

LAPORAN TUGAS KECIL

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

Ditujukan untuk memenuhi salah satu tugas kecil mata kuliah IF2211 Strategi Algoritma
pada Semester II Tahun Akademik 2021/2022

Disusun oleh:

Adiyansa Prasetya Wicaksana (K2)

13520044



**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022**

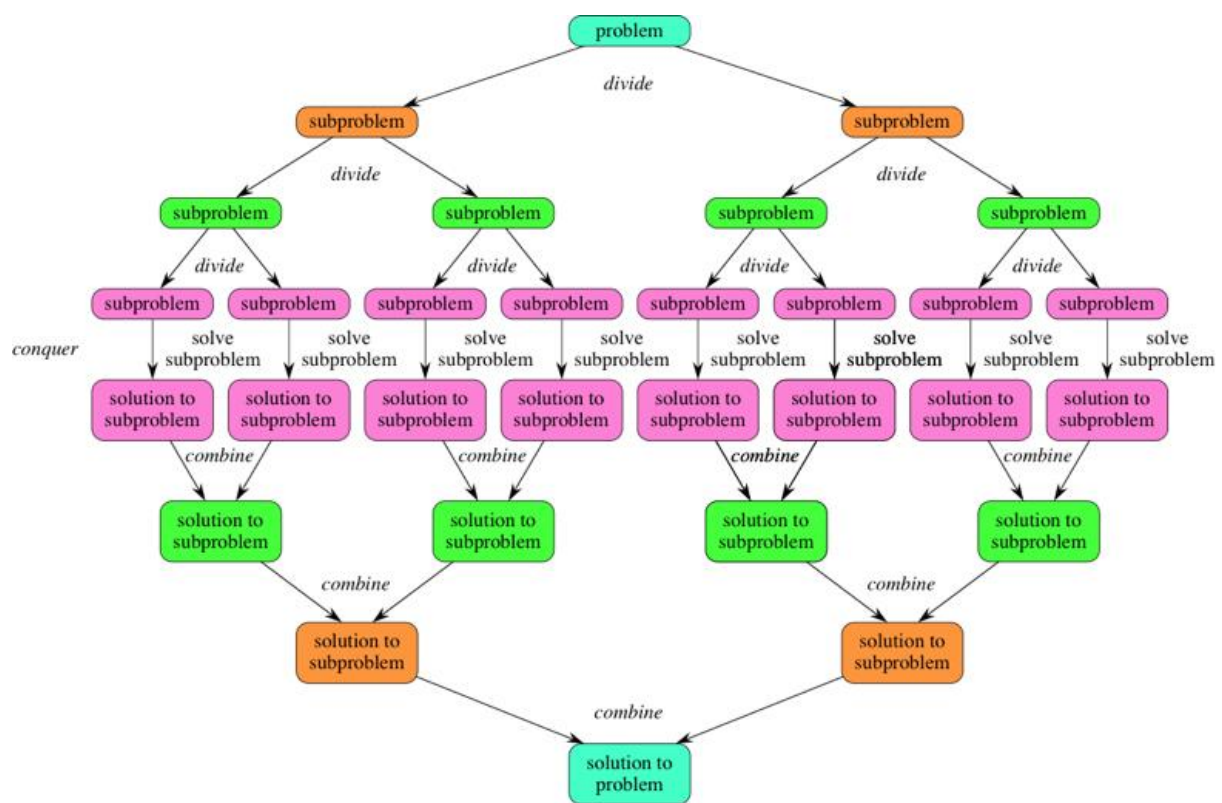
DAFTAR ISI

DAFTAR ISI	i
BAB I PENJELASAN ALGORITMA PROGRAM.....	1
BAB II IMPLEMENTASI PROGRAM.....	4
BAB III HASIL PERCOBAAN	12
LAMPIRAN	ii
REFERENSI	iii

BAB I

PENJELASAN ALGORITMA PROGRAM

Divide and Conquer pada awalnya merupakan strategi militer yang dikenal dengan nama *divide ut imperes*. Namun, sekarang strategi tersebut menjadi strategi yang dipakai dalam ilmu komputer sebagai *Divide and Conquer*. Seperti namanya, *Divide* membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula yang berukuran lebih kecil. Lalu, *Conquer* menyelesaikan masing-masing upa-persoalan, akan diselesaikan secara langsung jika berukuran cukup kecil atau secara rekursif jika masih berukuran besar. Akhirnya, setelah upa-persoalan dibagi menjadi lebih kecil dan diselesaikan, nanti hasil upa-persoalan akan digabungkan menjadi satu hasil utama.



Gambar 1.1 Ilustrasi Divide and Conquer

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

Dalam implementasinya ke dalam algoritma, strategi *Divide and Conquer* pada umumnya diselesaikan menggunakan fungsi/prosedur yang dipanggil terus menerus secara rekursif, dan hasil akhirnya akan digabungkan di fungsi utamanya.

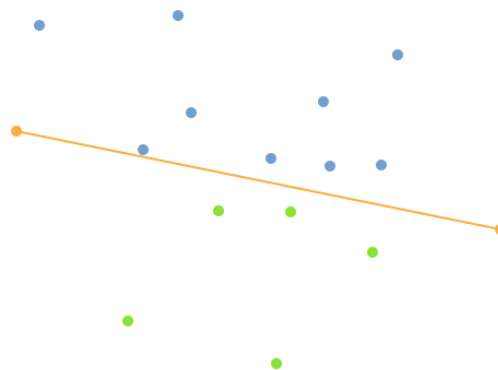
```
procedure DIVIDEandCONQUER(input  $P$  : problem,  $n$  : integer)  
{ Menyelesaikan persoalan  $P$  dengan algoritma divide and conquer  
Masukan: masukan persoalan  $P$  berukuran  $n$   
Luaran: solusi dari persoalan semula }  
Deklarasi  
   $r$  : integer  
  
Algoritma  
  if  $n \leq n_0$  then {ukuran persoalan  $P$  sudah cukup kecil }  
    SOLVE persoalan  $P$  yang berukuran  $n$  ini  
  else  
    DIVIDE menjadi  $r$  upa-persoalan,  $P_1, P_2, \dots, P_r$ , yang masing-masing berukuran  $n_1, n_2, \dots, n_r$   
    for masing-masing  $P_1, P_2, \dots, P_r$ , do  
      DIVIDEandCONQUER( $P_i, n_i$ )  
    endfor  
    COMBINE solusi dari  $P_1, P_2, \dots, P_r$  menjadi solusi persoalan semula  
  endif
```

Gambar 1.2 Skema Umum Strategi Divide and Conquer

(Sumber: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-\(2021\)-Bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf))

Convex merupakan himpunan titik pada bidang planar jika sembarang dua titik pada bidang tersebut (misal p dan q), seluruh segmen garis yang berakhir di p dan q berada pada himpunan tersebut. Convex Hull dari himpunan titik S adalah himpunan *convex* terkecil yang mengandung S . Penerapan strategi algoritma *Divide and Conquer* ini dilakukan dalam menyelesaikan permasalahan *Convex Hull*. Berikut adalah beberapa tahapan yang dilalui untuk menemukan solusi dari permasalahan ini, yaitu :

1. Dari sekumpulan titik yang menjadi input, tentukan dua titik dengan nilai absis paling kecil dan paling maksimum. Jika nilai absis minimum atau maksimumnya sama, maka lihat dari nilai ordinatnya. Titik dengan absis minimum akan disebut sebagai P_1 dan titik dengan absis maksimum akan disebut sebagai P_n .
2. Hubungkan titik P_1 dan P_n menjadi sebuah garis. Lalu, pisahkan titik berdasarkan garis yang dihubungkan antara P_1 dan P_n tersebut.



Gambar 1.3 Ilustrasi ConvexHull (1)

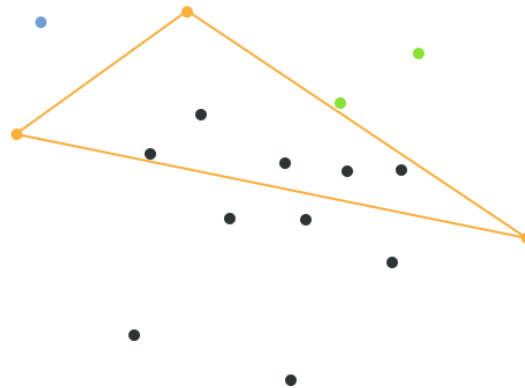
3. Seperti yang terlihat pada Gambar 1.3, pisahkan berdasarkan arah titik ke garis, jika titik berada di bagian atas atau kiri garis maka himpunan titik tersebut dimasukkan ke dalam S_1 , dan jika

titik berada di bagian bawah atau kanan garis maka himpunan titik tersebut dimasukkan ke dalam S1. Langkah untuk menentukan posisi titik terhadap garis bisa dilakukan dari nilai determinan tiga titik tersebut.

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

Dimana (x1, y1) adalah titik P1, (x2, y2) adalah titik Pn, dan (x3, y3) adalah titik yang ingin diuji terhadap garis yang dihubungkan oleh P1 dengan Pn.

4. Untuk setiap S1 dan S2 akan dilakukan:
 - a. Jika himpunan dari S1 atau S2 kosong, maka P1 dan Pn merupakan salah satu sisi dari Convex Hull (basis rekursi).
 - b. Tetapi, jika himpunan tidak kosong maka, cari titik dengan jarak terjauh dari P1 dan Pn tersebut. Untuk beberapa titik dengan jarak yang sama dan sama-sama merupakan yang terjauh, maka ditentukan dengan sudut terbesar yang terbentuk dari P1, Pn dan titik terjauh tersebut. Titik terjauh ini disebut sebagai pMax.



Gambar 1.4 Ilustrasi ConvexHull (2)

- c. Dari Gambar 1.4, pisahkan titik yang berada di luar segitiga yang dibentuk dari P1, Pn, dan pMax. Titik yang berada di luar bagian kiri dari segitiga menjadi S1 berikutnya dan titik yang berada di luar bagian kanan segitiga menjadi S2 berikutnya. Untuk menentukan arah dari titik tersebut bisa dilakukan pengecekan terhadap P1 dan pMax serta terhadap pMax dan Pn.
 - d. Untuk setiap S1 dan S2 yang terbuat, lakukan kembali langkah 4 dengan P1 menjadi P1 lagi untuk S1 dan pMax menjadi Pn untuk S1. Lalu, pMax menjadi P1 untuk S2 dan Pn menjadi Pn lagi untuk S2.
5. Hasil dari setiap rekursi tersebut akan terus dilakukan hingga himpunan S1 dan S2 nya kosong, lalu titik-titik yang menjadi Convex Hull akan dikembalikan dari rekursi tersebut.

BAB II

IMPLEMENTASI PROGRAM

1. main.py

File main.py berfungsi menampilkan Command Line Interface. Input yang akan diterima dari program adalah jenis data yang akan digunakan, antara menggunakan file eksternal atau *datasets* yang disediakan dari program. Lalu, input berikutnya untuk memasukkan *datasets* yang akan dipakai.

```
1 from convexHull import *
2 from utils import *
3 from random import randint
4
5 colorList = ['b', 'r', 'g', 'c', 'm', 'y', 'k']
6
7 if __name__ == "__main__":
8     # ask for whether using external file or load from provided datasets
9     print("1. external file")
10    print("2. provided datasets")
11    dataType = input("Choose data type (1-2): ")
12    dataType = checkInput(dataType, [1, 2])
13
14    if (dataType == 1):
15        # datasets from external file
16        df, X, Y = inputDataSets()
17        # NaN might raise an error in function so it's better to clear dataset
18        df.dropna(inplace=True)
19        # some datasets might not use integer as their target
20        # so it's better to use the original name from dataset
21        df['Target'] = df.iloc[:, -1]
22    else:
23        data, X, Y = providedDataSets()
24        df = pd.DataFrame(data.data, columns=data.feature_names)
25        try:
26            # data with no target might raise an error
27            # will be handled using generalized convex
28            df['Target'] = pd.DataFrame(data.target)
29        except:
30            pass
31
32        try:
33            # only works if datasets is categorized (have a target)
34            linearSepDataSet(df, X, Y)
35        except:
36            # generalized convex, only display one convex
37            # color will be randomized
38            plt.clf()
39            color = randint(0, len(colorList)-1)
40            color = colorList[color]
41            print(color)
42
43            print("\nCant use categorized convex")
44            print("Will take all point to create convex hull")
45            showConvexFromTable(df, X, Y, color)
46
```

Gambar 2.1 main.py

2. **utils.py**

File `utils.py` berfungsi sebagai fungsi-fungsi pembantu yang akan digunakan di dalam `main.py`. Fungsi-fungsi yang terdapat di dalam *file* ini:

- `showConvexFromTable`
Fungsi ini berfungsi untuk melakukan *convex hull* terhadap *datasets* yang tidak terkategori (tidak memiliki *target*).
- `linearSepDataSet`
Fungsi ini berfungsi untuk melakukan *convex hull* untuk *datasets* yang terkategori sehingga bisa dilakukan visualisasi tes *linear separability datasets*.
- `loadDatasets`
Fungsi ini berfungsi untuk melakukan *load* terhadap beberapa *datasets* dari *sklearn* untuk dapat dipilih oleh *user*.
- `checkInput`
Fungsi ini berfungsi untuk melakukan pengecekan input. Pengecekan ini dilakukan agar input *user* selalu valid.
- `providedDataSets`
Fungsi ini berfungsi untuk menerima input dari *user* jika *user* memilih untuk tidak menggunakan file eksternal.
- `inputDataSets`
Fungsi ini berfungsi untuk menerima input dari *user* jika *user* memilih untuk menggunakan file eksternal sebagai *datasets*.

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 from sklearn import datasets
4 from convexHull import *
5 from pathlib import Path
6
7 colorList = ['tab:blue', 'tab:orange', 'tab:green', 'tab:red', 'b', 'r', 'g', 'c', 'm', 'y', 'k']
8
9 def showConvexFromTable(df, column1, column2, colors):
10     """
11     generalized convex from any dataset
12     """
13     # create plot
14     plt.figure(num="Convex Hull", figsize=(10, 6))
15     plt.title(df.columns[column1] +
16             " vs " + df.columns[column2])
17     plt.xlabel(df.columns[column1])
18     plt.ylabel(df.columns[column2])
19
20     # take the value from specified column
21     bucket = df.iloc[:, [column1, column2]].values
22     # use convexHull from the bucket
23     hull = convexHull(bucket)
24     # plot all the point to the graph
25     plt.scatter(bucket[:, 0], bucket[:, 1], color=colors)
26
27     # plot convex point to the graph
28     for i in range(len(hull)):
29         plt.plot(hull[i][0], hull[i][1], colors)
30
31     plt.show()
32
33 def linearSepDataSet(df, column1, column2):
34     """
35     use case for linear separability dataset
36     """
37     # create plot
38     plt.figure(num="Convex Hull", figsize=(10, 6))
39     colors = colorList
40     plt.title(df.columns[column1] +
41             " vs " + df.columns[column2])
42     plt.xlabel(df.columns[column1])
43     plt.ylabel(df.columns[column2])
44     target = df.Target.unique()
45
46     for i in range(len(target)):
47         bucket = df[df['Target'] == target[i]]
48         bucket = bucket.iloc[:, [column1, column2]].values
49         # use convex at each target
50         myHull = convexHull(bucket)
51
52         # scatter and plot colors
53         plt.scatter(bucket[:, 0], bucket[:, 1],
54                 label=target[i], color=colors[i])
55
56         # plot convex point
57         for j in range(len(myHull)):
58             plt.plot(myHull[j][0], myHull[j][1], colors[i])
59
60     plt.legend()
61     plt.show()
62
63 def loadDatasets():
64     """
65     load datasets from sklearn
66     """
67     iris = datasets.load_iris()
68     wine = datasets.load_wine()
69     breastCancer = datasets.load_breast_cancer()
70     linerrud = datasets.load_linnerud()
71     diabetes = datasets.load_diabetes()
72     datasetsArray = [iris, wine, breastCancer, linerrud, diabetes]
73     dsName = ["iris", "wine", "breastCancer", "linerrud", "diabetes"]
74
75     return dsName, datasetsArray
```

```
1 def checkInput(var, arr):
2     """
3     check if input is a number and is in range of array's length
4     """
5     # check whether the input is a number
6     while (not var.isnumeric()):
7         var = input(f"Wrong input, choose a number 1-{len(arr)}: ")
8
9     var = int(var)
10
11     # check if the input is correct
12     while (var < 1 or var > len(arr)):
13         var = int(input(f"Wrong input, number in range 1-{len(arr)}: "))
14
15     return var
16
17 def providedDataSets():
18     """
19     ask user for input from the provided data sets
20     """
21     # used datasets
22     dsName, dsArray = loadDatasets()
23
24     print("List of database: ")
25     for i in range(len(dsName)):
26         print(f"{i+1}. {dsName[i]}")
27
28     # ask user for database
29     dsUsed = input(f"Choose a database (1-{len(dsArray)}): ")
30     # validate input
31     dsUsed = checkInput(dsUsed, dsArray)
32
33     # ask user for X and Y
34     column = dsArray[dsUsed-1].feature_names
35     print("List of column: ")
36     for i in range(len(column)):
37         print(f"{i+1}. {column[i]}")
38
39     X = input(f"Choose a column for X value (1-{len(column)}): ")
40     X = checkInput(X, column)
41     X -= 1
42     Y = input(f"Choose a column for Y value (1-{len(column)}): ")
43     Y = checkInput(Y, column)
44     Y -= 1
45
46     return dsArray[dsUsed-1], X, Y
47
48 def inputDataSets():
49     """
50     ask user for input from external file
51     """
52     # get file path and redirect it to test folder
53     filePath = input("Input file name: ")
54     filePath = str(Path(__file__).resolve().parent) + "/../test/" + filePath
55
56     # check for file
57     try:
58         df = pd.read_csv(filePath)
59     except:
60         print("File not found")
61         inputDataSets()
62
63     # ask user for X and Y
64     column = df.columns
65     print("List of column: ")
66     for i in range(len(column)-1):
67         print(f"{i+1}. {column[i]}")
68
69     X = input(f"Choose a column for X value (1-{len(column)}): ")
70     X = checkInput(X, column)
71     X -= 1
72     Y = input(f"Choose a column for Y value (1-{len(column)}): ")
73     Y = checkInput(Y, column)
74     Y -= 1
75
76     return df, X, Y
```

Gambar 2.2 utils.py

3. **convexHull.py**

File `convexHull.py` merupakan *library* untuk membentuk Convex Hull dari *datasets* yang digunakan. Implementasi *library* ini menggunakan algoritma *Divide and Conquer*. Beberapa fungsi yang digunakan dalam *file* ini:

- **findAngle**
Fungsi ini memiliki tiga parameter yaitu *a*, *b*, dan *c*. Tiap parameter tersebut merupakan *numpy array* yang berperan sebagai titik *x* dan *y*. Fungsi ini berfungsi untuk menentukan sudut dari tiga titik *a*, *b*, dan *c* yang dimasukkan sebagai parameter. Fungsi ini digunakan untuk menentukan titik terjauh pada `splittedConvex` jika terdapat beberapa titik dengan jarak yang sama.
- **determinantBetweenPoint**
Fungsi ini memiliki tiga parameter yaitu *p1*, *p2*, dan *p3*. Yang ketiganya juga merupakan *numpy array* yang juga berperan sebagai titik *x* dan *y*. Fungsi ini berfungsi untuk memberikan nilai determinan yang menjadi penentuan arah dari titik *p3* terhadap garis yang dibentuk oleh *p1* dan *p2*. Jika nilai determinan positif maka *p3* berada di kiri (atas) dari garis *p1* dan *p2*, untuk nilai determinan negative maka *p3* berada di kanan (bawah) dari garis *p1* dan *p2*. Jika nilai determinan nol, maka titik tersebut tepat berada di garis yang dibentuk oleh *p1* dan *p2*. Fungsi ini digunakan di fungsi `splittedConvex` dan juga `convexHull` untuk memisahkan himpunan *S1* dan *S2*.
- **pointDistanceMax**
Fungsi ini memiliki tiga parameter yaitu *S*, *P1*, dan *Pn*. *S* merupakan himpunan titik (*array of numpy array*) dan *P1* dan *Pn* yang merupakan titik (*numpy array*). Fungsi ini digunakan untuk menentukan titik terjauh dari himpunan *S* terhadap garis yang dibentuk oleh titik *P1* dan *Pn*.
- **splittedConvex**
Fungsi ini memiliki empat parameter yaitu *S*, *P1*, *Pn*, dan *pivot*. *S* merupakan himpunan titik (*array of numpy array*) dan *P1* dan *Pn* yang merupakan titik (*numpy array*), dan *pivot* merupakan integer (1 dan -1) untuk menentukan arah dari convex hull. Fungsi ini berfungsi sebagai fungsi rekursi pada langkah 4 dari Bab 2. Fungsi ini mengimplementasikan algoritma *Divide and Conquer*.
- **convexHull**
Fungsi ini memiliki satu parameter yaitu *listOfPoint* yang merupakan himpunan titik (*array of numpy array*). Fungsi ini berfungsi sebagai implementasi dari langkah awal dari Bab 2. Fungsi ini akan mengembalikan *array of numpy array* yang merupakan sisi dari Convex Hull.

```
1 import numpy as np
2
3 def findAngle(a, b, c):
4     """
5     find the angle between three point (return in degrees)
6     """
7     ba = a - b
8     bc = c - b
9
10    cosine_angle = np.dot(ba, bc) / (np.linalg.norm(ba) * np.linalg.norm(bc))
11    angle = np.arccos(cosine_angle)
12
13    return np.degrees(angle)
14
15 def determinantBetweenPoint(p1, p2, p3):
16     """
17     find the determinant between three point
18     if > 0 then it is left/upper from p1 and p2
19     if = 0 then it is in p1 and p2
20     if < 0 then it is right/below from p1 and p2
21     """
22     x1 = p1[0]
23     y1 = p1[1]
24     x2 = p2[0]
25     y2 = p2[1]
26     x3 = p3[0]
27     y3 = p3[1]
28     return x1*y2 + x3*y1 + x2*y3 - x3*y2 - x2*y1 - x1*y3
29
30 def pointDistanceMax(S, P1, Pn):
31     """
32     find the furthest point from a line (P1, pn) to a point from array S
33     if the distance is the same then maximize the angle
34     """
35     maxD = 0
36     pointMaxD = []
37     for i in S:
38         d = np.abs(np.cross(Pn-P1, i-P1)/np.linalg.norm(Pn-P1))
39         if (d > maxD):
40             maxD = d
41             pointMaxD = i
42
43     elif (d == maxD):
44         # maximize the angle if distance is the same
45         dAngle = findAngle(P1, i, Pn)
46         maxDAngle = findAngle(P1, pointMaxD, Pn)
47
48         if (dAngle > maxDAngle):
49             maxD = d
50             pointMaxD = i
51
52     return pointMaxD
```

```
1 def splittedConvex(S, P1, Pn, pivot):
2     """
3     divide and conquer algorithm using quickHull
4     """
5     if (len(S) == 0):
6         # there's no point in S, then P1 and Pn is convexPoint
7         convexPoint.append(
8             [np.array([P1[0], Pn[0]]), np.array([P1[1], Pn[1]])])
9
10    # edge case for only one point in S
11    elif (len(S) == 1):
12        pMax = S[0]
13        convexPoint.append(
14            [np.array([P1[0], pMax[0]]), np.array([P1[1], pMax[1]])])
15        convexPoint.append(
16            [np.array([P1[0], Pn[0]]), np.array([P1[1], Pn[1]])])
17
18    else:
19        # find the max distances between point in S to the Line between P1 and Pn
20        pointMaxD = pointDistanceMax(S, P1, Pn)
21
22        # define S1 and S2
23        S1 = []
24        S2 = []
25        # split to S1 and S2
26        for i in S:
27            # if pivot is -1 then it is upside down of the original function
28            # multiply with -1 so that the function will work properly
29            if (pivot == -1):
30                dir = -1
31            else:
32                dir = 1
33
34            # check if point outside of the left triangle of P1, pointMaxD, and Pn
35            if (pointMaxD[0] > i[0]):
36                dir == determinantBetweenPoint(pointMaxD, P1, i)
37
38            if (dir < 0):
39                S1.append(i)
40
41            # check if point outside of the right triangle of P1, pointMaxD, and Pn
42            elif (pointMaxD[0] < i[0]):
43                dir == determinantBetweenPoint(Pn, pointMaxD, i)
44
45            if (dir < 0):
46                S2.append(i)
47
48        splittedConvex(S1, P1, pointMaxD, pivot)
49        splittedConvex(S2, pointMaxD, Pn, pivot)
50
51 def convexHull(listOfPoint):
52     """
53     main function from convex hull
54     """
55     # initialize convexPoint
56     global convexPoint
57     convexPoint = []
58
59     # sort array by the absis
60     listOfPoint = listOfPoint[listOfPoint[:, 1].argsort(kind='mergesort')]
61     listOfPoint = listOfPoint[listOfPoint[:, 0].argsort(kind='mergesort')]
62     # take the minimum of absis as P1
63     P1 = listOfPoint[0]
64     # take the maximum of absis as Pn
65     Pn = listOfPoint[-1]
66
67     # define S1 and S2
68     S1 = []
69     S2 = []
70
71     # split point by the line of P1 and Pn
72     for i in listOfPoint[1:-1]:
73         dir = determinantBetweenPoint(Pn, P1, i)
74         if (dir < 0):
75             S1.append(i)
76         elif (dir > 0):
77             S2.append(i)
78
79     # divide and conquer for S1 and S2
80     splittedConvex(S1, P1, Pn, 1)
81     splittedConvex(S2, P1, Pn, -1)
82     return convexPoint
```

Gambar 2.3 convexHull.py

BAB III

HASIL PERCOBAAN

1. Input/Output

```
1. external file
2. provided datasets
Choose data type (1-2): 2
List of database:
1. iris
2. wine
3. breastCancer
4. linerrud
5. diabetes
Choose a database (1-5): 1
List of column:
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)
Choose a column for X value (1-4): 1
Choose a column for Y value (1-4): 2
```

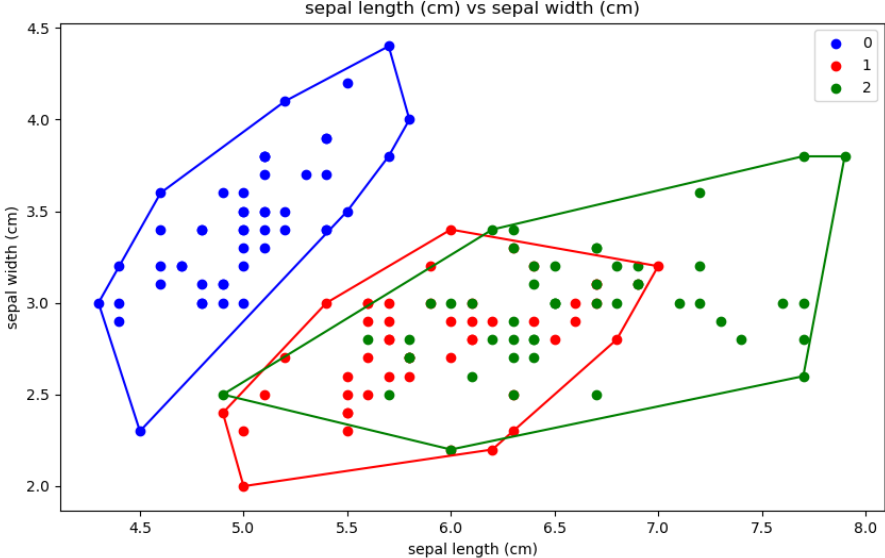
Gambar 3.1 Commnad Line Interface Provided Datasests

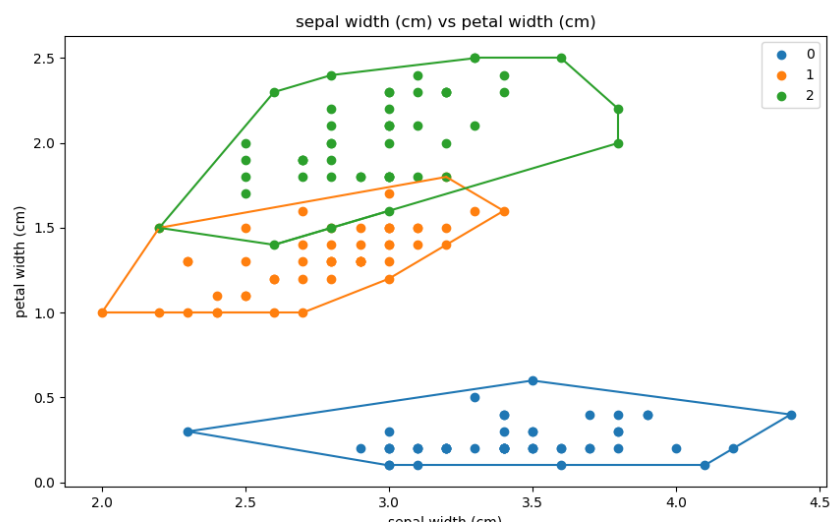
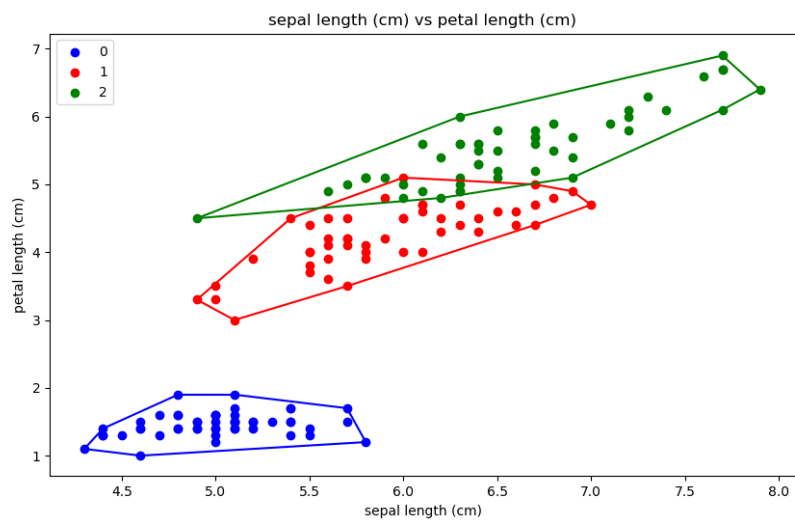
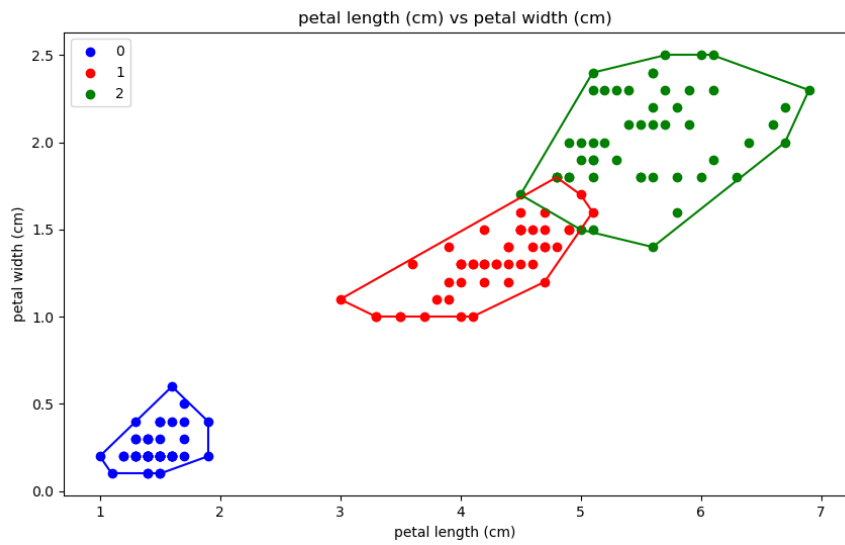
```
1. external file
2. provided datasets
Choose data type (1-2): 1
Input file name: heart.csv
List of column:
1. age
2. sex
3. cp
4. trestbps
5. chol
6. fbs
7. restecg
8. thalach
9. exang
10. oldpeak
11. slope
12. ca
13. thal
Choose a column for X value (1-14): 4
Choose a column for Y value (1-14): 5
```

Gambar 3.1 Commnad Line Interface File Eksternal

2. Percobaan Dataset Sklearn

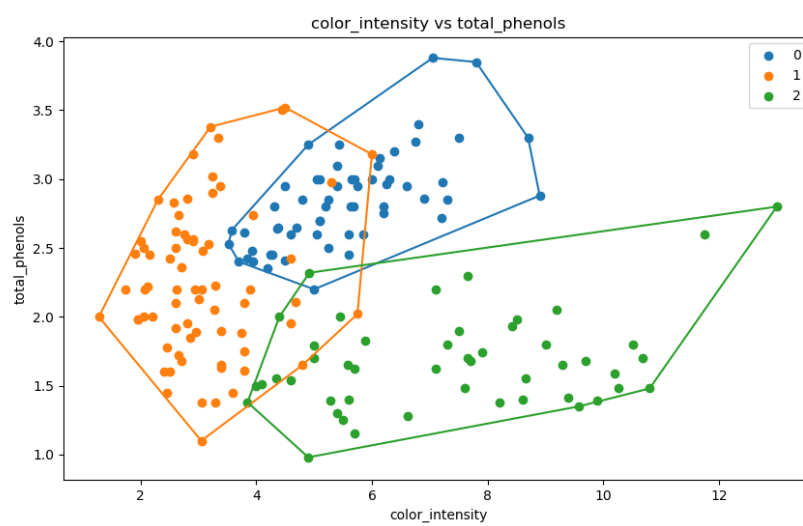
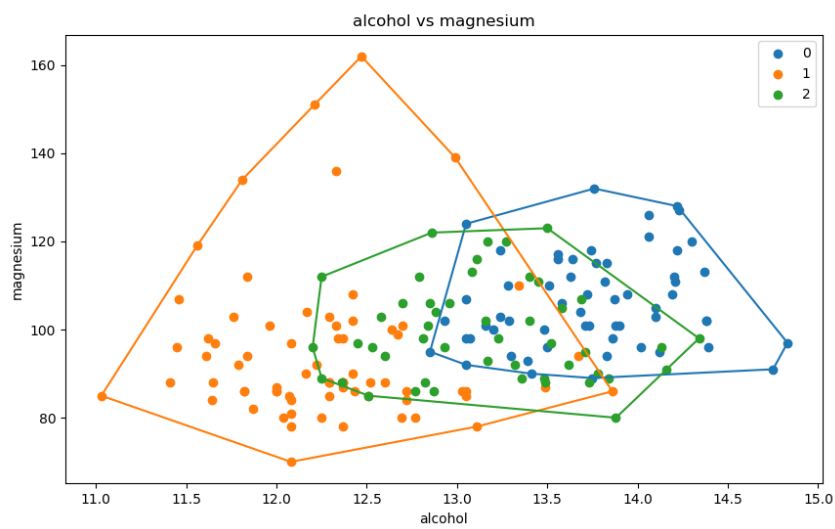
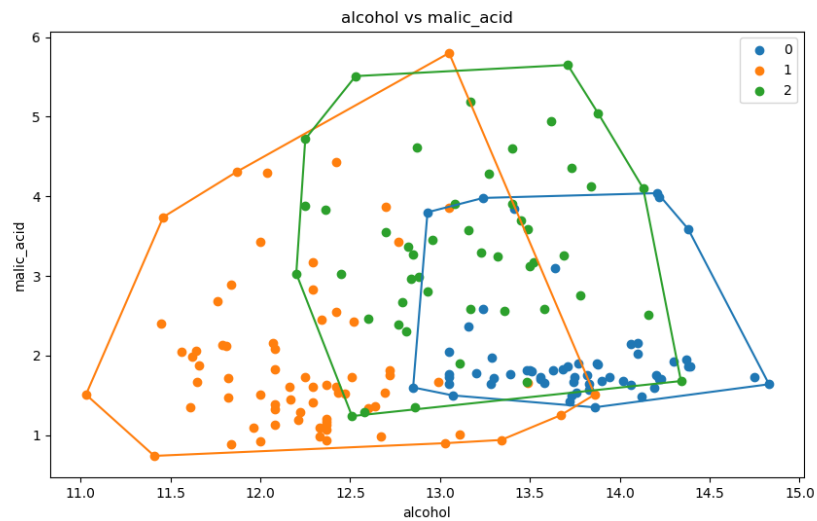
Percobaan 1

<p>Nama Dataset : Iris</p>	<p>Pasangan Atribut : sepal-length, sepal-width petal-length, petal-width sepal-length, petal-length sepal-width, petal-width</p>
Input	
<pre> sepal length (cm) sepal width (cm) petal length (cm) petal width (cm) \ 0 5.1 3.5 1.4 0.2 1 4.9 3.0 1.4 0.2 2 4.7 3.2 1.3 0.2 3 4.6 3.1 1.5 0.2 4 5.0 3.6 1.4 0.2 145 6.7 3.0 5.2 2.3 146 6.3 2.5 5.0 1.9 147 6.5 3.0 5.2 2.0 148 6.2 3.4 5.4 2.3 149 5.9 3.0 5.1 1.8 Target 0 0 1 0 2 0 3 0 4 0 145 2 146 2 147 2 148 2 149 2 [150 rows x 5 columns]</pre>	
Output	
	



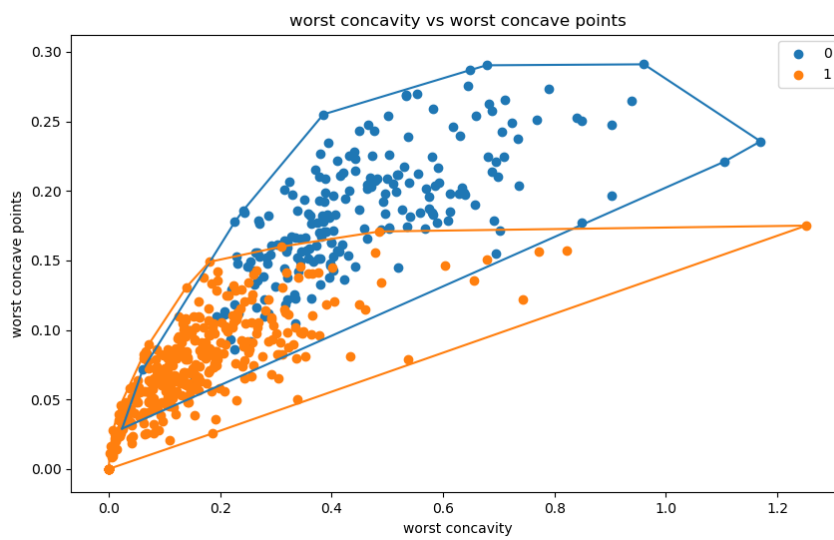
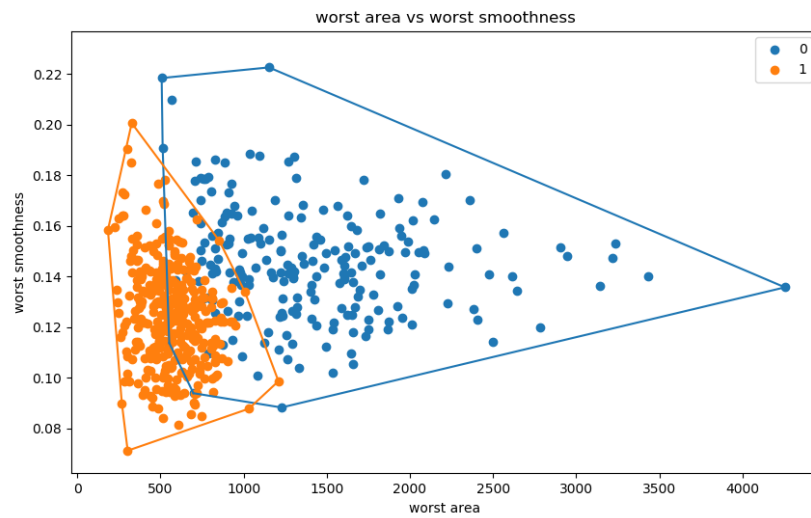
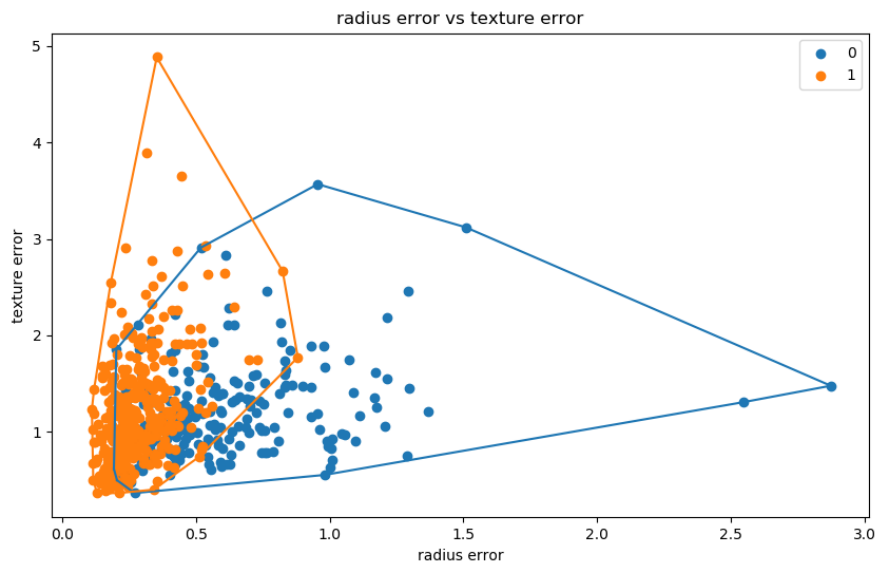
Percobaan 2

The scatter plot, titled "ash vs alcalinity_of_ash", displays the relationship between "ash" (x-axis) and "alcalinity_of_ash" (y-axis) for three categories: 0 (blue), 1 (orange), and 2 (green). The x-axis ranges from approximately 1.25 to 3.25, and the y-axis ranges from 10.0 to 30.0. Each category is represented by a set of data points and a convex hull. Category 1 (orange) has the highest values for both variables, followed by Category 2 (green), and then Category 0 (blue). There is a significant overlap between the data points of Categories 0 and 2.



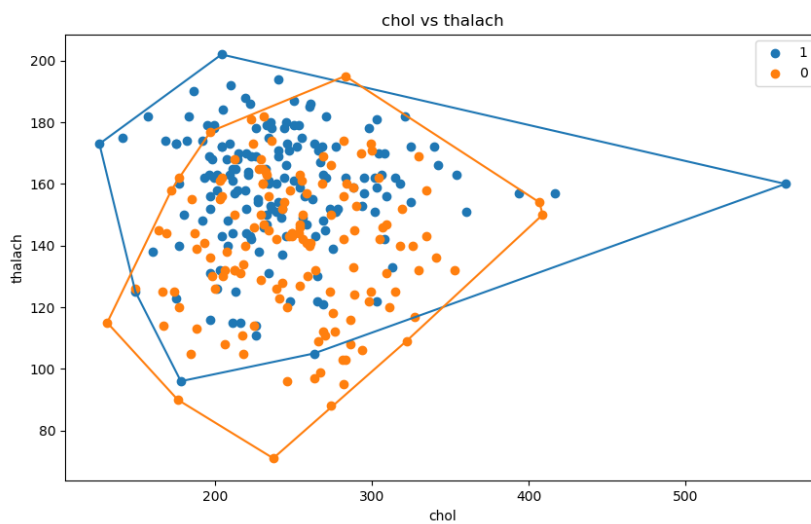
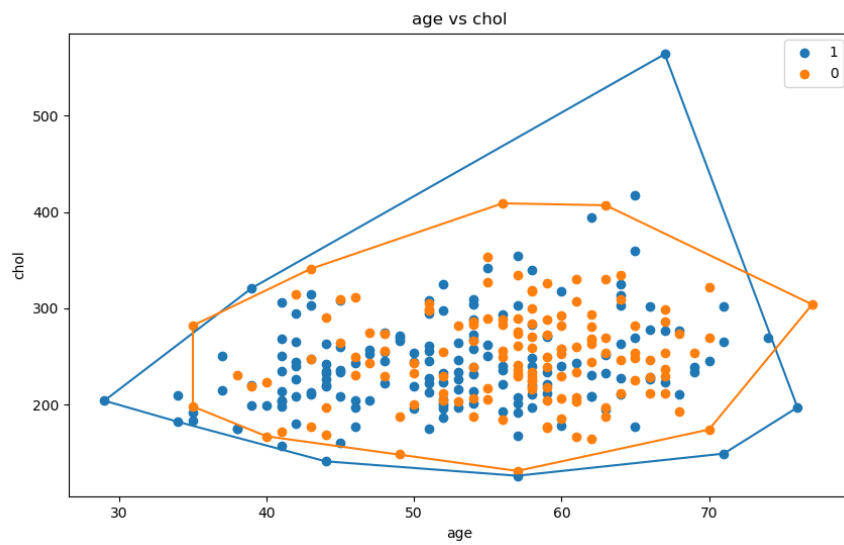
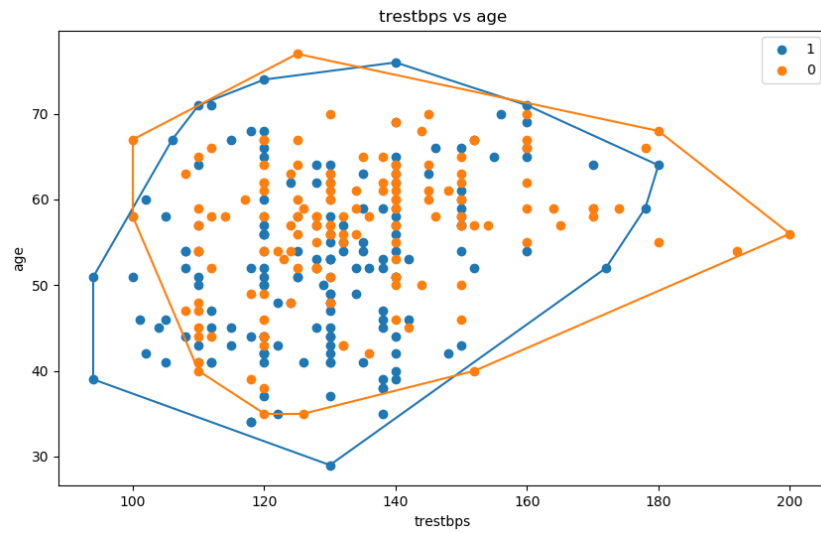
Percobaan 3

<i>Nama Dataset :</i> Breast Cancer	<i>Pasangan Atribut :</i> mean concave points, mean compactness texture error, radius error worst smoothness, worst area worst concavity, worst concave points				
Input					
	mean radius	mean texture	mean perimeter	mean area	mean smoothness \
0	17.99	10.38	122.80	1001.0	0.11840
1	20.57	17.77	132.90	1326.0	0.08474
2	19.69	21.25	130.00	1203.0	0.10960
3	11.42	20.38	77.58	386.1	0.14250
4	20.29	14.34	135.10	1297.0	0.10030
..
564	21.56	22.39	142.00	1479.0	0.11100
565	20.13	28.25	131.20	1261.0	0.09780
566	16.60	28.08	108.30	858.1	0.08455
567	20.60	29.33	140.10	1265.0	0.11780
568	7.76	24.54	47.92	181.0	0.05263
	mean compactness	mean concavity	mean concave points	mean symmetry \	
0	0.27760	0.30010	0.14710	0.2419	
1	0.07864	0.08690	0.07017	0.1812	
2	0.15990	0.19740	0.12790	0.2069	
3	0.28390	0.24140	0.10520	0.2597	
4	0.13280	0.19800	0.10430	0.1809	
..	
564	0.11590	0.24390	0.13890	0.1726	
565	0.10340	0.14400	0.09791	0.1752	
566	0.10230	0.09251	0.05302	0.1590	
567	0.27700	0.35140	0.15200	0.2397	
568	0.04362	0.00000	0.00000	0.1587	
Output					
<p>mean concave points vs mean compactness</p>					

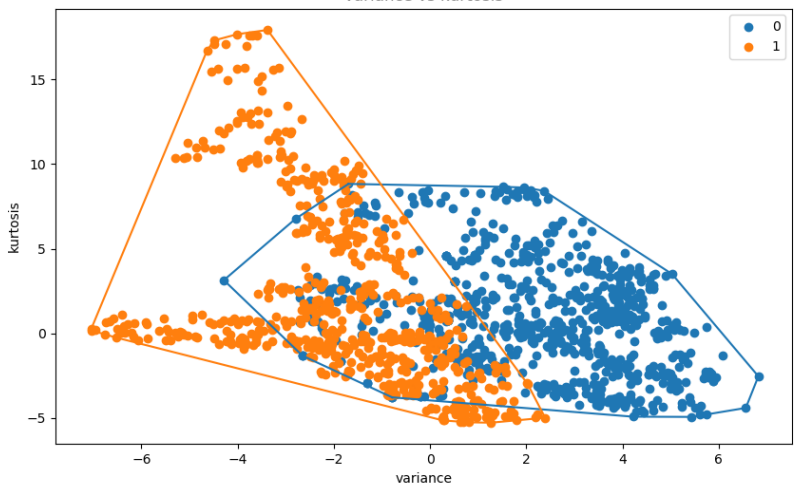


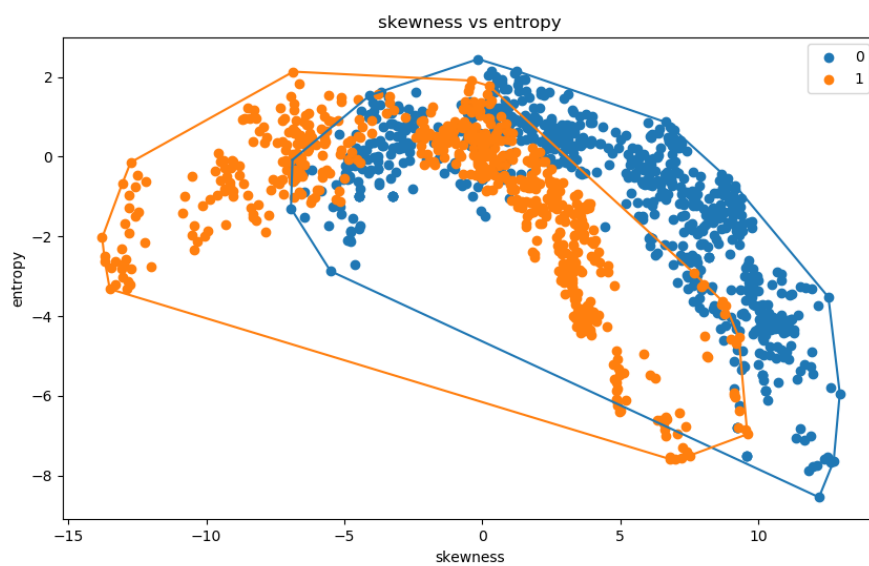
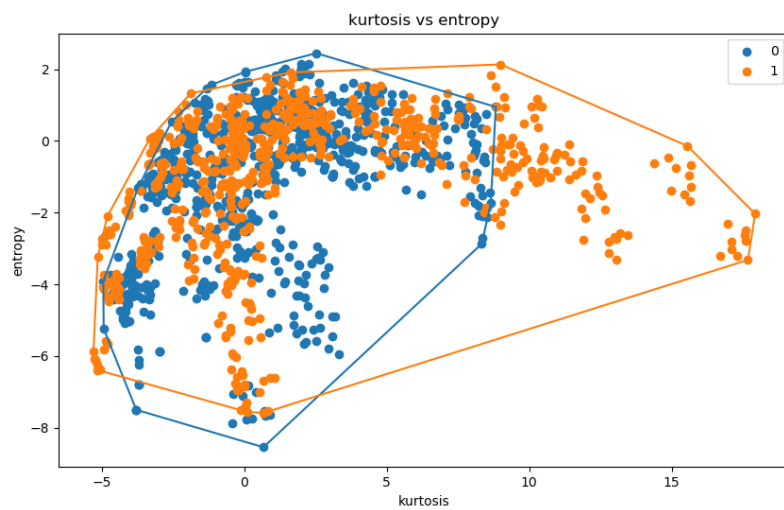
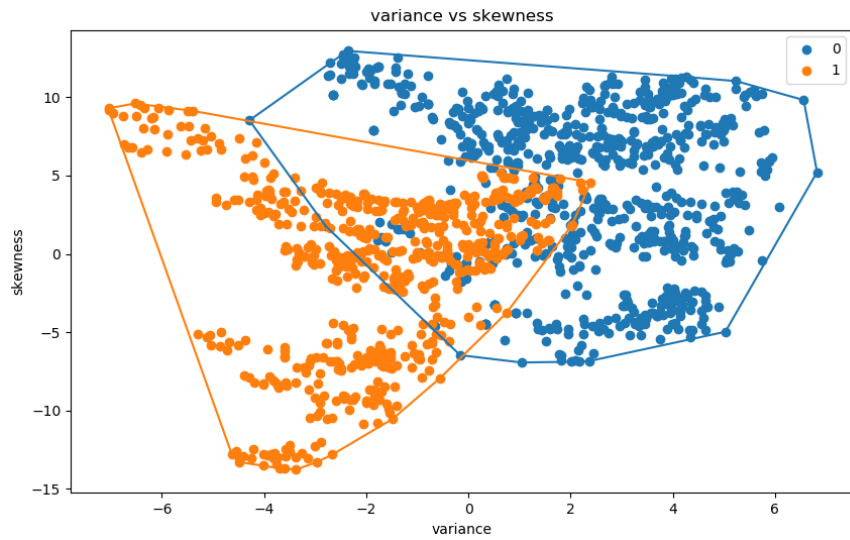
3. Percobaan Dataset File Eksternal

Percobaan 1



Percobaan 2

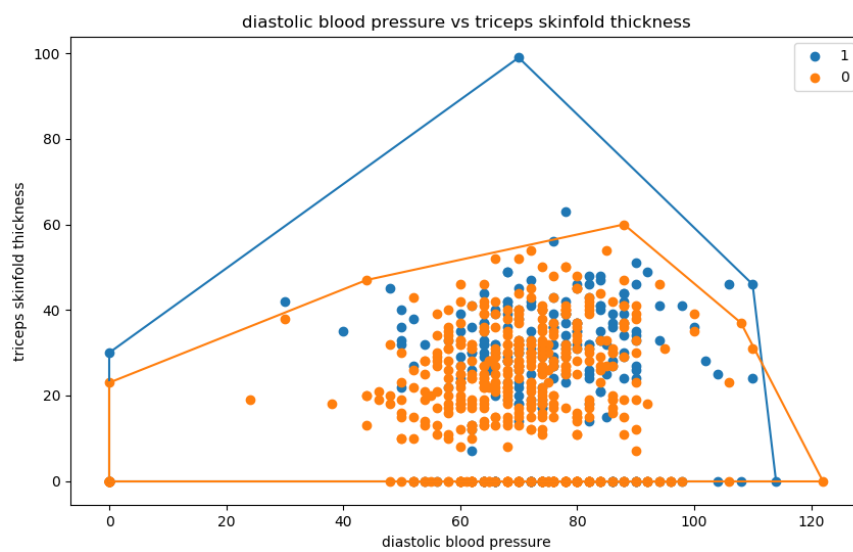
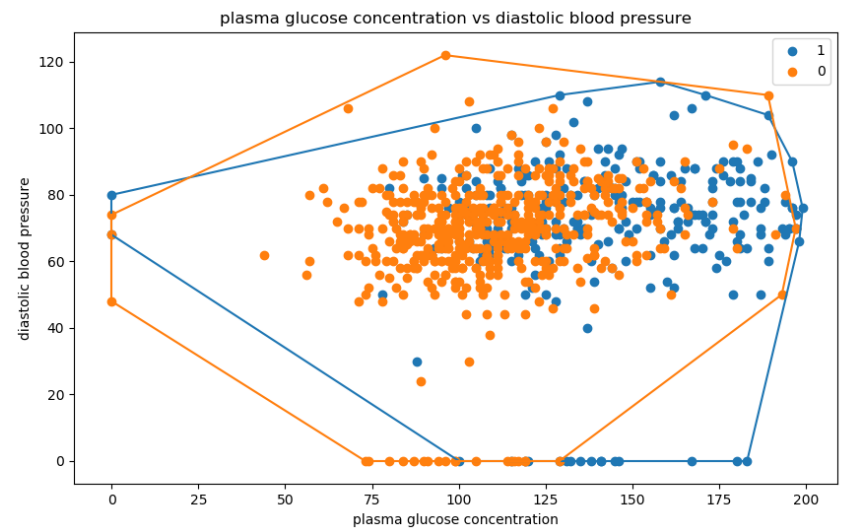
<p>Nama Dataset : banknote.csv</p>	<p>Pasangan Atribut : kurtosis, variance variance, skewness entropy, kurtosis skewness, entropy</p>
Input	
<pre> variance skewness kurtosis entropy target 0 3.62160 8.66610 -2.8073 -0.44699 0 1 4.54590 8.16740 -2.4586 -1.46210 0 2 3.86600 -2.63830 1.9242 0.10645 0 3 3.45660 9.52280 -4.0112 -3.59440 0 4 0.32924 -4.45520 4.5718 -0.98880 0 1367 0.40614 1.34920 -1.4501 -0.55949 1 1368 -1.38870 -4.87730 6.4774 0.34179 1 1369 -3.75030 -13.45860 17.5932 -2.77710 1 1370 -3.56370 -8.38270 12.3930 -1.28230 1 1371 -2.54190 -0.65804 2.6842 1.19520 1 [1372 rows x 5 columns] </pre>	
Output	
	



Percobaan 3

<i>Nama Dataset :</i> indiansdiabetes.csv	<i>Pasangan Atribut :</i> plasma glucose concentration, diastolic blood pressure diastolic blood pressure, triceps skinfold thickness
<i>Input</i>	
<pre>number of times pregnant plasma glucose concentration \ 0 6 148 1 1 85 2 8 183 3 1 89 4 0 137 763 10 101 764 2 122 765 5 121 766 1 126 767 1 93 diastolic blood pressure triceps skinfold thickness \ 0 72 35 1 66 29 2 64 0 3 66 23 4 40 35 763 76 48 764 70 27 765 72 23 766 60 0 767 70 31 2-Hour serum insulin body mass index diabetes pedigree function age \ 0 0 33.6 0.627 50 1 0 26.6 0.351 31 2 0 23.3 0.672 32 3 94 28.1 0.167 21 4 168 43.1 2.288 33 763 180 32.9 0.171 63 764 0 36.8 0.340 27 765 112 26.2 0.245 30 766 0 30.1 0.349 47 767 0 30.4 0.315 23 target 0 1 1 0 2 1 3 0 4 1 763 0 764 0 765 0 766 1 767 0</pre>	
[768 rows x 9 columns]	

Output



LAMPIRAN

1. Repository Github :

<https://github.com/apwic/convex-hull-visualizer>

2. Checklist :

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.	✓	

REFERENSI

Informatika.stei.itb.ac.id/~rinaldi.munir. (2022). Algoritma Divide and Conquer Bagian 1. Diakses pada 26 Februari 2022, dari

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-\(2021\)-Bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf)

Informatika.stei.itb.ac.id/~rinaldi.munir. (2022). Algoritma Divide and Conquer Bagian 4. Diakses pada 26 Februari 2022, dari

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-\(2022\)-Bagian4.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf)