LAPORAN TUGAS KECIL 2

IF2211 STRATEGI ALGORITMA

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

Disusun oleh:

13520051 – Flavia Beatrix Leoni A. S.

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

Daftar Isi

Daftar Gambar

BAB 1
Algoritma *Divide and Conquer*

Algoritma *divide and conquer* terdiri atas tiga tahap, yaitu *divide*, *conquer (solve)*, dan *combine*. Tahap *divide* adalah tahap membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya berukuran hampir sama). Tahap *conquer* adalah tahap menyelesaikan masing-masing upa-persoalan secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar. Dan tahap *combine* adalah tahap menggabungkan solusi masing-masing upa-persoalan sehingga membentuk solusi persoalan semula.

Menentukan *convex hull* merupakan salah satu hal penting dalam komputasi geometri. Himpunan titik pada bidang planar disebut *convex* jika untuk sembarang dua titik pada bidang tersebut (missal p dan q), seluruh segmen garis yang berakhir di p dan q berada pada himpunan tersebut.

Penulis membuat sebuah pustaka (*library*) *myConvexHull* dalam bahasa Python yang dapat mengembalikan *convex hull* dari kumpulan data 2 dimensi. Pada pustaka ini, *myConvexHull* menerima sebuah parameter bernama *points* yang merupakan array dari kumpulan array titik dan mengembalikan *hull_simplices*, sebuah array of array yang menunjukkan pasangan indeks pada array masukan dari titik-titik yang membentuk garis pada *convex hull*.

Array masukan akan diurutkan membesar terlebih dahulu berdasarkan nilai absisnya. Jika ditemukan terdapat lebih dari satu titik yang memiliki nilai absis yang sama, akan diurutkan membesar pula berdasarkan nilai ordinatnya. Untuk memudahkan, array masukan yang telah diurutkan akan disebut sebagai array terurut pada dokumen ini.

Kemudian, akan dibentuk garis yang menghubungkan titik p_1 dengan nilai absis dan ordinat terkecil dengan titik p_2 dengan nilai absis dan ordinat terbesar, atau dengan kata lain titik pada indeks pertama dan terakhir dari array terurut. Indeks dari pasangan dari kedua titik ini pada array masukan kemudian dimasukkan ke dalam array simplices. Garis tersebut akan menjadi acuan dalam algoritma *divide and conquer* ini. Kumpulan titik di sebelah kiri atau atas garis p_1p_2 akan dimasukkan ke dalam array bernama *a* dan kumpulan titik di sebelah kanan atau bawah garis pp_2 akan dimasukkan ke dalam array bernama *b*. Untuk menentukkan posisi sebuah titik dari suatu garis yang dibentuk oleh dua titik, akan dicari determinannya terlebih

dahulu. Apabila hasil determinan bernilai positif, titik tersebut berada di sebelah kiri garis. Sementara jika hasil determinan bernilai negatif, titik tersebut berada di sebelah kanan garis.

Selanjutnya, akan dicari *convex hull* bagian kiri dan kanan dari garis dengan menggunakan fungsi *findConvexHull*. Fungsi ini menerima empat buah parameter, yaitu array masukan, array terurut yang berisi kumpulan titik yang berada di kiri atau kanan garis, dua buah titik p_1 dan p_2 yang membentuk garis, boolean yang menunjukkan posisi kumpulan titik dari garis sebelumnya, serta array simplices. Jika array masukan merupakan array kosong, maka akan dikembalikan array simplices masukan yang tidak diubah sedikitpun. Sementara jika array masukan tidak kosong, akan dicari titik yang memiliki jarak terjauh dari garis. Hapus pasangan indeks dari kedua titik yang membentuk garis sebelumnya karena telah ditemukan garis terluar baru, yaitu garis p_1p_{max} dan garis $p_{max}p_2$. Masukkan kedua pasangan indeks dari garis tersebut ke dalam array simplices.

Jika kumpulan titik masukan berada di sebelah kiri garis p_1p_2, akan dicari kumpulan titik yang berada di sebelah kiri garis yang baru saja karena titik-titik yang berada di sebelah kanan kedua garis baru tidak mungkin membentuk *convex hull*. Hal yang sama juga diterapkan pada kumpulan titik masukan yang berada di sebelah kanan garis p_1p_2, yaitu akan dicari kumpulan titik yang berada di sebelah kanan garis yang baru. Kumpulan titik yang berada di sebelah kiri/kanan garis p_1p_{max} dimasukkan ke dalam array *a* dan kumpulan titik yang berada di sebelah kiri/kanan garis $p_{max}p_1$ dimasukkan ke dalam array *b*.

Pencarian *convex hull* akan dilanjutkan secara rekursif dengan memanggil fungsi *findConvexHull* untuk array *a* dan *b*. Parameter boolean akan disesuaikan berdasarkan parameter boolean masukan. Jika *convex hull* yang sedang dicari adalah *convex hull* bagian kiri, maka akan dicari *convex hull* bagian kiri kembali, dan sebaliknya. Pemanggilan fungsi secara rekursif ini akan dilakukan hingga bagian kiri dan kanan telah kosong.

BAB 2

Source Code Program

Pustaka myConvexHull

```
import numpy as np
from math import sqrt, acos, degrees, isclose

def myConvexHull(points):
  hull_simplices = []
  # array diurutkan membesar
  sortedPoints = points[points[:,1].argsort()] # sort by ordinat
  sortedPoints = sortedPoints[sortedPoints[:,0].argsort(kind='mergesort')] #
sort by absis

  # titik terkecil dan terbesar akan menjadi titik dalam convex hull
  p1 = sortedPoints[0]
  p2 = sortedPoints[len(sortedPoints)-1]
  temp = [findIndex(points,p1), findIndex(points,p2)]
  hull_simplices.append(temp)

  # bagi kumpulan titik selain p1 dan p2 menjadi 2 bagian
  a = [] # kumpulan titik di sebelah kiri atau bagian atas
  b = [] # kumpulan titik di sebelah kanan atau bagian bawah
  for i in range(len(sortedPoints)):
    if (not((sortedPoints[i][0] == p1[0] and sortedPoints[i][1] == p1[1]) or
(sortedPoints[i][0] == p2[0] and sortedPoints[i][1] == p2[1]))):
      if (determinan(p1, p2, sortedPoints[i]) > 0):
        a.append(sortedPoints[i])
      elif (determinan(p1, p2, sortedPoints[i]) < 0):
        b.append(sortedPoints[i])

  # mencari convex hull bagian kiri dan kanan
  hull_simplices = findConvexHull(points,a,p1,p2,True, hull_simplices)
  hull_simplices = findConvexHull(points,b,p1,p2,False, hull_simplices)

  return hull_simplices

def findConvexHull(points_in,points,p1,p2,kiri,simplices):
  if (len(points) == 0):
    return simplices
  else:
    # cari sebuah titik yang memiliki jarak terjauh dari garis p1p2
    pmax = farthestPoint(points,p1,p2)

    # hapus simplices p1 p2 dan tambahkan simplices p1 pmax dan pmax p2
    idx_p1 = findIndex(points_in,p1)
    idx_p2 = findIndex(points_in,p2)
    idx_pmax = findIndex(points_in,pmax)
```

```
    prevSimplices = [idx_p1, idx_p2]
    idx_prevSimplices = findIndex(simplices,prevSimplices)
    if (idx_prevSimplices == -1): # jika tidak ditemukan, coba cari dengan
urutan terbalik
      prevSimplices = [idx_p2, idx_p1]
      idx_prevSimplices = findIndex(simplices,prevSimplices)
    if (idx_prevSimplices != -1): # belum dihapus dari simplices
      simplices = (np.delete(np.array(simplices), idx_prevSimplices,
axis=0)).tolist()
    temp1_1 = [idx_p1,idx_pmax]
    temp1_2 = [idx_pmax,idx_p1]
    temp2_1 = [idx_pmax,idx_p2]
    temp2_2 = [idx_p2,idx_pmax]
    # tambahkan ke simplices jika belum ada
    if (findIndex(simplices,temp1_1) == -1 and findIndex(simplices,temp1_2)
== -1): # belum ada di simplices
      simplices.append(temp1_1)
    if (findIndex(simplices,temp2_1) == -1 and findIndex(simplices,temp2_2)
== -1): # belum ada di simplices
      simplices.append(temp2_1)

    # bagi kumpulan titik selain pmax menjadi 2 bagian
    a = [] # kumpulan titik di sebelah kiri/kanan garis p1 pmax
    b = [] # kumpulan titik di sebelah kiri/kanan garis pmax p2
    # jika kumpulan titik masukan berada di sebelah kiri garis p1 p2, maka
akan dicari convex hull bagian kiri dari p1 pmax dan pmax p2
    if kiri:
      for i in range(len(points)):
        if (points[i][0] != pmax[0] or points[i][1] != pmax[1]):
          if (determinan(p1, pmax, points[i]) > 0):
            a.append(points[i])
          elif (determinan(pmax, p2, points[i]) > 0):
            b.append(points[i])
      # mencari convex hull secara rekursif
      simplices = findConvexHull(points_in,a,p1,pmax,True,simplices)
      simplices = findConvexHull(points_in,b,pmax,p2,True,simplices)
    # jika kumpulan titik masukan berada di sebelah kanan garis p1 p2, maka
akan dicari convex hull bagian kanan dari p1 pmax dan pmax p2
    else:
      for i in range(len(points)):
        if (points[i][0] != pmax[0] or points[i][1] != pmax[1]):
          if (determinan(p1, pmax, points[i]) < 0):
            a.append(points[i])
          elif (determinan(pmax, p2, points[i]) < 0):
            b.append(points[i])
      # mencari convex hull secara rekursif
      simplices = findConvexHull(points_in,a,p1,pmax,False,simplices)
      simplices = findConvexHull(points_in,b,pmax,p2,False,simplices)
```

```
    return simplices

def determinan(p1,p2,p3):
# Mencari determinan dari p1, p2, p3
  det = p1[0]*p2[1] + p3[0]*p1[1] + p2[0]*p3[1] - p3[0]*p2[1] - p2[0]*p1[1]
- p1[0]*p3[1]
  if (isclose(det,0)): # jika hasil determinan sangat kecil, dibulatkan ke 0
    return 0
  return det

def lineEquation(p1,p2):
# Mencari persamaan garis lurus yang melewati titik p1 dan p2
# ax + by + c = 0
  a = p2[1] - p1[1]   # a = y2 - y1
  b = p1[0] - p2[0]   # b = x1 - x2
  c = (a*p1[0] + b*p1[1])*-1  # c = -(ax1 + by1)
  return (a,b,c)

def distance(a,b,c,point):
# Mencari jarak suatu titik point dengan garis ax + by + c = 0
  return abs(a*point[0] + b*point[1] + c) / sqrt(a**2 + b**2)

import warnings
warnings.filterwarnings('error')

def angle(p,p1,p2):
# Mencari sudut dari p1 p p2
  pp1 = np.subtract(p1,p)
  pp2 = np.subtract(p2,p)
  norm1 = np.sqrt(pp1[0]**2 + pp1[1]**2)
  norm2 = np.sqrt(pp2[0]**2 + pp2[1]**2)
  try:
    cos = np.dot(pp1,pp2) / (norm1 * norm2)
  # untuk menangani kasus pembagian dengan nol
  except Warning:
    if (np.dot(pp1,pp2) > 0):
      cos = 1
    elif (np.dot(pp1,pp2) < 0):
      cos = -1
    else:
      cos = 1
  # untuk menangani kasus hasil pembulatan yang menyebabkan nilai cos tidak
valid
  cos = np.clip(cos, -1, 1)
  return degrees(acos(cos))

def farthestPoint(points,p1,p2):
# Mencari titik terjauh dari garis p1 p2
```

```
  (a,b,c) = lineEquation(p1,p2)
  pmax = points[0]
  dmax = distance(a,b,c,pmax)
  for i in range(1, len(points)):
    dist = distance(a,b,c,points[i])
    if (dist > dmax):
      pmax = points[i]
      dmax = dist
    elif (dist == dmax):
      # cari titik yang memaksimalkan sudut p p1 p2
      if (angle(p1,points[i],p2) > angle(p1,pmax,p2)):
        pmax = points[i]
        dmax = dist
  return pmax

def findIndex(points,p):
  # Mencari indeks p pada array points
  for i in range(len(points)):
    if (points[i][0] == p[0] and points[i][1] == p[1]):
      return i
  # jika p tidak terdapat pada points
  return -1
```

Visualisasi hasil Convex Hull pada Dataset Iris

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


# Data Iris Sepal Width vs Sepal Length
data_iris = datasets.load_iris()
# create a DataFrame
df = pd.DataFrame(data_iris.data, columns=data_iris.feature_names)
df['Target'] = pd.DataFrame(data_iris.target)
# visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Sepal Width vs Sepal Length')
plt.xlabel(data_iris.feature_names[0])
plt.ylabel(data_iris.feature_names[1])
for i in range(len(data_iris.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:,[0,1]].values
  simplices = myConvexHull(bucket)
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data_iris.target_names[i])
  for simplex in simplices:
    plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

BAB 3

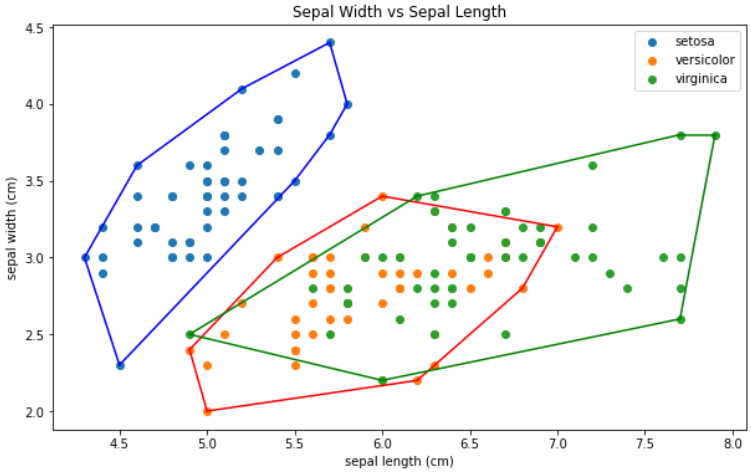
Pengujian Program

3.1 Iris Plants Dataset

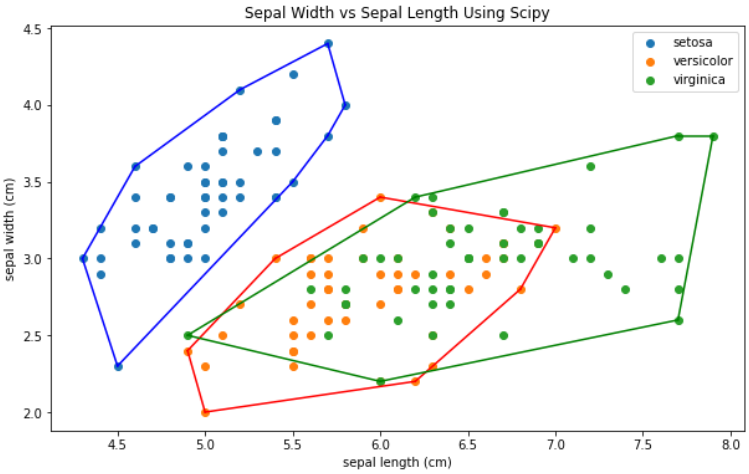
3.1.1 Sepal Width vs Sepal Length

```
# Data Iris Sepal Width vs Sepal Length
data_iris = datasets.load_iris()
# create a DataFrame
df = pd.DataFrame(data_iris.data, columns=data_iris.feature_names)
df['Target'] = pd.DataFrame(data_iris.target)
# visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Sepal Width vs Sepal Length')
plt.xlabel(data_iris.feature_names[0])
plt.ylabel(data_iris.feature_names[1])
for i in range(len(data_iris.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:,[0,1]].values
  simplices = myConvexHull(bucket)
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data_iris.target_names[i])
  for simplex in simplices:
    plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

Gambar 3.1 Kode Program Untuk Visualisasi Sepal Width vs Sepal Length



Gambar 3.2 Hasil Visualisasi Sepal Width vs Sepal Length dengan Pustaka myConvexHull

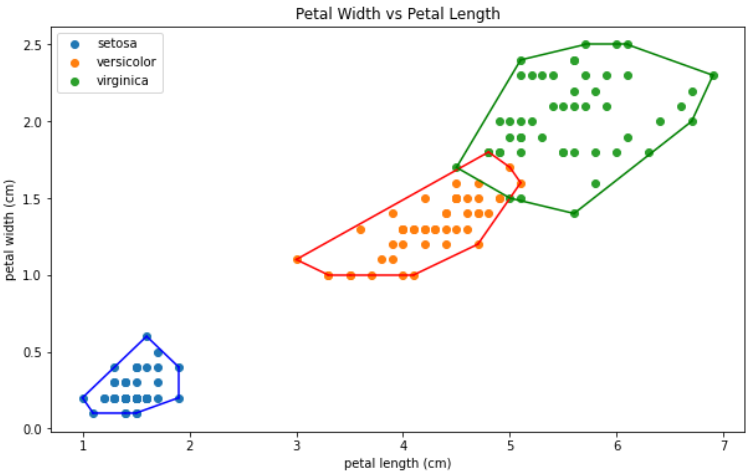


Gambar 3.3 Hasil Visualisasi Sepal Width vs Sepal Length dengan Pustaka ConvexHull SciPy

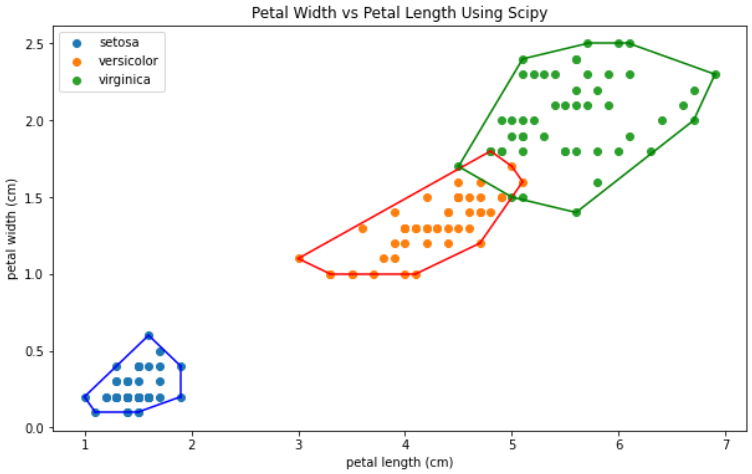
3.1.2 Petal Width vs Petal Length

```
# Data Iris Petal Width vs Petal Length
# visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal Length')
plt.xlabel(data_iris.feature_names[2])
plt.ylabel(data_iris.feature_names[3])
for i in range(len(data_iris.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:,[2,3]].values
  simplices = myConvexHull(bucket)
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data_iris.target_names[i])
  for simplex in simplices:
    plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

Gambar 3.4 Kode Program Untuk Visualisasi Petal Width vs Petal Length



Gambar 3.5 Hasil Visualisasi Petal Width vs Petal Length dengan Pustaka myConvexHull



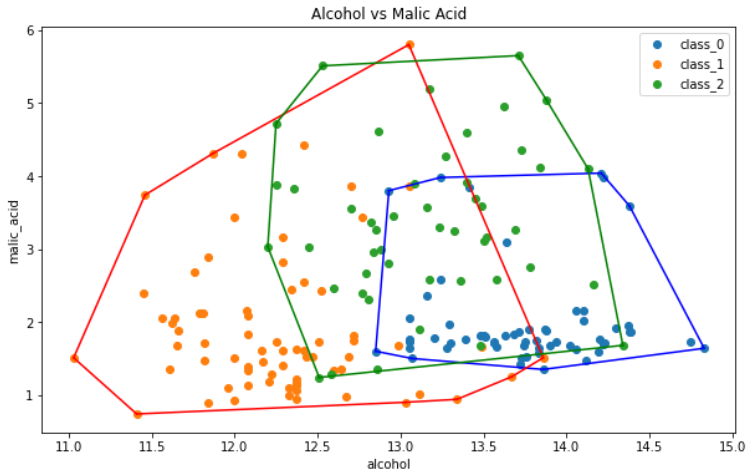
Gambar 3.6 Hasil Visualisasi Petal Width vs Petal Length dengan Pustaka ConvexHull SciPy

3.2 Wine Recognition Dataset

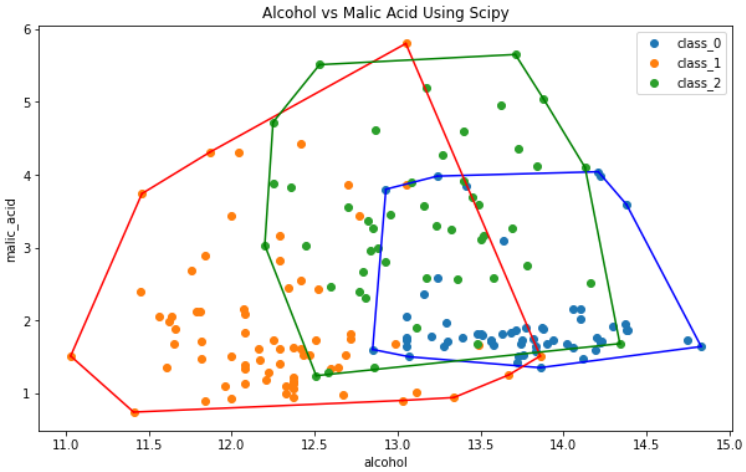
3.2.1 Alcohol vs Malic Acid

```
# Data Wine Alcohol vs Malic Acid
data_wine = datasets.load_wine()
# create a DataFrame
df = pd.DataFrame(data_wine.data, columns=data_wine.feature_names)
df['Target'] = pd.DataFrame(data_wine.target)
# visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Alcohol vs Malic Acid')
plt.xlabel(data_wine.feature_names[0])
plt.ylabel(data_wine.feature_names[1])
for i in range(len(data_wine.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:,[0,1]].values
  simplices = myConvexHull(bucket)
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data_wine.target_names[i])
  for simplex in simplices:
    plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

Gambar 3.7 Kode Program Untuk Visualisasi Alcohol vs Malic Acid



Gambar 3.8 Hasil Visualisasi Alcohol vs Malic Acid dengan Pustaka myConvexHull

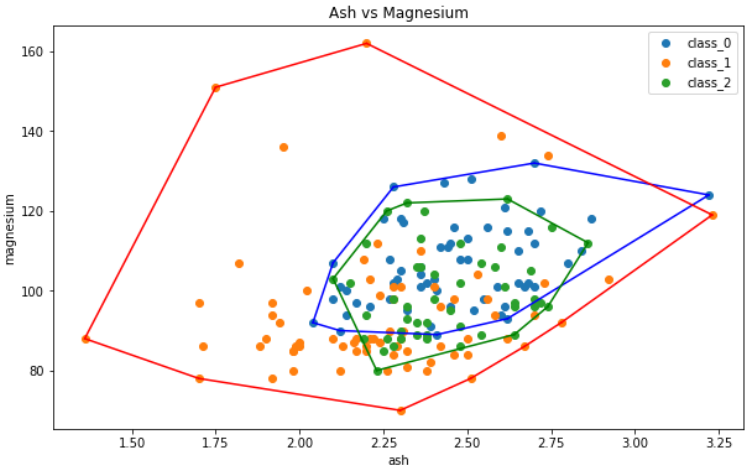


Gambar 3.9 Hasil Visualisasi Alcohol vs Malic Acid dengan Pustaka ConvexHull SciPy

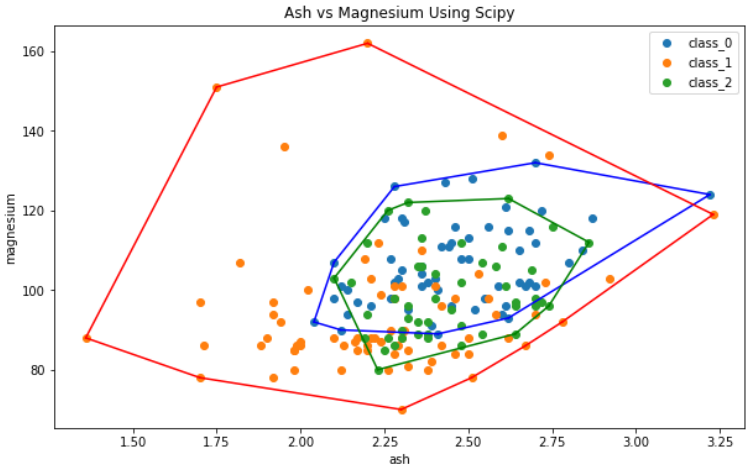
3.2.2 Ash vs Magnesium

```
# Data Wine Ash vs Magnesium
# visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Ash vs Magnesium')
plt.xlabel(data_wine.feature_names[2])
plt.ylabel(data_wine.feature_names[4])
for i in range(len(data_wine.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:,[2,4]].values
  simplices = myConvexHull(bucket)
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data_wine.target_names[i])
  for simplex in simplices:
    plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

Gambar 3.10 Kode Program Untuk Visualisasi Ash vs Magnesium



Gambar 3.11 Hasil Visualisasi Ash vs Magnesium dengan Pustaka myConvexHull



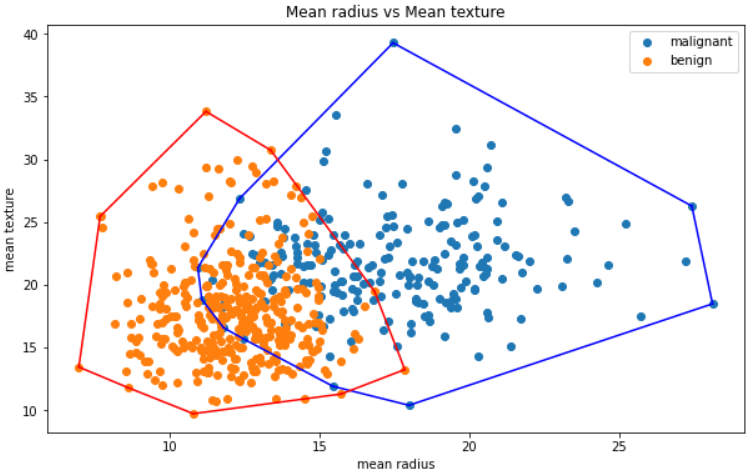
Gambar 3.12 Hasil Visualisasi Ash vs Magnesium dengan Pustaka ConvexHull SciPy

3.3 Breast Cancer Wisconsin (Diagnostic) Dataset

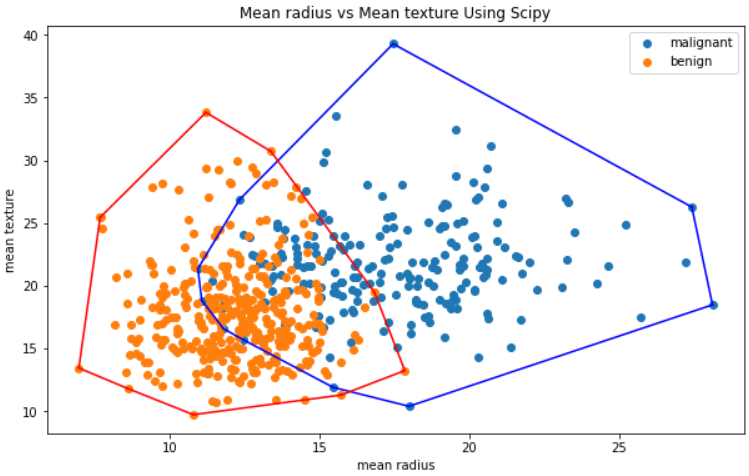
3.3.1 Mean Radius vs Mean Texture

```
# Data Breast Cancer Mean Radius vs Mean Texture
data_breast_cancer = datasets.load_breast_cancer()
# create a DataFrame
df = pd.DataFrame(data_breast_cancer.data, columns=data_breast_cancer.feature_names)
df['Target'] = pd.DataFrame(data_breast_cancer.target)
# visualisasi hasil ConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Mean radius vs Mean texture')
plt.xlabel(data_breast_cancer.feature_names[0])
plt.ylabel(data_breast_cancer.feature_names[1])
for i in range(len(data_breast_cancer.target_names)):
  bucket = df[df['Target'] == i]
  bucket = bucket.iloc[:,[0,1]].values
  simplices = myConvexHull(bucket)
  plt.scatter(bucket[:, 0], bucket[:, 1], label=data_breast_cancer.target_names[i])
  for simplex in simplices:
    plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

Gambar 3.13 Kode Program Untuk Visualisasi Mean Radius vs Mean Texture



Gambar 3.14 Hasil Visualisasi Mean Radius vs Mean Texture dengan Pustaka myConvexHull



Gambar 3.15 Hasil Visualisasi Mean Radius vs Mean Texture dengan Pustaka ConvexHull SciPy

Lampiran

Link kode program:

https://github.com/leoniantoinette/Convex-Hull.git

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	√	