Laporan Tugas Kecil 2
IF2211 Strategi Algoritma
Semester 2 Tahun 2021/2022

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

Disusun oleh

Gerald Abraham Sianturi **13520138**

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

Link *source code*: https://github.com/geraldabrhm/TucilStima02

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. **Bonus**: program dapat menerima input dan menuliskan output untuk dataset lainnya.	✓	

Algoritma *Divide and Conquer* yang Digunakan

Algoritma yang dipakai sebagian besar mengacu pada *slide* mata kuliah IF2211 Strategi Algoritma Algoritma *Divide and Conquer* (Bagian 4). Berikut paparannya

1. Pertama, dilakukan pencarian titik terkiri dan terkanan dari sekumpulan titik-titik yang ada, misal, titik-titiknya adalah $P_1(x_1, x_2), P_2(x_2, y_2), \dots, P_n(x_n, y_n)$. Maka, akan dipilih titik dengan x_n terkecil (terkiri, misal P_a)dan titik yang lain dengan x_n terbesar (terkanan, misal P_b).
2. Mengumpulkan titik-titik yang ada pada dua daerah bagi. Kumpulan titik-titik pertama merupakan titik yang berada di atas (atau kiri) garis P_aP_b, sedangkan kumpulan titik lain sebaliknya (di bawah atau kanan garis).
3. Akan diproses kumpulan titik-titik pertama (daerah atas P_aP_b)
    a. Jika kumpulan titik-titik pertama tidak memiliki isi, maka P_a dan P_b merupakan titik-titik yang menyusun *convex hull*. Oleh karena itu, P_a dan $P_b$dimasukan ke dalam suatu NumPy Array ($arrOfConvexHull$)
    b. Jika ada isi, maka akan dilakukan pencarian titik terjauh yang ada di atas garis (P_{max}. Dan dibuat dua garis hubung, yakni P_aP_{max} dan $P_{max}P_b$. Jika ada titik-titik yang memiliki jarak sama ke garis acuan, akan dicari titik yang memaksimumkan sudut $\angle P_1P_{max}P_n$
4. Dicari titik-titik yang ada di atas P_aP_{max} dan $P_{max}P_b$. Kemudian lakukan proses poin ketiga pada kedua garis tersebut hingga kasus basis pada poin 3a terpenuhi.
5. Lakukan proses yang sama apda poin ketiga dan keempat pada kumpulan titik-titik kedua (daerah bawah P_aP_b)
6. Lakukan penyatuan hasil dari poin 3 dan 4 dengan poin 5
7. Lakukan *ploting* terhadap setiap garis yang menghubungkan tiap ujung pada *convex hull*.

Fungsi-fungsi dalam Modul myConvexHull

1. findcoefconst(point1, point2)
   Mengembalikan tiga nilai yang merupakan koefisien dan konstanta dari persamaan garis yang menghubungkan point1 dan point2. Nilai ini nanti akan digunakan pada fungsi-fungsi lainnya
2. measuredistpoints(p1, p2)
   Mengembalikan besar jarak dua titik, yakni p1 dan p2.
3. measuredist(p1, p2, pMeas)
   Mengembalikan besar jarak dari pMeas ke garis yang menghubungkan p1 dan p2
4. measureAngle (p1, p2, pRef)
   Menghitubg besar sudut $\angle P_1P_{Ref}P_2$
5. leftestRightest(arrOfPoint)
   Dari array dua dimensi, yang merupakan kumpulan titik berupa array 1 dimensi, akan dicari titik terkiri dan terkanan
6. farthestPoint(arrOfPoint, p1,p2)
   Dari sekumpulan array dua dimensi, akan diambil titik terjauh yang jaraknya ke garis yang menghubungkan p1 dan p2 merupakan yang terbesar
7. divideArea(arrOfPoint, p1, p2)

Mengembalikan kumpulan titik, masing-masing yang ada di atas garis yang menghubungkan p1 dan p2 dan di bawahnya

8. convexHullParticular(arrOfPoint, lp, rp, category)
   Mengumpulkan titik-titik *convex hull* dari array dua dimensi dengan lp dan rp adalah titik-titik ujung dari garis yang akan diproses. Category berupa string yang menerima "above" untuk mengembalikan tiap titik *convex hull* di atasnya dan "below" untuk sebaliknya
9. convexHull(arrOfPoint, lp, rp)
   Mengembalikan gabungan dari fungsi poin kesembilan dengan kategori "above" dan "below"

Kode program

1. myConvexHull.py

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import math

def findcoefconst(point1, point2):
# Determine the equation of a line through point1 and point2
    x1 = point1[0]
    y1 = point1[1]
    x2 = point2[0]
    y2 = point2[1]

    a = y2-y1
    b = -(x2-x1)
    c=(x2*y1)-(x1*y2)
    return a,b,c

def measuredistpoints(p1,p2): # Measure distance between two points
    x1 = p1[0]
    y1 = p1[1]
    x2 = p2[0]
    y2 = p2[1]

    return math.sqrt((y2-y1)**2 + (x2-x1)**2)

def measuredist(p1,p2,pMeas):
# Measure distance from a point to a line that lies on p1 p2
    x0 = pMeas[0]
    y0 = pMeas[1]
    a, b, c = findcoefconst(p1,p2)
    d = abs((a*x0)+(b*y0)+c)/(math.sqrt(a**2 + b**2))
    return d

def measureAngle(p1,p2,pRef): # Measure angle
    b = measuredistpoints(pRef, p1)
    c = measuredistpoints(pRef, p2)
    a = measuredistpoints(p1, p2)
    if(b != 0 and c != 0):
        cosAngle = (b**2+c**2-a**2)/(2*b*c)
        return math.degrees(math.acos(cosAngle))
    else:
        return 0
```

```python
def leftestRightest(arrOfPoint):
# Determine the most left and most right point in an array of points
    min = arrOfPoint[0][0]
    max = arrOfPoint[0][0]
    indeksMin = 0
    indeksMaks = 0
    for i in range(1, len(arrOfPoint)):
        if(arrOfPoint[i][0]<min):
            min = arrOfPoint[i][0]
            indeksMin = i
        if(arrOfPoint[i][0]>max):
            max = arrOfPoint[i][0]
            indeksMaks = i
    return arrOfPoint[indeksMin], arrOfPoint[indeksMaks]

'''Test case leftestRightest'''
# reskiri, reskanan = leftestRightest(arrOfPoint)
# print(f"Leftest: {reskiri}\n \nRightest: {reskanan}\n ")

def farthestPoint(arrOfPoint, p1, p2):
# Find farthest point in arrOfPoint from a line that lies on p1 p2
    max_dist = measuredist(p1, p2, arrOfPoint[0])
    indeksMaxDist = 0

    for i in range(1, len(arrOfPoint)):
        temp = measuredist(p1,p2, arrOfPoint[i])
        if(temp >= max_dist):
            if(temp == max_dist):
                angleTemp = measureAngle(p1, p2, arrOfPoint[i])
                angleCurrMax = measureAngle(p1, p2, arrOfPoint[indeksMaxDist])
                if(angleTemp >= angleCurrMax):
                    max_dist = temp
                    indeksMaxDist = i
                # else indeksMaxDist remain the same
            else: # temp > max_dist
                max_dist = temp
                indeksMaxDist = i
    return arrOfPoint[indeksMaxDist]

def divideArea(arrOfPoint, p1, p2):
# Divide area of an array of points where separate between a line that lies on p1 p2
    x1 = p1[0]
    y1 = p1[1]
    x2 = p2[0]
    y2 = p2[1]

    leftSidePoint = [] # Left(above)
    rightSidePoint = [] # Right(below)

    for i in range(len(arrOfPoint)):
        xp3 = arrOfPoint[i][0]
        yp3 = arrOfPoint[i][1]

        valDet = (x1*y2)+(xp3*y1)+(x2*yp3)-(xp3*y2)-(x2*y1)-(x1*yp3)

        if(valDet>0):
            leftSidePoint.append(arrOfPoint[i])
        elif(valDet<0):
            rightSidePoint.append(arrOfPoint[i])
    return leftSidePoint, rightSidePoint
```

```python
def convexHullParticular(arrOfPoint, lp, rp, category):
    arrOfHullPoint=np.array([])
    lp1 = lp.tolist()
    rp1 = rp.tolist()
    arrOfPoint = np.array([x for x in arrOfPoint if list(x) != lp1])
    arrOfPoint = np.array([x for x in arrOfPoint if list(x) != rp1])
    arrOfPointLeft, arrOfPointRight = divideArea(arrOfPoint, lp, rp)
    if(category == "above"):
        if(len(arrOfPointLeft) == 0):
            a = lp[0]
            b = lp[1]
            c = rp[0]
            d = rp[1]
            arrOfHullPoint = np.append(arrOfHullPoint, [[a, c],[b, d]])
        else:
            far = farthestPoint(arrOfPointLeft, lp, rp)
            list1 = convexHullParticular(arrOfPointLeft, lp, far, "above")
            list2 = convexHullParticular(arrOfPointLeft, far, rp, "above")
            arrOfHullPoint = np.append(arrOfHullPoint, list1)
            arrOfHullPoint = np.append(arrOfHullPoint, list2)
    elif(category == "below"):
        if(len(arrOfPointRight) == 0):
            a = lp[0]
            b = lp[1]
            c = rp[0]
            d = rp[1]
            arrOfHullPoint = np.append(arrOfHullPoint, [[a, c],[b, d]])
        else:
            far = farthestPoint(arrOfPointRight, lp, rp)
            list3 = convexHullParticular(arrOfPointRight, lp, far, "below")
            list4 = convexHullParticular(arrOfPointRight, far, rp, "below")
            arrOfHullPoint = np.append(arrOfHullPoint, list3)
            arrOfHullPoint = np.append(arrOfHullPoint, list4)
    arrOfHullPoint = np.reshape(arrOfHullPoint, (int(arrOfHullPoint.size/4), 2, 2))
    return arrOfHullPoint

def convexHull(arrOfPoint, lp, rp):
    resultAbove = convexHullParticular(arrOfPoint, lp, rp, "above")
    resultBelow = convexHullParticular(arrOfPoint, lp, rp, "below")
    result = np.vstack((resultAbove, resultBelow))
    return result

'''TestCase for a small-size case'''
# np.random.seed(0)
# arrOfPoint = np.random.randint(1,10,size=(10,2))
# lp, rp = leftestRightest(arrOfPoint)
# result = convexHull(arrOfPoint, lp, rp)
# print(result)

# for pair in result:
#     plt.plot(pair[0], pair[1])
# plt.show()
```

2. testing.py

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import math
from myConvexHull import *


from sklearn import datasets
data = datasets.load_wine()

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

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('flavanoids vs nonflavanoid_phenols')
plt.xlabel(data.feature_names[6])
plt.ylabel(data.feature_names[7])

for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    lp, rp = leftestRightest(bucket)
    hull = convexHull(bucket, lp, rp)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(simplex[0], simplex[1], colors[i])
plt.legend()
plt.show()
```

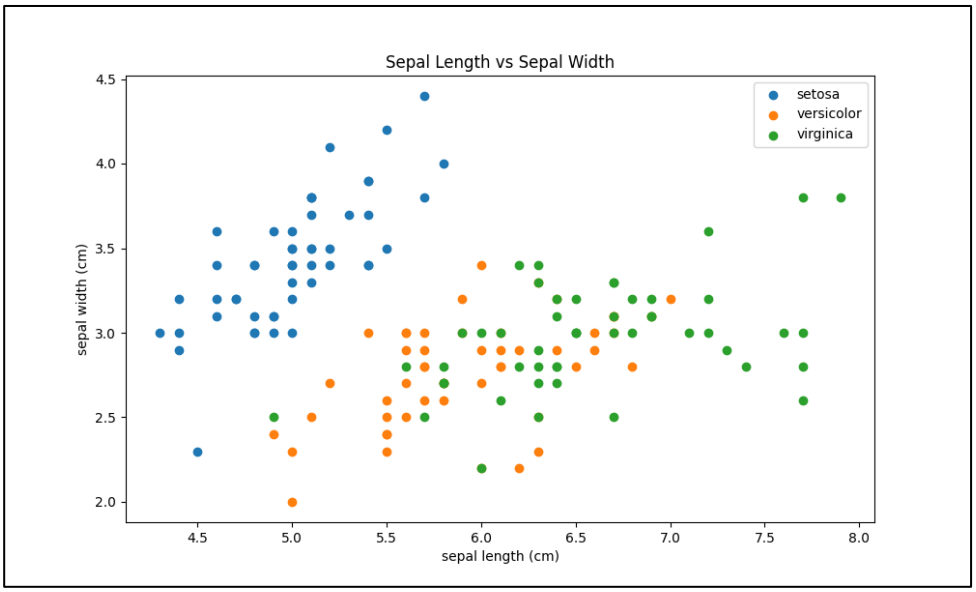
Pengetesan

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

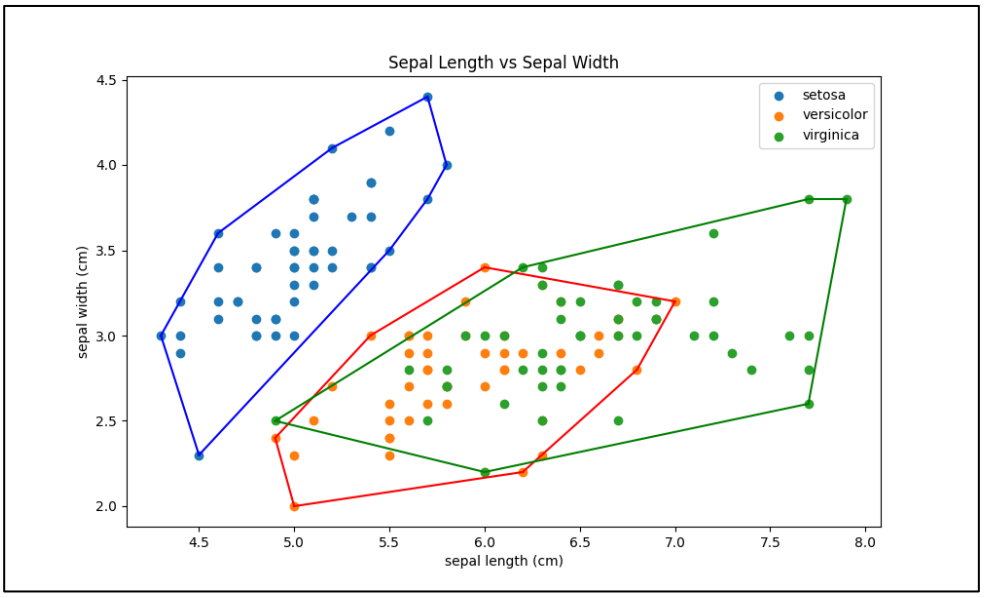
150 rows × 5 columns

Gambar 1 Data Tabular Dataset Iris

A. Sepal Length-Sepal Width

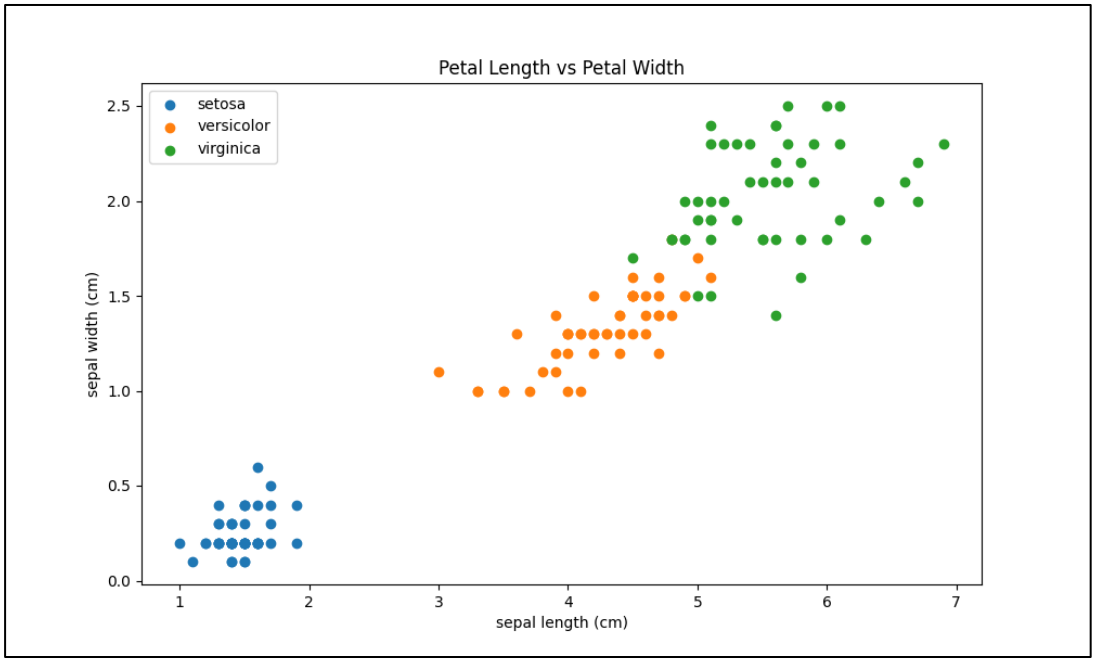


Gambar 2 Scatter Plot (Sepal Length – Sepal Width)

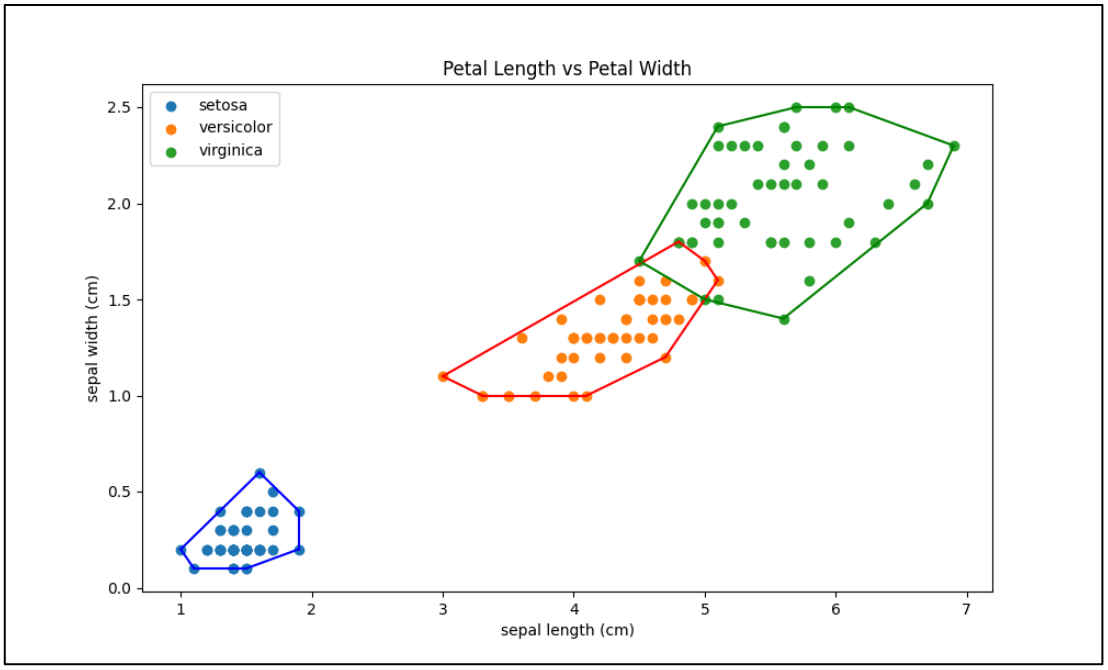


Gambar 3 Scatter Plot (Sepal Length – Sepal Width) dengan Convex Hull

B. Petal Length-Petal Width



Gambar 4 Scatter Plot (Petal Length – Petal Width)



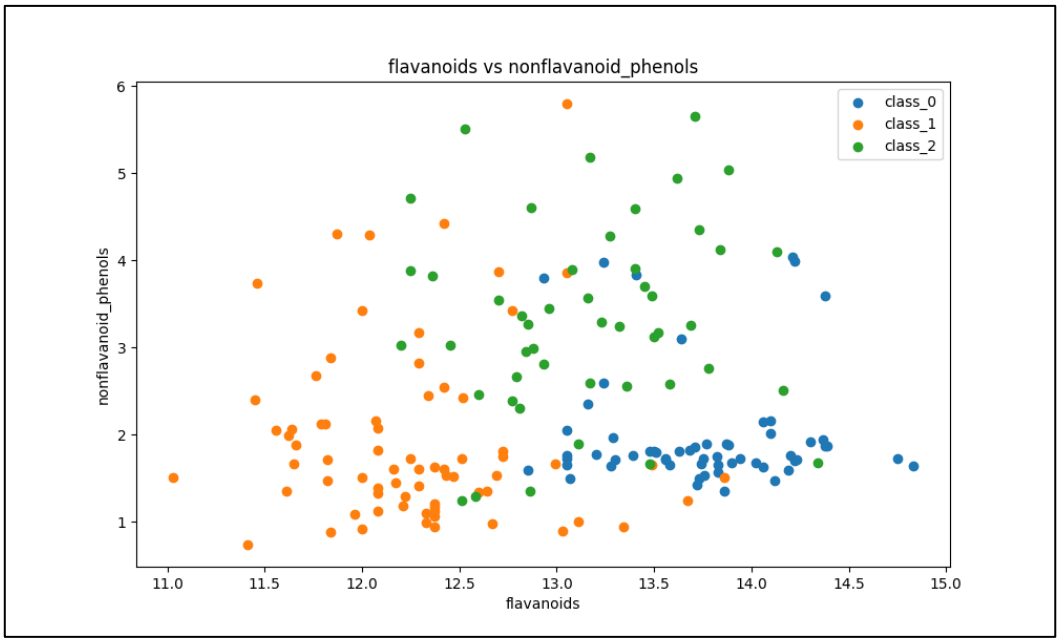
Gambar 5 Scatter Plot (Petal Length – Petal Width) dengan Convex Hull

C. Flavonoid – nonflavanoid_phenols pada *dataset wine*

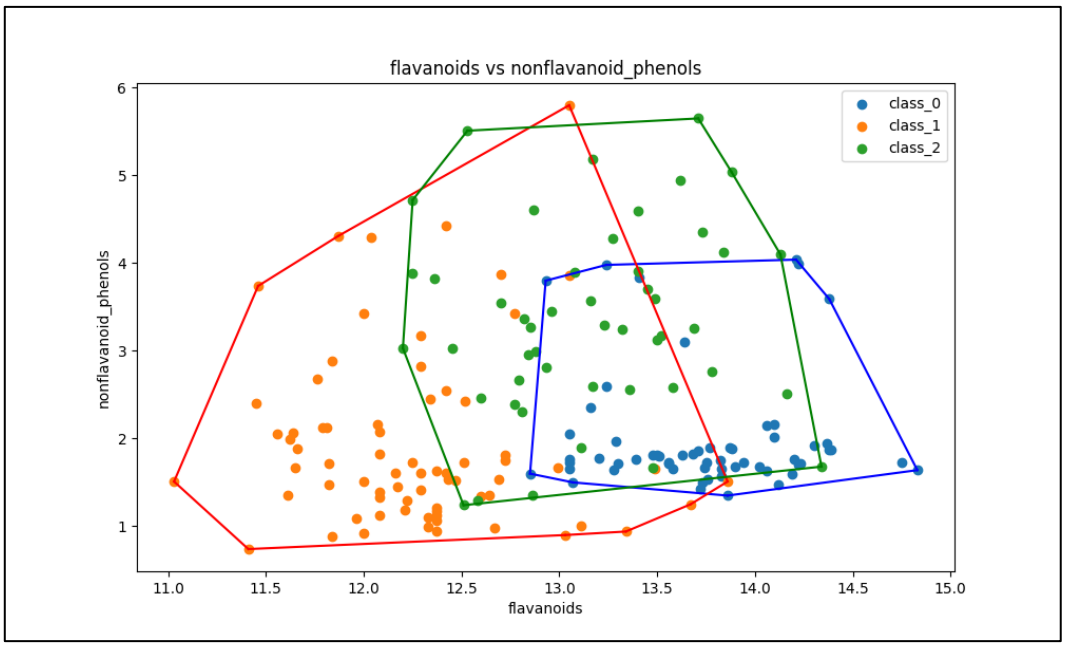
	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline	Target
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065.0	0
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050.0	0
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185.0	0
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480.0	0
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	0.52	1.06	7.70	0.64	1.74	740.0	2
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	0.43	1.41	7.30	0.70	1.56	750.0	2
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.35	10.20	0.59	1.56	835.0	2
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.46	9.30	0.60	1.62	840.0	2
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.35	9.20	0.61	1.60	560.0	2

178 rows × 14 columns

Gambar 6 Data Tabular Dataset Wine



Gambar 7 Scatter Plot (flavanoids – nonflavanoid_phenols)



Gambar 8 Scatter Plot (flavanoids – nonflavanoid_phenols) dengan Convex Hull