LAPORAN TUGAS KECIL
IF2211 – STRATEGI ALGORITMA

ALGORITMA PEMBENTUK *CONVEX HULL* DENGAN METODE *DIVIDE AND CONQUER*

Dipersiapkan oleh:

Dzaky Fattan Rizqullah

13520003

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESHA 10, BANDUNG 40132

Daftar Isi

A. Algoritma *Divide-and-Conquer*

Library ini merupakan pembentuk *convex hull* yang menerima data yang dapat dibentuk menjadi tuple bernilai dua, atau merupakan data dua dimensi (dalam hal ini data dapat dianggap sebagai titik dua dimensi). Himpunan titik pada bidang planar disebut *convex* jika untuk sembarang dua titik pada bidang tersebut (misal p dan q), seluruh segmen garis yang berakhir di p dan q berada pada himpunan tersebut. Algoritma yang digunakan yaitu algoritma *Divide-and-Conquer*.

Untuk prosesnya, pertama, data kumpulan titik akan diurutkan berdasarkan absis, kemudian ordinatnya, agar didapat dua titik ekstrim (terletak di ujung-ujung bidang). Dua titik ini dapat membentuk garis yang memisahkan kumpulan titik lainnya menjadi dua bagian. Di sini lah *Divide-and-Conquer* dilakukan untuk masing-masing bagian.

Setiap bagian akan dicari titik yang letaknya paling jauh dari garis (bila ada titik dengan jarak yang sama, pilih titik yang bila ditarik garis ke salah satu dari dua titik awal sebelumnya akan membentuk sudut yang paling besar), kemudian tarik garis dari titik tersebut ke dua titik awal sebelumnya sehingga membentuk dua garis dan akhirnya membentuk segitiga. Titik-titik di dalam segitiga diabaikan, sementara titik di luar segitiga (masih di dalam bagian yang dimaksud) dapat dibagi menjadi dua bagian lagi (sebelah kiri dan kanan), yang kemudian di cari kembali titik terjauh dan tarik dua garis hingga membentuk segitiga kembali.

Proses ini dilakukan hingga tidak ada lagi titik di luar segitiga. Dua titik tersisa untuk setiap bagian yang mencapai kasus ini disatukan sedemikian rupa agar titk-titik ini bila dihubungkan dengan garis maka akan membentuk *convex hull*.

B. Source Program

Berikut dilampirkan kode library `myConvexHull` yang sudah mengimplementasi metode *Divide-and-Conquer*. Program ini ditulis dalam bahasa Python 3.

```
# Tugas Kecil 2
## Implementasi Library myConvexHull Menggunakan Algoritma Divide and Conquer

### Nama  : Dzaky Fattan Rizqullah
### NIM   : 13520003
### Kelas : K3

# Import seluruh modul yang diperlukan
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets

# Modul myConvexHull
# Dapat dipanggil oleh program utama dengan memanggil fungsi myConvexHull()

def sortPoints(x):
```

```
    # Mengurutkan titik2 pada array agar didapat urutan menaik pada x, kemudian pada
y
    x = x[x[:,1].argsort()]
    x = x[x[:,0].argsort(kind='mergesort')]
    return x

def makeUniqueList(x):
    # membuat list yang elemen2nya unik
    uniqueList = []
    for e in x:
        flag = 0
        for elem in uniqueList:
            if e[0] == elem[0] and e[1] == elem[1]:
                    flag = 1
        if flag == 0:
            if type(e) != list:
                e = e.tolist()
            uniqueList.append(e)
    return uniqueList

def determinePos(a, p1, p2):
    # menentukan posisi titik a apakah di atas atau dibawah garis p1p2, dengan
mencari determinan matriks
    # [p1x p1y 1]
    # [p2x p2y 1]
    # [ax  ay  1]
    return (p1[0] * p2[1]) + (p2[0] * a[1]) + (a[0] * p1[1]) - (a[0] * p2[1]) -
(p2[0] * p1[1]) - (p1[0] * a[1])

def distancePointToLine(x, p1, p2):
    # mengembalikan nilai jarak terpendek titik x ke garis p1p2
    return abs((p2[0] - p1[0]) * (p1[1] - x[1]) - (p1[0] - x[0]) * (p2[1] - p1[1])) /
math.sqrt(((p2[0] - p1[0])**2) + ((p2[1] - p1[1])**2))

def getAngle(p1, p2, p3):
    # mengembalikan besar sudut p1p2p3 dalam derajat
    angle = math.degrees(math.atan2(p3[1]-p2[1], p3[0]-p2[0]) - math.atan2(p1[1]-
p2[1], p1[0]-p2[0]))
    return angle + 360 if angle < 0 else angle

# Modul myConvexHull
# Dapat dipanggil oleh program utama dengan memanggil fungsi myConvexHull()

def sortPoints(x):
    # Mengurutkan titik2 pada array agar didapat urutan menaik pada x, kemudian pada
y
```

```
    x = x[x[:,1].argsort()]
    x = x[x[:,0].argsort(kind='mergesort')]
    return x

def makeUniqueList(x):
    # membuat list yang elemen2nya unik
    uniqueList = []
    for e in x:
        flag = 0
        for elem in uniqueList:
            if e[0] == elem[0] and e[1] == elem[1]:
                    flag = 1
        if flag == 0:
            if type(e) != list:
                e = e.tolist()
            uniqueList.append(e)
    return uniqueList

def determinePos(a, p1, p2):
    # menentukan posisi titik a apakah di atas atau dibawah garis p1p2, dengan
mencari determinan matriks
    # [p1x p1y 1]
    # [p2x p2y 1]
    # [ax  ay  1]
    return (p1[0] * p2[1]) + (p2[0] * a[1]) + (a[0] * p1[1]) - (a[0] * p2[1]) -
(p2[0] * p1[1]) - (p1[0] * a[1])

def distancePointToLine(x, p1, p2):
    # mengembalikan nilai jarak terpendek titik x ke garis p1p2
    return abs((p2[0] - p1[0]) * (p1[1] - x[1]) - (p1[0] - x[0]) * (p2[1] - p1[1])) /
math.sqrt(((p2[0] - p1[0])**2) + ((p2[1] - p1[1])**2))

def getAngle(p1, p2, p3):
    # mengembalikan besar sudut p1p2p3 dalam derajat
    angle = math.degrees(math.atan2(p3[1]-p2[1], p3[0]-p2[0]) - math.atan2(p1[1]-
p2[1], p1[0]-p2[0]))
    return angle + 360 if angle < 0 else angle

# IMPLEMENTASI DIVIDE AND CONQUER myConvexHull
def DnC(hull, sgmt, p1, p2, upper):
    # argument upper digunakan untuk membedakan apakah separuh convexhull yang
    # dikerjakan adalah bagian atas (S1) atau bawah (S2)

    # Kalau sgmt kosong (tidak ada titik lain di atas p1p2),
    if not sgmt:
        # maka p1p2 menjadi pembentuk convex hull S1
```

```
        hull.append(p1)
        hull.append(p2)

    # Kalau jumlah elemen sgmt lebih dari satu
    elif len(sgmt) > 1:
        # Tentukan titik yang jarak terpendeknya paling jauh dari garis p1p2
        temp = sgmt[0]
        for i in sgmt:
            if distancePointToLine(temp, p1, p2) < distancePointToLine(i, p1, p2):
                temp = i
            # Jika ada lebih dari satu titik yang jarak terpendeknya terjauh,
            elif distancePointToLine(temp, p1, p2) == distancePointToLine(i, p1, p2):
                # maka pilih yang membentuk sudut yang lebih besar
                if  getAngle(temp, p1, p2) <= getAngle(i, p1, p2):
                    temp = i
            # Tidak perlu meninjau kasus untuk titik di dalam segitiga temp-p1-p2
        Sub1 = []
        Sub2 = []
        # Setelah itu menentukan titik2 di atas garis p1temp dan temp-p2
        for i in sgmt:
            # math.isclose() digunakan untuk menghindari kegagalan saat
            # membandingkan nilai bertipe data float
            point = i
            if upper:
                if (determinePos(point, p1, temp) > 0 and not
(math.isclose(determinePos(point, p1, temp), 0, abs_tol=1e-10))):
                    Sub1.append(point)
                elif (determinePos(point, temp, p2) > 0 and not
(math.isclose(determinePos(point, temp, p2), 0, abs_tol=1e-10))):
                    Sub2.append(point)
            else:
                if (determinePos(point, p1, temp) < 0 and not
(math.isclose(determinePos(point, p1, temp), 0, abs_tol=1e-10))):
                    Sub1.append(point)
                elif (determinePos(point, temp, p2) < 0 and not
(math.isclose(determinePos(point, temp, p2), 0, abs_tol=1e-10))):
                    Sub2.append(point)
        # dibagi menjadi dua kasus rekursif, Sub1 dan Sub2
        hull1 = DnC(hull, Sub1, p1, temp, upper)
        hull2 = DnC(hull, Sub2, temp, p2, upper)
        for e in hull1:
            hull.append(e)
        for f in hull2:
            hull.append(f)

    # Kalau jumlah elemen sgmt hanya satu
```

```
    else:
        # p1p2 dan satu2nya elemen pada sgmt menjadi pembentuk separuh bagian convex
hull
        hull.append(p1)
        hull.append(p2)
        hull.append(sgmt[0])
    hull = makeUniqueList(hull)
    return hull

# IMPLEMENTASI FUNGSI UTAMA myConvexHull
def myConvexHull(hull):
    # menentukan titik2 pembentuk convex hull secara manual

    # Pertama, melakukan sort pada data S
    sortedPlot = sortPoints(hull)

    # Kemudian menentukan dua titik (p1, p2) sedemikian sehingga S terbagi
    # menjadi S1 (di atas garis p1p2) dan S2 (di atas garis p1p2)
    p1 = sortedPlot[0]
    p2 = sortedPlot[len(sortedPlot)-1]

    # membuat list kosong untuk menampung S1 dan S2 secara terpisah
    upper = []
    lower = []

    # mengelompokkan titik2 yang posisinya di atas ataupun di bawah garus p1p2
    for i in sortedPlot:
        point = i.tolist()
        if (determinePos(point, p1, p2) > 0):
            upper.append(point)
        elif (determinePos(point, p1, p2) < 0):
            lower.append(point)

    # membuat list kosong yang menerima titik2 yang membentuk convexHull
    sortedUpper = []
    sortedLower = []
    # DnC dilakukan di sini, terpisah untuk upper dan lower
    sortedUpper = DnC(sortedUpper, upper, p1, p2, True)
    sortedUpper.sort(key = lambda i: i[0])
    sortedLower = DnC(sortedLower, lower, p1, p2, False)
    sortedLower.sort(key = lambda i: -i[0])

    # Mengambungkan upper dan lower dan menyesuaikan urutan agar dapat dibentuk garis
    for i in sortedLower:
        sortedUpper.append(i)
```

```
    # Membuat list sortedHull yang sudah di cek agar tidak ada duplikasi titik
    # serta meng-append titik paling pertama di ujung list agar dapat dibentuk
    # garis convex hull
    sortedHull = makeUniqueList(sortedUpper)
    sortedHull.append(p1)

    # kembalikan tipe list menjadi ndarray
    nphull = np.array(sortedHull)
    return nphull

def myConvexHull(hull):
    # menentukan titik2 pembentuk convex hull secara manual

    # Pertama, melakukan sort pada data S
    sortedPlot = sortPoints(hull)

    # Kemudian menentukan dua titik (p1, p2) sedemikian sehingga S terbagi
    # menjadi S1 (di atas garis p1p2) dan S2 (di atas garis p1p2)
    p1 = sortedPlot[0]
    p2 = sortedPlot[len(sortedPlot)-1]

    # membuat list kosong untuk menampung S1 dan S2 secara terpisah
    upper = []
    lower = []

    # mengelompokkan titik2 yang posisinya di atas ataupun di bawah garus p1p2
    for i in sortedPlot:
        point = i.tolist()
        if (determinePos(point, p1, p2) > 0):
            upper.append(point)
        elif (determinePos(point, p1, p2) < 0):
            lower.append(point)

    # membuat list kosong yang menerima titik2 yang membentuk convexHull
    sortedUpper = []
    sortedLower = []
    # DnC dilakukan di sini, terpisah untuk upper dan lower
    sortedUpper = DnC(sortedUpper, upper, p1, p2, True)
    sortedUpper.sort(key = lambda i: i[0])
    sortedLower = DnC(sortedLower, lower, p1, p2, False)
    sortedLower.sort(key = lambda i: -i[0])

    # Mengambungkan upper dan lower dan menyesuaikan urutan agar dapat dibentuk garis
    for i in sortedLower:
        sortedUpper.append(i)
```

```
    # Membuat list sortedHull yang sudah di cek agar tidak ada duplikasi titik
    # serta meng-append titik paling pertama di ujung list agar dapat dibentuk
    # garis convex hull
    sortedHull = makeUniqueList(sortedUpper)
    sortedHull.append(p1)

    # kembalikan tipe list menjadi ndarray
    nphull = np.array(sortedHull)
    return nphull
```

Berikut dilampirkan implementasi library `myConvexHull` untuk ujicoba visualisasi data untuk dataset `iris`, `wine`, dan `diabetes`.

```
# IMPORT DATA IRIS (MENGIKUTI SPEK TUCIL 2)
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()

# PERSIAPAN DATA, PEMANGGILAN FUNGSI CONVEXHULL
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
    # bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for e in hull:
        plt.plot(hull[:, 0], hull[:, 1], colors[i])
plt.legend()

# PERSIAPAN DATA, PEMANGGILAN FUNGSI CONVEXHULL
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
```

```
    # bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for e in hull:
        plt.plot(hull[:, 0], hull[:, 1], colors[i])
plt.legend()
# IMPORT DATASET LAIN #2 (DATASET WINE)
data2 = datasets.load_wine()
# create a DataFrame
df2 = pd.DataFrame(data2.data, columns=data2.feature_names)
df2['Target'] = pd.DataFrame(data2.target)
print(df2.shape)
df2.head()

# PERSIAPAN DATA, PEMANGGILAN FUNGSI CONVEXHULL
# SUMBU X = ALCOHOL, SUMBU Y = COLOR INTENSITY
plt.figure(figsize=(10, 6))
colors = ['b', 'r', 'g']
plt.title('Alcohol vs Color Intensity')
plt.xlabel(data2.feature_names[0])
plt.ylabel(data2.feature_names[9])
for i in range(len(data2.target_names)):
    bucket2 = df2[df2['Target'] == i]
    bucket2 = bucket2.iloc[:, [0, 9]].values
    # bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull2 = myConvexHull(bucket2)
    plt.scatter(bucket2[:, 0], bucket2[:, 1], label=data2.target_names[i])
    for e in hull2:
        plt.plot(hull2[:, 0], hull2[:, 1], colors[i])
plt.legend()
# IMPORT DATASET LAIN #3 (DATASET DIABETES)
data3 = datasets.load_diabetes()
# create a DataFrame
df3 = pd.DataFrame(data3.data, columns=data3.feature_names)
print(df3.shape)
df3.head()

# PERSIAPAN DATA, PEMANGGILAN FUNGSI CONVEXHULL
# SUMBU X = AGE, SUMBU Y = BLOOD SUGAR LEVEL
plt.figure(figsize=(10, 6))
plt.title('Age vs blood sugar level')
plt.xlabel("Age")
plt.ylabel("Blood Sugar Level")
bucket3 = df3.iloc[:, [0, 9]].values
hull3 = myConvexHull(bucket3)
plt.scatter(bucket3[:, 0], bucket3[:, 1])
```

```
for e in hull3:
    plt.plot(hull3[:, 0], hull3[:, 1])
```

C. Dokumentasi *Input* dan *Output* Program

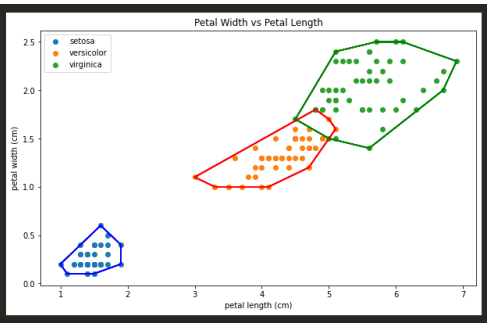
Proses ini dilakukan hingga tidak ada lagi titik di luar segitiga. Dua titik tersisa untuk setiap bagian yang mencapai kasus ini disatukan sedemikian rupa agar titk-titik ini bila dihubungkan dengan garis maka akan membentuk *convex hull*.

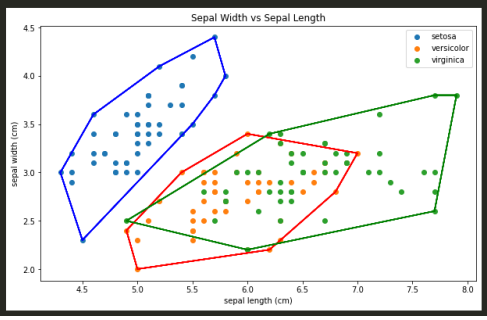
1. Dataset Iris, “Petal width vs Petal length” dan ”Sepal width vs Sepal length”

```
# IMPORT DATA IRIS (MENGIKUTI SPEK TUCIL 2)
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()

# PERSIAPAN DATA, PEMANGGILAN FUNGSI CONVEXHULL
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
    # bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for e in hull:
        plt.plot(hull[:, 0], hull[:, 1], colors[i])
plt.legend()

# PERSIAPAN DATA, PEMANGGILAN FUNGSI CONVEXHULL
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
    # bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for e in hull:
        plt.plot(hull[:, 0], hull[:, 1], colors[i])
plt.legend()
```

✓ 0.5s Python

```
(150, 5)
<matplotlib.legend.Legend at 0x18c12760f10>
```



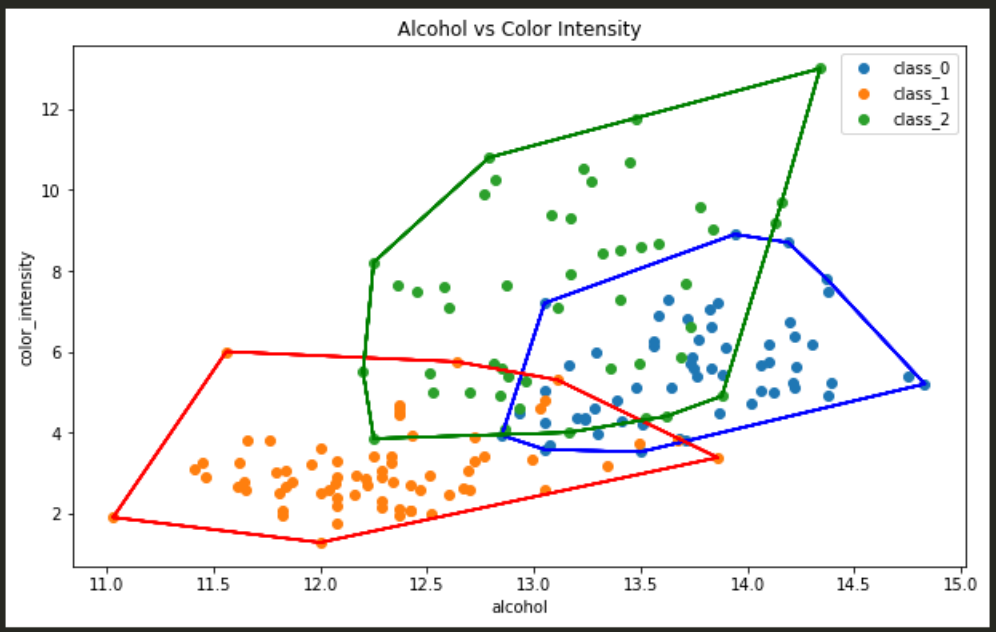


2. Dataset Wine, “Alcohol vs Color intensity”

```python
# IMPORT DATASET LAIN #2 (DATASET WINE)
data2 = datasets.load_wine()
# create a DataFrame
df2 = pd.DataFrame(data2.data, columns=data2.feature_names)
df2['Target'] = pd.DataFrame(data2.target)
print(df2.shape)
df2.head()

# PERSIAPAN DATA, PEMANGGILAN FUNGSI CONVEXHULL
# SUMBU X = ALCOHOL, SUMBU Y = COLOR INTENSITY
plt.figure(figsize=(10, 6))
colors = ['b', 'r', 'g']
plt.title('Alcohol vs Color Intensity')
plt.xlabel(data2.feature_names[0])
plt.ylabel(data2.feature_names[9])
for i in range(len(data2.target_names)):
    bucket2 = df2[df2['Target'] == i]
    bucket2 = bucket2.iloc[:, [0, 9]].values
    # bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull2 = myConvexHull(bucket2)
    plt.scatter(bucket2[:, 0], bucket2[:, 1], label=data2.target_names[i])
    for e in hull2:
        plt.plot(hull2[:, 0], hull2[:, 1], colors[i])
plt.legend()
```

✓ 0.2s Python

```
(178, 14)

<matplotlib.legend.Legend at 0x18c1284e1c0>
```

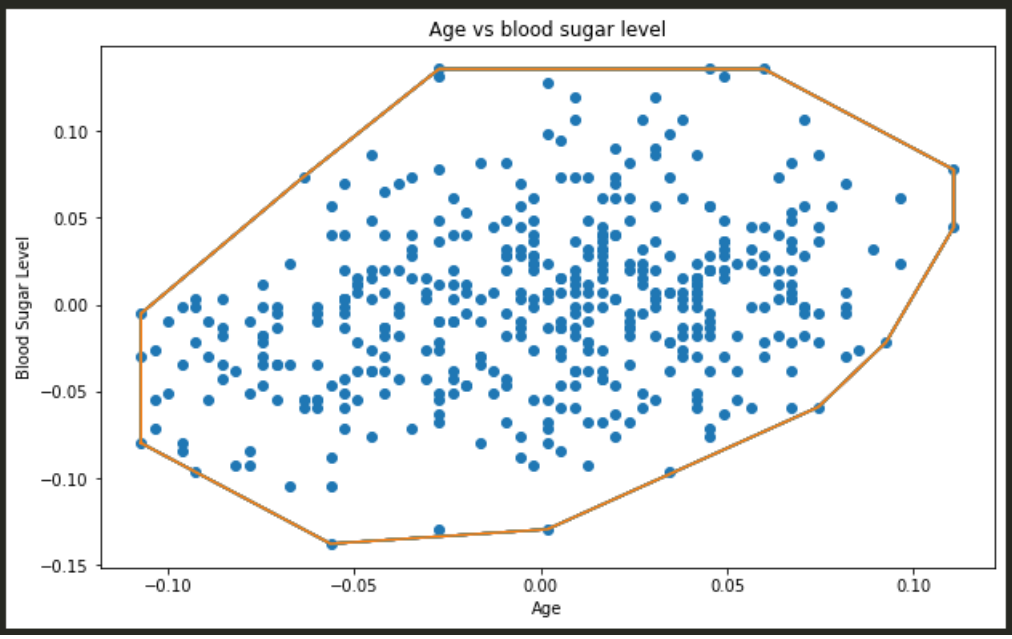


3. Dataset Diabetes, “Age vs Blood sugar level”

```
# IMPORT DATASET LAIN #3 (DATASET DIABETES)
data3 = datasets.load_diabetes()
# create a DataFrame
df3 = pd.DataFrame(data3.data, columns=data3.feature_names)
print(df3.shape)
df3.head()

# PERSIAPAN DATA, PEMANGGILAN FUNGSI CONVEXHULL
# SUMBU X = AGE, SUMBU Y = BLOOD SUGAR LEVEL
plt.figure(figsize=(10, 6))
plt.title('Age vs blood sugar level')
plt.xlabel("Age")
plt.ylabel("Blood Sugar Level")
bucket3 = df3.iloc[:, [0, 9]].values
hull3 = myConvexHull(bucket3)
plt.scatter(bucket3[:, 0], bucket3[:, 1])
for e in hull3:
    plt.plot(hull3[:, 0], hull3[:, 1])
```

✓ 0.1s Python

```
(442, 10)
```



D. Alamat Drive

Drive: https://drive.google.com/drive/folders/1k_I6AMrPUFXo2ZV-vmL0qAJWhumiGhGx?usp=sharing

Github: https://github.com/DzakyFattan/dnc-myConvexHull/

Link di atas akan menampilkan drive berisi kode program, executable, testcase, serta file laporan ini.

E. Check 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.	√	