LAPORAN
TUGAS KECIL 2

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

IF2211 STRATEGI ALGORITMA

Oleh:

Ng Kyle / 13520040

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

2021

Bagian I
Deskripsi Algoritma dan Kompleksitas Algoritma

1. **Algoritma *Divide and Conquer* dalam Convex Hull (QuickHull)**
   Algoritma *Divide and Conquer* sebagaimana menjadi dasar algoritma Convex Hull (QuickHull) merupakan algoritma yang memecah persoalan menjadi sub-sub persoalan yang lebih kecil (divide) lalu menyelesaikan setiap sub persoalan secara rekursif (conquer), dengan hasil penyelesaian setiap sub persoalan digabungkan kembali menjadi penyelesaian lengkap (merge). QuickHull merupakan algoritma Convex Hull yang menggunakan teknik *Divide and* Conquer, yang mana pada awalnya, kumpulan titik dibagi menjadi 2 subset berdasarkan letaknya realtif dengan garis yang menghubungkan 2 titik terjauh kiri dan kanan. Lalu kedua subset tersebut dilakukan rekursi pemanggilan penyelesaian QuickHull yang membagi kembali mejadi 3 subset berdasarkan letaknya relatif terhadap garis penghubung titik terjauh subset dengan ujung-ujung subset, hingga subset tersisa satu atau nol titik.

2. **Perincian dan Penjelasan Algoritma**
   Dalam pengaplikasian algoritma *Divide and Conquer* dalam QuickHull, dapat dibagi menjadi 2 tahap utama:
   a. Proses pembagian dasar
      Tahap ini merupakan melakukan pencarian 2 titik dengan masing-masing nilai x terkecil dan x terbesar, misalkan P_Min dan P_Max. Lalu kumpulan titik dibagi menjadi 2 buah subset yaitu subset berisi titik yang berada di atas dari garis penghubung P_Min dan P_Max dan subset berisi titik yang berada di bawah dari garis penghubung P_Min dan P_Max. Kedua subset kemudian dilakukan operasi rekursif QuickHull.
   b. Proses Rekursif QuickHull
      Proses rekursif Quickhull memiliki masukan 2 titik acuan (P_Awal dan P_Akhir) serta sebuah set, lalu dilakukan beberapa tahap:
      1. Basis rekursif : Set berisi 1 titik atau 0 titik, maka kembalikan isi set.
      2. Melakukan pencarian titik terjauh pada set dari P_Awal dan P_Akhir (Menggunakan rumus cross product 2 dimensi), sebut P_Far.
      3. Membagi set menjadi 3 subset :
         i. Subset I yaitu subset berisi kumpulan titik yang berada dalam daerah segitiga yang dibentuk P_Awal, P_Akhir, dan P_Far.
         ii. Subset II yaitu subset berisi kumpulan titik yang berada di “kanan” dari garis berarah (sinar) dari P_Awal ke P_Far
         iii. Subset III yaitu subset berisi kumpulan titik yang berada di “kanan” dari garis berarah (sinar) dari P_Far ke P_Akhir
      4. Subset II dan Subset III dilakukan operasi rekursif QuickHull lalu digabungkan untuk dikembalikan yaitu :
         QuickHull(P_Awal,P_Far,Subset II) + P_Far + QuickHull(P_Far,P_Akhir,Subset III)
   c. Hasil QuickHull
      Algoritma QuickHull ini akan menghasilkan dari tahap a:

P_Min+QuickHull(P_Min,P_Max,Subset_Bawah)+P_Max
+QuickHull(P_Max,P_Min, Subset_Atas)

Keterangan :

- Orientasi dari titik sangat penting, sehingga penentuan P_Min, P_Max serta subset harus konsisten (dalam hal ini selalu membagi subset kanan—kiri secara counter-clockwise.
- Optimasi pembagian subset dengan melakukan hanya 2 kali pembagian subset (bukan 3), dengan melakukan pembagian subset II terlebih dahulu, lalu titik yang berada pada "kiri" subset II dilakukan pembagian kembali menjadi Subset I dan Subset III dengan jumlah titik yang dicari lebih sedikit.
- Dalam visualisasi, perlu ditambahkan kembali Point pertama untuk membuat closed hull, namun set of point hull sudah complete.

3. Elemen Divide and Conquer QuickHull

Divide: Membagi set menjadi 3 subset (tahap b.2 dan b.3)
Conquer (solve): Mengembalikan set jika berisi 1 atau 0 titik (tahap b.1)
Combine: Menggabungkan hasil QuickHull dan titik acuan (tahap b.4)

4. Analisis Kompleksitas Algoritma

Pada tahap Conquer terdapat dua pemanggilan rekursif, dengan asumsi set terbagi rata maka masing-masing subset memiliki $\frac{n}{2}$ titik. Tiap pemanggilan rekursif dilakukan 2 kali pembagian subset dengan perkiraan pencarian dilakukan sebanyak $\frac{3}{2}$n. Maka kompleksitas algoritma :

$$\mathrm{T(n)} = \begin{cases} 1, & \mathrm{n} \leq 1 \\ 2\mathrm{T}\left(\frac{\mathrm{n}}{2}\right) + \frac{3}{2}n, & n > 1 \end{cases}$$

Dengan teorema master $a = 2, b = 2, d = 1$, kompleksitas waktu QuickHull $O(n \log n)$.

Bagian II
Source Program

Program dibuat menjadi 2 bagian, yaitu file myConvexHull berisi implementasi QuickHull serta file main merupakan driver program visualisasi dengan data set dari scikit-learn.

1. myConvexHull
   a. function ConvexHull
      Input: bucket berupa list 2D dimensi Nx2 dengan N banyak data.
      Proses: Melakukan proses a. Proses pembagian dasar.
      Output: List 2D berukuran Mx2 berisi M buah titik pembentuk ConvexHull (terurut Counter ClockWise)

```
def ConvexHull(bucket):
    bucket = bucket.tolist()
    min_x = bucket[0]
    max_x = bucket[0]

    #Find point with minimum x and maximum x => O(N)
    for point in bucket:
        if (point[0] > max_x[0]):
            max_x = point
        if (point[0] < min_x[0]):
            min_x = point

    #Divide into two set (left and right of line) => O(N)
    left_batch,right_batch = divideSet(min_x,max_x,bucket)

    #Conquer (Counter ClockWise to maintain order always use right set of oriented line)
    return [min_x] + QuickHull(min_x,max_x, right_batch) + [max_x] + QuickHull(max_x,min_x,left_batch)
```

   b. function QuickHull
      Input:
      - begin_point: List 2 elemen (point), sebagai titik awal set of points QuickHull
      - begin_point: List 2 elemen (point), sebagai titik akhir set of points QuickHull
      - bucket: list 2D dimensi Nx2 dengan N banyak data / point.

      Proses: Melakukan Divide and Conquer proses rekursif (proses b.1 – b.4)
      Output: List 2D berukuran Mx2 berisi M buah titik pembentuk bagian ConvexHull (terurut Counter ClockWise)

```python
def QuickHull(begin_point, end_point,bucket):
    n_points = len(bucket)
    if n_points == 0:
        return []
    elif n_points == 1:
        return [bucket[0]]
    else:
        furthest_dist = 0
        furthest_point = bucket[0]

        #find furthest point from oriented line => O(N)
        for point in bucket:
            if (abs(distance(begin_point,end_point,point)) > furthest_dist):
                furthest_dist = abs(distance(begin_point,end_point,point))
                furthest_point = point

        #divide into two triangle parts ("upper left" and "upper right")
        #Get "upper right" set
        temp_set, upright_set = divideSet(begin_point,furthest_point,bucket)
        #Get "upper left" set
        temp_set, upleft_set = divideSet(furthest_point,end_point,temp_set)
        return QuickHull(begin_point,furthest_point,upright_set) + [furthest_point] + QuickHull(furthest_point,end_point,upleft_set)
```

c. function divideSet
Input:
- begin_point: List 2 elemen (point), sebagai titik awal set of points QuickHull
- end_point: List 2 elemen (point), sebagai titik akhir set of points QuickHull
- bucket: list 2D dimensi Nx2 dengan N banyak data / point.

Proses: Melakukan Divide set of points (bucket) menjadi subset ‘kanan’ dan ‘kiri’
Output:
- left_batch: subset kiri dari garis berarah (berorientasi) begin_point – end_point
- right_batch: subset kiri dari garis berarah (berorientasi) begin_point – end_point

```python
def divideSet(begin_point,end_point,bucket):
    left_batch = []
    right_batch =[]
    for point in bucket:
        sideVal = distance(begin_point,end_point,point)
        if(sideVal > 0): #Point on the right side of line begin_point--end_point
            right_batch+=[point]
        if(sideVal < 0): #Point on the left side of line min_x--max_x
            left_batch+=[point]
    return left_batch,right_batch
```

d. function distance
Input:
- begin_point: List 2 elemen (point), sebagai titik acuan awal garis
- end_point: List 2 elemen (point), sebagai titik akhir awal garis
- eval_point: List 2 elemen (point), sebagai titik untuk diperiksa

Proses: Mendapatkan jarak eval_point dengan garis berorientasi begin_point – end_point (digunakan juga untuk memeriksa posisi titik eval_point relative dengan garis tersebut).
Output: Mengembalikan jarak berarah eval_point terhadap garis

```
def distance(begin_point, end_point, eval_point):
#if on the right side, value > 0 ; if left side value < 0 ; else on the line => O(1)
    #Using Cross-Product on 2D
    return ((end_point[0] - begin_point[0])*(begin_point[1]-eval_point[1]) - (end_point[1] -
begin_point[1])*(begin_point[0]-eval_point[0]))
```

2. Main
   Driver untuk visualisasi dengan data set dari scikit-learn.
   Dataset yang didukung:
   Test Separability:
   1. iris
   2. wine
   3. breast_cancer

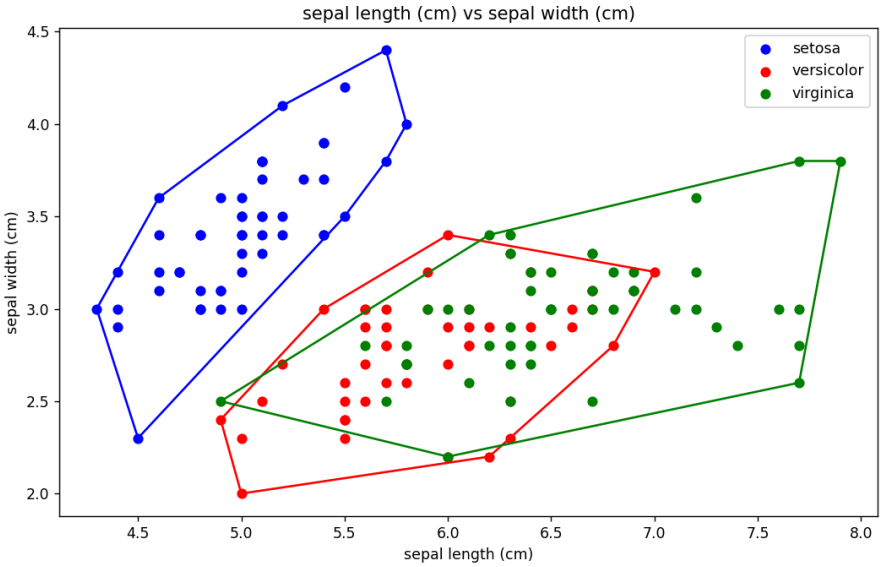
   Test single Convex Hull:
   1. boston
   2. diabetes

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from random import uniform
from sklearn import datasets

print("\nChoose number for your dataset to Test Linear Separability Dataset")
print("(Below chosen datasets from sklearn which has class / separable)")
print(
    '''
    1. iris
    2. wine
    3. breast_cancer
    '''
)
print("(Below chosen datasets from sklearn which is not separable (no classes))")
print(
    '''
    4. boston
    5. diabetes
    '''
)

while(True):
    n = int(input("Chosen data number: "))
    if(n == 1):
        data = datasets.load_iris()
        break
    elif(n == 2):
        data = datasets.load_wine()
        break
    elif(n == 3):
        data = datasets.load_breast_cancer()
        break
    elif(n == 4):
        data = datasets.load_boston()
        break
    elif(n == 5):
        data = datasets.load_diabetes()
        break
    else: print("Invalid Number!")

print("\nHere are the list of features of this data collection:\n")
for i in range(len(data.feature_names)):
    print(str(i+1) + "." +data.feature_names[i])
first_feature = int(input("Choose first feature: "))-1
second_feature = int(input("Choose second feature: "))-1
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)

from myConvexHull import ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g','c','m','y']
plt.title(str(data.feature_names[first_feature])+" vs "+str(data.feature_names[second_feature]))
plt.xlabel(data.feature_names[first_feature])
plt.ylabel(data.feature_names[second_feature])

try :
    for i in range(len(data.target_names)):
        bucket = df[df['Target'] == i]
        bucket = bucket.iloc[:,[first_feature,second_feature]].values
        hull = ConvexHull(bucket)
        plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i],color =colors[i%6])
        hul_len = len(hull)
        plt.plot([row[0] for row in hull]+[hull[0][0]], [row[1] for row in hull]+[hull[0][1]],color =colors[i%6])
    plt.legend()
except:
    bucket = df
    bucket = bucket.iloc[:,[first_feature,second_feature]].values
    hull = ConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1],color =colors[0])
    hul_len = len(hull)
    plt.plot([row[0] for row in hull]+[hull[0][0]], [row[1] for row in hull]+[hull[0][1]],color =colors[0])
plt.show()
```

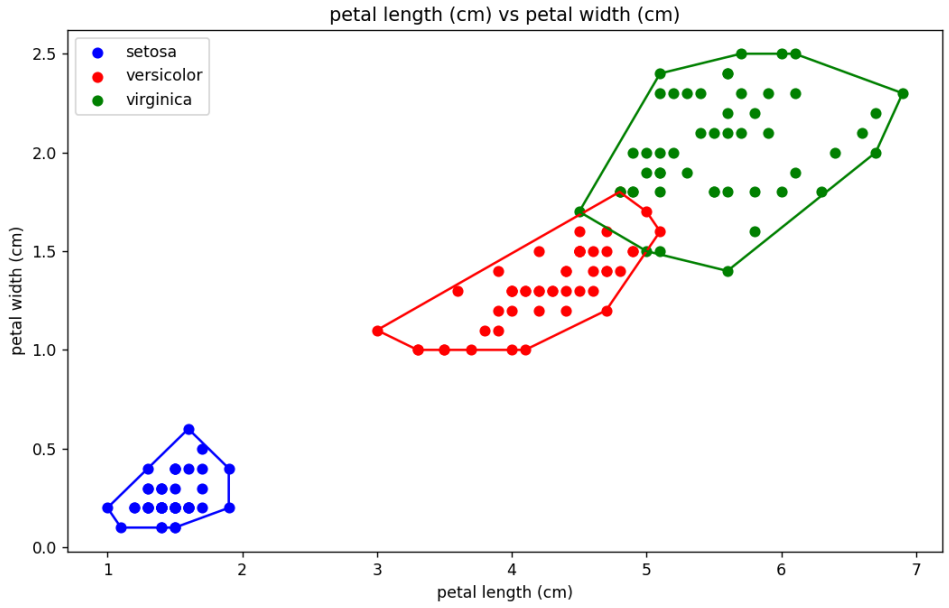
Bagian III
Screenshoot Program Testing

Berikut merupakan kumpulan hasil penjalanan program, sesuai data set:

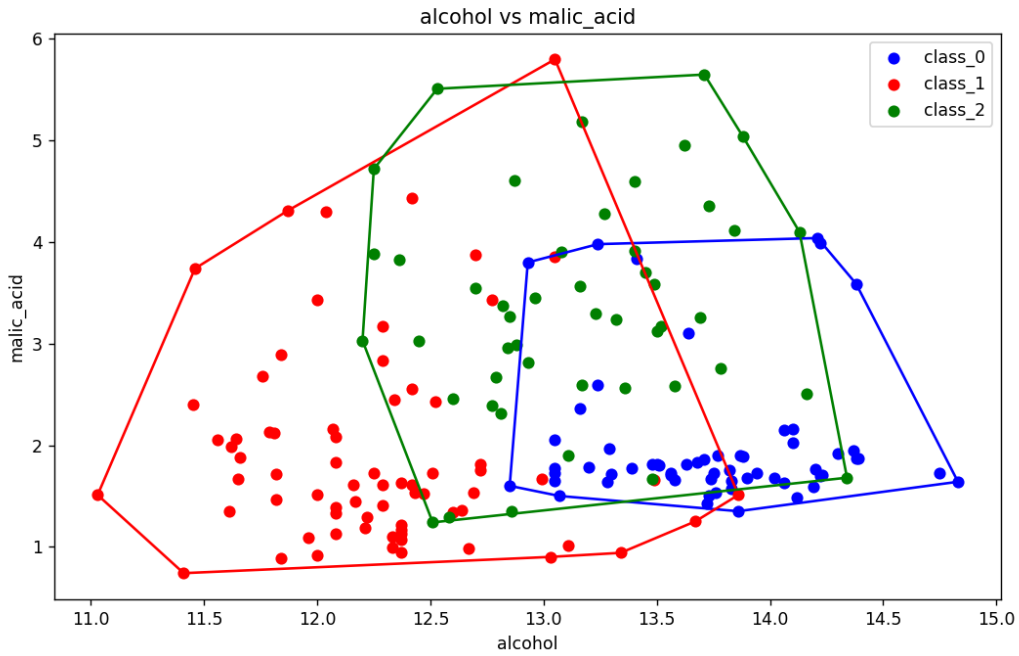
Data Set	Iris
First Feature	sepal length (cm)
Second Feature	sepal width (cm)
Resulting Convex Hull	



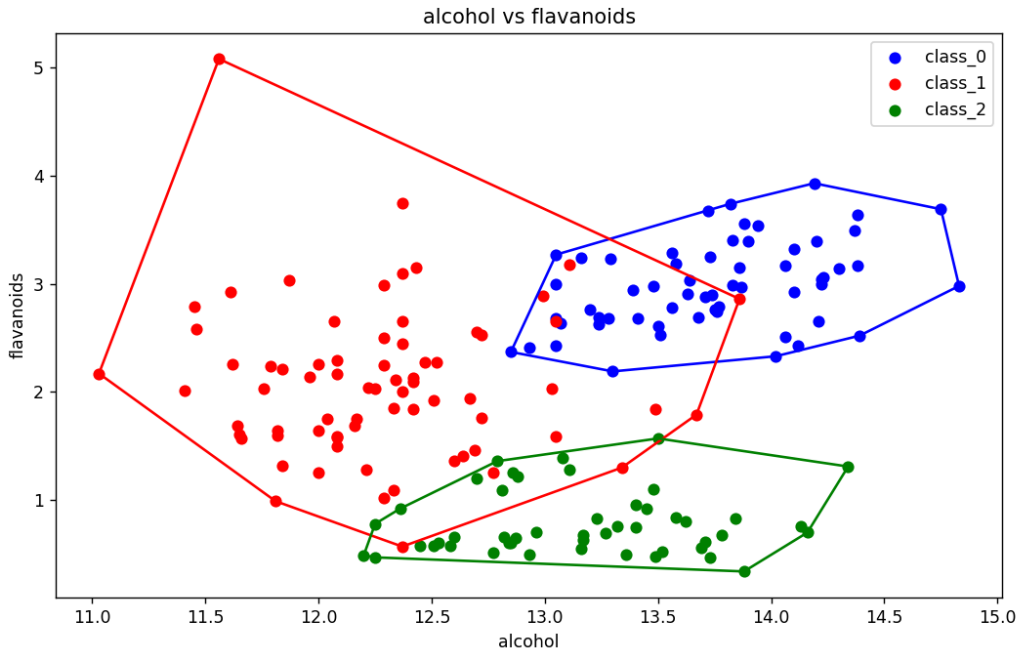
Data Set	Iris
First Feature	petal length (cm)
Second Feature	petal width (cm)
Resulting Convex Hull	



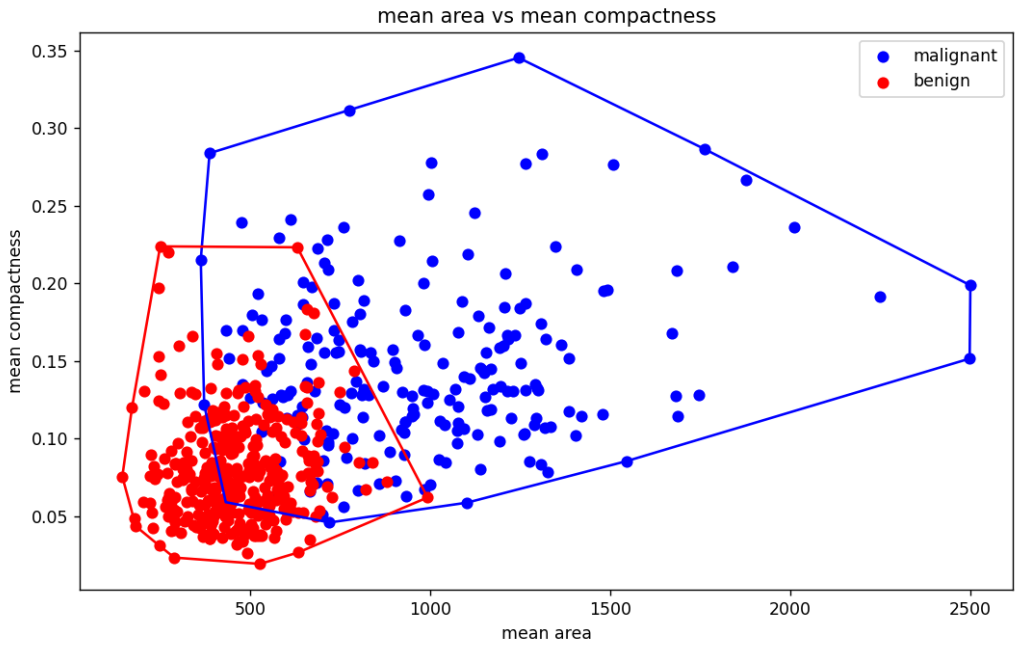
Data Set	Wine
First Feature	alcohol
Second Feature	malic_acid
Resulting Convex Hull	



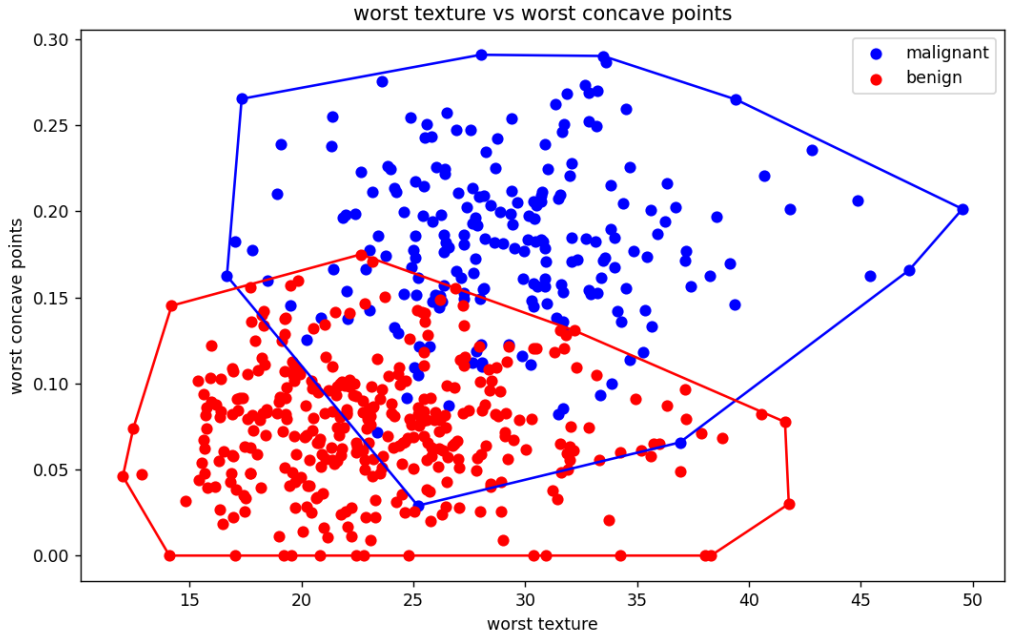
Data Set	Wine
First Feature	alcohol
Second Feature	flavonoids
Resulting Convex Hull	



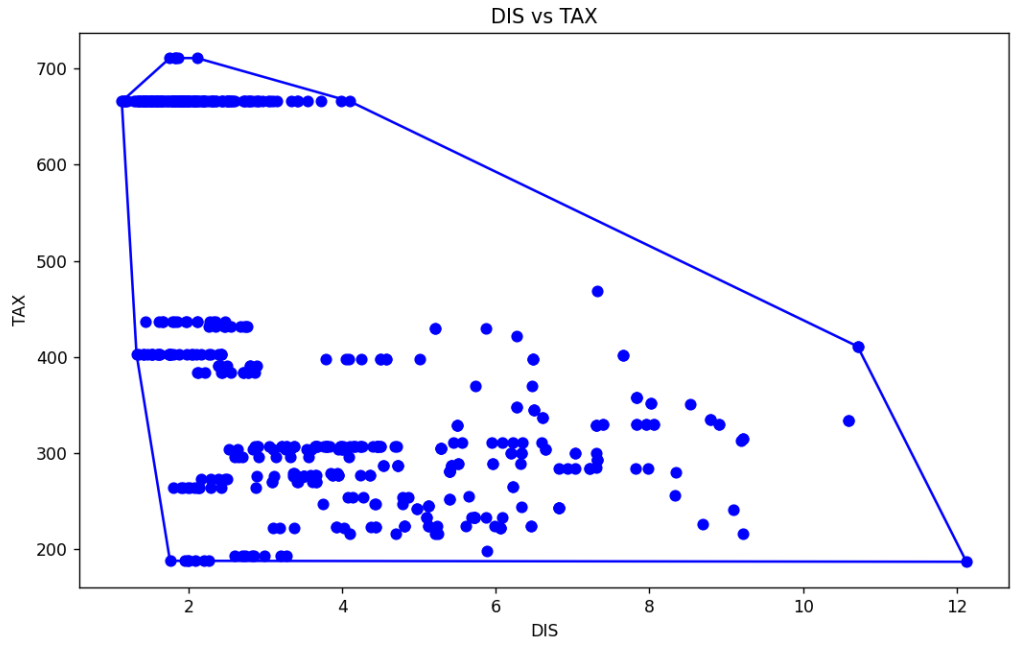
Data Set	breast_cancer
First Feature	mean area
Second Feature	mean compactness
Resulting Convex Hull	



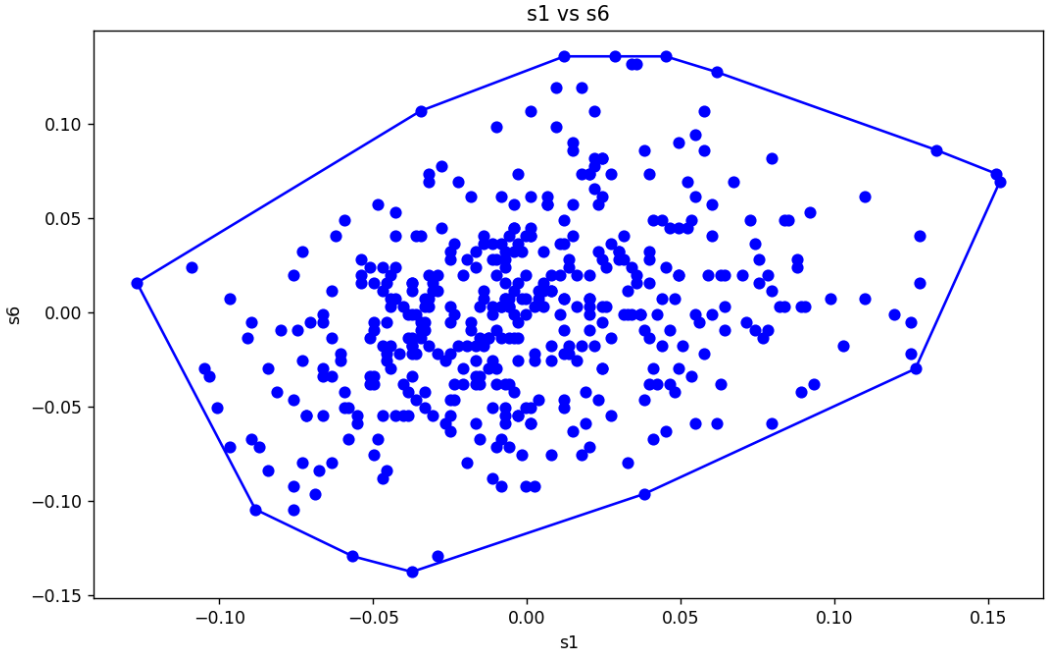
Data Set	breast_cancer
First Feature	Worst texture
Second Feature	Worst concave points
Resulting Convex Hull	



Data Set	boston
First Feature	DIS
Second Feature	TAX
Resulting Convex Hull	



Data Set	diabetes
First Feature	S1
Second Feature	S6
Resulting Convex Hull	



Bagian IV
Link Source Code / Github

Berikut terlampir link drive source code serta isinya sesuai dengan spesifikasi :
https://drive.google.com/drive/folders/1yskNVeuTaLi9yG3toXUM0XPLJjeCM_z3?usp=sharing

Berikut Repository Github:
https://github.com/Nk-Kyle/ConvexHull

Poin	Ya	Tidak
1. Pustaka *myConvexHull* berhasil dibuat dan tidak ada kesalahan	✓	
2. *Convex Hull* yang dihasilkan sudah benar	✓	
3. Pustaka *myConvexHull* dapat digunakan untuk menampilkan *convex hull* setiap label dengan warna yang berbeda	✓	
4. Program dapat menerima input dan menuliskan output untuk dataset lainnya	✓	