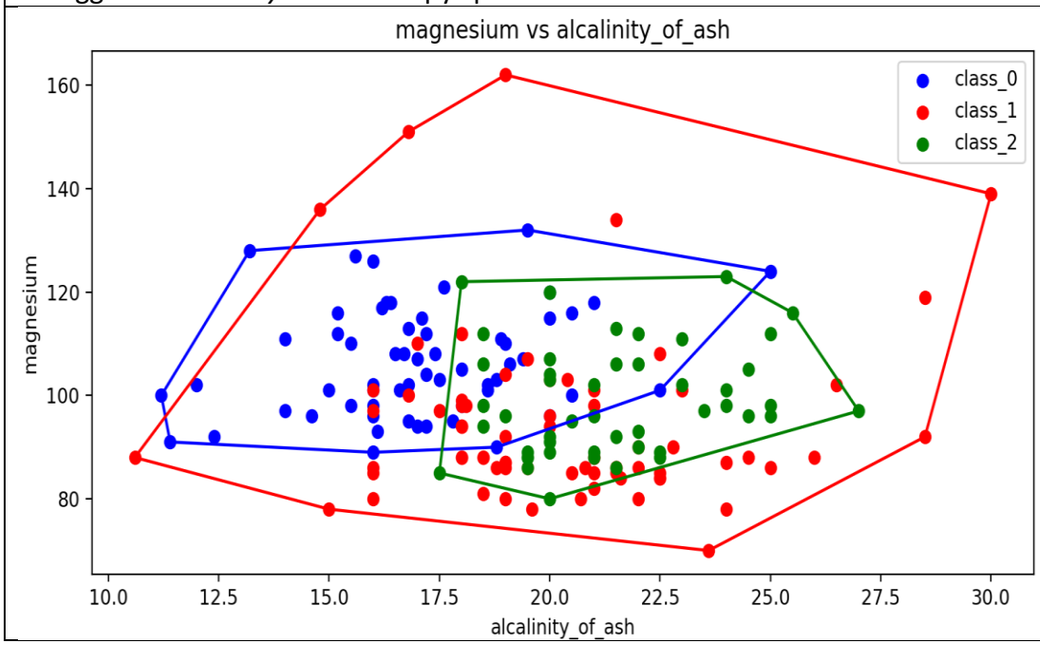
Menggunakan *library* myConvexHull



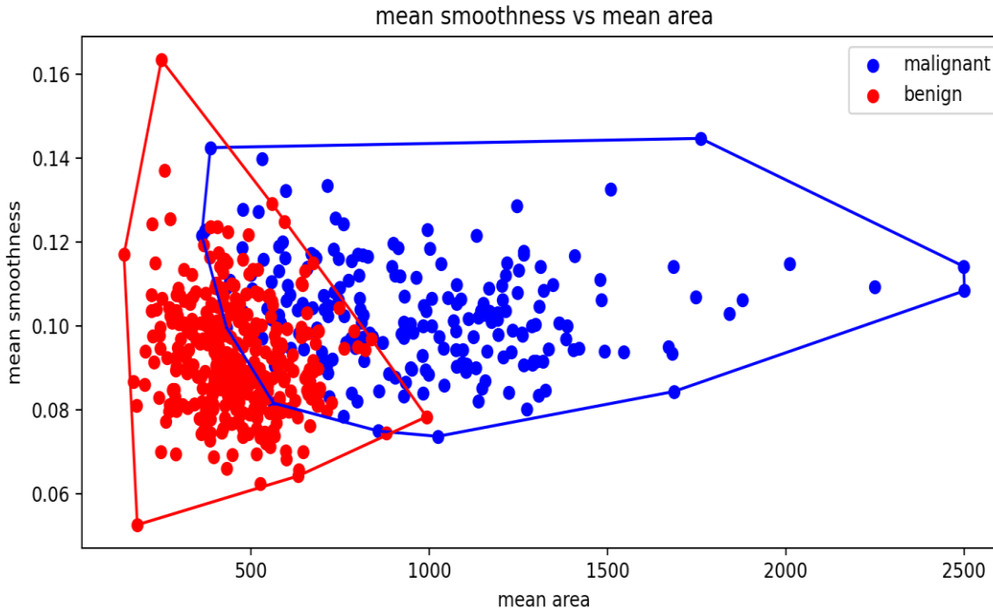
Menggunakan *library* bawaan scipy.spatial

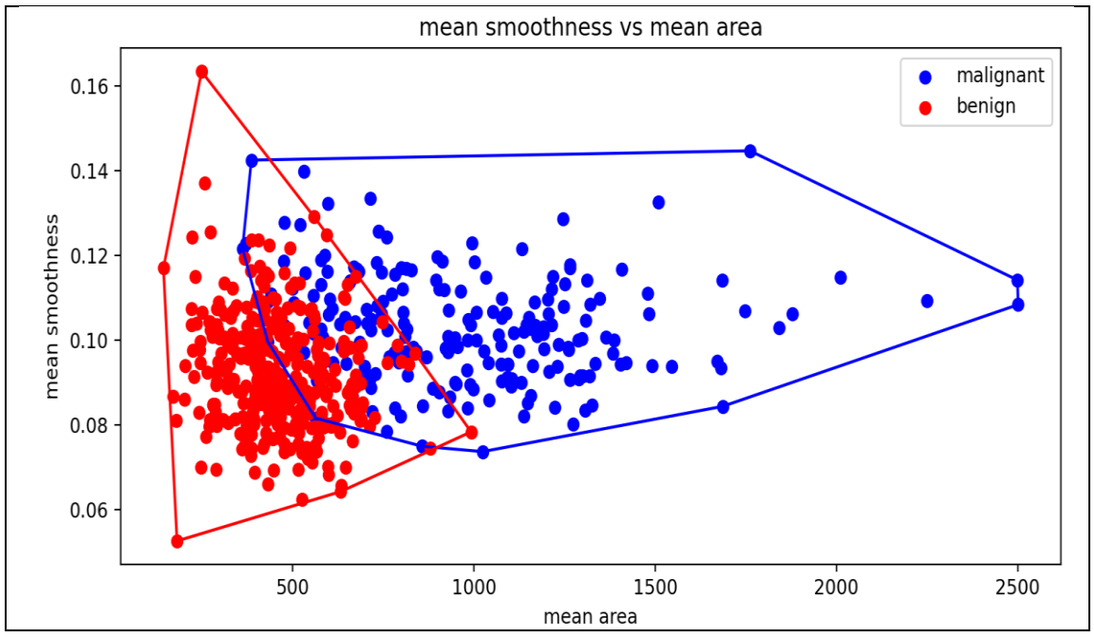


3) Tabel 3.3 Dataset wine (magnesium vs alcanity_of_ash)

Menggunakan *library* myConvexHull

Menggunakan *library* bawaan scipy.spatial


4) Tabel 3.4 Dataset breast cancer (mean smoothness vs mean area)

Menggunakan *library* myConvexHull

Menggunakan *library* bawaan scipy.spatial



IV. Alamat Kode Program

Alamat repositori kode program:

https://github.com/Wiradhika6051/Tucil-2-Stima-Convex-Hull