

LAPORAN TUGAS KECIL

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

Ditujukan untuk memenuhi Tugas Kecil 2 mata kuliah IF2211 Strategi Algoritma pada Semester II Tahun Akademik 2021/2022

Disusun oleh:

Sarah Azka Arief (K2)

13520083



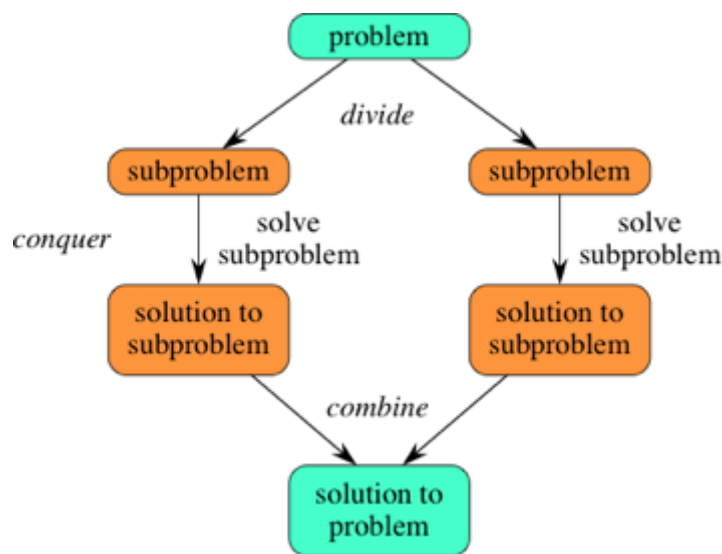
**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG**

2022

BAB I

Deskripsi Algoritma *Divide and Conquer*

Algoritma *divide and conquer* merupakan algoritma yang secara rekursif memecah suatu permasalahan menjadi upa-permasalahan yang lebih kecil dan sederhana dengan tipe persoalan yang masih identik dengan permasalahan awalnya. Metode pemecahan masalah ini akan mencari penyelesaian dari setiap upa-permasalahan dan menggabungkannya agar didapat penyelesaian dari permasalahan awal.



Gambar 1. Divide and Conquer

(sumber: <https://www.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/divide-and-conquer-algorithms>)

Metode *divide and conquer* dapat diterapkan untuk algoritma *sorting* (*quicksort* dan *merge sort*) hingga *Fast Fourier Transform* (FFT). Metode ini juga bisa dipakai pada visualisasi tes *linear separability dataset* seperti pada tugas kecil ini.

Pada tugas kecil kali ini, algoritma *divide and conquer* diimplementasikan dalam pembuatan suatu *library* bernama *myConvexHull*. *Library* tersebut terdapat pada *file* ‘*myConvexHull.py*’ dan berfungsi untuk mencari *convex hull* dari suatu dataset. Terdapat 8 fungsi pada *file* tersebut yang rinciannya sebagai berikut:

No.	Fungsi	Keterangan
1.	<code>getExtremes(arr)</code>	Mencari titik ekstrem (ujung kiri dan kanan)
2.	<code>inLine(p1, p2, p3)</code>	Melihat apakah titik p3 berada pada garis yang dibentuk oleh titik p1 dan p2

3.	checkPosition(p1, p2, p3)	Melihat apakah titik p3 berada pada kiri dari garis yang dibentuk oleh p1 dan p2 atau sebaliknya
4.	getAngle(p1, p2, p3)	Mencari sudut yang dibuat antara garis p1 dan p3 serta garis p3 dan p2
5.	getFarthest(p1, p2, list)	Mencari titik terjauh dari garis p1 dan p2 dari suatu list berisi titik
6.	insideTriangle(area, p1, p2, pmax)	Menghapus titik pada array of points berupa area yang berada di dalam segitiga yang dibentuk oleh titik p1, p2, dan pmax
7.	divideArea(area, p1, p2, direction)	Mencari titik-titik yang membentuk garis convex hull pada suatu area dengan titik acuan p1 dan p2 serta arah pengecekan berdasarkan direction
8.	convexHull(bucket)	Mencari convex hull dari suatu kumpulan data

Adapun *file* 'main.py' yang memungkinkan pengguna untuk memilih dataset yang tersedia atau memilih dataset lainnya dalam bentuk *file* CSV, dengan catatan *file* CSV tersebut harus memiliki atribut berupa target (tipe klasifikasi). Setelah memilih dataset, pengguna dapat memilih dua dari seluruh atribut yang tersedia untuk diproses sehingga didapat hasil visualisasi dari tes *linear separability dataset* tersebut.

Pencarian *convex hull* menggunakan *library* myConvexHull memanfaatkan algoritma *divide and conquer* serta sifat rekursif untuk menyelesaikan permasalahan. Secara umum, bagian rekursif dari penyelesaian ini dapat dibagi menjadi tiga bagian yaitu rekursi bagian umum yang pasti akan dijalankan, rekursi bagian atas yang hanya dijalankan apabila rekursi dilakukan saat memeriksa bagian atas atau saat garis acuan masih dibentuk oleh titik ekstrem keseluruhan, dan rekursi bagian bawah yang hanya dijalankan apabila rekursi dilakukan saat memeriksa bagian bawah atau saat garis acuan masih dibentuk oleh titik ekstrem keseluruhan. Adapun rincian dari penyelesaiannya sebagai berikut:

1. Fungsi convexHull menerima *ndarray* berisi titik yang dicopy menjadi array baru bernama 'area' dan menginisialisasi suatu *list* global bernama 'result' yang akan menyimpan hasil pencarian dalam bentuk *array* berisi dua titik yang membentuk garis dari *convex hull*
2. Titik ekstrem dari kumpulan titik pada 'area' dicari dengan menggunakan fungsi getExtremes yang melakukan *sorting* berdasarkan absis (ekstrem kiri dan kanan)
3. Fungsi rekursif yakni divideArea dipanggil dengan memasukkan parameter area berupa 'area', p1 berupa 'minPoint', p2 berupa 'maxPoint', dan direction berupa 0

a. Rekursi Bagian Umum

1. Titik p1 dan p2 akan membentuk suatu garis yang merupakan garis acuan, sehingga kedua titik tersebut dihapus dari 'area' agar tidak perlu diproses
2. Dua buah *list* kosong bernama 'upper' dan 'lower' diinisialisasi. List ini akan menampung titik-titik yang berada di atas dan di bawah garis acuan
3. Proses pemeriksaan untuk setiap titik dilakukan dengan terlebih dahulu memastikan bahwa titik tidak berada pada garis acuan dengan menggunakan fungsi *inLine*. Apabila titik berada pada garis acuan, titik dihapus dari 'area' agar tidak diproses
4. Apabila titik tidak berada pada garis acuan, titik tersebut akan ditentukan posisinya relatif terhadap garis acuan dengan menggunakan fungsi *checkPosition* yang mencari determinan dari matriks dengan *minPoint* berupa (x1, y1), *maxPoint* berupa (x2, y2), dan titik tersebut berupa (x3, y3) dan visualisasi matriks sebagai berikut

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = x_1y_2 + x_3y_1 + x_2y_3 - x_3y_2 - x_2y_1 - x_1y_3$$

Apabila didapat determinan berupa bilangan positif, maka titik tersebut akan dimasukkan ke dalam *list* 'upper' dan sebaliknya akan dimasukkan ke *list* 'lower'. Setelah semua titik telah diperiksa, akan didapat dua bagian berbeda yang diwakili oleh kumpulan titik 'upper' dan kumpulan titik 'lower'.

b. Rekursi Bagian Atas

Catatan: apabila parameter 'direction' yang dimasukkan pada fungsi *divideArea* kurang dari 0, maka bagian ini tidak dijalankan

1. Selanjutnya, dilakukan pemeriksaan terhadap bagian atas garis acuan. Apabila 'upper' kosong alias tidak ada titik lagi di bagian atas dari garis acuan, maka dua titik yang membentuk garis acuan (p1 dan p2) akan dimasukkan ke 'result'.
2. Namun, apabila 'upper' tidak kosong, maka:
 - Akan dicari titik pada 'upper' yang jaraknya paling jauh dan membentuk sudut paling besar terhadap garis acuan dengan menggunakan fungsi *getFarthest*
 - Untuk setiap titik pada 'upper', akan dihapus titik yang berada di dalam segitiga yang dibentuk oleh garis acuan dan titik terjauh dengan

menggunakan fungsi `insideTriangle` yang juga menghapus titik yang berada pada garis dari segitiga

- Setelah itu, dilakukan rekursi untuk memeriksa bagian kiri dari bagian atas dari garis acuan dengan memanggil `divideArea` dengan parameter area berupa 'upper', p1 berupa p1, p2 berupa 'farthest', dan direction berupa 1
- Selain itu, dilakukan juga rekursi untuk memeriksa bagian kanan dari bagian atas dari garis acuan dengan memanggil `divideArea` dengan parameter area berupa 'upper', p1 berupa 'farthest', p2 berupa p2, dan direction berupa 1

c. Rekursi Bagian Bawah

Catatan: apabila parameter 'direction' yang dimasukkan pada fungsi `divideArea` lebih dari 0, maka bagian ini tidak dijalankan

1. Selanjutnya, dilakukan pemeriksaan terhadap bagian bawah garis acuan. Apabila 'lower' kosong alias tidak ada titik lagi di bagian atas dari garis acuan, maka dua titik yang membentuk garis acuan (p1 dan p2) dimasukkan ke 'result'.
2. Namun, apabila 'lower' tidak kosong, maka:
 - Akan dicari titik pada 'lower' yang jaraknya paling jauh dan membentuk sudut paling besar terhadap garis acuan dengan menggunakan fungsi `getFarthest`
 - Untuk setiap titik pada 'lower' akan dihapus titik yang berada di dalam segitiga yang dibentuk oleh garis acuan dan titik terjauh dengan menggunakan fungsi `insideTriangle` yang juga menghapus titik yang berada pada garis dari segitiga
 - Setelah itu, dilakukan rekursi untuk memeriksa bagian kiri dari bagian bawah dari garis acuan dengan memanggil `divideArea` dengan parameter area berupa 'lower', p1 berupa p1, p2 berupa 'farthest', dan direction -1
 - Selain itu, dilakukan juga rekursi untuk memeriksa bagian kanan dari bagian bawah dari garis acuan dengan memanggil `divideArea` dengan parameter area berupa 'lower', p1 berupa 'farthest', p2 berupa p2, dan direction -1

BAB II

Kode Program

2.1 myConvexHull.py

```
1 import numpy as np
2 import math
3
4 def getExtremes(arr):
5     """Returns leftmost and rightmost point"""
6
7     arr = arr[arr[:, 0].argsort()] # sort based on x - coordinate
8     minPoint = arr[0]
9     maxPoint = arr[-1]
10    return minPoint, maxPoint
11
12 def inLine(p1, p2, p3):
13     """Returns true if p3 is in line (check by distance)"""
14
15     return ((math.dist(p1, p3) + math.dist(p2, p3)) == math.dist(p1, p2))
16
17 def checkPosition(p1, p2, p3):
18     """Checks position of p3 relative to p1 and p2 through determinant
19
20     Return 1 when p3 is in rightside and -1 when p3 is in leftside"""
21
22     # init matrix
23     p1 = np.append(p1, 1)
24     p2 = np.append(p2, 1)
25     p3 = np.append(p3, 1)
26     temp = [p1, p2, p3]
27
28     det = np.linalg.det(temp)
29     if det < 0:
30         return 1 # p3 in rightside
31     else:
32         return -1 # p3 in leftside
33
34 def getAngle(p1, p2, p3):
35     """Return angle between p1, p3, and p2"""
36
37     p1x, p1y = p1[0] - p3[0], p1[1] - p3[1]
38     a1 = math.atan2(p1x, p1y) # find arctan
39     if a1 < 0: # negative case
40         a1 += math.pi * 2
41     p2x, p2y = p2[0] - p3[0], p2[1] - p3[1]
42     a2 = math.atan2(p2x, p2y) # find arctan
43     if a2 < 0: # negative case
44         a2 += math.pi * 2
45     if (a1 > a2): # negative case
46         return (math.pi * 2 + a2 - a1)
47     else:
48         return a2 - a1
49
50 def getFarthest(p1, p2, list):
51     """Return farthest point from p1 and p2"""
52
53     farthest = [0, 0] # store distance and angle for comparison
54     for arr in list:
55         angle = getAngle(p1, p2, arr) # get angle for pmax comparison
56         dist = math.dist(p1, arr) + math.dist(p2, arr) # get distance p1-point-p2
57         if (dist > farthest[0] or (dist == farthest and angle > farthest[1])):
58             farthest[0] = dist
59             farthest[1] = angle
60             point = arr
61     return point # return point # return farthest point only
```

Gambar 2. myConvexHull.py (line 1 – 61)

```

1 def insideTriangle(area, p1, p2, pmax):
2     """Remove points inside triangle with points p1, p2, and pmax"""
3
4     for arr in area:
5         # check if point in triangle line
6         if not(inLine(p1, p2, arr) or inLine(p1, pmax, arr) or inLine(pmax, p2, arr)):
7             # check if point in leftside of pmax and rightside of line pmax-p2
8             if (arr[0] <= pmax[0] and checkPosition(p1, pmax, arr) == 1):
9                 np.delete(area, np.argwhere(area == arr))
10            # check if point in rightside of pmax and leftside of line pmax-p2
11            elif (arr[0] >= pmax[0] and checkPosition(pmax, p2, arr) == -1):
12                np.delete(area, np.argwhere(area == arr))
13        else:
14            np.delete(area, np.argwhere(area == arr))
15
16 def divideArea(area, p1, p2, direction):
17     """Find sets by dividing areas based on line with points p1 and p2
18
19     area is filled with all points to be checked and direction is based on recursion phase
20     (0 for first recursion, 1 for upper area recursion, -1 for lower area recursion)"""
21
22     np.delete(area, np.argwhere(area == p1))
23     np.delete(area, np.argwhere(area == p2))
24     # List for storing points to be checked
25     upper = []
26     lower = []
27
28     for point in area:
29         if not(inLine(p1, p2, point)):
30             # if point above line, add point to upper list
31             if (checkPosition(p1, p2, point) == -1):
32                 upper.append(point)
33             # if point below line, add point to lower list
34             else:
35                 lower.append(point)
36         else:
37             np.delete(area, np.argwhere(area == point))
38
39     # continue checking upper area for init and upper area recursion
40     if (direction >= 0):
41         if (not(upper)):
42             result.append([p1, p2]) # add line to end result if upper list is empty
43         else:
44             farthest = getFarthest(p1, p2, upper) # get pmax
45             insideTriangle(upper, p1, p2, farthest) # remove inside triangle
46             divideArea(upper, p1, farthest, 1) # leftside upper area recursion
47             divideArea(upper, farthest, p2, 1) # rightside upper area recursion
48
49     # continue checking lower area for init and lower area recursion
50     if (direction <= 0):
51         if (not(lower)):
52             result.append([p1, p2]) # add line to end result if lower list is empty
53         else:
54             farthest = getFarthest(p1, p2, lower) # get pmax
55             insideTriangle(lower, p1, p2, farthest) # remove inside triangle
56             divideArea(lower, p1, farthest, -1) # leftside lower area recursion
57             divideArea(lower, farthest, p2, -1) # rightside lower area recursion
58
59 def convexHull(bucket):
60     """Returns list of lines that make up convex hull
61     by implementing divide and conquer algorithm"""
62
63     # initialize variables
64     global result
65     result = []
66     area = np.copy(bucket)
67     # get two starting points
68     minPoint, maxPoint = getExtremes(area)
69     # find lines for convex hull
70     divideArea(area, minPoint, maxPoint, 0) # first recursion
71     return result

```

Gambar 3. myConvexHull.py (line 63 – 133)

2.2 main.py

```
1 import pandas as pd
2 from sklearn import datasets
3 import matplotlib.pyplot as plt
4 from myConvexHull import convexHull
5
6 def pickData():
7     """Get user chosen data and columns out of 3 available datasets"""
8
9     # print list of available datasets
10    print("List of available datasets")
11    print("1. CSV File")
12    print("2. Iris")
13    print("3. Wine")
14    print("4. Breast Cancer")
15    choice = int(input("Enter dataset number: "))
16
17    # initialize csv dataset
18    if (choice == 1):
19        print("-----")
20        print("[Make sure CSV file is in 'test' folder & has target]")
21        datasetname = input("Enter filename (<FILENAME>.csv): ")
22        try:
23            df = pd.read_csv("./test/" + datasetname)
24        except:
25            print("File not found")
26            return
27        # check if file has target
28        if (('target' not in df) and ('Target' not in df)):
29            print("Unable to process file (no target found)")
30            return
31        # initialize target
32        target = (df.target.unique())
33        # initialize sklearn dataset
34        else:
35            if (choice == 2):
36                data = datasets.load_iris()
37                datasetname = "Iris"
38            elif (choice == 3):
39                data = datasets.load_wine()
40                datasetname = "Wine"
41            else:
42                data = datasets.load_breast_cancer()
43                datasetname = "Breast Cancer"
44            df = pd.DataFrame(data.data, columns=data.feature_names)
45            # initialize target
46            df['Target'] = pd.DataFrame(data.target)
47            target = data.target_names
48
49        # initialize column choices
50        print("-----")
51        print("List of available columns in", datasetname, "dataset")
52        i = 1
53        for col in df.columns:
54            if (col == 'target' or col == 'Target'):
55                continue
56            print(str(i) + ". " + str(col))
57            i += 1
58        print("Enter two column numbers to test linear separability")
59        x = int(input("First column number: ")) - 1
60        y = int(input("Second column number: ")) - 1
61        title = (df.columns[x] + " vs " + df.columns[y])
62
63        # process data through acquired choices
64        processData(target, df, x, y, title)
65
66    def processData(target, df, x, y, title):
67        """Visualize linear separability dataset test"""
68
69        # initialize plot and color
70        plt.figure(figsize=(10, 6))
71        colors = ['b', 'r', 'g']
72        plt.title(title)
73        plt.xlabel(df.columns[x])
74        plt.ylabel(df.columns[y])
75
76        # plot convex hull per target
77        for i in range(len(target)):
78            try:
79                bucket = df[df['Target'] == i]
80            except:
81                bucket = df[df['target'] == i]
82            bucket = bucket.iloc[:, [x, y]].values
83            hull = convexHull(bucket)
84            plt.scatter(bucket[:, 0], bucket[:, 1], label=target[i])
85            for simplex in hull:
86                plt.plot([simplex[0][0], simplex[1][0]], [simplex[0][1], simplex[1][1]], colors[i % 3])
87        plt.legend()
88        plt.show()
89
90    pickData()
```

Gambar 4. main.py

BAB III

Screenshot Input/Output

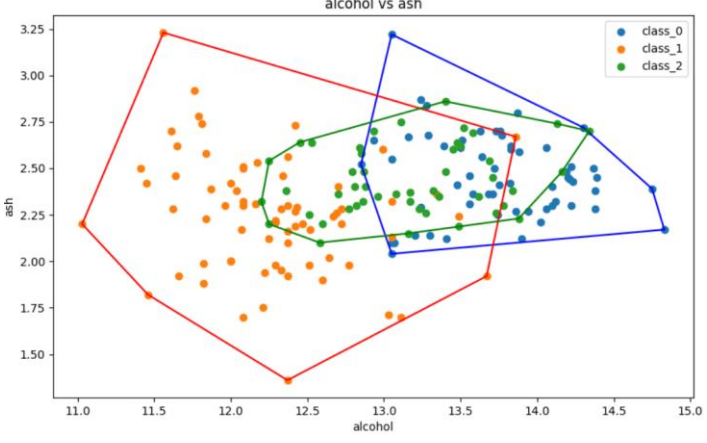
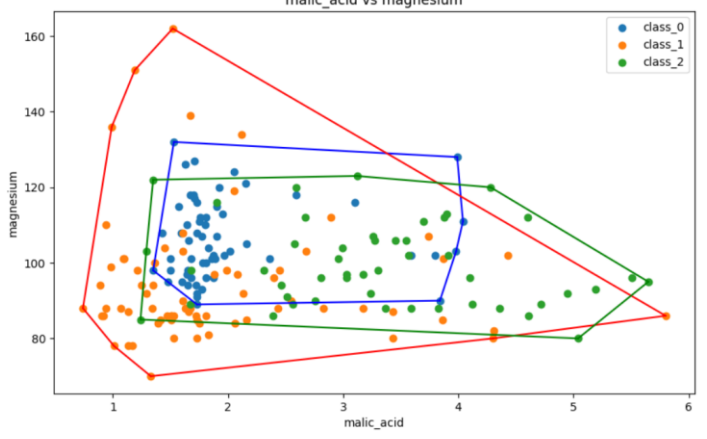
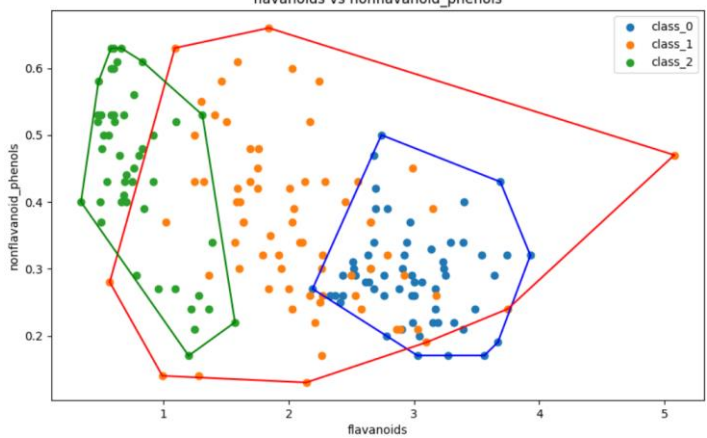
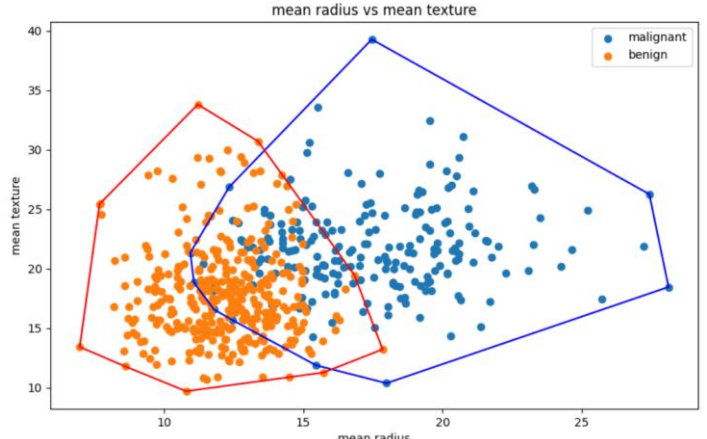
```
List of available datasets
1. CSV File
2. Iris
3. Wine
4. Breast Cancer
Enter dataset number: 2
-----
List of available columns in Iris dataset
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)
Enter two column numbers to test linear separability
First column number: 1
Second column number: 3
```

Gambar 5. Input dataset sklearn (Iris)

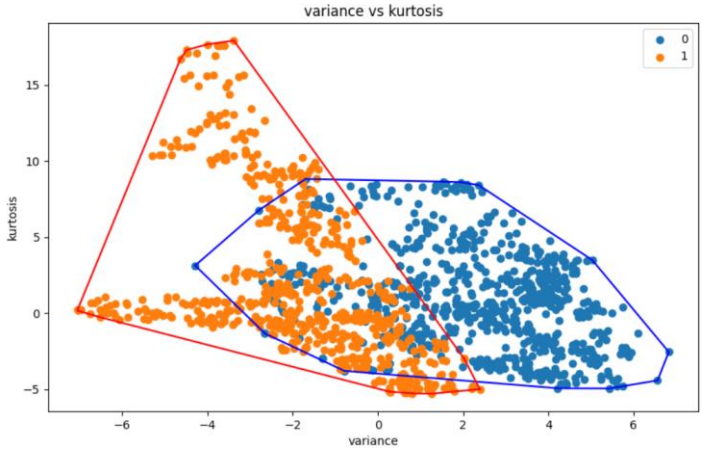
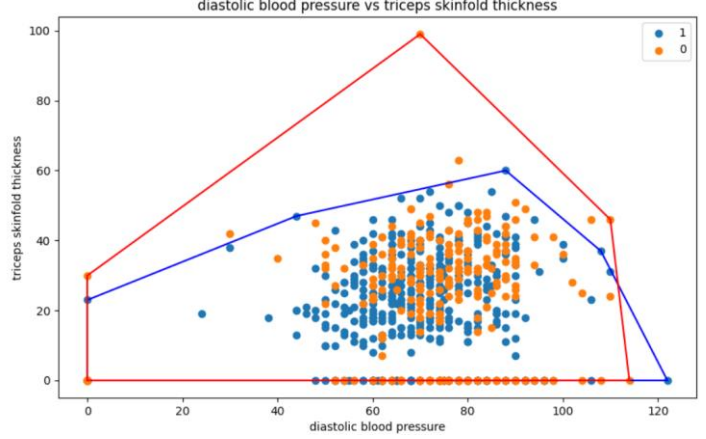
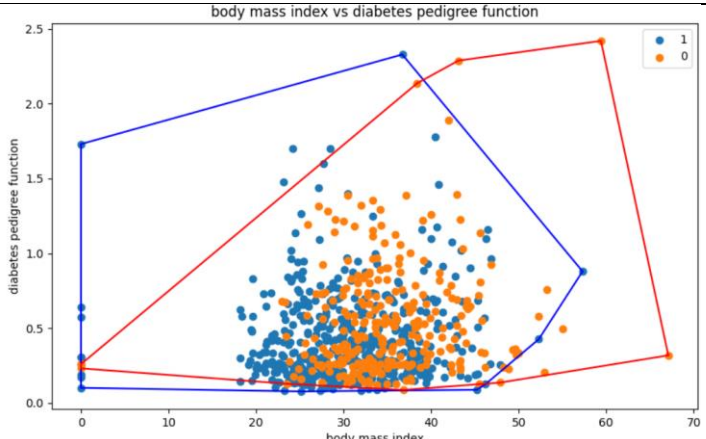
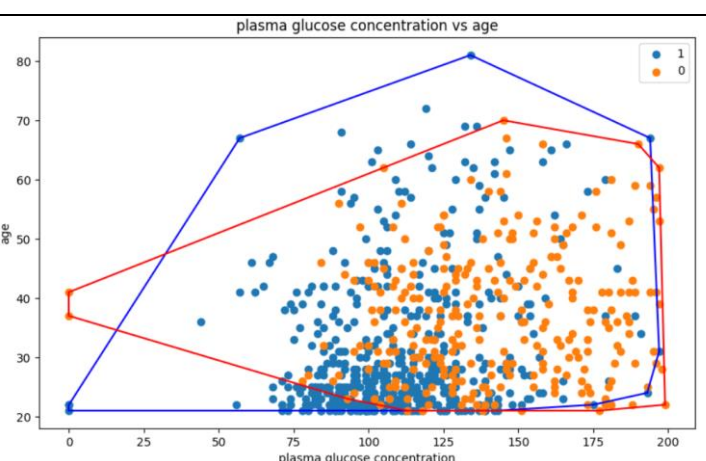
```
List of available datasets
1. CSV File
2. Iris
3. Wine
4. Breast Cancer
Enter dataset number: 1
-----
[Make sure CSV file is in 'test' folder & has target]
Enter filename (<FILENAME>.csv): banknote.csv
-----
List of available columns in banknote.csv dataset
1. variance
2. skewness
3. kurtosis
4. entropy
Enter two column numbers to test linear separability
First column number: 1
Second column number: 3
```

Gambar 6. Input dataset CSV (banknote.csv)

No.	Nama Dataset	Atribut	Screenshot
1.	Iris (sklearn)	- sepal length - sepal width	
2.		- petal length - petal width	
3.		- sepal length - petal length	

4.	Wine (sklearn)	- alcohol - ash	
5.		- malic acid - magnesium	
6.		- flavonoid - non flavanoid phenols	
7.	Breast Cancer (sklearn)	- mean radius - mean texture	

8.		<ul style="list-style-type: none"> - mean perimeter - mean area 	
9.		<ul style="list-style-type: none"> - mean compactness - mean concavity 	
10.	Banknote (CSV)	<ul style="list-style-type: none"> - variance - skewness 	
11.		<ul style="list-style-type: none"> - kurtosis - entropy 	

12.		<ul style="list-style-type: none"> - variance - kurtosis 	
13.		<ul style="list-style-type: none"> - diastolic blood pressure - triceps skinfold thickness 	
14.	Indiansdiabetes (CSV)	<ul style="list-style-type: none"> - body mass index - diabetes pedigree function 	
15.		<ul style="list-style-type: none"> - plasma glucose concentration - age 	

LAMPIRAN

Github Repository: <https://github.com/azkazkazka/convex-hull>

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