**LAPORAN TUGAS KECIL-2 STRATEGI ALGORITMA**

**Program library *ConvexHull* menggunakan algortima *divide and conquer***



**Disusun Oleh:**

Muhammad Gerald Akbar Giffera 13520143

**Institut Teknologi Bandung**

**2022**

1. **Algoritma *Divide and Conquer***

Pada pembuatan library ConvexHull ini digunakan pendekatan *divide and conquer,* dengan garis besar tahap-tahapnya sebagai berikut:

1. Baca masukkan dataset yang ingin dianalisis
2. Pilih 2 kolom yang akan dicari ConvexHull nya
3. Urutkan titik koordinat yang diperoleh (x,y) menaik berdasarkan absis
4. Ambil titik paling kanan dan kiri dari himpunan koordinat(A,B) , tarik garis yang memisahkan koordinat menjadi 2 bagian (atas/bawah) S1 dan S2, masukkan garis AB kedalam himpunan solusi
5. Cari titik yang memiliki jarak paling jauh dari garis AB (C)
6. Buat garis baru dari titik yang baru dan dua titik pembentuk garis sebelumnya (A,C) dan (C,B), masukkan kedua garis ke himpunan solusi, dan hilangkan garis AB dari himpunan solusi
7. Ulangi Langkah d sampai f untuk setiap garis baru yang terbuat, sampai tidak ada titik yang tersisa di atas garis pada daerah S1 dan dibawah garis pada daerah S2
8. **Kode Program**

# Fungsi - fungsi pembentuk convexHull  
  
import pandas as pd  
import math  
import matplotlib.pyplot as plt  
import numpy as np  
  
from sklearn import datasets  
  
  
data = datasets.load\_iris()  
df = pd.DataFrame(data.data, columns=data.feature\_names)  
df['target'] = pd.DataFrame(data.target)  
labels = data.target\_names  
column\_name = data.feature\_names  
solution = []  
  
  
def quickSortPartition(arrayAbsis, low, high):  
 i = low - 1;  
 pivot = arrayAbsis[high][0]  
  
 # loop untuk melakukan partisi pada array  
 for j in range(low, high):  
 if arrayAbsis[j][0] < pivot:  
 # increment nilai i untuk melakukan penukaran  
 i = i + 1  
 arrayAbsis[i], arrayAbsis[j] = arrayAbsis[j], arrayAbsis[i]  
  
 # memastikan bahwa niali pivot berada di tengah  
 arrayAbsis[i + 1], arrayAbsis[high] = arrayAbsis[high], arrayAbsis[i + 1]  
  
 # mengembalikan nilai indeks pivot / indeks dimana array sudah terurut  
 return i + 1  
  
  
def quickSort(arrayAbsis, low, high):  
 if len(arrayAbsis) == 1:  
 return arrayAbsis  
 if low < high:  
 pi = quickSortPartition(arrayAbsis, low, high)  
  
 quickSort(arrayAbsis, low, pi - 1)  
 quickSort(arrayAbsis, pi + 1, high)  
  
 # return sorted array  
  
  
def findPointDistance(A,B,p):  
 #mencari jarak antara koordinat p dengan garis AB  
  
 x1, y1 = A[0], A[1]  
 x2, y2 = B[0], B[1]  
  
 # membuat persamaan garis dengan bentuk:  
 # ax + by + c = 0  
  
 x,y = p[0],p[1]  
  
 a = y2 - y1  
 b = x1 - x2  
 c = x2 \* y1 - x1 \* y2  
  
 return a\*x + b\*y + c  
  
def findRightMostPoint(pointArray):  
 # asumsi array selalu terurut  
 return pointArray[-1]  
  
  
def findLeftMostPoint(pointArray):  
 # asumsi array selalu terurut  
 return pointArray[0]  
  
  
def getYBoundary(p1, p2):  
 # untuk memisahkan titik menjadi dua bagian antara (atas/bawah)  
 y1 = p1[1]  
 y2 = p2[2]  
  
 return min(y1, y2)  
  
  
def getCoordinateSide(A, B, p):  
 # AB --> Garis, p--> koordinat  
 f = findPointDistance(A,B,p)  
  
 # Memeriksa jika point p berada di bawah/atas garis  
 if f < 0:  
 return 'above'  
 elif f > 0:  
 return 'below'  
 else:  
 return 'on line'  
  
  
def findHull(pointArray, A, B):  
 if len(pointArray) == 0:  
 return  
 else:  
  
 farthestDistance = -1  
  
 # proses mencari titik dengan jarak paling jauh  
 C = [0 for i in range(2)]  
 for point in pointArray:  
 f = abs(findPointDistance(A,B, point))  
 if f > farthestDistance:  
 farthestDistance = f  
 C[0], C[1] = point[0],point[1]  
  
  
 # update array solution agar terisi garis yang paling baru  
 solution.remove([A, B])  
 solution.append([A, C])  
 solution.append([C, B])  
  
 # menghilangkan koordinat dari pointArray agar tidak digunakan lagi  
 pointArray.remove(C)  
  
 # memisahkan sisa koordinat menjadi kiri dan kanan  
 rightCoordinate = []  
 leftCoordinate = []  
  
 for point in pointArray:  
 if (getCoordinateSide(A, C, point) == 'above'):  
 rightCoordinate.append(point)  
  
 for point in pointArray:  
 if (getCoordinateSide(C, B, point) == 'above'):  
 leftCoordinate.append(point)  
  
 findHull(rightCoordinate, A, C)  
 findHull(leftCoordinate, C, B)  
  
  
# Main convexHull function  
def convexHull(pointArray):  
 # Sort array berdasarkan absis  
 # Jika absis sama, berdasarkan ordinat  
 pointArray = sorted(pointArray, key=lambda k: [k[0], k[1]])  
  
 A = findLeftMostPoint(pointArray)  
 B = findRightMostPoint(pointArray)  
  
 # membagi koordinat menjadi 2 array atas/bawah  
 topHalf = []  
 lowHalf = []  
  
  
 for point in pointArray:  
 if (getCoordinateSide(A, B, point) == 'above'):  
 topHalf.append(point)  
 elif (getCoordinateSide(A, B, point) == 'below'):  
 lowHalf.append(point)  
 else:  
 pass  
  
 solution.append([A, B])  
 solution.append([B, A])  
  
 # divide and conquer  
 findHull(topHalf, A, B)  
 findHull(lowHalf, B, A)  
  
  
# Sort the solution array  
# Hal ini diperlukan agar koordinat bisa di plot dengan sesuai  
def sortSolution(solutionArr):  
 n = len(solutionArr)  
  
 for i in range(0,n):  
 line = solutionArr[i]  
 next\_line\_idx = min(i+1, n-1)  
 next\_line = solution[next\_line\_idx]  
  
 # Jika urutan pada data tidak sesuai lakukan penukaran posisi koordinat  
 if (line[1] != next\_line[0]):  
 Pass = next\_line\_idx  
 next\_line = solution[Pass]  
 while Pass < n-1 and (line[1] != next\_line[0]):  
 Pass += 1  
 next\_line = solution[Pass]  
  
 solutionArr[next\_line\_idx], solutionArr[Pass] = solutionArr[Pass], solutionArr[next\_line\_idx]  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print('Pilih dataset yang ingin digunakan')  
 print('1. Iris dataset')  
 print('2. Diabetes dataset')  
 print('3. Wine dataset')  
 dataset = int(input('Masukkan angka 1-3: '))  
  
 x\_value = 0  
 y\_value = 0  
  
 Title = ""  
 if dataset == 1:  
 data = datasets.load\_iris()  
 Title += "Iris dataset: "  
 elif dataset == 2:  
 data = datasets.load\_diabetes()  
 Title += "Diabetes dataset: "  
 elif dataset == 3:  
 data = datasets.load\_wine()  
 Title += "Wine dataset: "  
  
  
  
 df = pd.DataFrame(data.data, columns=data.feature\_names)  
 df['target'] = pd.DataFrame(data.target)  
 labels = data.target\_names  
 column\_name = data.feature\_names  
 solution = []  
  
 for j in range(len(column\_name)):  
 print(f'{j+1}. {column\_name[j]}')  
  
 x\_value = int(input('Masukkan nomor untuk kolom atribut-x: '))  
 y\_value = int(input('Maskkan nomor untuk kolom atribut-y: '))  
  
 Title += column\_name[x\_value-1] + " Vs " + column\_name[y\_value-1]  
  
 for i in range(len(labels)):  
 # mengambil data untuk setiap fitur (target)  
 data\_new = df[df['target'] == i]  
  
 #array solution yang akan digunakan  
 solution = []  
 plt.scatter(data\_new[column\_name[x\_value-1]], data\_new[column\_name[y\_value-1]],label=labels[i])  
 plt.title(Title)  
 plt.xlabel(column\_name[x\_value-1])  
 plt.ylabel(column\_name[y\_value-1])  
 plt.legend()  
 data\_new = data\_new[[column\_name[x\_value-1], column\_name[y\_value-1]]].values.tolist()  
 convexHull(data\_new)  
 # sort solution  
 sortSolution(solution)  
  
 # menyesuaikan format dari himpunan solusi agar bisa di tampilkan  
 hull = []  
 for line in solution:  
 hull.append(line[0])  
 hull.append(line[1])  
  
 # menampilkan hasil convexHull  
 df\_solution = pd.DataFrame(hull, columns=['x', 'y'])  
 plt.plot(df\_solution['x'], df\_solution['y'])  
  
  
 plt.show()

def quickSort(arrayAbsis, low, high):  
 if len(arrayAbsis) == 1:  
 return arrayAbsis  
 if low < high:  
 pi = quickSortPartition(arrayAbsis, low, high)  
  
 quickSort(arrayAbsis, low, pi - 1)  
 quickSort(arrayAbsis, pi + 1, high)  
  
 # return sorted array  
  
  
def findPointDistance(A,B,p):  
 #mencari jarak antara koordinat p dengan garis AB  
  
 x1, y1 = A[0], A[1]  
 x2, y2 = B[0], B[1]  
  
 # membuat persamaan garis dengan bentuk:  
 # ax + by + c = 0  
  
 x,y = p[0],p[1]  
  
 a = y2 - y1  
 b = x1 - x2  
 c = x2 \* y1 - x1 \* y2  
  
 return a\*x + b\*y + c  
  
def findRightMostPoint(pointArray):  
 # asumsi array selalu terurut  
 return pointArray[-1]  
  
  
def findLeftMostPoint(pointArray):  
 # asumsi array selalu terurut  
 return pointArray[0]  
  
  
def getYBoundary(p1, p2):  
 # untuk memisahkan titik menjadi dua bagian antara (atas/bawah)  
 y1 = p1[1]  
 y2 = p2[2]  
  
 return min(y1, y2)  
  
  
def getCoordinateSide(A, B, p):  
 # AB --> Garis, p--> koordinat  
 f = findPointDistance(A,B,p)  
  
 # Memeriksa jika point p berada di bawah/atas garis  
 if f < 0:  
 return 'above'  
 elif f > 0:  
 return 'below'  
 else:  
 return 'on line'  
  
  
def findHull(pointArray, A, B):  
 if len(pointArray) == 0:  
 return  
 else:  
  
 farthestDistance = -1  
  
 # proses mencari titik dengan jarak paling jauh  
 C = [0 for i in range(2)]  
 for point in pointArray:  
 f = abs(findPointDistance(A,B, point))  
 if f > farthestDistance:  
 farthestDistance = f  
 C[0], C[1] = point[0],point[1]  
  
  
 # update array solution agar terisi garis yang paling baru  
 solution.remove([A, B])  
 solution.append([A, C])  
 solution.append([C, B])  
  
 # menghilangkan koordinat dari pointArray agar tidak digunakan lagi  
 pointArray.remove(C)  
  
 # memisahkan sisa koordinat menjadi kiri dan kanan  
 rightCoordinate = []  
 leftCoordinate = []  
  
 for point in pointArray:  
 if (getCoordinateSide(A, C, point) == 'above'):  
 rightCoordinate.append(point)  
  
 for point in pointArray:  
 if (getCoordinateSide(C, B, point) == 'above'):  
 leftCoordinate.append(point)  
  
 findHull(rightCoordinate, A, C)  
 findHull(leftCoordinate, C, B)  
  
  
# Main convexHull function  
def convexHull(pointArray):  
 # Sort array berdasarkan absis  
 # Jika absis sama, berdasarkan ordinat  
 pointArray = sorted(pointArray, key=lambda k: [k[0], k[1]])  
  
 A = findLeftMostPoint(pointArray)  
 B = findRightMostPoint(pointArray)  
  
 # membagi koordinat menjadi 2 array atas/bawah  
 topHalf = []  
 lowHalf = []  
  
  
 for point in pointArray:  
 if (getCoordinateSide(A, B, point) == 'above'):  
 topHalf.append(point)  
 elif (getCoordinateSide(A, B, point) == 'below'):  
 lowHalf.append(point)  
 else:  
 pass  
  
 solution.append([A, B])  
 solution.append([B, A])  
  
 # divide and conquer  
 findHull(topHalf, A, B)  
 findHull(lowHalf, B, A)  
  
  
# Sort the solution array  
# Hal ini diperlukan agar koordinat bisa di plot dengan sesuai  
def sortSolution(solutionArr):  
 n = len(solutionArr)  
  
 for i in range(0,n):  
 line = solutionArr[i]  
 next\_line\_idx = min(i+1, n-1)  
 next\_line = solution[next\_line\_idx]  
  
 # Jika urutan pada data tidak sesuai lakukan penukaran posisi koordinat  
 if (line[1] != next\_line[0]):  
 Pass = next\_line\_idx  
 next\_line = solution[Pass]  
 while Pass < n-1 and (line[1] != next\_line[0]):  
 Pass += 1  
 next\_line = solution[Pass]  
  
 solutionArr[next\_line\_idx], solutionArr[Pass] = solutionArr[Pass], solutionArr[next\_line\_idx]  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print('Pilih dataset yang ingin digunakan')  
 print('1. Iris dataset')  
 print('2. Diabetes dataset')  
 print('3. Wine dataset')  
 dataset = int(input('Masukkan angka 1-3: '))  
  
 x\_value = 0  
 y\_value = 0  
  
 Title = ""  
 if dataset == 1:  
 data = datasets.load\_iris()  
 Title += "Iris dataset: "  
 elif dataset == 2:  
 data = datasets.load\_diabetes()  
 Title += "Diabetes dataset: "  
 elif dataset == 3:  
 data = datasets.load\_wine()  
 Title += "Wine dataset: "  
  
  
  
 df = pd.DataFrame(data.data, columns=data.feature\_names)  
 df['target'] = pd.DataFrame(data.target)  
 labels = data.target\_names  
 column\_name = data.feature\_names  
 solution = []  
  
 for j in range(len(column\_name)):  
 print(f'{j+1}. {column\_name[j]}')  
  
 x\_value = int(input('Masukkan nomor untuk kolom atribut-x: '))  
 y\_value = int(input('Maskkan nomor untuk kolom atribut-y: '))  
  
 Title += column\_name[x\_value-1] + " Vs " + column\_name[y\_value-1]  
  
 for i in range(len(labels)):  
 # mengambil data untuk setiap fitur (target)  
 data\_new = df[df['target'] == i]  
  
 #array solution yang akan digunakan  
 solution = []  
 plt.scatter(data\_new[column\_name[x\_value-1]], data\_new[column\_name[y\_value-1]],label=labels[i])  
 plt.title(Title)  
 plt.xlabel(column\_name[x\_value-1])  
 plt.ylabel(column\_name[y\_value-1])  
 plt.legend()  
 data\_new = data\_new[[column\_name[x\_value-1], column\_name[y\_value-1]]].values.tolist()  
 convexHull(data\_new)  
 # sort solution  
 sortSolution(solution)  
  
 # menyesuaikan format dari himpunan solusi agar bisa di tampilkan  
 hull = []  
 for line in solution:  
 hull.append(line[0])  
 hull.append(line[1])  
  
 # menampilkan hasil convexHull  
 df\_solution = pd.DataFrame(hull, columns=['x', 'y'])  
 plt.plot(df\_solution['x'], df\_solution['y'])  
  
  
 plt.show()

def findHull(pointArray, A, B):  
 if len(pointArray) == 0:  
 return  
 else:  
  
 farthestDistance = -1  
  
 # proses mencari titik dengan jarak paling jauh  
 C = [0 for i in range(2)]  
 for point in pointArray:  
 f = abs(findPointDistance(A,B, point))  
 if f > farthestDistance:  
 farthestDistance = f  
 C[0], C[1] = point[0],point[1]  
  
  
 # update array solution agar terisi garis yang paling baru  
 solution.remove([A, B])  
 solution.append([A, C])  
 solution.append([C, B])  
  
 # menghilangkan koordinat dari pointArray agar tidak digunakan lagi  
 pointArray.remove(C)  
  
 # memisahkan sisa koordinat menjadi kiri dan kanan  
 rightCoordinate = []  
 leftCoordinate = []  
  
 for point in pointArray:  
 if (getCoordinateSide(A, C, point) == 'above'):  
 rightCoordinate.append(point)  
  
 for point in pointArray:  
 if (getCoordinateSide(C, B, point) == 'above'):  
 leftCoordinate.append(point)  
  
 findHull(rightCoordinate, A, C)  
 findHull(leftCoordinate, C, B)  
  
  
# Main convexHull function  
def convexHull(pointArray):  
 # Sort array berdasarkan absis  
 # Jika absis sama, berdasarkan ordinat  
 pointArray = sorted(pointArray, key=lambda k: [k[0], k[1]])  
  
 A = findLeftMostPoint(pointArray)  
 B = findRightMostPoint(pointArray)  
  
 # membagi koordinat menjadi 2 array atas/bawah  
 topHalf = []  
 lowHalf = []  
  
  
 for point in pointArray:  
 if (getCoordinateSide(A, B, point) == 'above'):  
 topHalf.append(point)  
 elif (getCoordinateSide(A, B, point) == 'below'):  
 lowHalf.append(point)  
 else:  
 pass  
  
 solution.append([A, B])  
 solution.append([B, A])  
  
 # divide and conquer  
 findHull(topHalf, A, B)  
 findHull(lowHalf, B, A)  
  
  
# Sort the solution array  
# Hal ini diperlukan agar koordinat bisa di plot dengan sesuai  
def sortSolution(solutionArr):  
 n = len(solutionArr)  
  
 for i in range(0,n):  
 line = solutionArr[i]  
 next\_line\_idx = min(i+1, n-1)  
 next\_line = solution[next\_line\_idx]  
  
 # Jika urutan pada data tidak sesuai lakukan penukaran posisi koordinat  
 if (line[1] != next\_line[0]):  
 Pass = next\_line\_idx  
 next\_line = solution[Pass]  
 while Pass < n-1 and (line[1] != next\_line[0]):  
 Pass += 1  
 next\_line = solution[Pass]  
  
 solutionArr[next\_line\_idx], solutionArr[Pass] = solutionArr[Pass], solutionArr[next\_line\_idx]  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print('Pilih dataset yang ingin digunakan')  
 print('1. Iris dataset')  
 print('2. Diabetes dataset')  
 print('3. Wine dataset')  
 dataset = int(input('Masukkan angka 1-3: '))  
  
 x\_value = 0  
 y\_value = 0  
  
 Title = ""  
 if dataset == 1:  
 data = datasets.load\_iris()  
 Title += "Iris dataset: "  
 elif dataset == 2:  
 data = datasets.load\_diabetes()  
 Title += "Diabetes dataset: "  
 elif dataset == 3:  
 data = datasets.load\_wine()  
 Title += "Wine dataset: "  
  
  
  
 df = pd.DataFrame(data.data, columns=data.feature\_names)  
 df['target'] = pd.DataFrame(data.target)  
 labels = data.target\_names  
 column\_name = data.feature\_names  
 solution = []  
  
 for j in range(len(column\_name)):  
 print(f'{j+1}. {column\_name[j]}')  
  
 x\_value = int(input('Masukkan nomor untuk kolom atribut-x: '))  
 y\_value = int(input('Maskkan nomor untuk kolom atribut-y: '))  
  
 Title += column\_name[x\_value-1] + " Vs " + column\_name[y\_value-1]  
  
 for i in range(len(labels)):  
 # mengambil data untuk setiap fitur (target)  
 data\_new = df[df['target'] == i]  
  
 #array solution yang akan digunakan  
 solution = []  
 plt.scatter(data\_new[column\_name[x\_value-1]], data\_new[column\_name[y\_value-1]],label=labels[i])  
 plt.title(Title)  
 plt.xlabel(column\_name[x\_value-1])  
 plt.ylabel(column\_name[y\_value-1])  
 plt.legend()  
 data\_new = data\_new[[column\_name[x\_value-1], column\_name[y\_value-1]]].values.tolist()  
 convexHull(data\_new)  
 # sort solution  
 sortSolution(solution)  
  
 # menyesuaikan format dari himpunan solusi agar bisa di tampilkan  
 hull = []  
 for line in solution:  
 hull.append(line[0])  
 hull.append(line[1])  
  
 # menampilkan hasil convexHull  
 df\_solution = pd.DataFrame(hull, columns=['x', 'y'])  
 plt.plot(df\_solution['x'], df\_solution['y'])  
  
  
 plt.show()

elif (getCoordinateSide(A, B, point) == 'below'):  
 lowHalf.append(point)  
 else:  
 pass  
  
 solution.append([A, B])  
 solution.append([B, A])  
  
 # divide and conquer  
 findHull(topHalf, A, B)  
 findHull(lowHalf, B, A)  
  
  
# Sort the solution array  
# Hal ini diperlukan agar koordinat bisa di plot dengan sesuai  
def sortSolution(solutionArr):  
 n = len(solutionArr)  
  
 for i in range(0,n):  
 line = solutionArr[i]  
 next\_line\_idx = min(i+1, n-1)  
 next\_line = solution[next\_line\_idx]  
  
 # Jika urutan pada data tidak sesuai lakukan penukaran posisi koordinat  
 if (line[1] != next\_line[0]):  
 Pass = next\_line\_idx  
 next\_line = solution[Pass]  
 while Pass < n-1 and (line[1] != next\_line[0]):  
 Pass += 1  
 next\_line = solution[Pass]  
  
 solutionArr[next\_line\_idx], solutionArr[Pass] = solutionArr[Pass], solutionArr[next\_line\_idx]  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print('Pilih dataset yang ingin digunakan')  
 print('1. Iris dataset')  
 print('2. Breast Cancer dataset')  
 print('3. Wine dataset')  
 dataset = int(input('Masukkan angka 1-3: '))  
  
 x\_value = 0  
 y\_value = 0  
  
 Title = ""  
 if dataset == 1:  
 data = datasets.load\_iris()  
 Title += "Iris dataset: "  
 elif dataset == 2:  
 data = datasets.load\_diabetes()  
 Title += "Breast Cancer dataset: "  
 elif dataset == 3:  
 data = datasets.load\_wine()  
 Title += "Wine dataset: "  
  
 df = pd.DataFrame(data.data, columns=data.feature\_names)  
 df['target'] = pd.DataFrame(data.target)  
 labels = data.target\_names  
 column\_name = data.feature\_names  
 solution = []  
  
 for j in range(len(column\_name)):  
 print(f'{j+1}. {column\_name[j]}')  
  
 x\_value = int(input('Masukkan nomor untuk kolom atribut-x: '))  
 y\_value = int(input('Maskkan nomor untuk kolom atribut-y: '))  
  
 Title += column\_name[x\_value-1] + " Vs " + column\_name[y\_value-1]  
  
 for i in range(len(labels)):  
 # mengambil data untuk setiap fitur (target)  
 data\_new = df[df['target'] == i]  
  
 #array solution yang akan digunakan  
 solution = []  
 plt.scatter(data\_new[column\_name[x\_value-1]], data\_new[column\_name[y\_value-1]],label=labels[i])  
 plt.title(Title)  
 plt.xlabel(column\_name[x\_value-1])  
 plt.ylabel(column\_name[y\_value-1])  
 plt.legend()  
 data\_new = data\_new[[column\_name[x\_value-1], column\_name[y\_value-1]]].values.tolist()  
 convexHull(data\_new)  
 # sort solution  
 sortSolution(solution)  
  
 # menyesuaikan format dari himpunan solusi agar bisa di tampilkan  
 hull = []  
 for line in solution:  
 hull.append(line[0])  
 hull.append(line[1])  
  
 # menampilkan hasil convexHull  
 df\_solution = pd.DataFrame(hull, columns=['x', 'y'])  
 plt.plot(df\_solution['x'], df\_solution['y'])  
  
  
 plt.show()

df = pd.DataFrame(data.data, columns=data.feature\_names)  
 df['target'] = pd.DataFrame(data.target)  
 labels = data.target\_names  
 column\_name = data.feature\_names  
 solution = []  
  
 for j in range(len(column\_name)):  
 print(f'{j+1}. {column\_name[j]}')  
  
 x\_value = int(input('Masukkan nomor untuk kolom atribut-x: '))  
 y\_value = int(input('Maskkan nomor untuk kolom atribut-y: '))  
  
 Title += column\_name[x\_value-1] + " Vs " + column\_name[y\_value-1]  
  
 for i in range(len(labels)):  
 # mengambil data untuk setiap fitur (target)  
 data\_new = df[df['target'] == i]  
  
 #array solution yang akan digunakan  
 solution = []  
 plt.scatter(data\_new[column\_name[x\_value-1]], data\_new[column\_name[y\_value-1]],label=labels[i])  
 plt.title(Title)  
 plt.xlabel(column\_name[x\_value-1])  
 plt.ylabel(column\_name[y\_value-1])  
 plt.legend()  
 data\_new = data\_new[[column\_name[x\_value-1], column\_name[y\_value-1]]].values.tolist()  
 convexHull(data\_new)  
 # sort solution  
 sortSolution(solution)  
  
 # menyesuaikan format dari himpunan solusi agar bisa di tampilkan  
 hull = []  
 for line in solution:  
 hull.append(line[0])  
 hull.append(line[1])  
  
 # menampilkan hasil convexHull  
 df\_solution = pd.DataFrame(hull, columns=['x', 'y'])  
 plt.plot(df\_solution['x'], df\_solution['y'])  
  
  
 plt.show()

1. **Hasil pengujian Program**

|  |  |  |
| --- | --- | --- |
| Dataset yang digunakan | Kolom atribut yang diuji (X,Y) | Bukti Screenshot |
| Iris | Sepal length,Sepal Width | Chart, scatter chart  Description automatically generated |
| Petal length, Petal width | Chart  Description automatically generated |
| Breast Cancer | Worst area,worst concave points | Chart, scatter chart  Description automatically generated |
| Worst symmetry ,worst fractal dimension | Chart, scatter chart  Description automatically generated |
| Wine | Alcohol, malic\_acid | Chart, radar chart  Description automatically generated |
| Magnesium, total\_phenols | Chart, radar chart  Description automatically generated |

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Pustaka myConvexHull berhasil dibuat dan tidak ada kesalahan | v |  |
| 2. Convex hull yang dihasilkan sudah benar | v |  |
| 3. Pustaka myConvexHull dapat digunakan untuk menampilkan convex hull setiap label dengan warna yang berbeda. | v |  |
| 4. Bonus: program dapat menerima input dan menuliskan output untuk dataset lainnya. | v  (hanya untuk beberapa toy datasets) |  |

Link Repository:

https://github.com/geraldakbar/Tucil2Stima-13520143