LAPORAN TUGAS KECIL

IF2211 STRATEGI ALGORITMA

IMPLEMENTASI CONVEX HULL UNTUK VISUALISASI TES LINEAR SEPARABILITY DATASET DENGAN ALGORITMA DIVIDE AND CONQUER

Disusun oleh:

Claudia 13520076

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

ALGORITMA DIVIDE AND CONQUER

1. Dari sekumpulan titik (x, y) yang diberikan akan dilakukan sorting terlebih dahulu sehingga x terurut menaik. Jika ada nilai x yang sama maka akan diurutkan berdasarkan nilai y yang menaik.
2. Setelah terurut maka diambil titik terkiri (misalkan P1) dan titik terkanan (misalkan P2) yaitu kedua titik terujung dan dimasukkan ke dalam solusi
3. Lakukan pengecekan untuk titik lain selain P1 dan P2 apakah terletak di atas atau di bawah garis yang menghubungkan titik P1 dan P2 (misalkan P1P2) dan kelompokkan titik berdasarkan lokasi di atas P1P2 atau di bawah P1P2. Titik yang berada di garis P1P2 tidak akan membentuk convex hull sehingga tidak dilakukan pengecekan.
4. Misalkan untuk kumpulan titik yang berada di bawah garis P1P2, cari titik yang mempunyai jarak terjauh dengan garis P1P2 (misalkan titik terjauh tersebut P3) dan titik P3 dimasukkan ke dalam solusi. Setelah itu lakukan perulangan untuk garis P1P3 dan garis P3P2 sampai tidak ada titik yang berada di bawah garis.
5. Untuk kumpulan titik yang berada di atas garis P1P2 akan dilakukan hal yang sama juga seperti di langkah 4. Pada implementasinya, karena kumpulan titik di atas garis P1P2 sama saja dengan kumpulan titik di bawah P2P1 maka digunakan 1 fungsi yang sama
6. Setelah dilakukan langkah-langkah di atas, maka kumpulan titik-titik solusi tersebut akan membentuk Convex Hull

KODE PROGRAM

myConvexHull.py

Library untuk mencari Convex Hull dengan algoritma divide and conquer

```
import numpy as np
from pandas import array


# sort the 2d array by x then by y
def sort(A: array) -> (array):
    array = np.array(A)
    indexSorted = np.lexsort((array[:,1],array[:,0]))


    sortedArray = []
    for i in indexSorted:
        sortedArray.append(A[i])


    return sortedArray


# find determinant, a helper function to find pointPosition
def determinant(P1: array, P2: array, P3: array) -> (float):
    x1, y1 = P1[0], P1[1]
    x2, y2 = P2[0], P2[1]
    x3, y3 = P3[0], P3[1]


    return x1 * y2 + x3 * y1 + x2 * y3 -  x3 * y2 - x2 * y1 - x1 * y3


# find whether a point P3 is above/below line P1P2
def pointPosition(P1: array, P2: array, P3: array) -> (int):
    if (determinant(P1, P2, P3) > 0): # above line P1P2
        return 1
    elif (determinant(P1, P2, P3) < 0): # below line P1P2
        return -1
    else: # in line P1P2
        return 0


# finding the farthest Point from line P1P2
def farthestPoint(P1: array, P2: array, arrayofPoints: array):

    # assume arrayofPoints is not empty
```

```
    x1, y1 = P1[0], P1[1]
    x2, y2 = P2[0], P2[1]
    tempFarthest = -999
    farthest = [-999,-999]
    for point in arrayofPoints:
        x, y = point[0], point[1]
        distance = (1/2) * abs((x1 - x) * (y2 - y1) - (x1 - x2) * (y -
y1))
        if (distance > tempFarthest):
            tempFarthest = distance
            farthest = point


    return farthest


# find the solution Points
def findPoints(P1, P2, arrayofPoints, solutionPoints):

    if (len(arrayofPoints) == 0):
        pass
    else:


        # get the farthest point
        P3 = farthestPoint(P1, P2, arrayofPoints)


        # add the farthest point to the solution
        solutionPoints.append(P3)
        # remove it from the collection of points to be checked
        arrayofPoints.remove(P3)
        # remove P2, will add it again to the end
        if (P2 in solutionPoints):
            solutionPoints.remove(P2)


        # check for points below the line P1P3 and P3P2
        belowP1P3 = []
        belowP3P2 = []
        for point in arrayofPoints:
            if (pointPosition(P1, P3, point) < 0):
                belowP1P3.append(point)
            if (pointPosition(P3, P2, point) < 0):
                belowP3P2.append(point)


        # recursive
```

```
        findPoints(P1, P3, belowP1P3, solutionPoints)
        findPoints(P3, P2, belowP3P2, solutionPoints)


        # append to the end of solution points
        solutionPoints.append(P2)


def myConvexHull(A: array) -> (array):

    # sort the points ascending by x, then by y
    sortedArray = sort(A)


    # take the left-end point
    P1 = sortedArray[0]
    # take the right-end point
    P2 = sortedArray[len(sortedArray)-1]


    # add the left-end and right-end points to solution
    solutionPoints = []
    solutionPoints.append(P1)
    solutionPoints.append(P2)


    # remove the left-end point and the right-end point from the
collection of points
    sortedArray.remove(P1)
    sortedArray.remove(P2)


    # divide the collection of points into above and below side
    above = []
    below = []
    for point in sortedArray:
        if (pointPosition(P1, P2, point) > 0): # if point is above line
P1P2
            above.append(point)
        elif (pointPosition(P1, P2, point) < 0): # if point is below
line P1P2
            below.append(point)

    # divide and conquer
    findPoints(P1, P2, below, solutionPoints) # find the convex hull
points above
    findPoints(P2, P1, above, solutionPoints) # find the convex hull
points below
```

```
    return solutionPoints
```

plot.py

Untuk visualisasi Convex Hull

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from myConvexHull import myConvexHull


# print dataset available to choose
print()
print("Which dataset do you want?")
print("1. iris")
print("2. wine")
print("3. breast_cancer")


# assume that user input is always valid
print()
choice = int(input("Enter the number of your choice: "))
print()


data = datasets


if (choice == 1):
    data = datasets.load_iris()
elif (choice == 2):
    data = datasets.load_wine()
else:
    data = datasets.load_breast_cancer()


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


# print the name of attributes
print("Attributes:")
for i in range (len(data.feature_names)):
```

```
    print(str(i+1) + ". " + str(data.feature_names[i]))
print()


# assume that user input is always valid
attribute1 = int(input("Enter the number of the first attribute: "))
attribute2 = int(input("Enter the number of the second attribute: "))


x = attribute1 - 1
y = attribute2 - 1


plt.figure(figsize=(10, 6))
colors = ['b','r','g']


title = str(data.feature_names[x]) + " vs " + str(data.feature_names[y])
plt.title(title)
plt.xlabel(data.feature_names[x])
plt.ylabel(data.feature_names[y])


for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[x,y]].values
    hull = myConvexHull(bucket.tolist()) # using the divide and conquer
implementation of convex hull
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])


    for j in range(1, len(hull) + 1):
        if j == len(hull):
            j = 0
        p0 = hull[j - 1]
        p1 = hull[j]
        plt.plot((p0[0], p1[0]), (p0[1], p1[1]), colors[i])


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

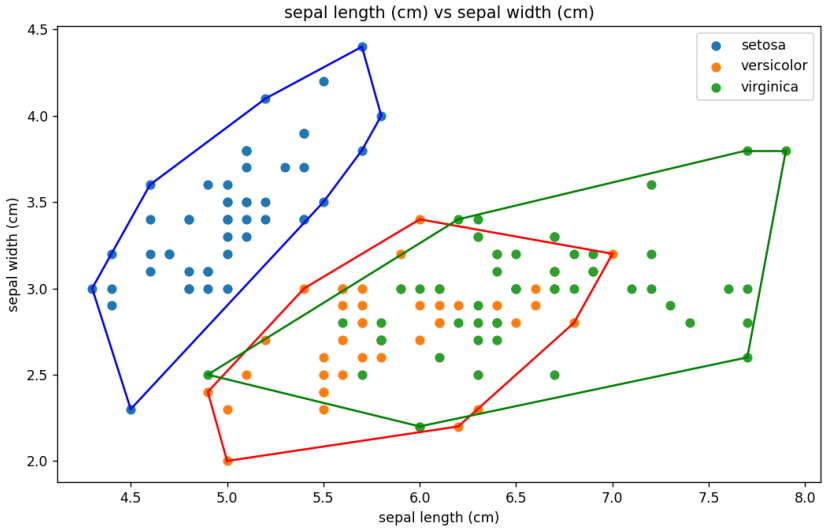
SKRINSUT INPUT-OUTPUT PROGRAM

Dataset Iris (sepal length vs sepal width)

Input

```
(venv) PS D:\Github\stima-convex-hull> python plot.py

Which dataset do you want?
1. iris
2. wine
3. breast_cancer

Enter the number of your choice: 1

Attributes:
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)

Enter the number of the first attribute: 1
Enter the number of the second attribute: 2
```

Output

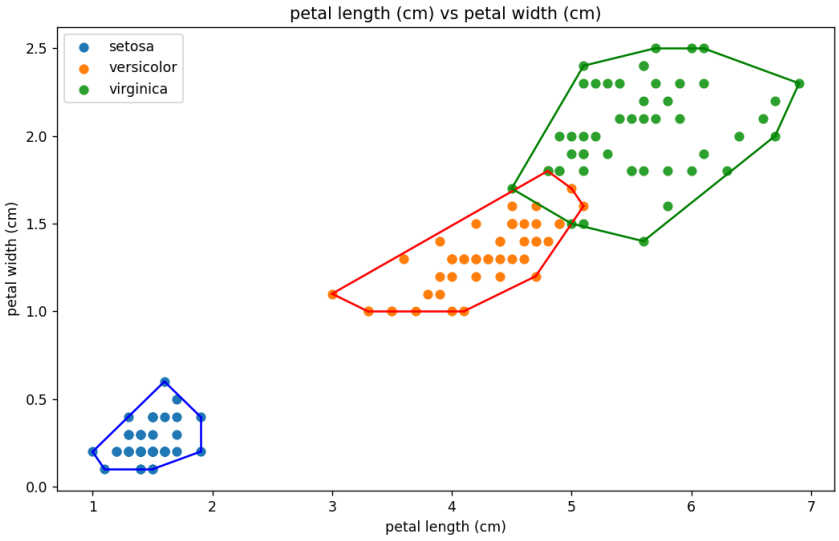


Dataset Iris (petal length vs petal width)

Input

```
(venv) PS D:\Github\stima-convex-hull> python plot.py

Which dataset do you want?
1. iris
2. wine
3. breast_cancer

Enter the number of your choice: 1

Attributes:
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)

Enter the number of the first attribute: 3
Enter the number of the second attribute: 4
```

Output

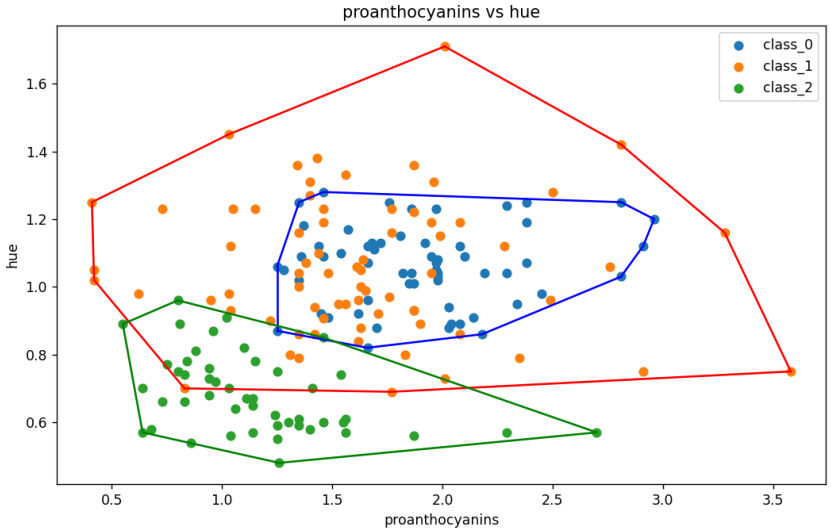


Dataset Wine (proanthocyanins vs hue)

Input

```
(venv) PS D:\Github\stima-convex-hull> python plot.py

Which dataset do you want?
1. iris
2. wine
3. breast_cancer

Enter the number of your choice: 2

Attributes:
1. alcohol
2. malic_acid
3. ash
4. alcalinity_of_ash
5. magnesium
6. total_phenols
7. flavanoids
8. nonflavanoid_phenols
9. proanthocyanins
10. color_intensity
11. hue
12. od280/od315_of_diluted_wines
13. proline

Enter the number of the first attribute: 9
Enter the number of the second attribute: 11
```

Output

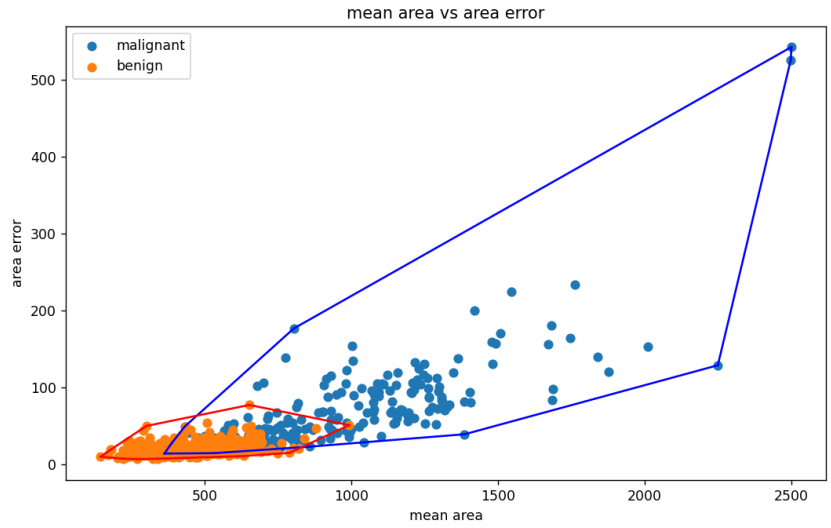


Dataset Breast Cancer (mean area vs area error)

Input

```
(venv) PS D:\Github\stima-convex-hull> python plot.py

Which dataset do you want?
1. iris
2. wine
3. breast_cancer

Enter the number of your choice: 3

Attributes:
1. mean radius
2. mean texture
3. mean perimeter
4. mean area
5. mean smoothness
6. mean compactness
7. mean concavity
8. mean concave points
9. mean symmetry
10. mean fractal dimension
11. radius error
12. texture error
13. perimeter error
14. area error
15. smoothness error
16. compactness error
17. concavity error
18. concave points error
19. symmetry error
20. fractal dimension error
21. worst radius
22. worst texture
23. worst perimeter
24. worst area
25. worst smoothness
26. worst compactness
27. worst concavity
28. worst concave points
29. worst symmetry
30. worst fractal dimension

Enter the number of the first attribute: 4
Enter the number of the second attribute: 14
```

Output



KELENGKAPAN PROGRAM

Poin	Ya	Tidak
Pustaka myConvexHull berhasil dibuat dan tidak ada kesalahan	Ya	
Convex hull yang dihasilkan sudah benar	Ya	
Pustaka myConvexHull dapat digunakan untuk menampilkan convex hull setiap label dengan warna yang berbeda.	Ya	
Bonus: program dapat menerima input dan menuliskan output untuk dataset lainnya.	Ya	

ALAMAT DRIVE

https://drive.google.com/drive/folders/1YYA3mlNY3V8A3KEcnebu9VOeBZo1tV6v?usp=sharing

ALAMAT GITHUB

https://github.com/clauculus/stima-convex-hull