LAPORAN TUGAS KECIL 2

IF2211 STRATEGI ALGORITMA SEMESTER II TAHUN 2021/2022

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

DANIEL SALIM
13520008
K02

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

DAFTAR ISI

BAB I **3**
DASAR TEORI **3**
1.1 ALGORITMA DIVIDE AND CONQUER 3
1.2 CONVEX HULL 3
BAB II **6**
SOURCE CODE PROGRAM **6**
2.1 Screenshot cvxhull.py 6
2.2 Screenshot main.py 9
BAB III **10**
INPUT DAN OUTPUT PROGRAM **10**
3.1 Dataset Iris 10
3.1.1 Sepal width vs Sepal length 10
3.1.2 Petal width vs Petal length 10
3.2 Dataset Breast Cancer 11
3.2.1 Radius vs Texture 11
3.2.2 Smoothness vs Compactness 11
3.3 Dataset Wine 12
3.3.1 Ash vs Magnesium 12
3.3.2 Flavanoids vs Proanthocyanins 12
LINK **12**
DAFTAR PUSTAKA **14**

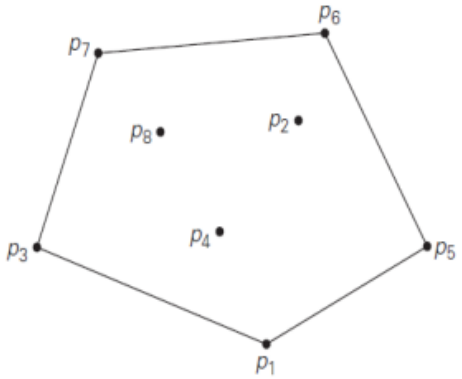
BAB I
DASAR TEORI

1.1 ALGORITMA DIVIDE AND CONQUER

Divide and conquer merupakan suatu teknik atau algoritma yang berguna dalam memecahkan berbagai masalah. Teknik ini terdiri dari tiga bagian, yang pertama adalah *divide,* yaitu algoritma akan mengawali prosesnya dengan membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya berukuran hampir sama). Kedua adalah *conquer*(solve), yaitu menyelesaikan masing-masing upa-persoalan (secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar). Ketiga adalah *combine*, yaitu menggabungkan solusi masing-masing upa-persoalan sehingga membentuk solusi persoalan semula. Algoritma ini digunakan dalam algoritma-algoritma lainnya, seperti *quicksort, merge sort, closest pair of points, strassen's algorithm,* dll.

1.2 CONVEX HULL

Dalam pengerjaan Tugas Kecil kali ini didasari oleh konsep algoritma *divide and conquer*, yaitu dalam mengimplementasi *convex hull* dalam visualisasi tes *linear seperability dataset*. Himpunan titik dapat dikatakan *convex* jika untuk dua buah titik sembarang, seluruh segmen garis yang berakhir pada dua titik sembarang tersebut mencakup seluruh himpunan titik tersebut.

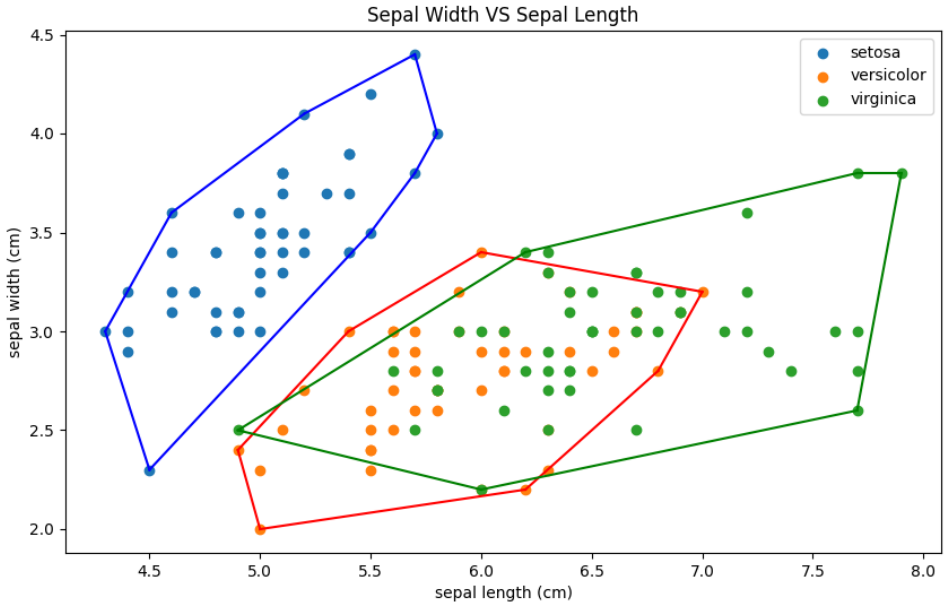


Gambar 1.1 Contoh convex hull 8 titik

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

Dengan demikian, kita dapat mengimplementasi konsep *convex hull* dalam menentukan *linear separability* dari sebuah dataset. Gambar di bawah ini adalah hasil visualisasi tes dari dataset iris yang menunjukkan hubungan *sepal width* dan *sepal length.* Sebuah kelas dikatakan

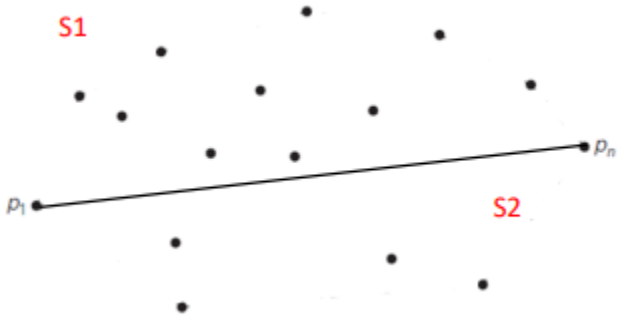
linearly separable apabila *convex hull* kelas tersebut tidak bertabrakan dengan *convex hull* kelas lain seperti kelas “setosa” pada gambar di bawah ini.



Gambar 1.2 Visualisasi dataset iris
sumber: Arsip pribadi

Kita dapat memperoleh *convex hull* seperti gambar di atas dengan menggunakan algoritma *divide and conquer* dengan tujuan yaitu menemukan kumpulan titik paling luar untuk membentuk *convex hull* dengan ide dasar menggunakan algoritma *quicksort.* Langkah pengerjaannya adalah sebagai berikut.

1. Pertama bentuklah sebuah garis yang menghubungkan p1 (titik paling kiri) dan pn (titik paling kanan) yang membagi himpunan titik S menjadi dua bagian yaitu S1 (kumpulan titik di sebelah kiri atau atas garis) dan S2 (kumpulan titik di sebelah kanan atau bawah garis).



Gambar 1.3 Ilustrasi langkah pertama
sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf

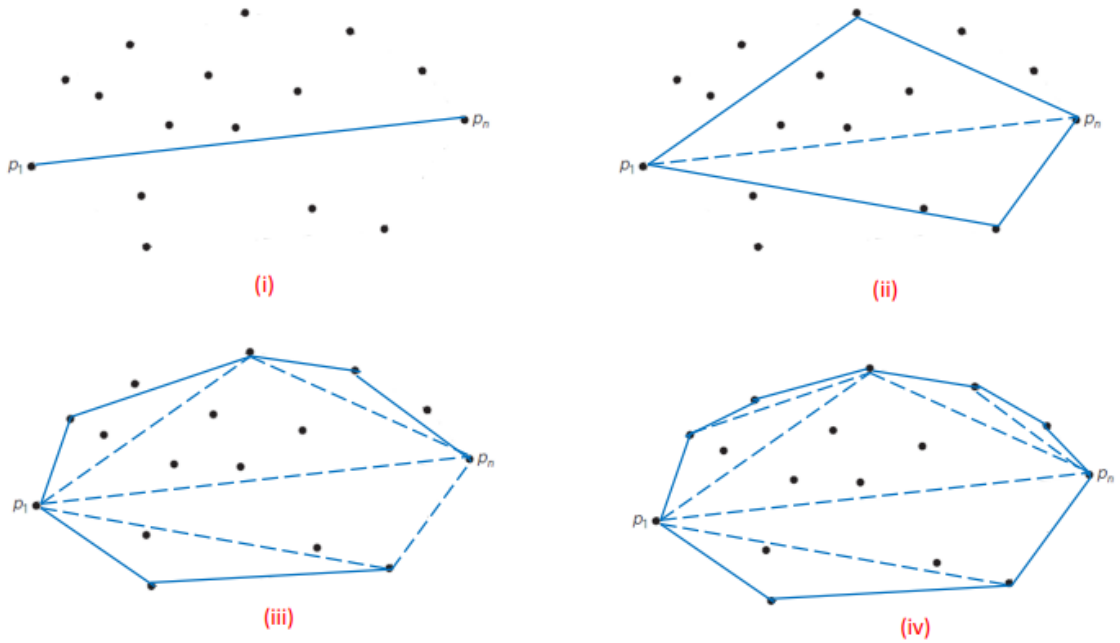
Untuk memeriksa apakah sebuah titik berada di sebelah atas atau bawah suatu garis yang dibentuk dua titik, dapat menggunakan rumus determinan.

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

Titik (x3,y3) berada di atas garis jika hasil di atas > 0 dan berada di bawah garis jika hasil diatas < 0.

2. Kumpulan titik pada S1 dapat membentuk *convex hull* bagian atas dan S2 dapat membentuk *convex hull* bagian bawah.

3. Misal untuk memperoleh bagian *convex hull* atas dapat diterapkan basis, yaitu jika tidak ada titik lain selain S1, maka titik *p*1 dan *pn* menjadi pembentuk *convex hull* bagian S1. Lalu, rekursinya adalah jika S1 tidak kosong, pilih sebuah titik yang memiliki jarak terjauh dari garis *p*1*pn* (misal pmax) dan periksa apakah masih ada kumpulan titik di area sebelah kiri/atas p1pmax dan sebelah kanan/atas pmaxpn. Jika ada, maka rekursi tetap berjalan dan akan berhenti sampai tidak ditemukan lagi adanya titik di luar area garis. Program akan mengembalikan pasangan titik yang dihasilkan.



Gambar 1.4 Ilustrasi tahapan penentuan titik *convex hull*

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

BAB II
SOURCE CODE PROGRAM

2.1 Screenshot cvxhull.py

```python
import numpy as np

def getAboveOrBelow(point, startPt, endPt):
    xS, yS = startPt; xE, yE = endPt; xP, yP = point
    det = (xS*yE + xP*yS + xE*yP - xP*yE - xE*yS - xS*yP)
    # ^Determinant formula to check whether xP,yP is above/below the line created by xS,yS and xE,yE

    if(det > 10**(-10)):
        return "ABOVE"
    elif(det < 0):
        return "BELOW"
# Obtain the location(above/below) of a point from a line (xP -> yP)

def getFarthest(arr,tailPt,headPt):
    curFarthestPoint = []; curDist = 0
    for point in arr:
        if(getDistance(point,tailPt,headPt) >= curDist):
            curDist = getDistance(point,tailPt,headPt)
            curFarthestPoint = point
    return curFarthestPoint
# Obtain the farthest point from a line (tailPt -> headPt)

def getDistance(point,tailPt,headPt):
    tailPt = np.asfarray(tailPt); headPt = np.asfarray(headPt); point = np.asfarray(point)
    distance = (np.linalg.norm(np.cross(headPt-tailPt, tailPt-point))/np.linalg.norm(headPt-tailPt))
    # ^Distance formula

    return distance
# Obtain the distance between a point and a line (tailPt -> headPt)

def isArrEmpty(arr):
    if len(arr) == 0:
        return "Empty"

```

```python
def getConvexHullPt(location, ptArr, startPt, endPt):
    # ptArr is either ptAbove or ptBelow
    # startPt is the starting point of the line, either ptLeft or ptRight
    # endPt is the end point of the line, either ptLeft or ptRight
    # location can be TOP or BOTTOM depends on ptArr

    #BASIS
    if(isArrEmpty(ptArr) == "Empty"):
        if(location =="A"):
            cvxHullPtAbove.extend([startPt])
            cvxHullPtAbove.extend([endPt])
        else:
            cvxHullPtBelow.extend([startPt])
            cvxHullPtBelow.extend([endPt])
    # The basis of the algorithm will assign 2 points that made the line as convex hull point if there's no point found
    # above/below the line

    #RECURSIVE
    else:
        furthestPoint = getFarthest(ptArr,startPt,endPt)

        ptAbove_leftLine=[]
        # Array for storing points that's above the line (ptLeft -> furthestPoint)

        ptAbove_rightLine=[]
        # Array for storing points that's above the line (furthestPoint -> ptRight)

        for point in ptArr:
            if(getAboveOrBelow(point,startPt,furthestPoint) == "ABOVE"):
                ptAbove_leftLine.extend([point])

        for point in ptArr:
            if(getAboveOrBelow(point,furthestPoint,endPt) == "ABOVE"):
                ptAbove_rightLine.extend([point])
        getConvexHullPt(location, ptAbove_leftLine,startPt,furthestPoint)
        getConvexHullPt(location, ptAbove_rightLine,furthestPoint,endPt)
        # Recursive until there's no point above/below the line
# Obtain the convex hull points using recursive
```

```python
def myConvexHull(arr):
    sortedArr = np.array(sorted(arr , key=lambda x:[x[0], x[1]]))

    global cvxHullPtAbove
    global cvxHullPtBelow
    cvxHullPtAbove = []
    cvxHullPtBelow = []
    # Assign empty array for storing convex hull points above and below the line created by the ptLeft and ptRight point

    ptAbove = []
    ptBelow = []
    # Assign empty array for storing points from above and below the line created by ptLeft and ptRight

    ptLeft = sortedArr[0]
    ptRight = sortedArr[-1]
    # Obtain the first and last point of the dataset

    for point in sortedArr:
        checkPos = getAboveOrBelow(point,ptLeft,ptRight)
        if checkPos == "ABOVE":
            ptAbove.append(point)
        else:
            ptBelow.append(point)
    # Divide the points from the array to 2 sides by checking their location (above or below the line)
    # and append it to ptAbove/ptBelow

    getConvexHullPt("A",ptAbove,ptLeft,ptRight)
    getConvexHullPt("B",ptBelow,ptRight,ptLeft)
    sortedHullAbove = sorted(cvxHullPtAbove , key=lambda x:[x[0], x[1]])
    # Obtain a sorted dataset (axis first then ordinate) in ascending order

    sortedHullBelow = sorted(cvxHullPtBelow , key=lambda x:[x[0], x[1]], reverse = True)
    # Obtain a sorted dataset (axis first then ordinate) in descending order

    finalConvexHull = np.concatenate((sortedHullAbove, sortedHullBelow), axis=0)
    # Obtain the final convex hull through merging these 2 arrays

    return(finalConvexHull)
```

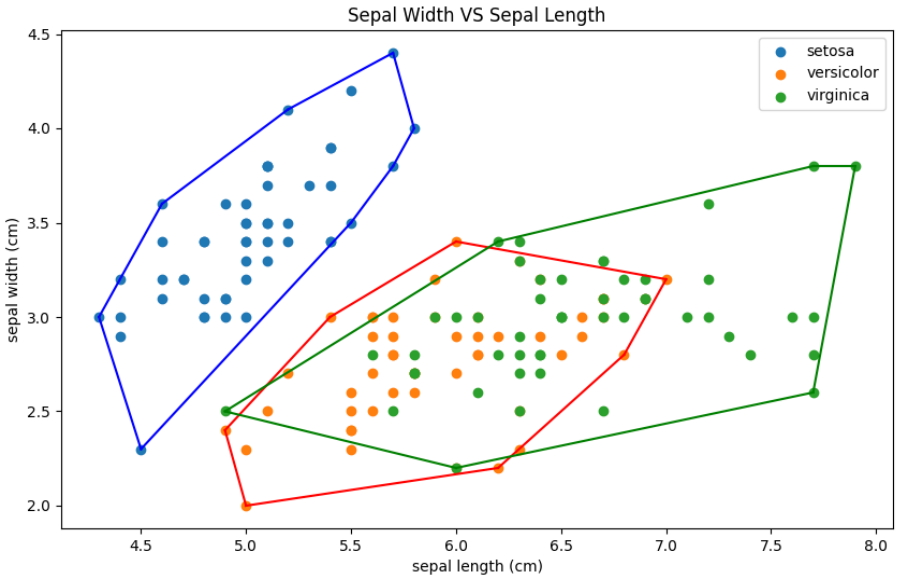
2.2 Screenshot main.py

```python
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import datasets
from cvxhull import *

iris = datasets.load_iris()
# bc = datasets.load_breast_cancer()
# wine = datasets.load_wine()
# Load datasets

iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_df['Target'] = pd.DataFrame(iris.target)

# bc_df = pd.DataFrame(bc.data, columns=bc.feature_names)
# bc_df['Target'] = pd.DataFrame(bc.target)

# wine_df = pd.DataFrame(wine.data, columns=wine.feature_names)
# wine_df['Target'] = pd.DataFrame(wine.target)
## Creating dataframe

plt.figure(figsize = (10, 6))
plt.title('Sepal Width VS Sepal Length')
colors = ['b','r','g']
# Configure the visualisation of convex hull

plt.xlabel(iris.feature_names[2])
plt.ylabel(iris.feature_names[3])
for i in range(len(iris.target_names)):
    bucket = iris_df[iris_df['Target'] == i]
    bucket = bucket.iloc[:,[2,3]].values
    cvxHull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=iris.target_names[i])
    for j in range(len(cvxHull)-1):
        plt.plot((cvxHull[j][0], cvxHull[j+1][0]), (cvxHull[j][1], cvxHull[j+1][1]), colors[i])
plt.legend()
plt.show()
# Replace the feature names and bucket value's index to desired attribute
```

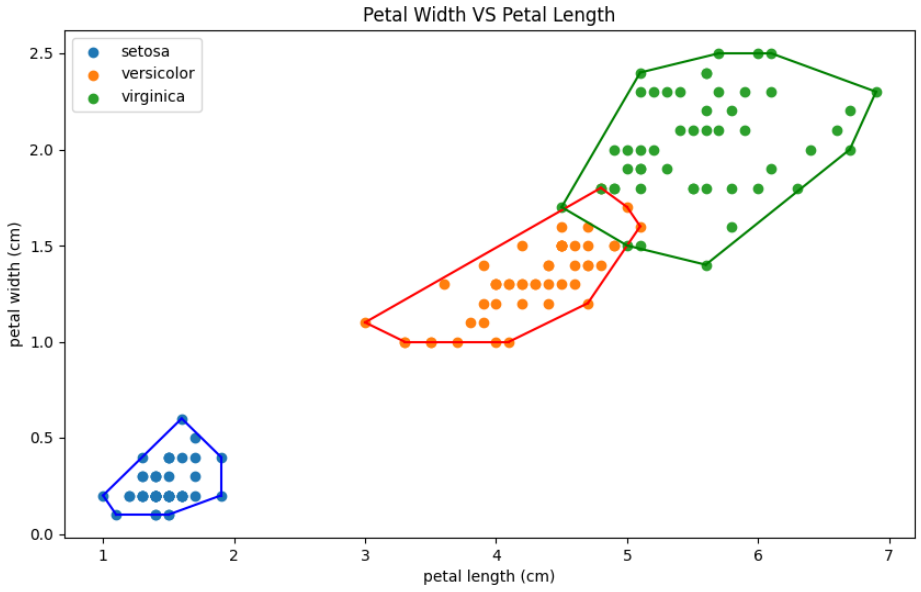
BAB III
INPUT DAN OUTPUT PROGRAM

3.1 Dataset Iris

3.1.1 Sepal width vs Sepal length

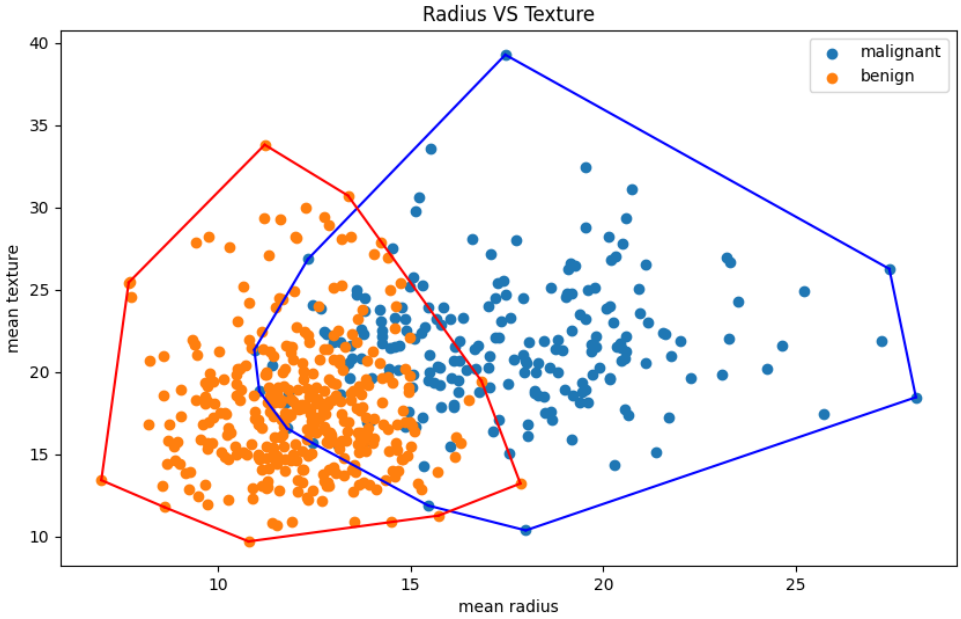


3.1.2 Petal width vs Petal length

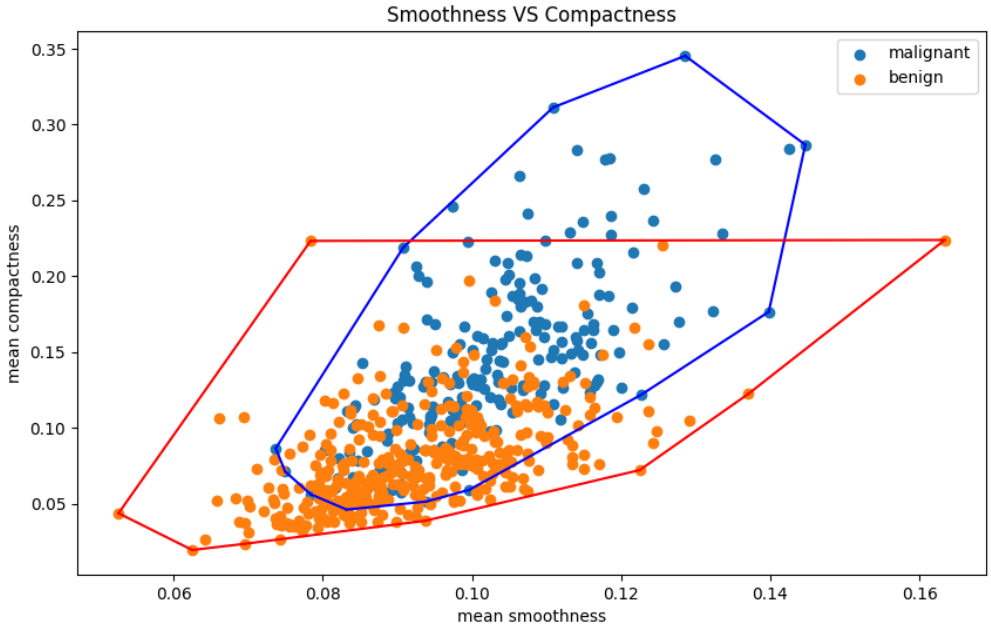


3.2 Dataset Breast Cancer

3.2.1 Radius vs Texture

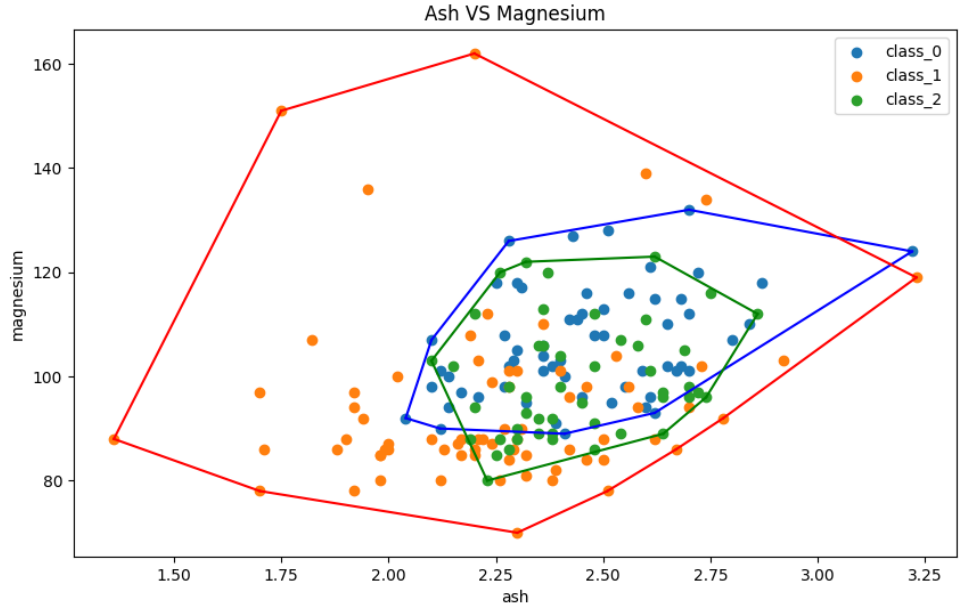


3.2.2 Smoothness vs Compactness

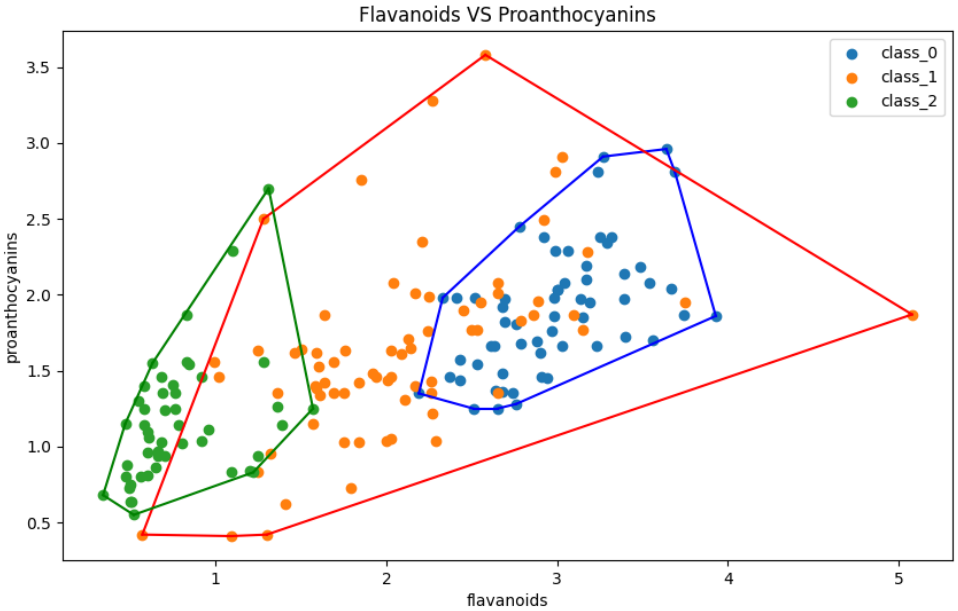


3.3 Dataset Wine

3.3.1 Ash vs Magnesium



3.3.2 Flavanoids vs Proanthocyanins



LINK

https://github.com/danielsalim/convexhull

DAFTAR PUSTAKA

Munir, Rinaldi. (2021). Algoritma Divide and Conquer. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf (diakses tanggal 28 Februari 2022).
(Materi tentang Algoritma Divide and Conquer)

(19 Juli 2021). Divide and Conquer Algorithm. https://www.geeksforgeeks.org/divide-and-conquer-algorithm-introduction/ (diakses tanggal 28 Februari 2022)
(Materi tentang Algoritma Divide and Conquer)

Munir, Rinaldi. (2021). Convex Hull. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf (diakses tanggal 28 Februari 2022).
(Materi tentang Convex Hull)