Tugas Kecil 2 IF 2211 Strategi Algoritma
Semester II Tahun 2021/2022

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

DISUSUN OLEH

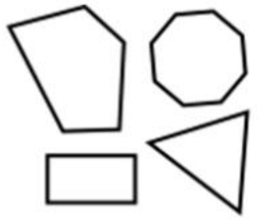
Kevin Roni	**13520114**

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

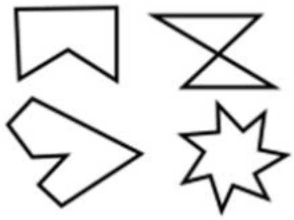
BAB I
ALGORITMA DIVIDE AND CONQUER

Deskripsi

Convex hull dapat didefinisikan sebagai himpunan convex terkecil dari sebuah himpunan titik, misalkan S. himpunan titik sendiri disebut convex apabila untuk sembarang dua titik pada himpunan S, misalkan P dan Q, seluruh segmen garis yang berakhir di P dan Q berada pada himpunan tersebut.



Gambar 1: convex



Gambar 2:non convex

Algoritma Divide and Conquer

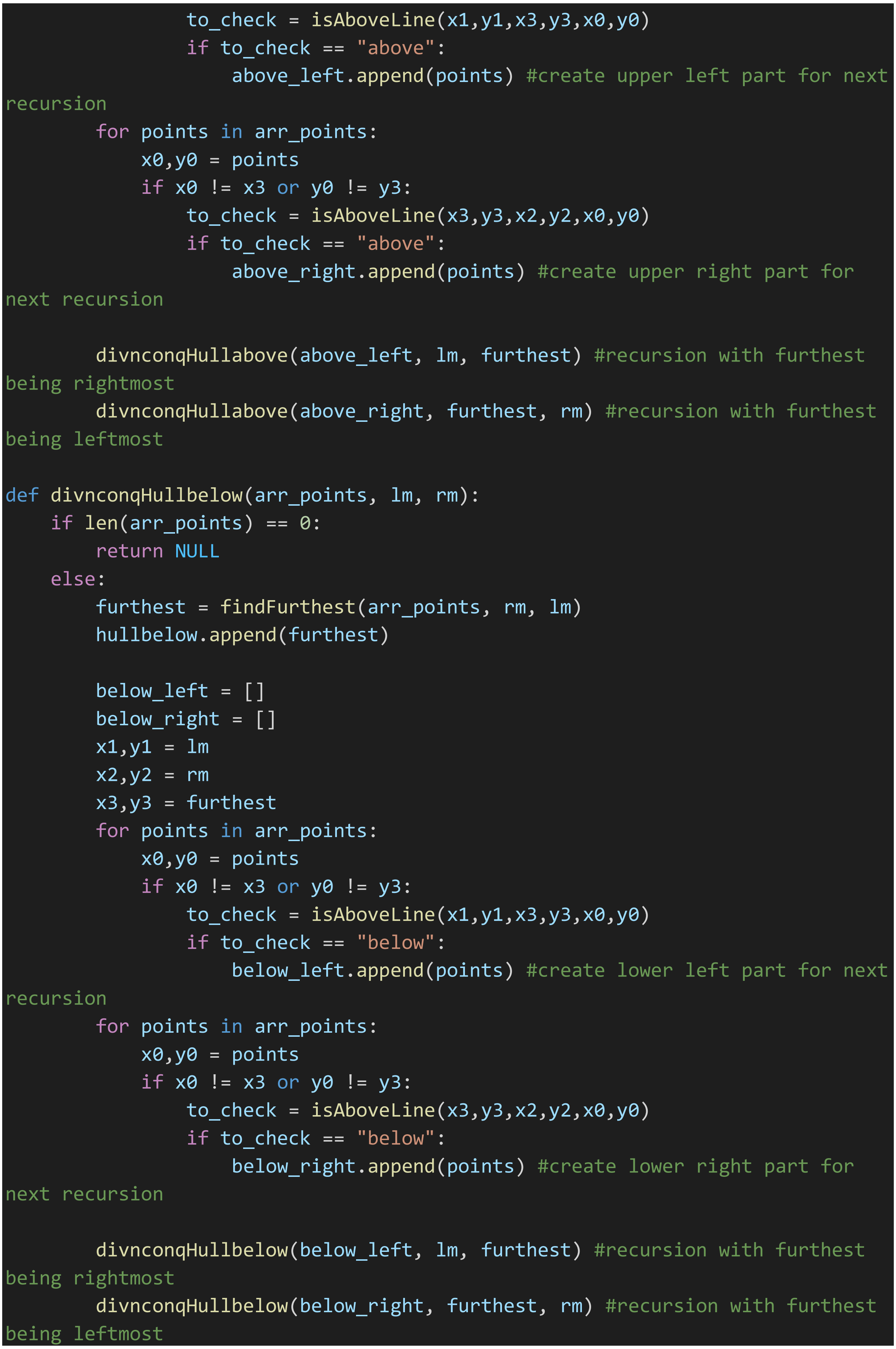
Pada dasarnya, algoritma *divide and conquer* memiliki ide untuk membagi persoalan yang sangat besar menjadi persoalan yang kecil yang memiliki karakteristik mirip. Solusi-solusi dari persoalan kecil ini kemudian digabung menjadi solusi untuk persoalan utama. Untuk persoalan mencari convex hull, idenya adalah membagi daerah menjadi dua bagian lalu dicari solusi hull untuk kedua daerah tersebut. Persoalan kemudian diulang hingga tidak ada lagi kemungkinan titik yang dapat menjadi solusi hull. Berikut adalah Langkah-langkah *divide and conquer* yang saya lakukan

1. Mula-mula cari dua titik terjauh dihitung berdasarkan sumbu-x. Titik terjauh di sebelah kanan akan disebut rightmost (rm) dan titik terjauh di sebelah kiri disebut leftmost (lm)
2. Tarik sebuah garik imajiner dari lm rm sehingga daerah terbagi menjadi dua, upper part dan lower part. Solusi convex dari upper part dan lower part akan digabung menjadi solusi utama
3. Titik lm dan rm pasti merupakan solusi sehingga dimasukkan ke dalam himpuanan solusi.
4. Apabila tidak ada himpunan titik di daerah upper part atau lower part, pencarian akan dihentikan untuk bagian tersebut.
5. Cari titik terjauh dari garis lmrm untuk kedua daerah, upper dan lower. Titik terjauh ini, misalkan titik P, pasti merupakan solusi dari convex hull
6. Untuk himpunan solusi upper part, tarik sebuah garis imajiner dari lm ke titik P. abaikan semua titik berada pada garis dan yang berada pada bawah garis. Lalu cari titik terjauh pada himpunan titik yang berada di atas garis. Titik terjauh ini merupakan solusi convex hull. Lakukan pengulangan terus menerus dengan mengganti titik P menjadi titik terjauh terbaru. Pencarian dihentikan ketika tidak ada himpunan titik yang berada di atas garis
7. Lakukan hal yang sama pada Langkah 5, tetapi garis dibuat dari titik P ke titik rm
8. Untuk himpunan solusi lower part, lakukan Langkah yang sama seperti 5 dan 6 tetapi himpunan yang akan dicari berada di bawah garis imajiner

BAB II

Hull.py

```
from asyncio.windows_events import NULL
import numpy as np
from cmath import sqrt

def isAboveLine(x1,y1,x2,y2,x3,y3):
    #is a point p(x3,y3) above line l(x2x1,y2y1)
    det = (x1*y2+x3*y1+x2*y3)-(x3*y2+x2*y1+x1*y3)
    if det > 0:
        return "above"
    else:
        return "below"

def findFurthest(arr_points, lm, rm):
    #find furthest points in arr_points from line lm rm
    x1, y1 = lm
    x2, y2 = rm
    a = y2 - y1
    b = x1 - x2
    c = (x2*y1) - (x1*y2)
    furthest_dist = -1
    furthest_point = None
    for points in arr_points:
        x3, y3 = points
        curr_dist = abs(a*x3 + b*y3 + c)/sqrt(a**2 + b**2)
        if curr_dist>furthest_dist:
            #update furthest point & distance
            furthest_point = points
            furthest_dist = curr_dist
    return furthest_point

def divnconqHullabove(arr_points, lm, rm):
    if len(arr_points) == 0:
        #base case, finish recursion
        return NULL
    else:
        furthest = findFurthest(arr_points, lm, rm)
        hullabove.append(furthest) #furthest point is a solution

        above_left = []
        above_right = []
        x1,y1 = lm
        x2,y2 = rm
        x3,y3 = furthest
        for points in arr_points:
            x0,y0 = points
            if x0 != x3 or y0 != y3:
```



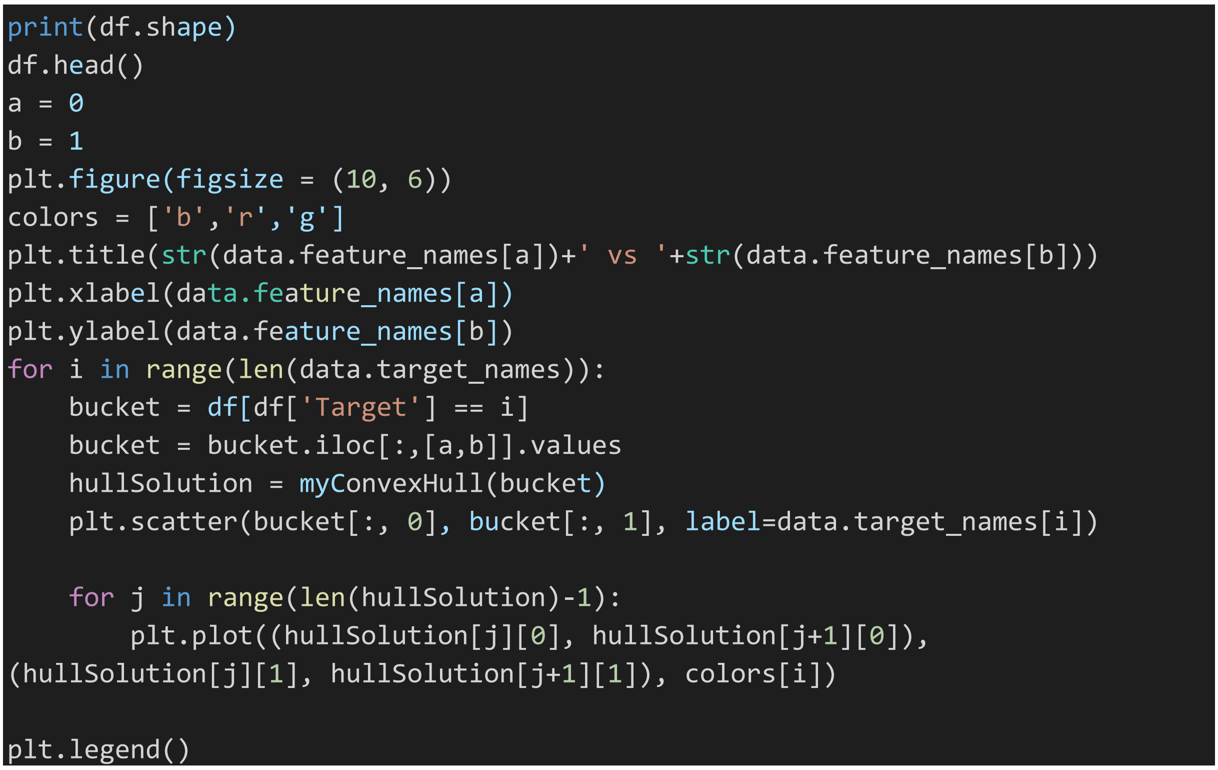
```
                to_check = isAboveLine(x1,y1,x3,y3,x0,y0)
                if to_check == "above":
                    above_left.append(points) #create upper left part for next
recursion
        for points in arr_points:
            x0,y0 = points
            if x0 != x3 or y0 != y3:
                to_check = isAboveLine(x3,y3,x2,y2,x0,y0)
                if to_check == "above":
                    above_right.append(points) #create upper right part for
next recursion

        divnconqHullabove(above_left, lm, furthest) #recursion with furthest
being rightmost
        divnconqHullabove(above_right, furthest, rm) #recursion with furthest
being leftmost

def divnconqHullbelow(arr_points, lm, rm):
    if len(arr_points) == 0:
        return NULL
    else:
        furthest = findFurthest(arr_points, rm, lm)
        hullbelow.append(furthest)

        below_left = []
        below_right = []
        x1,y1 = lm
        x2,y2 = rm
        x3,y3 = furthest
        for points in arr_points:
            x0,y0 = points
            if x0 != x3 or y0 != y3:
                to_check = isAboveLine(x1,y1,x3,y3,x0,y0)
                if to_check == "below":
                    below_left.append(points) #create lower left part for next
recursion
        for points in arr_points:
            x0,y0 = points
            if x0 != x3 or y0 != y3:
                to_check = isAboveLine(x3,y3,x2,y2,x0,y0)
                if to_check == "below":
                    below_right.append(points) #create lower right part for
next recursion

        divnconqHullbelow(below_left, lm, furthest) #recursion with furthest
being rightmost
        divnconqHullbelow(below_right, furthest, rm) #recursion with furthest
being leftmost
```

```
def myConvexHull(data):
    global hullSolution
    global hullabove
    global hullbelow
    hullSolution = [] #main solution
    hullabove = [] #solution for upper part
    hullbelow = [] #solution for lower part
    data = np.array(sorted(data, key=lambda k: [k[0], k[1]]))
    lm = data[0]
    rm = data[-1]

    points_above = []
    points_below = []
    hullSolution.append(lm)

    x1, y1 = lm
    x2, y2 = rm
    for points in data:
        x3, y3 = points
        tocheck = isAboveLine(x1,y1,x2,y2,x3,y3)
        if tocheck == "above":
            points_above.append(points)
        elif tocheck == "below":
            points_below.append(points)

    divnconqHullabove(points_above,lm,rm)
    divnconqHullbelow(points_below,lm,rm)
    hullabove = np.array(sorted(hullabove, key=lambda k: [k[0], k[1]]))
    hullbelow = np.array(sorted(hullbelow, reverse=True, key=lambda k: [k[0],
k[1]]))
    hullSolution.extend(hullabove)
    hullSolution.append(rm)
    hullSolution.extend(hullbelow)
    hullSolution.append(lm)
    return hullSolution
```

Main.ipynb

(hanya dilampirkan untuk satu percobaan file)

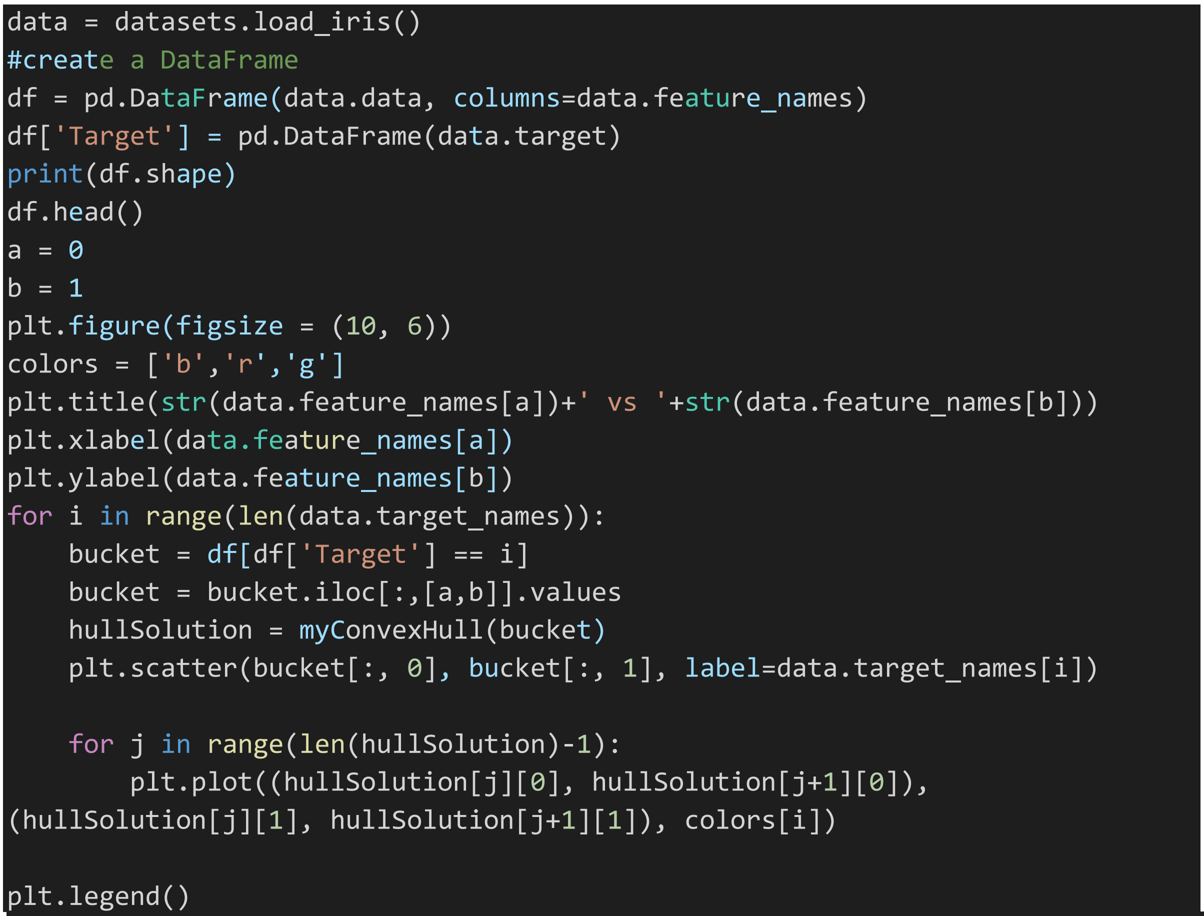
```
from scipy.spatial import ConvexHull
from sklearn import datasets
import matplotlib.pyplot as plt
import pandas as pd
from hull import *

data = datasets.load_iris()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
```



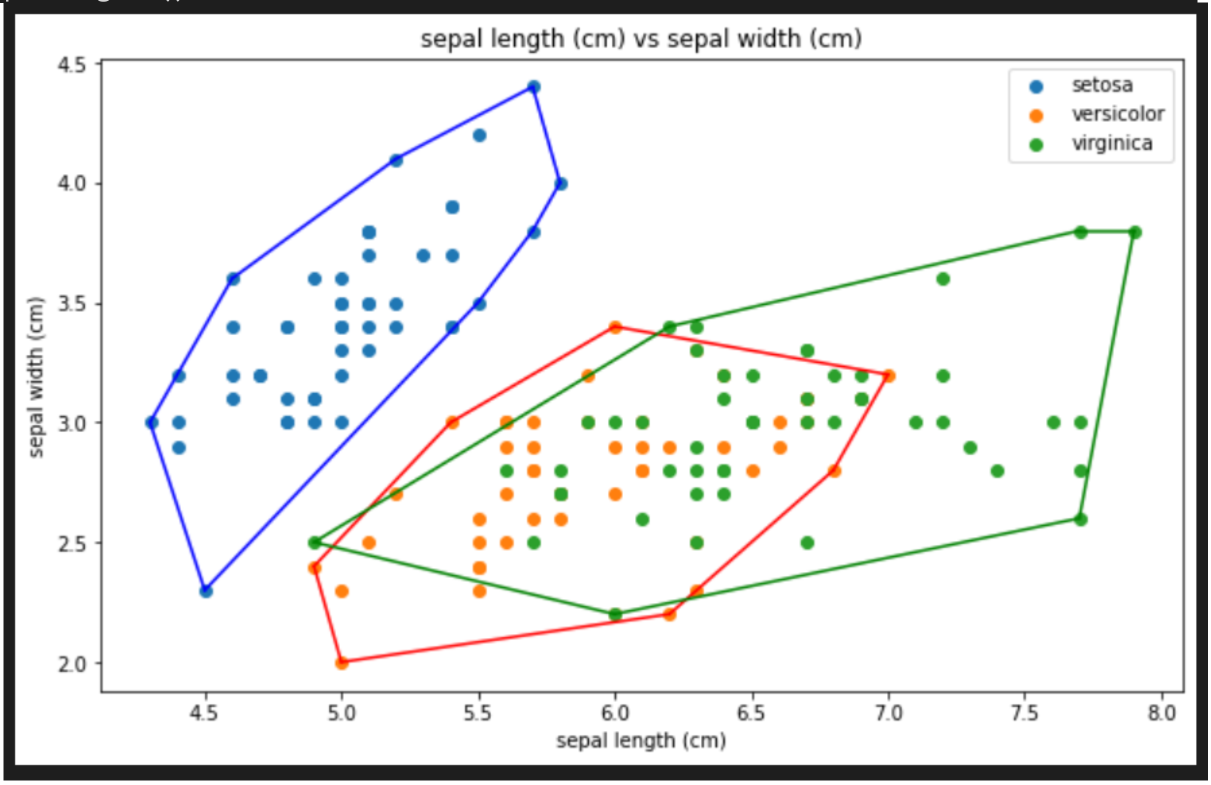
```
print(df.shape)
df.head()
a = 0
b = 1
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[a])+' vs '+str(data.feature_names[b]))
plt.xlabel(data.feature_names[a])
plt.ylabel(data.feature_names[b])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[a,b]].values
    hullSolution = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])

    for j in range(len(hullSolution)-1):
        plt.plot((hullSolution[j][0], hullSolution[j+1][0]),
(hullSolution[j][1], hullSolution[j+1][1]), colors[i])

plt.legend()
```

BAB III
INPUT OUTPUT

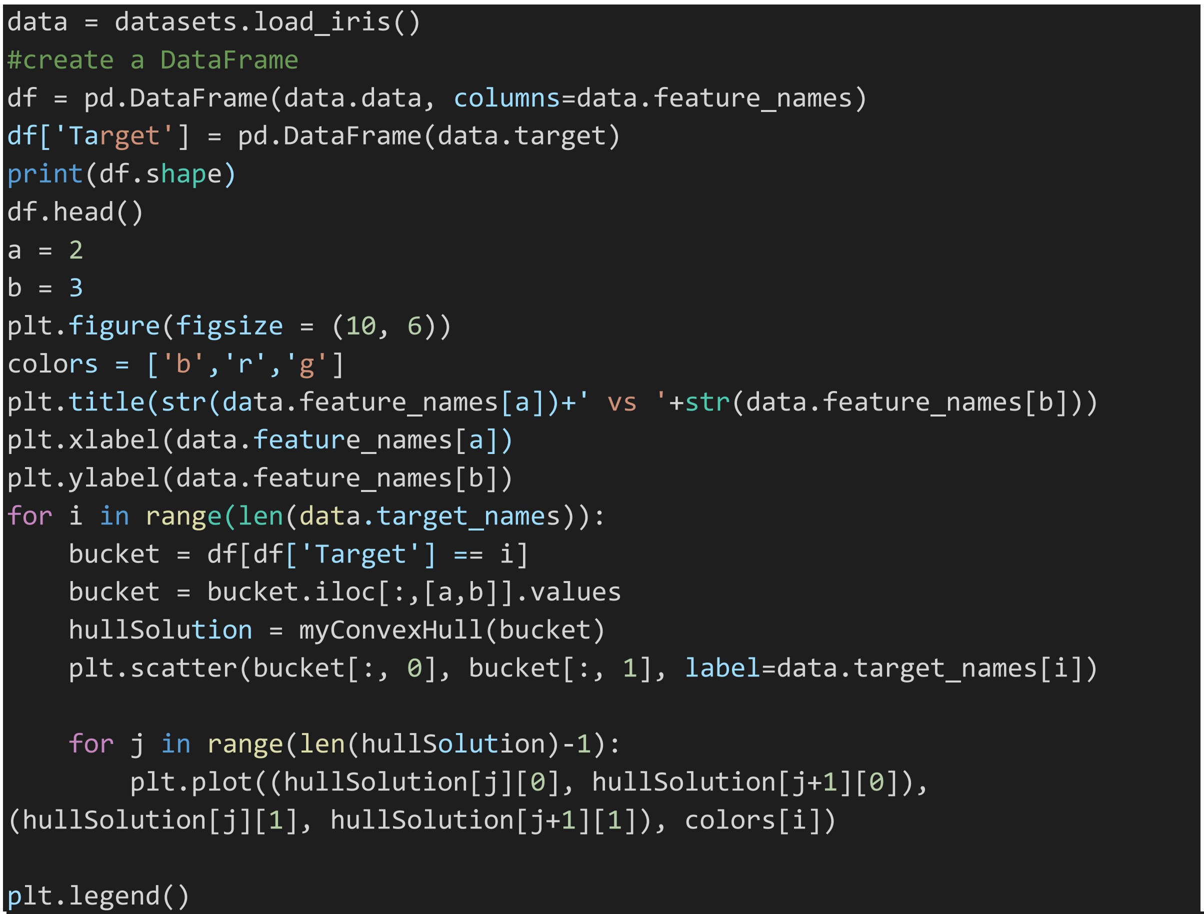
1. Sepal length vs sepal width



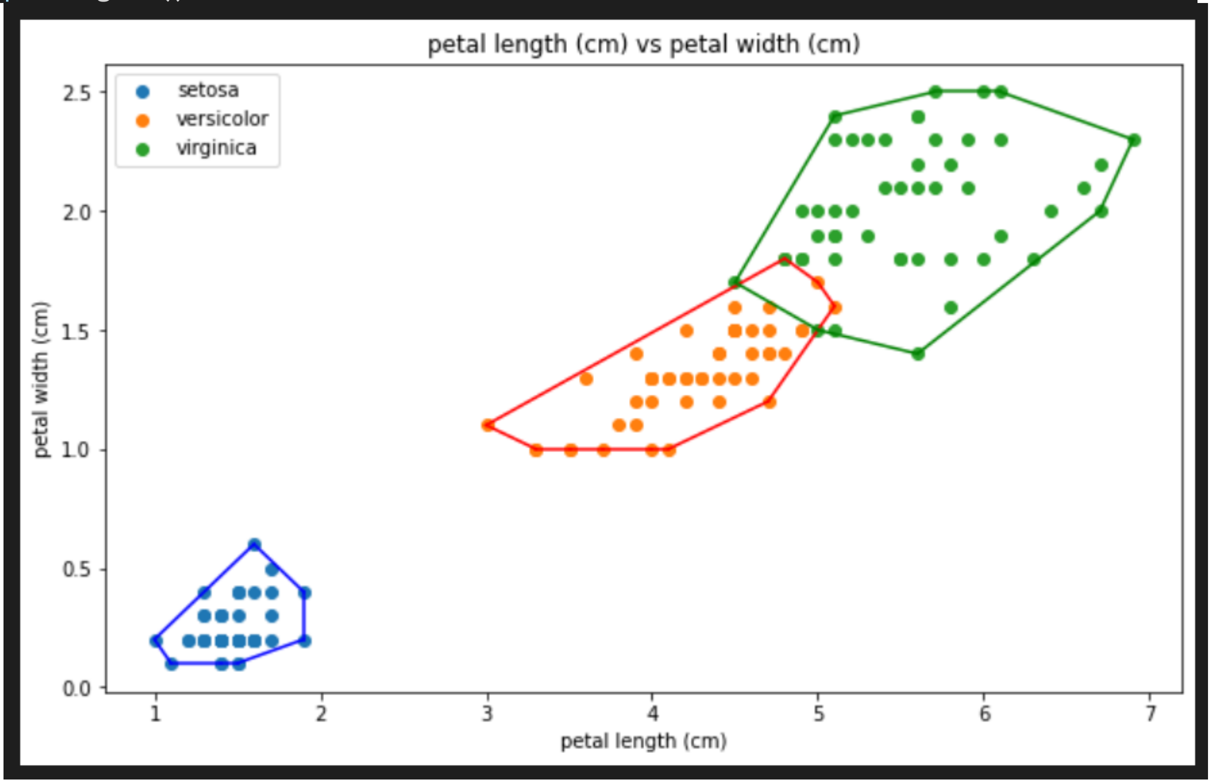
```
data = datasets.load_iris()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
a = 0
b = 1
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[a])+' vs '+str(data.feature_names[b]))
plt.xlabel(data.feature_names[a])
plt.ylabel(data.feature_names[b])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[a,b]].values
    hullSolution = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])

    for j in range(len(hullSolution)-1):
        plt.plot((hullSolution[j][0], hullSolution[j+1][0]),
(hullSolution[j][1], hullSolution[j+1][1]), colors[i])

plt.legend()
```



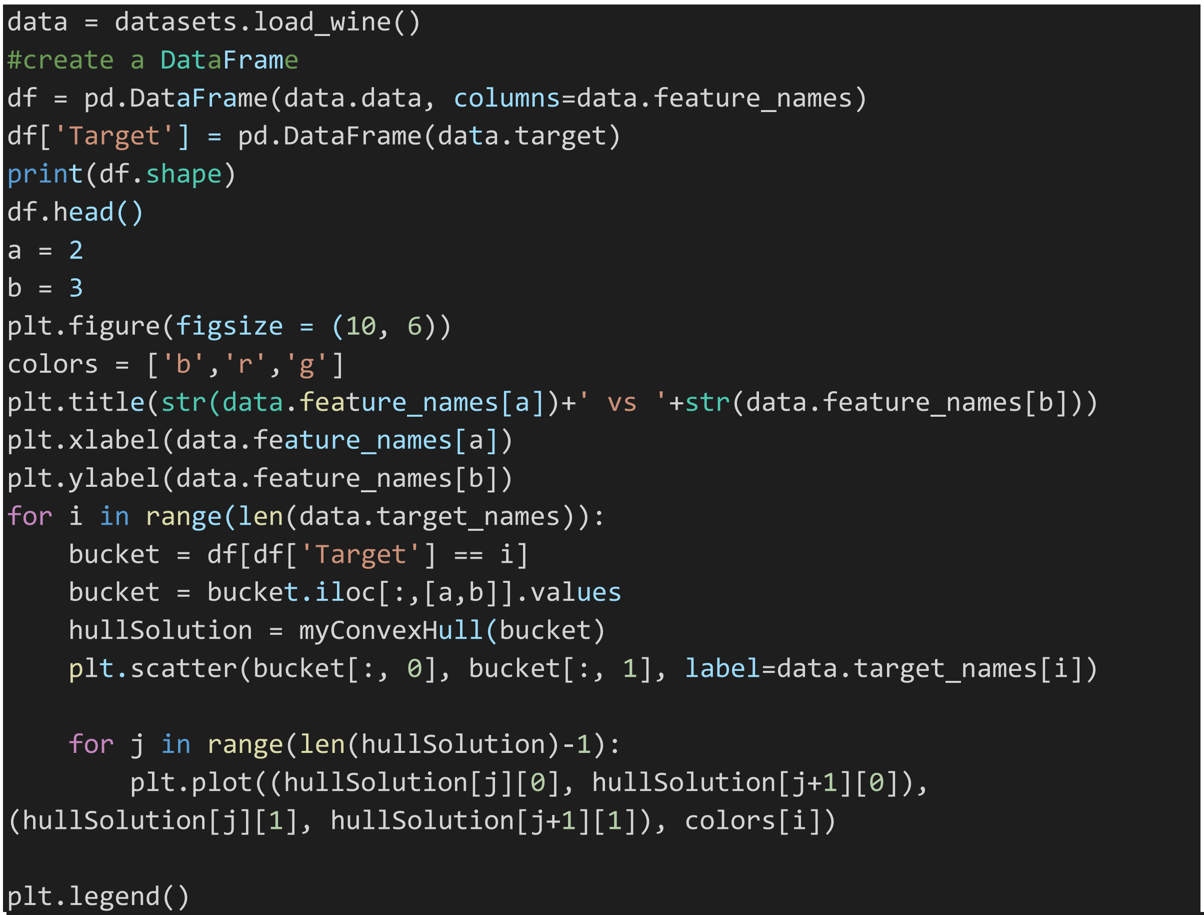
2. Petal width vs petal length



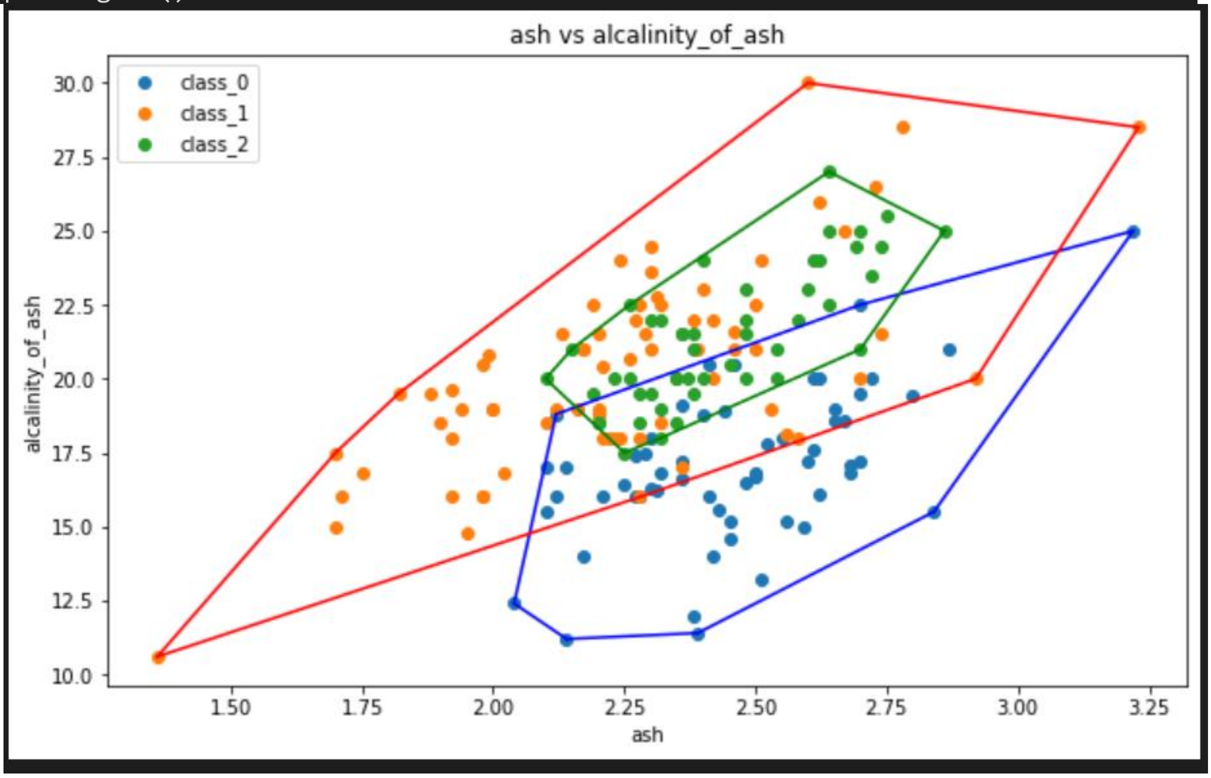
```
data = datasets.load_iris()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
a = 2
b = 3
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[a])+' vs '+str(data.feature_names[b]))
plt.xlabel(data.feature_names[a])
plt.ylabel(data.feature_names[b])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[a,b]].values
    hullSolution = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])

    for j in range(len(hullSolution)-1):
        plt.plot((hullSolution[j][0], hullSolution[j+1][0]),
(hullSolution[j][1], hullSolution[j+1][1]), colors[i])

plt.legend()
```



3. Ash vs alkalinity of ash

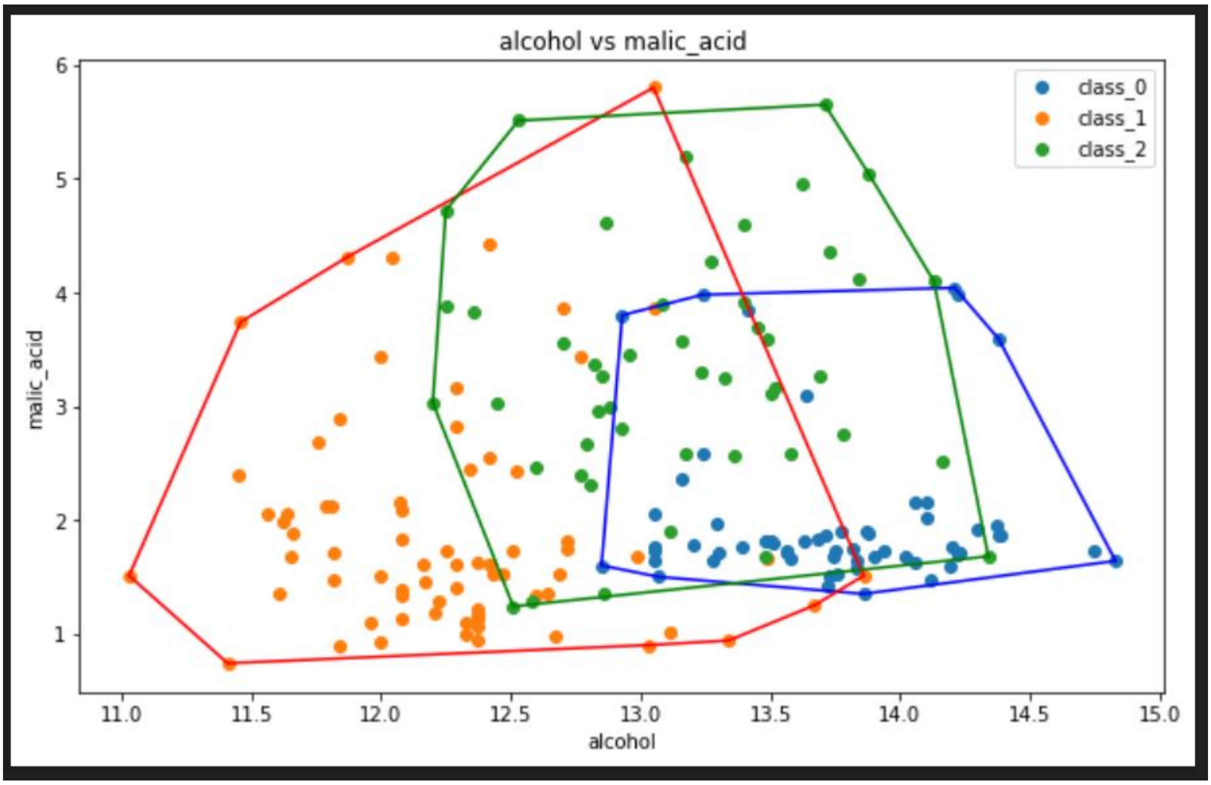


```
data = datasets.load_wine()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
a = 2
b = 3
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[a])+' vs '+str(data.feature_names[b]))
plt.xlabel(data.feature_names[a])
plt.ylabel(data.feature_names[b])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[a,b]].values
    hullSolution = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])

    for j in range(len(hullSolution)-1):
        plt.plot((hullSolution[j][0], hullSolution[j+1][0]),
(hullSolution[j][1], hullSolution[j+1][1]), colors[i])

plt.legend()
```



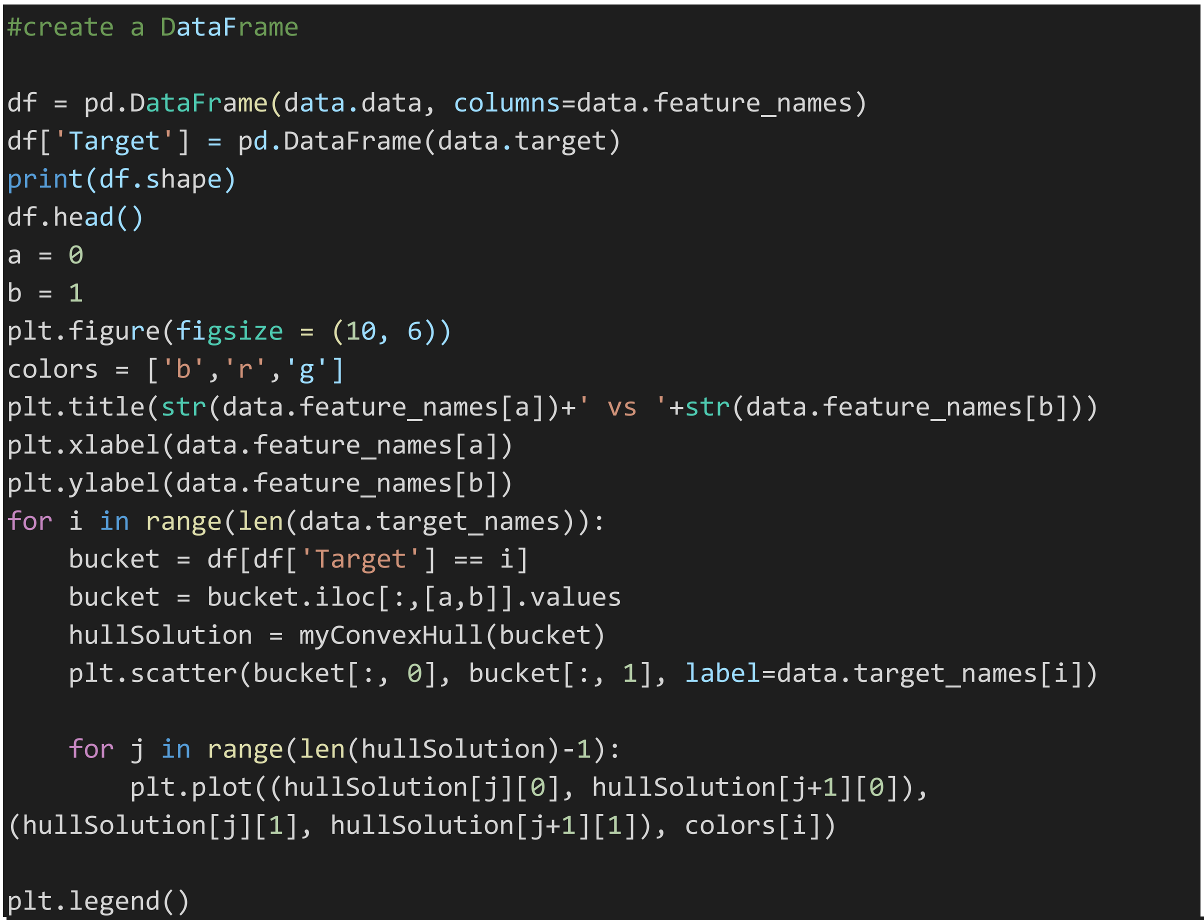
4. Alcohol vs malic acid

```
data = datasets.load_wine()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
a = 0
b = 1
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[a])+' vs '+str(data.feature_names[b]))
plt.xlabel(data.feature_names[a])
plt.ylabel(data.feature_names[b])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[a,b]].values
    hullSolution = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])

    for j in range(len(hullSolution)-1):
        plt.plot((hullSolution[j][0], hullSolution[j+1][0]),
(hullSolution[j][1], hullSolution[j+1][1]), colors[i])

plt.legend()
```

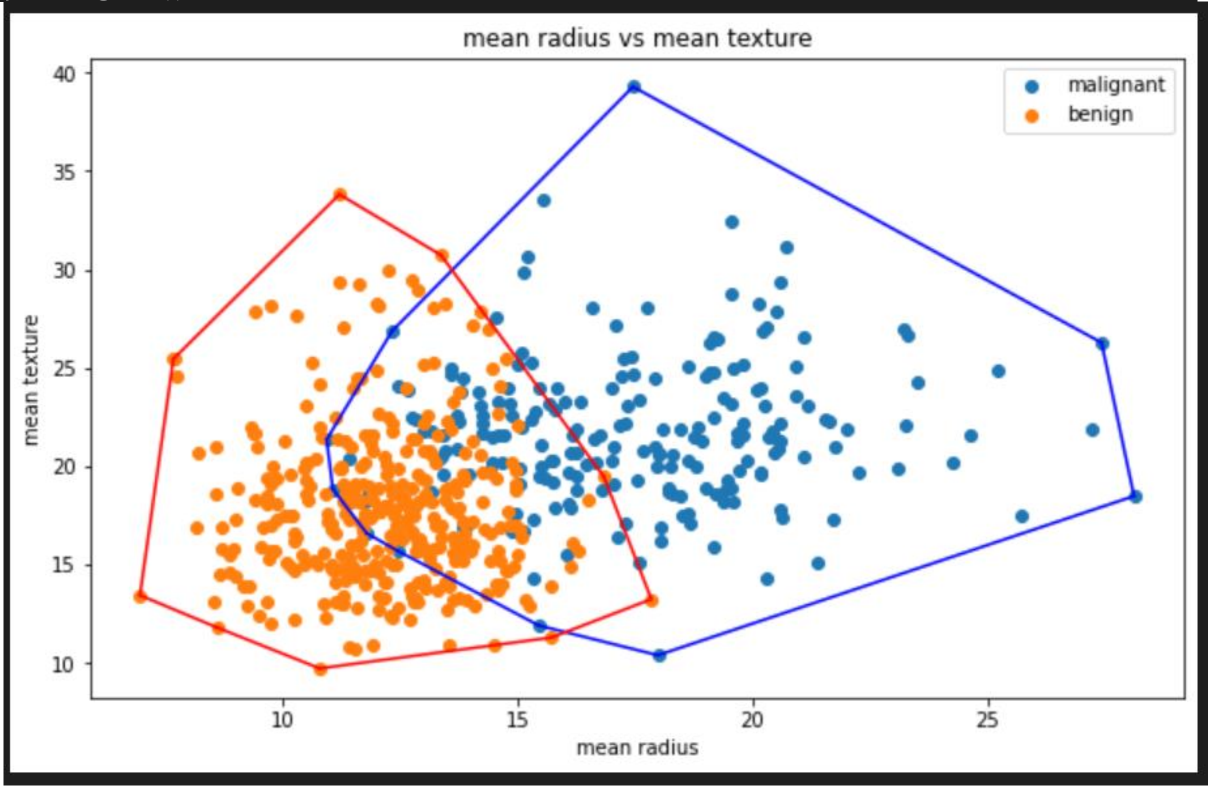


5. Mean radius vs mean texture

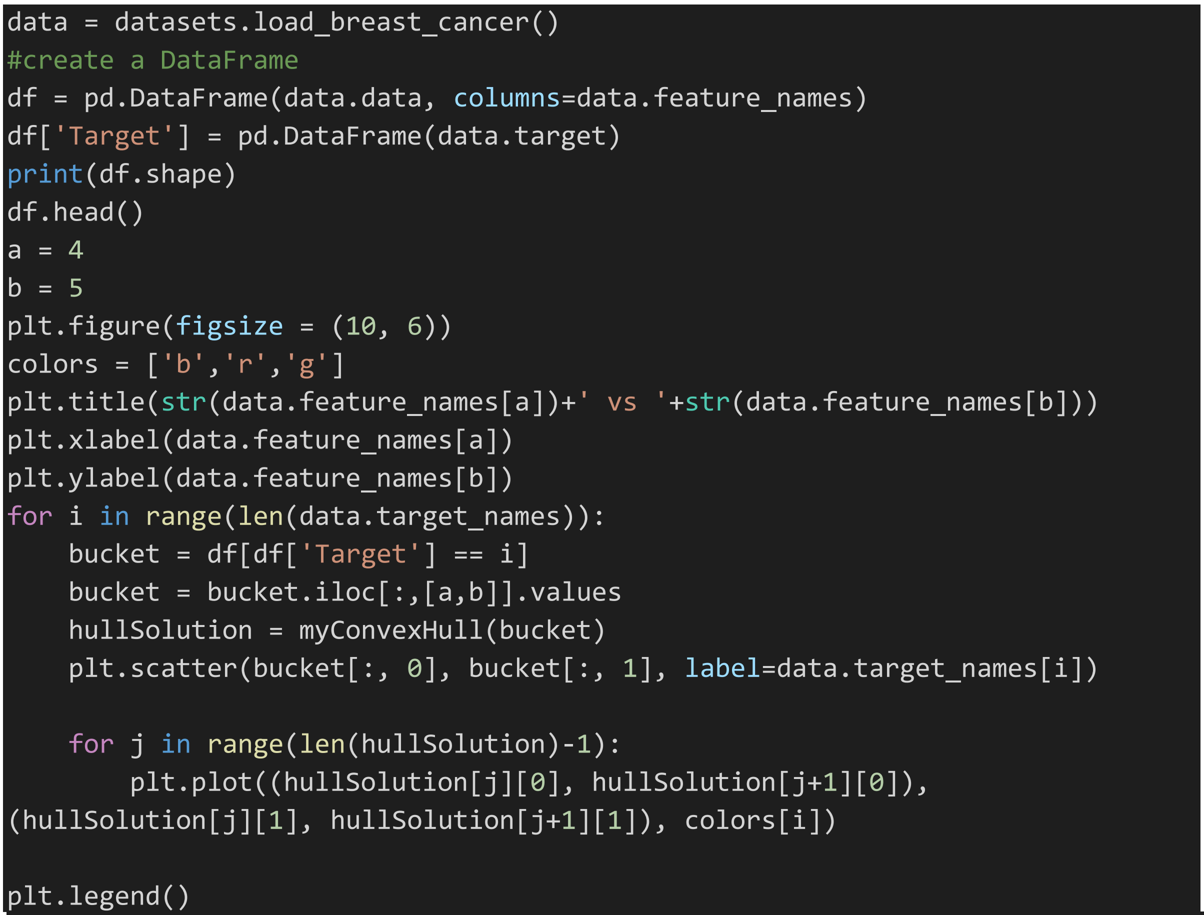
```
data = datasets.load_breast_cancer()
```



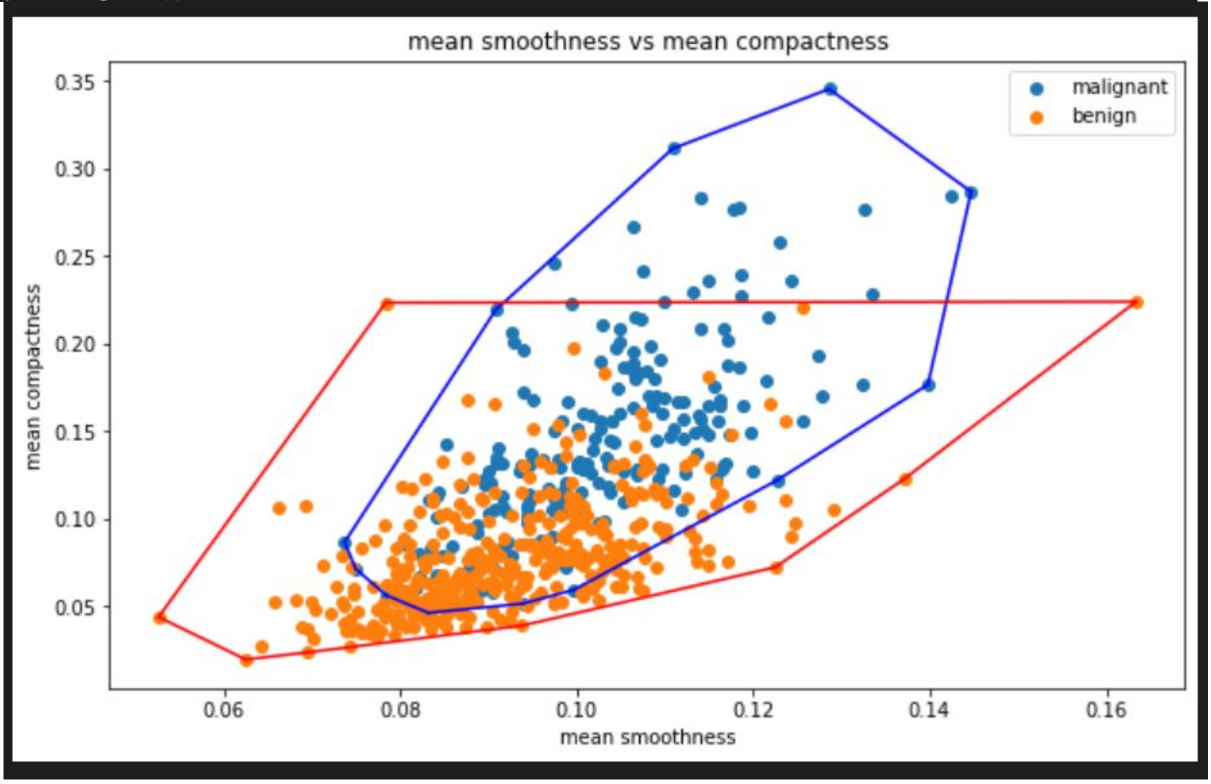
```
#create a DataFrame

df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
a = 0
b = 1
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[a])+' vs '+str(data.feature_names[b]))
plt.xlabel(data.feature_names[a])
plt.ylabel(data.feature_names[b])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[a,b]].values
    hullSolution = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])

    for j in range(len(hullSolution)-1):
        plt.plot((hullSolution[j][0], hullSolution[j+1][0]),
(hullSolution[j][1], hullSolution[j+1][1]), colors[i])

plt.legend()
```



6. Mean smoothness vs mean compactness



```
data = datasets.load_breast_cancer()
#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()
a = 4
b = 5
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title(str(data.feature_names[a])+' vs '+str(data.feature_names[b]))
plt.xlabel(data.feature_names[a])
plt.ylabel(data.feature_names[b])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[a,b]].values
    hullSolution = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])

    for j in range(len(hullSolution)-1):
        plt.plot((hullSolution[j][0], hullSolution[j+1][0]),
(hullSolution[j][1], hullSolution[j+1][1]), colors[i])

plt.legend()
```



BAB IV
PENUTUP

Link repository: https://github.com/jakartasipirok/convex-hull-with-divide-and-conquer

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

BAB V
LAMPIRAN

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