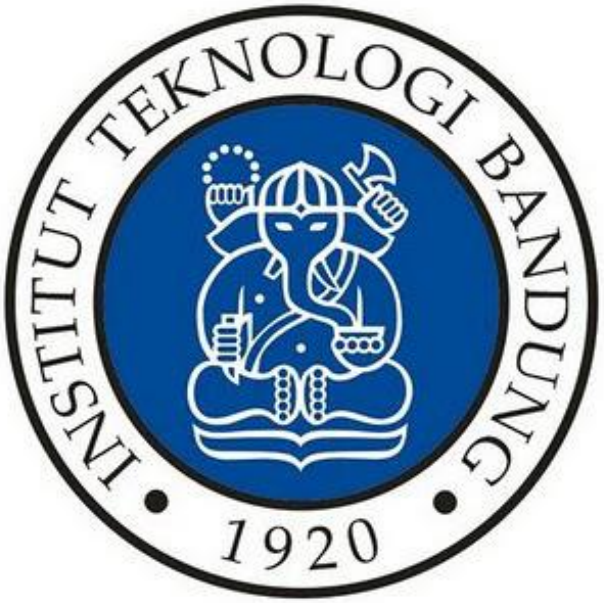
Tugas Kecil 2

IF2211 - Implementasi Convex Hull untuk Visualisasi Tes Linear Separability Datasetdengan Algoritma Divide and Conquer



Oleh

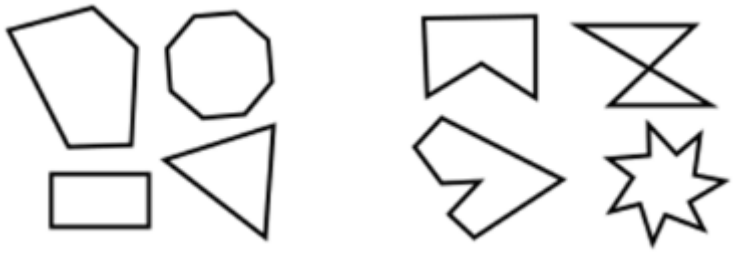
Bayu Samudra
13520128

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021

Daftar Isi

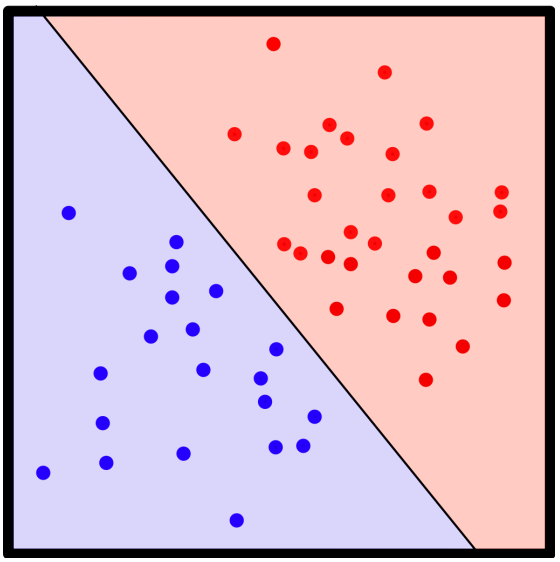
Bab 1
Pendahuluan

Convex Hull merupakan himpunan titik terkecil yang berisi kumpulan-kumpulan titik yang bersifat *convex*. Himpunan titik dikatakan convex apabila untuk sembarang dua buah titik yang berada pada himpunan tersebut, seluruh segmen garis yang berakhir pada dua titik tersebut berada pada himpunan tersebut. Berikut ini adalah ilustrasi mengenai poligon yang convex dan tidak.



Gambar 1.1. Gambar kiri merupakan gambar poligon yang konveks dan kanan bukan merupakan poligon yang konveks. (Sumber: Slide Kuliah)

Salah satu aplikasi dari convex hull adalah untuk melihat visualisasi dari tes *Linear Separability Dataset*. Dua buah himpunan dikatakan *linear separable* apabila terdapat sebuah garis yang dapat memisahkan dua buah dataset. Berikut ini adalah ilustrasinya



Gambar 1.2. Ilustrasi himpunan yang *Linear Separable* (Sumber : Wikipedia)

Salah satu cara untuk memvisualisasikan dua buah dataset ini *Linear Separable*, dapat digunakan convex hull dari dataset tersebut.

Pada tugas kecil kali ini, perlu diimplementasikan sebuah pustaka yang dapat mengembalikan *convex hull* dari kumpulan titik pada 2 dimensi. Tugas ini perlu dibuat dalam bahasa Python dan untuk pengujian perlu menggunakan dataset iris. Strategi algoritma yang perlu digunakan adalah *Divide and Conquer.*

Bab 2
Deskripsi Algoritma

Pada tugas ini, saya menggunakan algoritma *Divide and Conquer*. Fungsi yang mencari *convex hull* adalah `myconvexhull`. Fungsi ini memerlukan beberapa fungsi pembantu yaitu fungsi yang berperan sebagai pengurut data, penyortir data, penghitung jarak ke titik, dan fungsi yang dapat mencari titik terjauh dari sebuah dataset titik ke sebuah garis.

Langkah 1

Pada mulanya, fungsi menerima data berupa titik-titik yang akan dicari convex hull-nya. Titik ini pada mulanya dilakukan pengurutan dari kecil ke terbesar dengan aturan perbandingan sebagai berikut:

1. Titik dikatakan lebih besar dari pada titik lainnya apabila titik tersebut memiliki absis yang lebih besar
2. Apabila dua buah titik memiliki absis yang sama besar, dicari manakah diantara keduanya yang memiliki ordinat yang lebih besar.
3. Apabila titik memiliki absis dan ordinat sama besar, maka keduanya disebut memiliki sama besar

Setelah diurutkan, diambil dua buah titik yang paling kecil dan besar. Lalu dijadikan sebuah garis. Setelah itu, dipisahkan dua garis yang berada pada di atas garis dan di bawah garis. Letak sebuah titik terhadap garis dapat dicari dengan menghitung outer product dari titik beserta dua titik yang membentuk garis. Misalkan garis melewati titik (x_a, y_a) dan (x_b, y_b). Diketahui sebuah titik memiliki koordinat (x_p, y_p). Letak garis dapat dihitung dengan rumus berikut:

$$L = \begin{vmatrix} x_p & y_p & 1 \\ x_a & y_a & 1 \\ x_b & y_b & 1 \end{vmatrix}$$

Apabila nilai L > 0, titik berada di atas garis. Akan tetapi, bila L = 0, titik berada tepat pada garis. Pada L < 0, titik berada di bawah garis. Untuk setiap titik yang berada di atas garis, lakukan langkah 2. Untuk tiap titik yang berada di bawah garis lakukan langkah 3.

Langkah 2

Bila tidak terdapat titik atau hanya ada satu titik, itu merupakan *convex hull*. Jika tidak, Carilah titik yang memiliki jarak paling jauh dari garis. Bila terdapat dua titik yang jaraknya paling jauh, carilah titik yang lebih dekat terhadap titik (x_a, y_a). Hal ini dapat dihitung berdasarkan sudut yang dibentuknya. Titik terjauh ini merupakan anggota convex hull.

Setelah ditentukan titik terjauh misalkan titik p, buatlah garis dimulai dari awal titik ke p. Setelah itu, buatlah garis yang dimulai pada titik p ke titik akhir. Untuk garis baru pertama, ambil semua titik yang berada di atas garis itu dan tentukan *convex hull*-nya. Untuk garis baru kedua, tentukan titik yang berada di atas garis itu. Tentukan *convex hull*-nya. Pencarian konveks hull bisa kembali ke langkah 2.

Proses penggabungan dari tiap-tiap *convex hull* diawali dari hasil *convex hull* kiri ditambahkan dengan elemen titik maksimum dan digabung dengan *convex hull* kanan.

Langkah 3

Bila tidak terdapat titik atau hanya ada satu titik, itu merupakan *convex hull*. Jika tidak, Hal ini sama seperti pada langkah kedua. Setelah ditemukannya titik terjauh misalkan q, buatlah garis dimulai dari titik awal ke q dan garis yang melalui q ke titik akhir. Untuk setiap titik yang berada dibawah garis, dicari *convex hull*-nya. Cara pencariannya sama sebagaimana Langkah 3 ini (secara rekursif).

Proses penggabungan dari tiap *convex hull* yang didapat adalah menggabungkan hasil *convex hull* kanan dengan titik terjauh lalu digabungkan dengan convex hull kiri.

Langkah 4

Bila telah didapatkan *convex hull* dari langkah 2 dan 3, gabungkanlah titik terjauh kiri dengan hasil langkah 2 dilanjutkan dengan titik terjauh kanan dan diakhiri menggabungkan dengan hasil langkah 3. Itu merupakan *convex hull* total.

Bab 3

Kode Program

Berikut ini adalah kode program dari implementasi kode ini

```
# Library myConvexHull
# Mencari Convex Hull dari larik titik dua dimensi

import numpy as np

from util.line import max_point, point_position
from util.quicksort import quicksort
from util.compare import compare_point
import matplotlib.pyplot as plt

def myConvexHull(points: np.ndarray) -> list:
 """Mencari ConvexHull menggunakan Strategi DnC"""
 upper = []
 lower = []

 pointSorted = np.copy(points)
 quicksort(pointSorted, compare_point)
 line = (pointSorted[0], pointSorted[-1])

 for i in pointSorted:
   if(point_position(i, line) > 0):
     upper.append(i)
   elif(point_position(i,line) < 0):
     lower.append(i)

 upperPartition = convexHullPartition(upper, line)
 lowerPartition = convexHullPartition(lower, line, True)

 result = [pointSorted[0]] + upperPartition + [pointSorted[-1]] +
lowerPartition
 return result

def convexHullPartition(points: list, anchor: tuple, isBottom: bool =
False) -> list:
 """Partisi untuk perhitungan convexhull"""
 if len(points) <= 1:
```

```
    return points

  point = max_point(points, anchor)
  lineleft = (anchor[0], point)
  lineright = (point, anchor[1])
  upleft = []
  upright = []

  for i in points:
    if isBottom:
      if point_position(i, lineleft) < 0:
        upleft.append(i)
      elif point_position(i, lineright) < 0:
        upright.append(i)
    else:
      if point_position(i, lineleft) > 0:
        upleft.append(i)
      elif point_position(i, lineright) > 0:
        upright.append(i)

  leftCH = convexHullPartition(upleft, lineleft, isBottom)
  rightCH = convexHullPartition(upright, lineright, isBottom)

  result = []
  if isBottom:
    result = rightCH + [point] + leftCH
  else:
    result = leftCH + [point] + rightCH
  return result
```

Berikut ini adalah utilitas yang digunakan

```
# Quicksort
# Implementasi Quicksort

from pyclbr import Function
import numpy as np

def partition(arr: np.ndarray, comparator: Function) -> int:
  """Melakukan Partisi pada tiap state dari quicksort"""
  if len(arr) == 0:
```

```
    return 0

  p = 1
  q = len(arr) -1

  while(p <= q):
    while(p < len(arr) and comparator(arr[p],arr[0]) < 0):
      p += 1

    while(q > 0 and comparator(arr[q],arr[0]) > 0):
      q -= 1

    if p <= q:
      tmp = np.copy(arr[p])
      arr[p] = np.copy(arr[q])
      arr[q] = tmp
      p += 1
      q -= 1

  tmp = np.copy(arr[0])
  arr[0] = np.copy(arr[q])
  arr[q] = tmp
  return q

def quicksort(arr: np.ndarray, comparator: Function) -> None:
  """Pengurutan array menggunakan quicksort"""
  if len(arr) <= 1:
    return
   idx = partition(arr, comparator)
  quicksort(arr[:idx], comparator)
  quicksort(arr[idx+1:], comparator)

# Line Utility
# Utilitas yang berkaitan dengan garis dan titik

import numpy as np

def point_distance(p: np.ndarray, line:tuple[np.ndarray,np.ndarray]):
  """Menghitung jarak titik ke garis"""
  a = p - line[0]
  b = line[1] - line[0]
```

```
 factor = np.sum(a * b)/np.sum(b * b)
 v = a - factor * b

 return np.sqrt(np.sum(v * v))

def point_cosine(p: np.ndarray, line:tuple[np.ndarray,np.ndarray]):
 """Menghitung cosinus dari garis"""
 a = p - line[0]
 b = line[1] - line[0]
 na = np.sum(a * a)
 nb = np.sum(b * b)

 return np.sum(a * b), na * nb

def point_position(p: np.ndarray, line:tuple[np.ndarray,np.ndarray]):
 """Fungsi untuk menentukan posisi titik
  Menghasilkan > 0 jika diatas garis, = 0 jika tepat pada garis, dan
 < 0 jika dibawah garis
 """
  return (p[1] * line[1][0] + p[0] * line[0][1] + line[0][0] * line[1][1])
- \
     (p[0] * line[1][1] + line[0][0] * p[1] + line[0][1] * line[1][0])

def max_point(points: list, line: tuple) -> np.ndarray:
 """Mencari Titik dengan jarak terjauh dan sudut terbesar"""
 maxPoint = points[0]
 maxDist = point_distance(maxPoint, line)

 for i in points:
   dist = point_distance(i, line)

   if maxDist < dist:
     maxDist = dist
     maxPoint = i
   elif maxDist == dist:
     curCosA, curCosB = point_cosine(maxPoint, line)
     iCosA, iCosB = point_cosine(i, line)

     if(curCosA * iCosB < iCosA * curCosB):
       # Sudut lebih besar
```

```
        maxPoint = i
        maxDist = dist
  return maxPoint

# Graph Utility
# Utilitas untuk menampilkan grafik hasil ConvexHull

import matplotlib.pyplot as plt

def show_graph(points, convexhull, color, label):
 """Show result in graph"""
 plt.scatter(points[:,0], points[:,1], c=color, label=label)
 x = []
 y = []

 for i in convexhull:
   x.append(i[0])
   y.append(i[1])
  plt.plot(x, y, color)
 plt.plot([convexhull[-1][0],convexhull[0][0]],
           [convexhull[-1][1],convexhull[0][1]],
           color)

# Fungsi Compare
# Utilitas untuk komparasi

import numpy as np

def compare_point(p1: np.ndarray, p2: np.ndarray):
 """Membandingkan dua buah titik"""
 if(p1[0] == p2[0]):
   return p1[1] - p2[1]
 else:
   return p1[0] - p2[0]
```

Untuk menampilkan citra yang memvisualisasikan hasil, dapat digunakan kode sebagai berikut

```
from myConvexHull import myConvexHull
import matplotlib.pyplot as plt
import pandas as pd
```

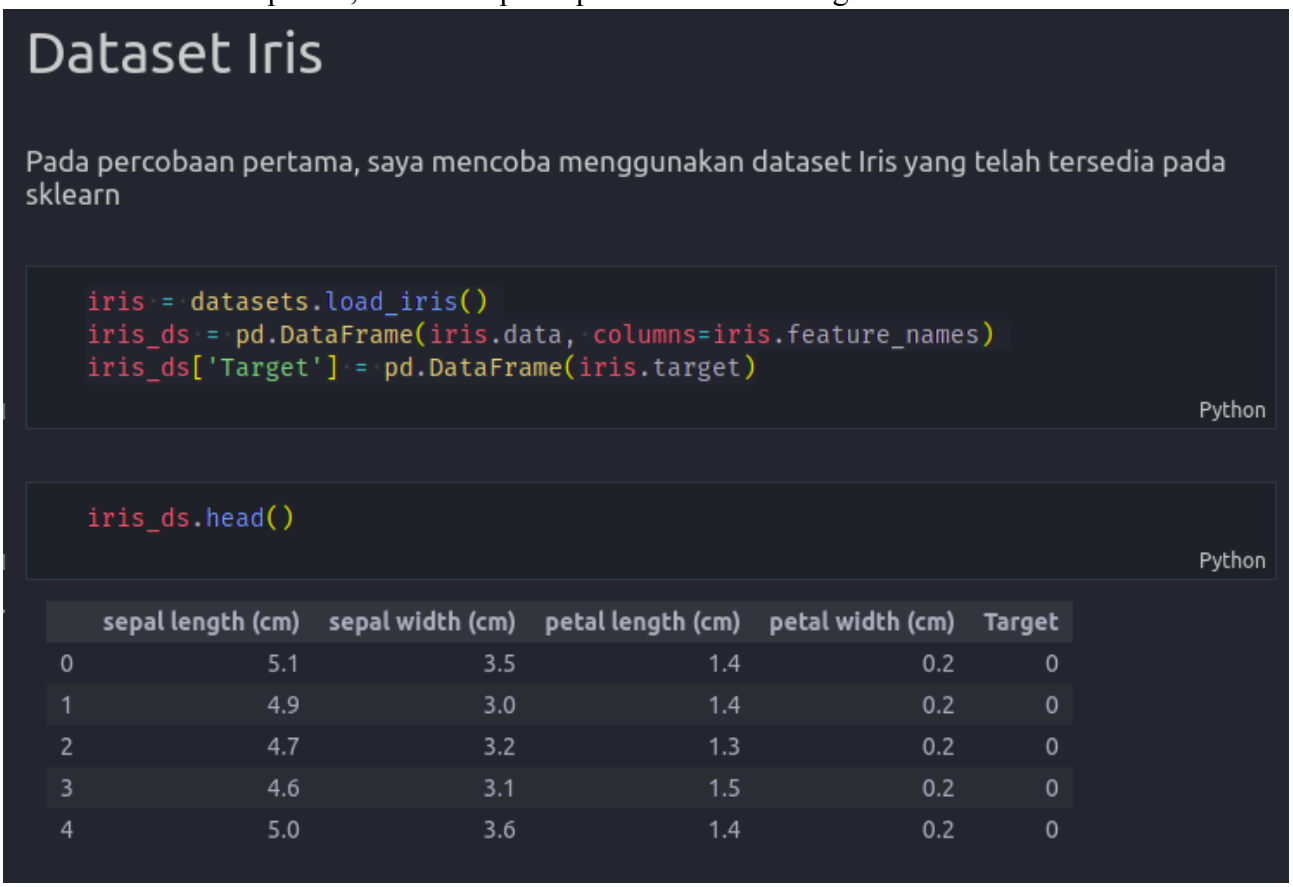
```
from sklearn import datasets
from util.graph import show_graph

# Ambil dataset
iris = datasets.load_iris()
iris_ds = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_ds['Target'] = pd.DataFrame(iris.target)

# Ilustrasikan hasil
plt.figure(figsize=(10,6))
colors = ['b','r','g']

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

for i in range(len(iris.target_names)):
 data = iris_ds[iris_ds['Target'] == i]
 data = data.iloc[:,[0,1]].values
 hull = myConvexHull(data)

 show_graph(data, hull, colors[i], iris.target_names[i])

plt.legend()
```

Bab 4
Hasil Eksekusi Program

4.1. Dataset Iris

4.1.1. Persiapan Data

Sebelum melakukan proses, dilakukan persiapan dataset iris sebagai berikut

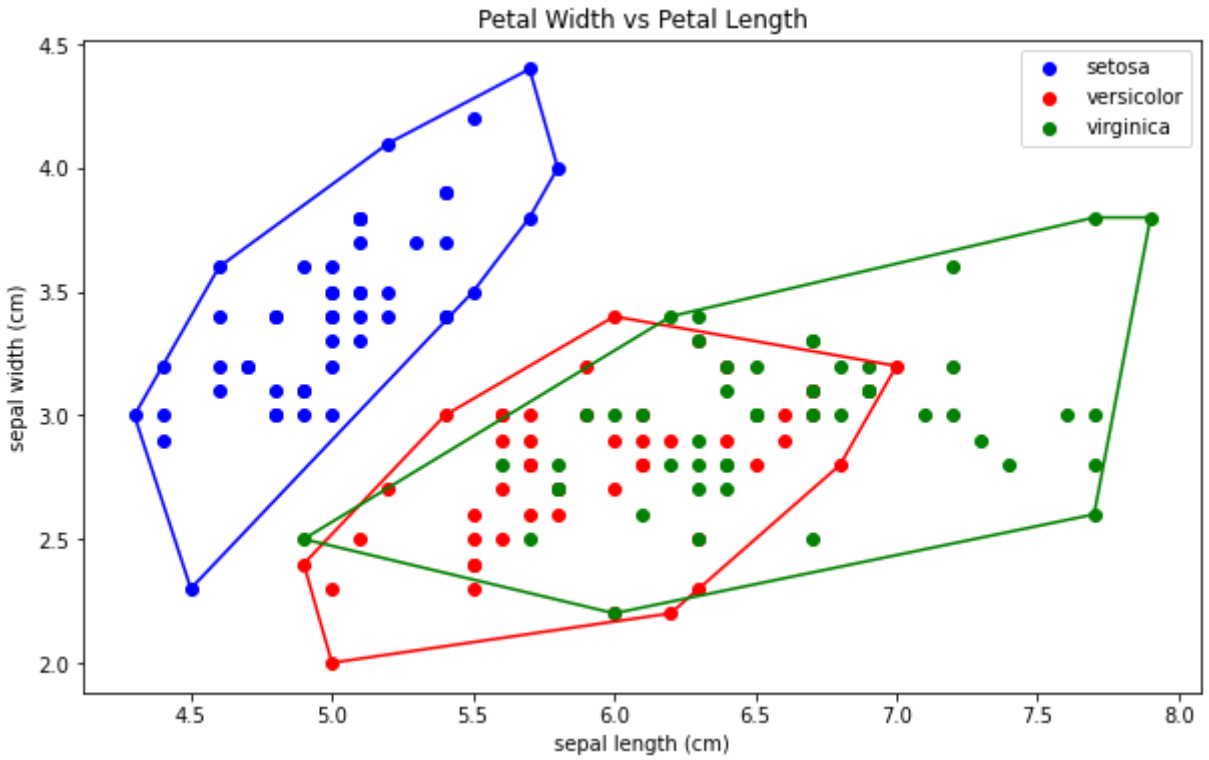


	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

Gambar 4.1. Persiapan Data

4.1.2. Hubungan *Petal Length* dengan *Petal Width*

Berikut ini adalah hasil dari ilustrasi program `myConvexHull`



Gambar 4.2. Hubungan Petal Length dan Petal Width

Dari hasil diatas terlihat bahwa Iris Versicolor dan Iris Virginica saling bersisian satu sama lain. Oleh karena itu, cukup sulit untuk mengklasifikasikan antara keduanya. Akan tetapi dengan Iris Sentosa, cukup jelas perbedaannya sehingga lebih mudah untuk dibuat pembatasnya. Berikut kode eksekusinya

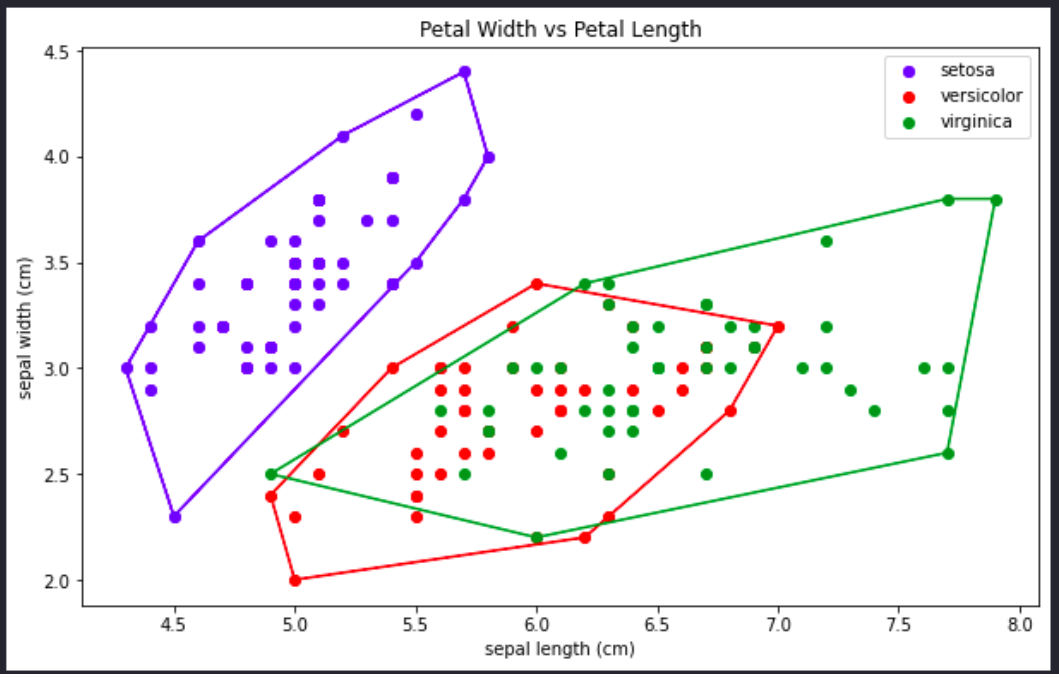
Hubungan Petal Length dan Petal Width

Pada bagian ini, saya mencoba untuk mengilustrasikan petal length dan width

```
plt.figure(figsize=(10,6))
colors = ['b','r','g']

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

for i in range(len(iris.target_names)):
    data = iris_ds[iris_ds['Target'] == i]
    data = data.iloc[:,[0,1]].values
    hull = myConvexHull(data)

    show_graph(data, hull, colors[i], iris.target_names[i])

plt.legend()
```

Python

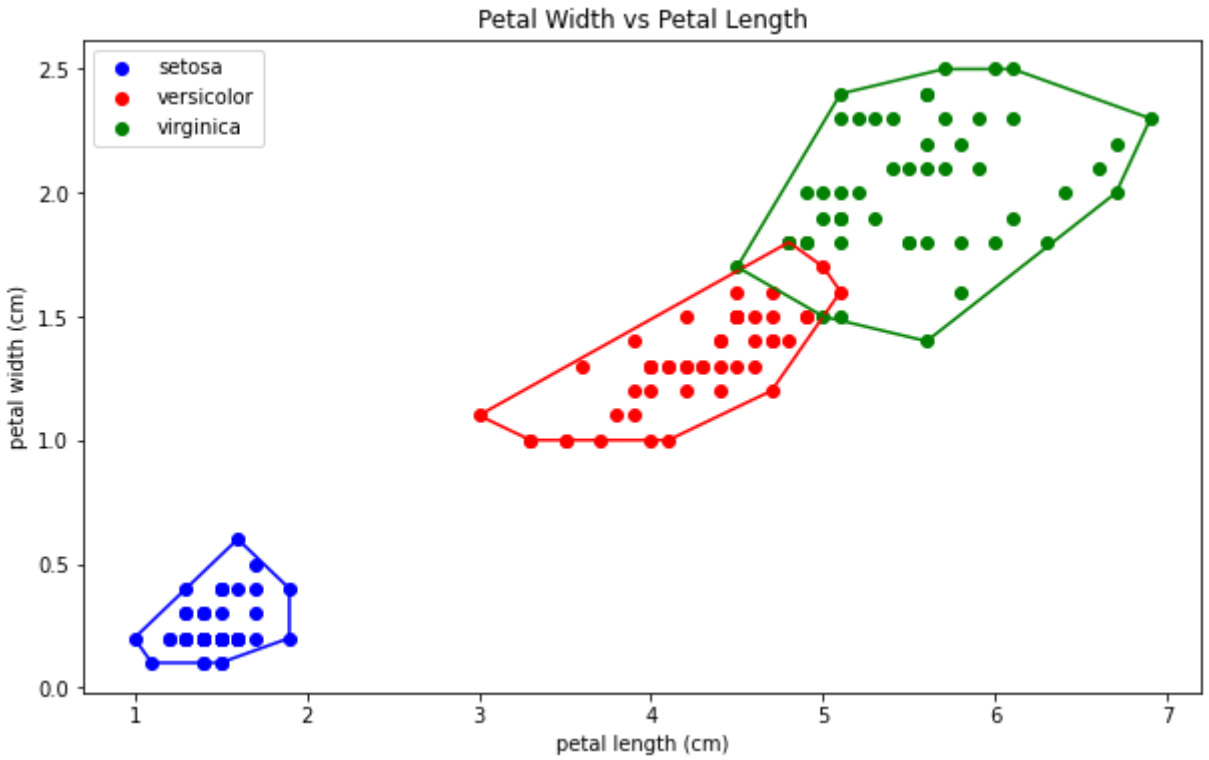
```
<matplotlib.legend.Legend at 0x7f48400e99a0>
```



Gambar 4.3. Hasil Tangkapan Layar Input dan Output

4.1.3. Hubungan *Sepal Length* dengan *Sepal Width*

Berikut ini adalah hasil dari



Gambar 4.4. Ilustrasi Hubungan *Petal Width* dengan *Petal Length*

Pada hasil ini, terlihat setiap jenis Iris cukup terpisah satu sama lain sehingga dapat diklasifikasikan berdasarkan dua hubungan ini. Berikut ini adalah kode untuk mendapatkan output diatas.

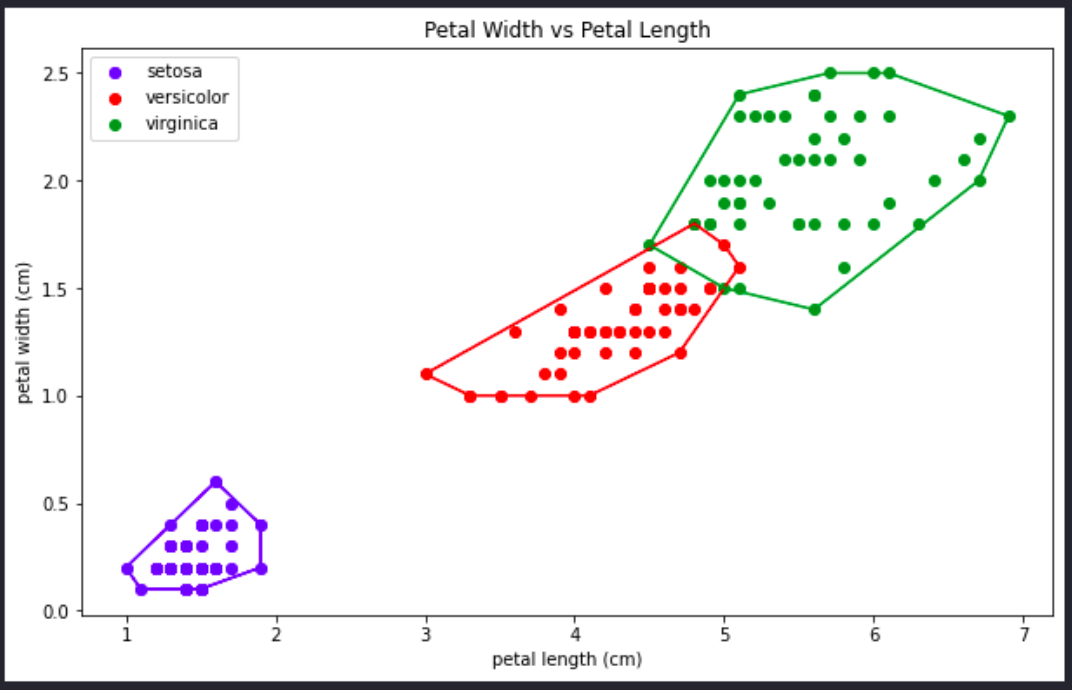
Hubungan Sepal Length dengan Sepal Width

Berikut ini kami tampilkan hubungan antara sepal length dengan sepal width

```
plt.figure(figsize=(10,6))
colors = ['b','r','g']

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

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

  show_graph(data, hull, colors[i], iris.target_names[i])

plt.legend()
```

Python

```
<matplotlib.legend.Legend at 0x7f483df49790>
```



Gambar 4.5. Tangkapan Layar Input dan Output

4.2. Dataset Wine

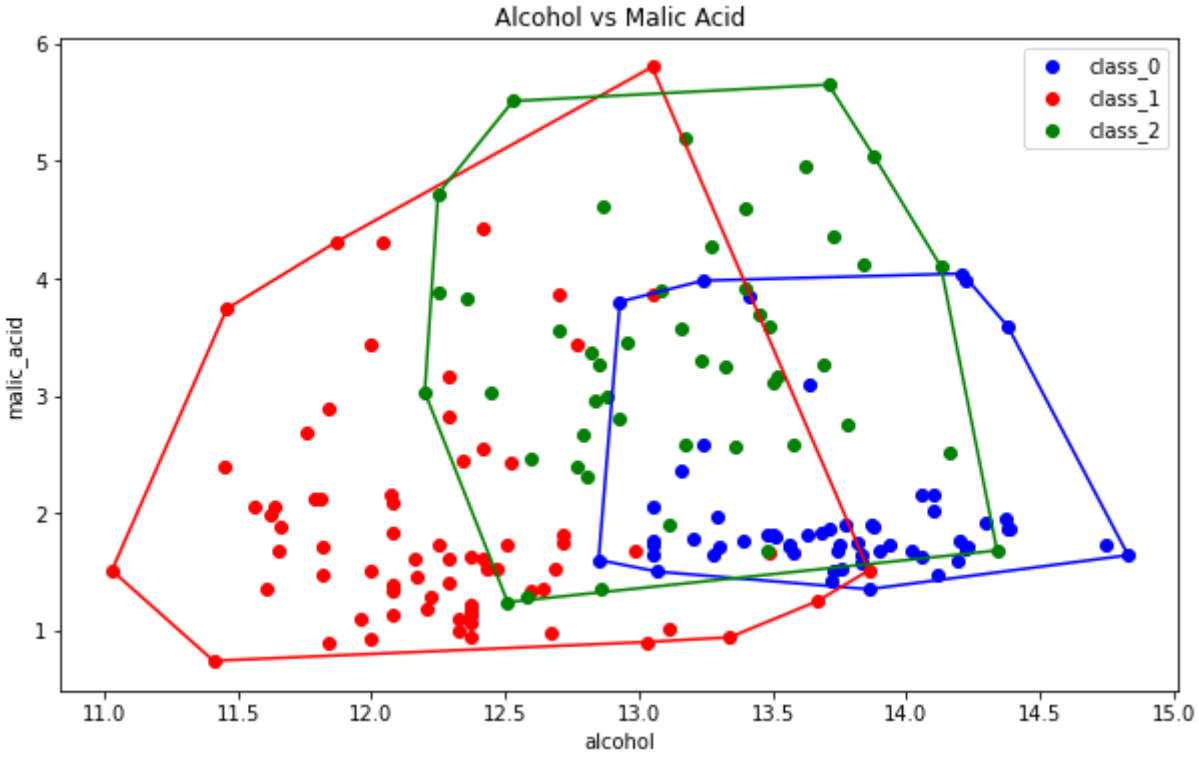
4.2.1. Persiapan Data

Sebelum melakukan proses, dilakukan persiapan sebagai berikut

Gambar 4.6. Persiapan Data

4.2.2. Hubungan Tingkat Alkohol dengan Asam Malic

Berikut ini adalah hasil dari hubungan Tingkat Alkohol dengan Asam Malic.



Gambar 4.7. Hubungan Asam Malic dengan tingkat alkohol untuk setiap kelas wine

Berikut ini adalah hasil tangkapan layar dari input dan output

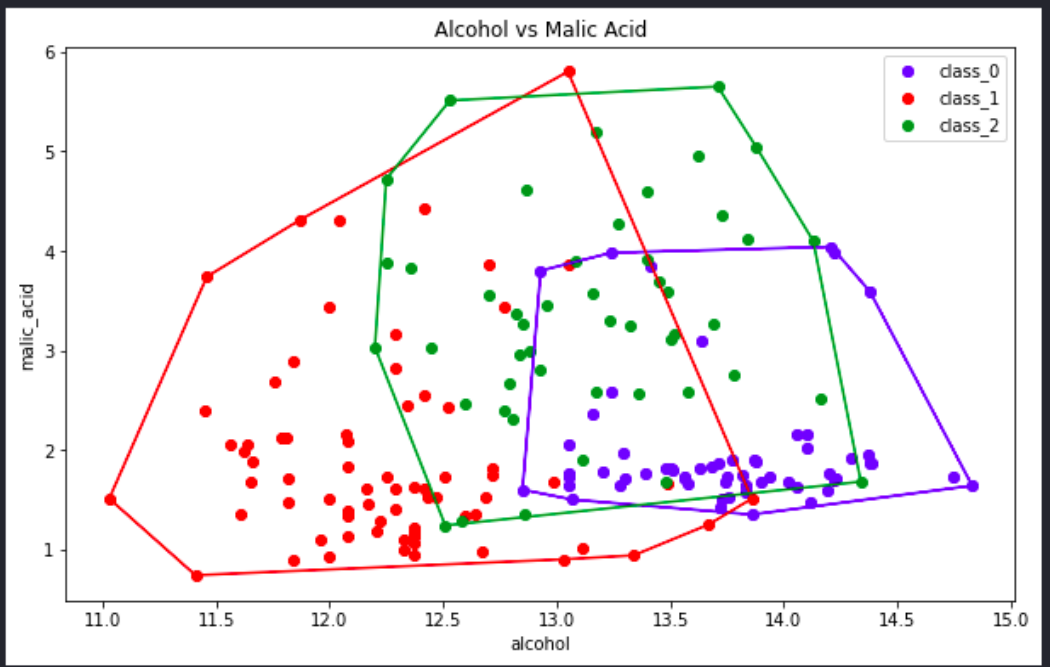
Hubungan Tingkat Alkohol dan Asam Malic

Berikut ini adalah hubungan tingkat alkohol dengan asam Malic

```
plt.figure(figsize=(10,6))
colors = ['b','r','g']

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

for i in range(len(wine.target_names)):
  data = wine_ds[wine_ds['Target'] == i]
  data = data.iloc[:,[0,1]].values
  hull = myConvexHull(data)

  show_graph(data, hull, colors[i], wine.target_names[i])

plt.legend()
```

[21] Python

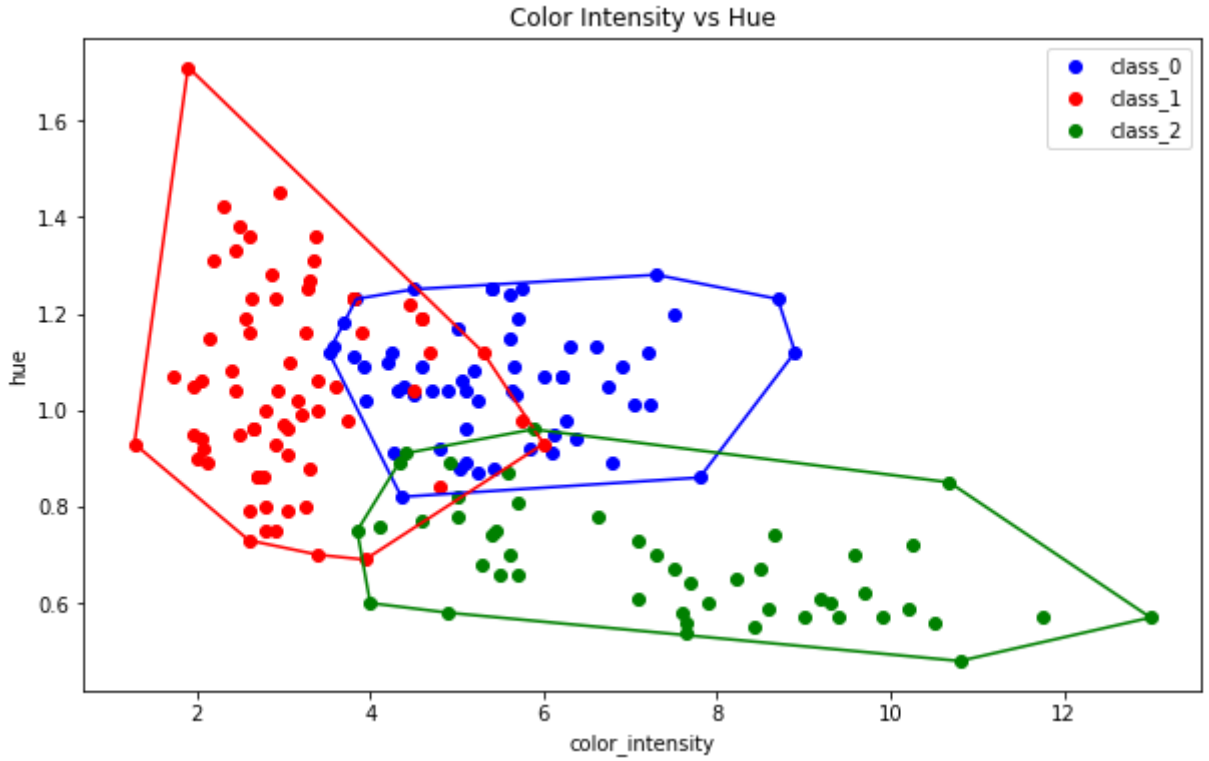
```
<matplotlib.legend.Legend at 0x7f486a9a5fd0>
```



Gambar 4.8. Hasil Tangkapan Layar

4.2.3. Hubungan Intensitas Warna dengan Hue

Berikut ini adalah hasil hubungan antara intensitas warna dengan hue



Gambar 4.9. Hubungan Intensitas Warna dengan Hue

Berikut ini adalah hasil tangkapan layar dari input dan output

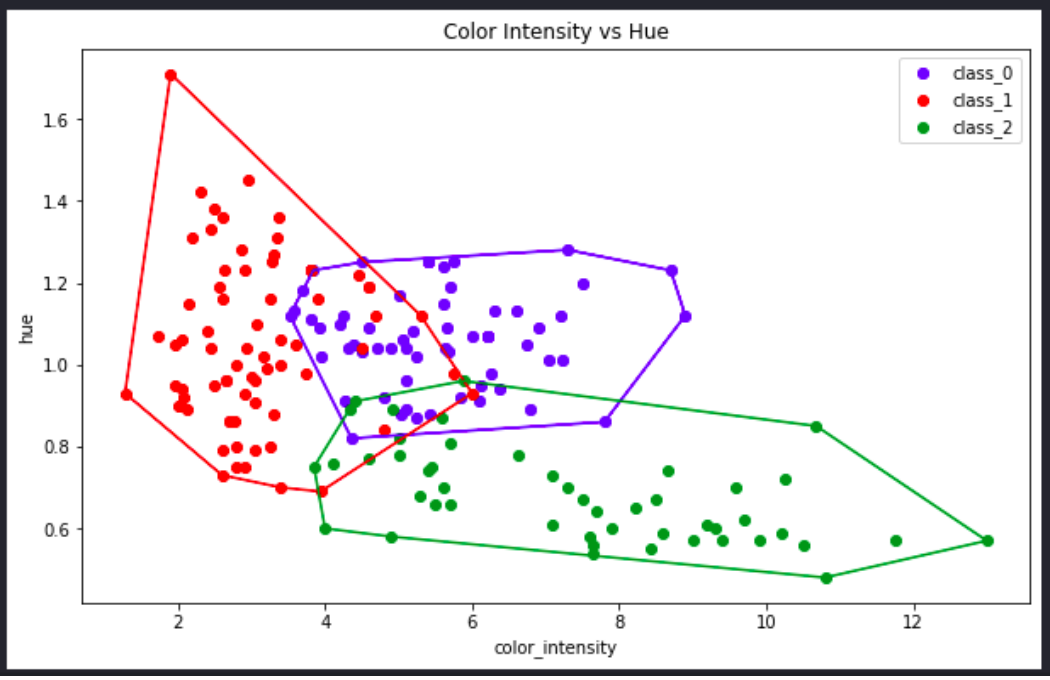
Hubungan Color Intensity dan Hue

Berikut ini adalah hubungan intensitas warna dan hue

```
plt.figure(figsize=(10,6))
colors = ['b','r','g']

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

for i in range(len(wine.target_names)):
  data = wine_ds[wine_ds['Target'] == i]
  data = data.iloc[:,[9,10]].values
  hull = myConvexHull(data)

  show_graph(data, hull, colors[i], wine.target_names[i])

plt.legend()
```

[23] Python

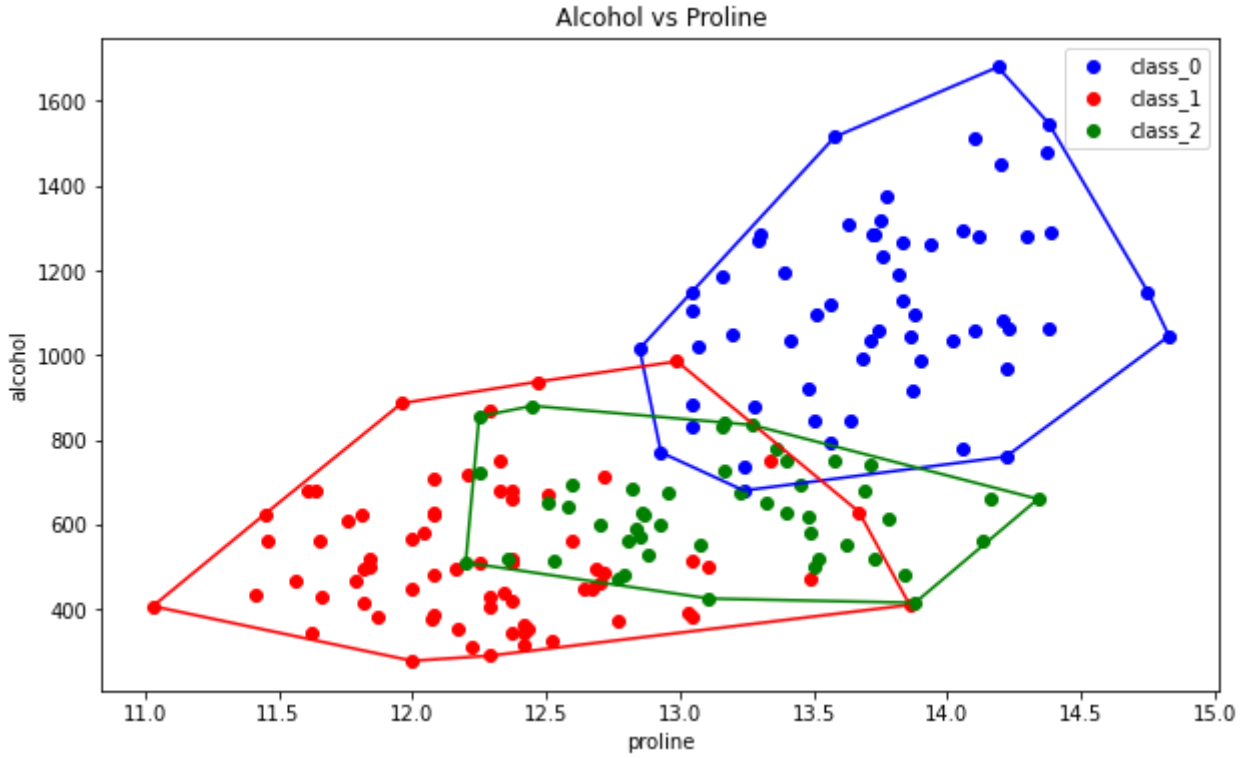
```
<matplotlib.legend.Legend at 0x7f486a8b9430>
```



Gambar 4.10 Hasil Tangkapan Layar

4.2.4. Hubungan Tingkat Alkohol dengan Proline

Berikut ini adalah output dari program dengan parameter diatas



Gambar 4.11 Hasil Hubungan antara tingkat alkohol dengan proline

Berikut adalah tangkapan layar dari input dan output

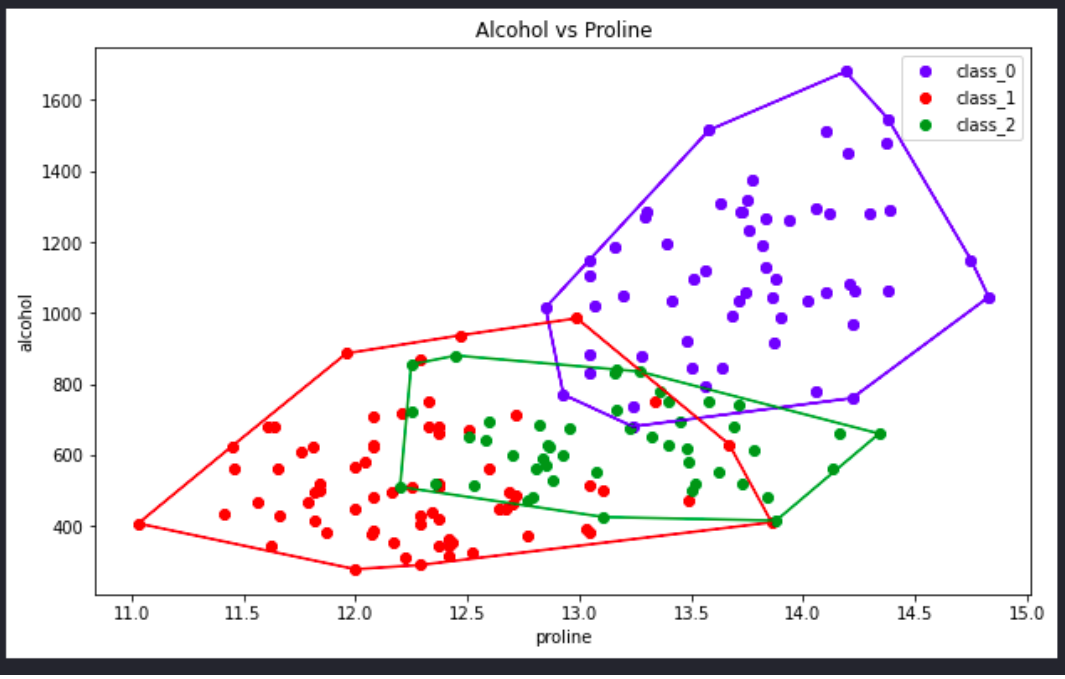
Hubungan Antara Alcohol dan Proline

Berikut ini adalah Alcohol dan Proline

```
plt.figure(figsize=(10,6))
colors = ['b','r','g']

plt.title('Alcohol vs Proline')
plt.xlabel(wine.feature_names[12])
plt.ylabel(wine.feature_names[0])

for i in range(len(wine.target_names)):
  data = wine_ds[wine_ds['Target'] == i]
  data = data.iloc[:,[0,12]].values
  hull = myConvexHull(data)

  show_graph(data, hull, colors[i], wine.target_names[i])

plt.legend()
```

[24] Python

```
<matplotlib.legend.Legend at 0x7f483d255f40>
```



Gambar 4.12 Hasil Tangkapan Layar

Tautan Penting

Hasil program pada laporan ini dapat anda akses pada link berikut ini:

https://github.com/bayusamudra5502/Stima-ConvexHull

Checklist Kemampuan Pustaka

Berikut ini adalah daftar cek dari kemampuan pustaka yang dibuat

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

Catatan:
***** Tidak ada interface khusus yang dapat menerima input langusng. Untuk contoh penggunaan library dapat dilihat pada bab 3 ataupun 4

Daftar Pustaka

Tim Dosen IF2211. 2022. “Slide Kuliah IF2211 Strategi Algoritma”. Bandung: Institut Teknologi Bandung. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/stima21-22.htm (Diakses pada 25 Februari 2022)

Wikipedia contributors. (2021, September 3). Linear separability. In Wikipedia, The Free Encyclopedia. Diakses 26 pada February 2022 dari https://en.wikipedia.org/w/index.php?title=Linear_separability&oldid=1042160240