Tugas Kecil Strategi Algoritma – IF2210

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

Disusun oleh:

Ignasius Ferry Priguna (13520126)

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

BAB I

ALGORITMA *DIVIDE AND CONQUER*

Pada program ini, dibuat suatu pustaka Convex Hull yang memanfaatkan algoritma Divide and Conquer. Pustaka diimplementasikan dengan fungsi utama myConvexHull dan fungsi pembantu lainnya. Fungsi menerima suatu array yang berisi koordinat titik-titik data yang akan dipetakan dan mengembalikan array pasangan-pasangan index titik yang akan membentuk Convex Hull.

Pertama, array berisi koordinat titik-titik data dicek jumlah datanya. Jika hanya ada 1 atau kurang titik koordinat, fungsi tidak akan mengembalikan array kosong. Jika terdapat lebih banyak titik dari itu, titik-titik tersebut akan diurutkan berdasarkan absis nya secara ascending. Jika absis sama, titik diurutkan berdasarkan ordinatnya secara ascending. Setelah itu, setiap titik diberi nomor urut sebagai elemen ketiganya selain absis dan ordinat. Dari titik-titik yang diurutkan tesebut, diperoleh titik ekstrim kiri (titik paling kiri dalam array) dan titik ekstrim kanan (titik paling kanan dalam array).

Kemudian, dilakukan pembagian permasalahan menjadi upper convex hull dan lower convex hull yang masing-masing membawa parameter 2 titik yang akan masuk dalam convex hull dan daftar titik. Pada upper convex hull, titik yang tidak berada di atas garis yang dibentuk oleh 2 titik parameter dihapuskan dari daftar titik. Dari titik-titik yang tersisa, akan dipilih titik dengan jarak terjauh dari garis. Jika jarak sama, akan dipilih titik yang membentuk sudut terbesar. Setelah itu, proses upper convex hall akan direkursi dengan parameter berupa 1 titik dengan titik yang dipilih sebelumnya dan daftar titik. Rekursi dilakukan untuk kedua titik parameter awal sehingga persoalan akan kembali terbagi 2. Rekursi dilakukan hingga tidak ada titik di daftar titik yang berada di atas garis. Saat itu, akan dikembalikan array indeks titik dari kedua titik yang menjadi parameter. Array indeks ditambahkan pada masing-masing rekurens sehingga hasil akhirnya berupa kumpulan indeks titik yang membentuk bagian atas convex hull.

Lower convex hull dilakukan dengan cara yang mirip dengan upper convex hall. Tapi, titik yang dihapuskan dari daftar titik pada setiap rekursi adalah titik yang tidak berada di bawah garis

yang dibentuk oleh kedua titik parameter. Hasil akhir dari proses rekurens ini berupa kumpulan indeks titik yang membentuk bagian bawah convex hull.

Setelah rekurens pembentuk bagian atas dan bawah convex hull selesai, hasilnya digabungkan menjadi satu array. Array ini akan dikembalikan ke program utama dan diproses menjadi grafik dengan convex hull.

BAB II

SOURCE CODE PROGRAM

Program ditulis dalam satu *file* myConvexHull.ipynb. Program dibagi menjadi beberapa blok program yaitu *import* module, pustaka fungsi *convex hull*, serta percobaan dan demonstrasi penggunaan *myConvexHull*. Implementasi *library convex hull* direalisasikan dalam bentuk fungsi *myConvexHull*.

Import Module

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

from sklearn import datasets
```

Pustaka Fungsi myConvexHull

```
# Fungsi Pembantu myConvexHull

def isPointAbove(point1, point2, testPoint):
    # Mengembalikan True jika testPoint berada di atas garis yang
      dibentuk point1

    # Menegatifkan determinan jika point2 berada di kanan point1
    multiplier = 1
    if point2[0] > point1[0]:
        multiplier = -1

    # Mencari determinan
    A = point1[0] * point2[1]
```

```
    B = testPoint[0] * point1[1]
    C = point2[0] * testPoint[1]
    D = testPoint[0] * point2[1]
    E = point2[0] * point1[1]
    F = point1[0] * testPoint[1]
    return np.round(multiplier * (A + B + C - D - E - F),9) > 0

def isPointBelow(point1, point2, testPoint):
    # Mengembalikan True jika testPoint berada di bawah garis yang
      dibentuk point1 dan

    # Menegatifkan determinan jika point2 berada di kanan point1
    multiplier = 1
    if point2[0] > point1[0]:
        multiplier = -1

    # Mencari determinan
    A = point1[0] * point2[1]
    B = testPoint[0] * point1[1]
    C = point2[0] * testPoint[1]
    D = testPoint[0] * point2[1]
    E = point2[0] * point1[1]
    F = point1[0] * testPoint[1]
    return np.round(multiplier * (A + B + C - D - E - F),9) < 0
```

```
def distanceToPoint(linePoint1, linePoint2, point3):
    # Mengembalikan jarak antara garis yang dibentuk linePoint1 dan
      linePoint2 dengan point3
    A = linePoint2[0] - linePoint1[0]
    B = linePoint1[1] - point3[1]
    C = linePoint1[0] - point3[0]
    D = linePoint2[1] - linePoint1[1]

    if (A == 0 and D == 0) :
        return math.sqrt((B*B) + (C*C))
    else:
        return np.round(abs((A*B) - (C*D)) / math.sqrt((A*A) +
        (D*D)),9)

def angleBetweenPoint(point1, point2, point3):
    # Mengembalikan sudut yang terbentuk dari point1, point3, dan
      point2
    ABx = point3[0] - point1[0]
    BCx = point2[0] - point3[0]
    ABy = point3[1] - point1[1]
    BCy = point2[1] - point3[1]

    lenAB = math.sqrt(ABx*ABx + ABy*ABy)
    lenBC = math.sqrt(BCx*BCx + BCy*BCy)

    if (lenAB == 0 or lenBC == 0):
        return 0
    else:
```

```
        return np.round(np.arccos( (ABx*BCx + ABy*BCy) / (lenAB*lenBC)
        ),9)

def upperConvexHull(point1, point2, points):
    # Mengembalikan Convex Hull bagian atas

    # Hapus point yang tidak berada di atas garis yang dibentuk point1
    dan point2
    aboveFilter = []
    for i in points:
        if isPointAbove(point1, point2, i) and i[2] != point1[2] and
        i[2] != point2[2]:
            aboveFilter.append(True)
        else:
            aboveFilter.append(False)
    points = points[aboveFilter]

    if (len(points) == 0):
        # Tidak ada point di atas garis
        return [[point1, point2]]
    else:
        # Cari point dengan jarak terbesar dari garis yang dibentuk
          point1 dan point2
        farthestPoint = points[0]
        farthestDistance = 0
        for i in range(len(points)):
            if distanceToPoint(point1, point2, points[i]) >
farthestDistance or (distanceToPoint(point1, point2, points[i]) ==
farthestDistance and angleBetweenPoint(point1, points[i], point2) >
angleBetweenPoint(point1, farthestPoint, point2)):
```

```
                farthestPoint = points[i]
                farthestDistance = distanceToPoint(point1, point2,
points[i])
        return upperConvexHull(point1, farthestPoint, points) +
upperConvexHull(point2, farthestPoint, points)



def lowerConvexHull(point1, point2, points):
    # Mengembalikan Convex Hull bagian bawah

    # Hapus point yang tidak berada di bawah garis yang dibentuk
      point1 dan point
    lowerFilter = []
    for i in points:
        if isPointBelow(point1, point2, i) and i[2] != point1[2] and
i[2] != point2[2]:
            lowerFilter.append(True)
        else:
            lowerFilter.append(False)
    points = points[lowerFilter]

    if (len(points) == 0):
        # Tidak ada point di atas garis
        return [[point1, point2]]
    else:
        # Cari point dengan jarak terbesar dari garis yang dibentuk
          point1 dan point2
        farthestPoint = points[0]
        farthestDistance = 0
        for i in range(len(points)):
```

```
            if distanceToPoint(point1, point2, points[i]) >
farthestDistance or (distanceToPoint(point1, point2, points[i]) ==
farthestDistance and angleBetweenPoint(point1, point2, points[i]) >
angleBetweenPoint(point1, point2, farthestPoint)):
                farthestPoint = points[i]
                farthestDistance = distanceToPoint(point1, point2,
points[i])
        return lowerConvexHull(point1, farthestPoint, points) +
lowerConvexHull(point2, farthestPoint, points)


# myConvexHull Function
def myConvexHull(bucket):
    # Mengembalikan garis yang membentuk Convex Hull
    # Garis dinyatakan dengan format [[indexTitik1,indexTitik2], ....]
    # Prasyarat: di luar fungsi, bucket harus diurutkan berdasarkan
      absis dan
    # ordinat secara ascending sebelum ditampilkan dalam grafik
    if (len(bucket) > 1):
        # Cari titik ekstrim
        bucket = sorted(bucket, key=lambda x: (x[0], x[1]))
        bucket = np.array([np.append(bucket[i],i) for i in range(0,
                 len(bucket))])
        leftExtremePoint = bucket[0]
        rightExtremePoint = bucket[-1]

        # Convex Hull bagian atas
        upperConvexHullPoints =
np.array(upperConvexHull(leftExtremePoint, rightExtremePoint, bucket))
        # Convex Hull bagian bawah
        lowerConvexHullPoints =
np.array(lowerConvexHull(leftExtremePoint, rightExtremePoint, bucket))
```

```
        # Gabung Convex Hull bagian atas dan bagian bawah
        return np.concatenate((upperConvexHullPoints[:,:,2],
lowerConvexHullPoints[:,:,2]), axis=0).astype('int32')
    else:
        # Jika jumlah titik <= 1, mengembalikan array kosong
        return []
```

Percobaan dan Demonstrasi Penggunaan myConvexHull

Dataset Iris

```
# Load iris dataset
data = datasets.load_iris()

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

# Visualisasi hasil ConvexHull dataset iris Petal Width vs Petal
Length
plt.figure(figsize = (10, 6))
colors = ['b','r','g']

plt.title('Petal Width vs Petal Length')
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])

for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[2,3]].values
    hull = myConvexHull(bucket)
```

```
    bucket = np.array(sorted(bucket, key=lambda x: (x[0], x[1])))
    plt.scatter(bucket[:, 0], bucket[:, 1],
label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])

plt.legend()

# Visualisasi hasil ConvexHull dataset iris Sepal Width vs Sepal
Length
plt.figure(figsize = (10, 6))
colors = ['b','r','g']

plt.title('Sepal Width vs Sepal Length')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])

for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    hull = myConvexHull(bucket)
    bucket = np.array(sorted(bucket, key=lambda x: (x[0], x[1])))
    plt.scatter(bucket[:, 0], bucket[:, 1],
label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])

plt.legend()
```

Dataset Digits

```
# Load digits dataset

digitsData = datasets.load_digits()


digitsDf = pd.DataFrame(digitsData.data,
columns=digitsData.feature_names)

digitsDf['Target'] = pd.DataFrame(digitsData.target)

digitsDf.head()


# Visualisasi hasil ConvexHull dataset digits


plt.figure(figsize = (10, 6))

colors =
['b','r','g','c','m','y','darkgoldenrod','lime','salmon','violet']


plt.title('Pixel 02 vs Pixel 03')

plt.xlabel(digitsData.feature_names[3])

plt.ylabel(digitsData.feature_names[2])


for i in range(len(digitsData.target_names)):

    bucket = digitsDf[digitsDf['Target'] == i]

    bucket = bucket.iloc[:,[3,2]].values

    hull = myConvexHull(bucket)

    bucket = np.array(sorted(bucket, key=lambda x: (x[0], x[1])))

    plt.scatter(bucket[:, 0], bucket[:, 1],
label=digitsData.target_names[i])

    for simplex in hull:

       plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])


plt.legend()
```

Dataset Wine

```
# Load wine dataset
wineData = datasets.load_wine()

wineDf = pd.DataFrame(wineData.data, columns=wineData.feature_names)
wineDf['Target'] = pd.DataFrame(wineData.target)
wineDf.head()

# Visualisasi hasil ConvexHull dataset wine Malic Acid vs Alcohol
plt.figure(figsize = (10, 6))
colors = ['b','r','g']

plt.title('Malic Acid vs Alcohol')
plt.xlabel(wineData.feature_names[0])
plt.ylabel(wineData.feature_names[1])

for i in range(len(wineData.target_names)):
    bucket = wineDf[wineDf['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    hull = myConvexHull(bucket)
    bucket = np.array(sorted(bucket, key=lambda x: (x[0], x[1])))
    plt.scatter(bucket[:, 0], bucket[:, 1],
label=wineData.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])

plt.legend()
```

```
# Visualisasi hasil ConvexHull dataset wine Hue vs Intensity
plt.figure(figsize = (10, 6))
colors = ['b','r','g']

plt.title('Hue vs Intensity')
plt.xlabel(wineData.feature_names[9])
plt.ylabel(wineData.feature_names[10])

for i in range(len(wineData.target_names)):
    bucket = wineDf[wineDf['Target'] == i]
    bucket = bucket.iloc[:,[9,10]].values
    hull = myConvexHull(bucket)
    bucket = np.array(sorted(bucket, key=lambda x: (x[0], x[1])))
    plt.scatter(bucket[:, 0], bucket[:, 1],
label=wineData.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])

plt.legend()
```

Dataset Breast Cancer

```
# Load breast cancer dataset
bcData = datasets.load_breast_cancer()

bcDf = pd.DataFrame(bcData.data, columns=bcData.feature_names)
bcDf['Target'] = pd.DataFrame(bcData.target)
bcDf.head()
```

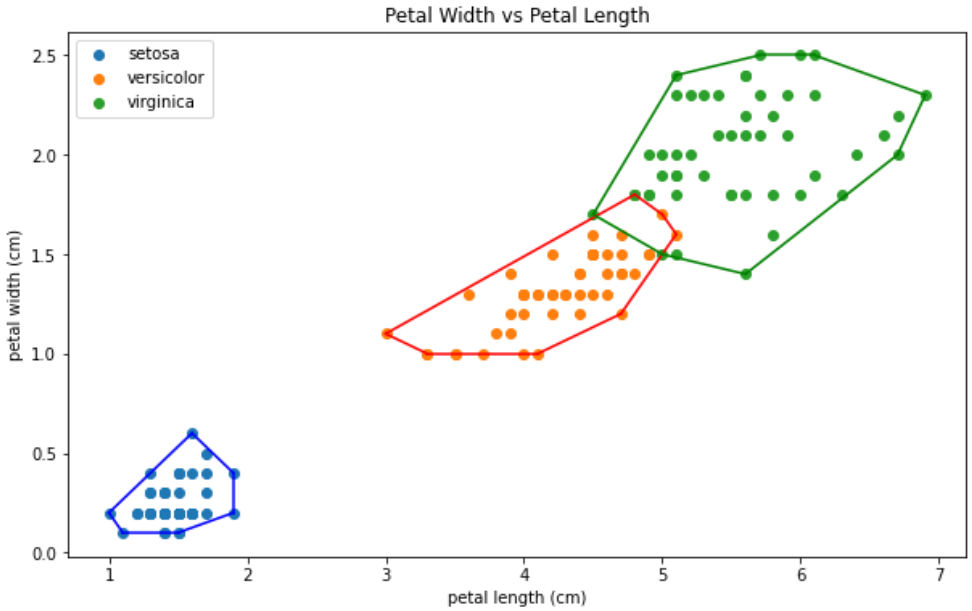
```
# Visualisasi hasil ConvexHull dataset breast cancer Mean Smoothness
vs Mean Texture

plt.figure(figsize = (10, 6))

colors = ['b','r']


plt.title('Mean Smoothness vs Mean Texture')

plt.xlabel(bcData.feature_names[1])

plt.ylabel(bcData.feature_names[4])


for i in range(len(bcData.target_names)):

    bucket = bcDf[bcDf['Target'] == i]

    bucket = bucket.iloc[:,[1,4]].values

    hull = myConvexHull(bucket)

    bucket = np.array(sorted(bucket, key=lambda x: (x[0], x[1])))

    plt.scatter(bucket[:, 0], bucket[:, 1],
label=bcData.target_names[i])

    for simplex in hull:

        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])


plt.legend()
```

BAB III

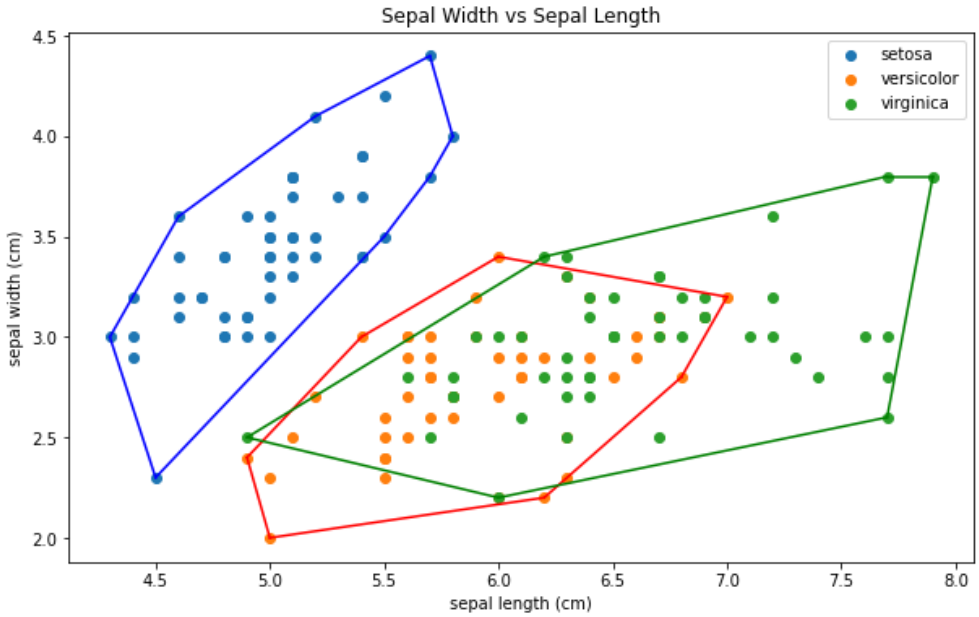
EKSPERIMEN

Seluruh dataset yang digunakan dalam eksperimen diambil dari *toy datasets* scikit learn

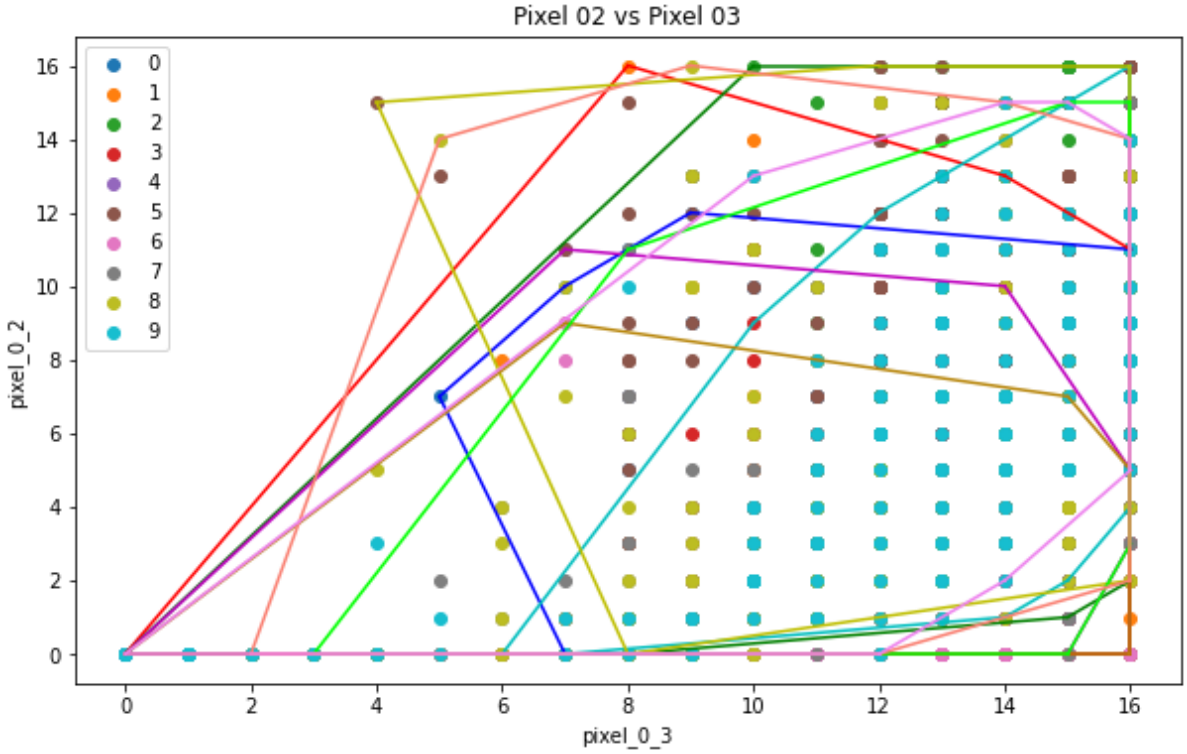
1. Eksperimen 1 : Dataset Iris – Petal Width vs Petal Length



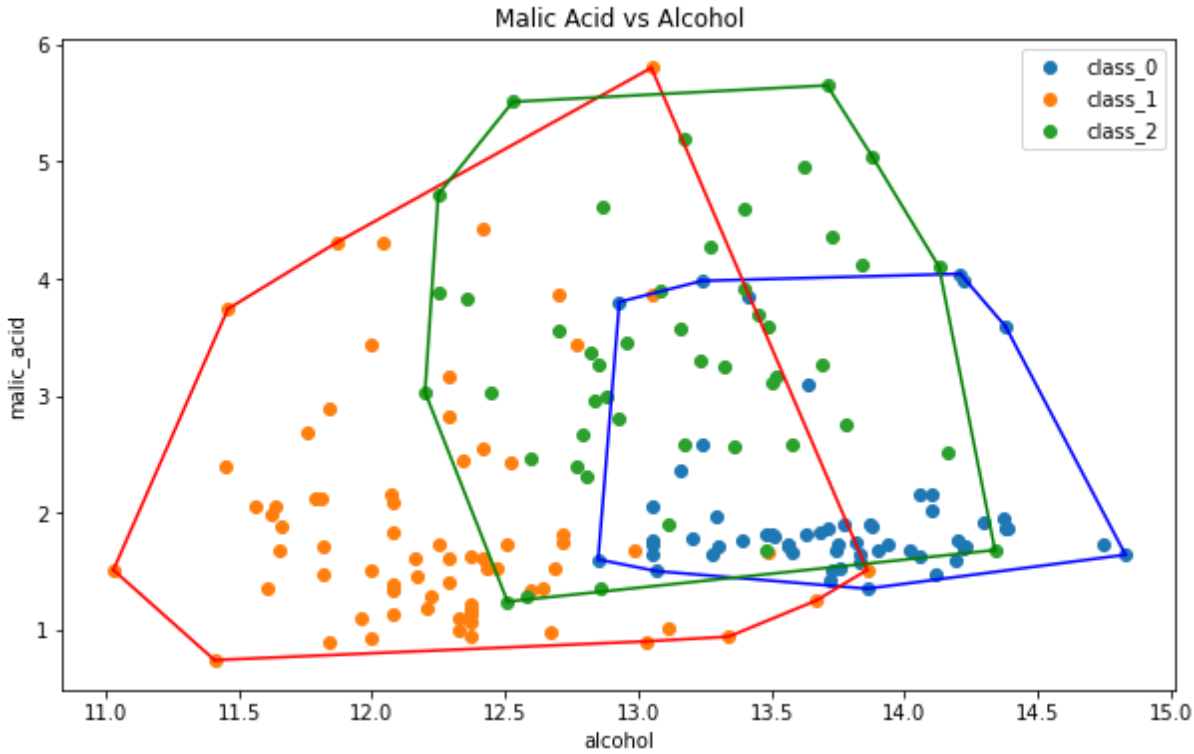
2. Eksperimen 2 : Dataset Iris – Sepal Width vs Sepal Length



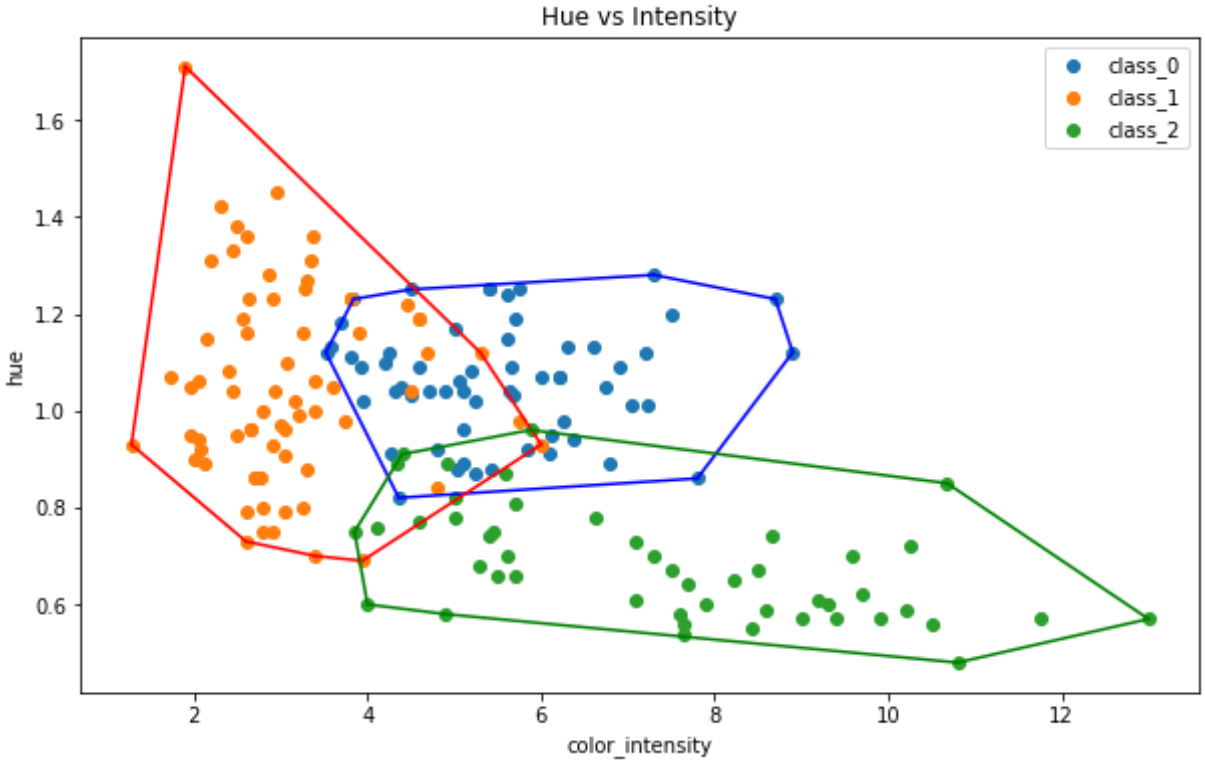
3. Eksperimen 3 : Dataset Digits – Pixel 02 vs Pixel 03



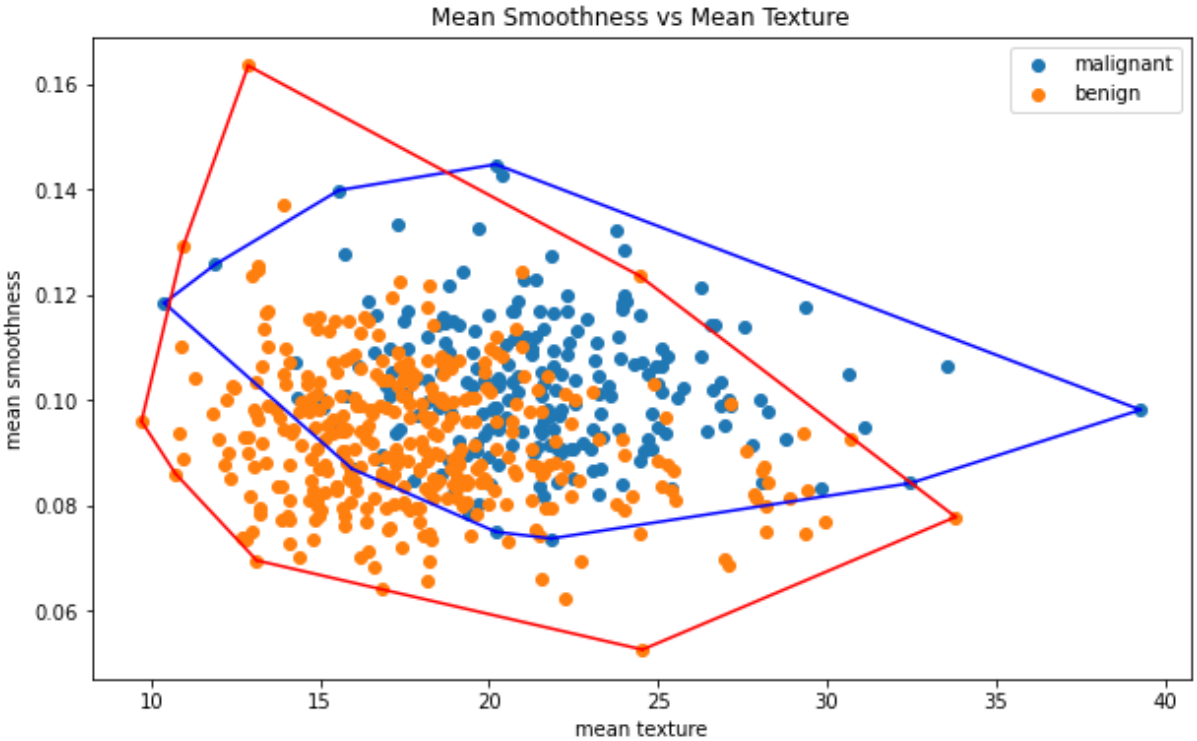
4. Eksperimen 4 : Dataset Wine – Malic Acid vs Alcohol



5. Eksperimen 5 : Dataset Wine – Hue vs Intensity



6. Eksperimen 6 : Dataset Breast Cancer – Mean Smoothness vs Mean Texture



LINK DRIVE DAN GITHUB

1, Link Drive

https://drive.google.com/drive/folders/1i3o845gb7cb5OW3V89RzfvxtgANmdLzU?usp=sharing

2. Link Github

https://github.com/ignferry/Tucil-2-Stima-Convex-Hull

CEK LIST

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..	✓	