IMPLEMENTASI CONVEX HULL UNTUK VISUALISASI TES *LINEAR SEPARABILITY DATASET* DENGAN ALGORITMA *DIVIDE AND CONQUER*

Laporan Tugas Kecil II

Disusun sebagai syarat kelulusan mata kuliah
IF2211/Strategi Algoritma

Oleh
HANA FATHIYAH
NIM : 13520047

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO & INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
Februari 2022

DAFTAR ISI

ALGORITMA DIVIDE AND CONQUER **2**

Definisi Divide and Conquer 3

Pemanfaatan Algoritma Divide and Conquer dalam Convex Hull 3

Implementasi Algoritma Divide and Conquer pada Convex Hull dalam Tugas Kecil Strategi Algoritma ke-2 4

KODE PROGRAM **5**

main.ipynb 5

file_myConvexHull.py 8

file_convex_hull_atas.py 12

file_convex_hull_bawah.py 14

file_membuat_area.py 17

file_jarak_titik_ke_garis.py 18

file_jarak_titik_ke_titik.py 19

file_titik_menjadi_indeks.py 19

file_membuat_tipe_numpy.py 20

SCREENSHOT INPUT DAN OUTPUT PROGRAM **20**

Pasangan Atribut (Petal-Length, Petal-Width) pada Datasets Iris 21

Pasangan Atribut (Sepal-Length, Sepal-Width) pada Datasets Iris 22

Pasangan Atribut (Alcohol vs Malic_Acid) pada Datasets Wine 23

Pasangan Atribut (Mean Radius vs Mean Area) pada Datasets Breast_Cancer 24

TABEL KELENGKAPAN KOMPONEN TUGAS **25**

LINK REPOSITORY GITHUB **26**

ALGORITMA *DIVIDE AND CONQUER*

I.1 Definisi *Divide and Conquer*

Dilansir dari Algoritma Divide and Conquer (itb.ac.id), *divide* berarti membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula, tetapi berukuran lebih kecil (idealnya berukuran hampir sama. *Conquer (solve)* berarti menyelesaikan masing-masing upa persoalan secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar. *Combine* berarti menggabungkan solusi masing-masing upa-persoalan, sehingga membentuk solusi persoalan semula

I.2 Pemanfaatan Algoritma *Divide and Conquer* dalam Convex Hull

Dilansir dari Convex Hull (itb.ac.id), salah satu hal penting dalam komputasi geometri adalah menentukan convex hull dari kumpulan titik. 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.

Untuk dua titik, maka convex hull berupa garis yang menghubungkan 2 titik tersebut. Untuk tiga titik yang terletak pada satu garis, maka convex hull adalah sebuah garis yang menghubungkan dua titik terjauh. Convex hull untuk tiga titik yang tidak terletak pada satu garis adalah sebuah segitiga yang menghubungkan ketiga titik tersebut. Untuk titik yang lebih banyak dan tidak terletak pada satu garis, maka convex hull berupa poligon convex dengan sisi berupa garis yang menghubungkan beberapa titik pada S.

Ide dasar dari convex hull ini menggunakan algoritma QuickSort. Kumpulan titik tersebut diurutkan berdasarkan nilai absis yang menaik, jika ada nilai abscess yang sama, diurutkan dengan nilai ordinat yang menaik.

I.3 Implementasi Algoritma *Divide and Conquer* pada Convex Hull dalam Tugas Kecil Strategi Algoritma ke-2

Implementasi algoritma *divide and conquer* pada convex hull dalam tugas kecil strategi algoritma ke-2 ini dituliskan di dalam *file* file_myConvexHull.py. Ide pengerjaannya dijabarkan di dalam poin-poin berikut ini.

1. Mengurutkan titik berdasarkan absis, jika ditemukan titik yang dengan absis yang sama, diurutkan berdasarkan ordinatnya.

   Pengurutan titik ini menggunakan algoritma quick_sort. Titik yang diurutkan adalah titik koordinat yang terdapat di dalam *datasets*

2. Pilih titik yang memiliki urutan pertama dan terakhir, bentuk suatu garis yang membagi area menjadi dua bagian (atas dan bawah)
3. Cari titik terjauh di area atas dan cari titik terjauh di area bawah, hubungkan kedua titik tersebut dengan titik-titik yang sudah ada di dalam himpunan penyelesaian convex hull

   Pencarian titik terjauh dilakukan dengan menggunakan fungsi jarak_titik_ke_garis

4. Lakukan secara rekursif hingga mencapai basis (tidak ada titik di atas garis maupun di bawah garis)
5. Diperoleh himpunan penyelesaian berupa kumpulan titik, urutkan kembali berdasarkan absisnya, kemudian cari indeksnya menggunakan fungsi titik_menjadi_indeks
6. Tambahkan setiap pasangan indeks titik tersebut ke dalam *list* hasil
7. Convex hull siap untuk divisualisasikan pada *file* notebook

KODE PROGRAM

Dalam tugas kecil Strategi Algoritma ke-2 ini, program dibuat dengan dibagi menjadi ke dalam beberapa *file* untuk setiap utilitas yang diperlukan. Program utama terdapat pada *file* main.ipynb dan file_myConvexHull.py. Selain itu, terdapat berbagai *file* lain untuk menunjang kebutuhan pemrograman. Secara lebih lengkap, *source code* dapat diakses di [hanafathiyah/Tugas-Kecil-Stima-2 (github.com)]

I.4 main.ipynb

I.4.1 *Import datasets* iris melalui sklearn dan disimpan di dalam *dataframe* df

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


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()
```

I.4.2 Visualisasi data untuk *datasets* iris bagian Petal Width vs Petal Length

```
import matplotlib.pyplot as plt
from file_myConvexHull import myConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
```

```
plt.title('Petal Width vs Petal 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
    result = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1],
label=data.target_names[i])
    for simplex in result:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1],
colors[i])
plt.legend()
```

from file_myConvexHull *import* myConvexHull dijalankan untuk memanggil fungsi myConvexHull yang telah dibuat menggunakan algoritma *divide and conquer* sebelumnya

I.4.3 Visualisasi data untuk *datasets* iris bagian Sepal Width vs Sepal Length

```
import matplotlib.pyplot as plt
from file_myConvexHull import myConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Sepal Width vs Sepal 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
    result = myConvexHull(bucket)
```

```
    plt.scatter(bucket[:, 0], bucket[:, 1],
label=data.target_names[i])
    for simplex in result:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1],
colors[i])
plt.legend()
```

from file_myConvexHull *import* myConvexHull dijalankan untuk memanggil fungsi myConvexHull yang telah dibuat menggunakan algoritma *divide and conquer* sebelumnya

I.4.4 *Import datasets* wine melalui sklearn dan disimpan di dalam *dataframe* df2 (BONUS)

```
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()
```

I.4.5 Visualisasi data untuk *datasets* wine bagian alcohol vs malic_acid (BONUS)

```
import matplotlib.pyplot as plt
from file_myConvexHull import myConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('alcohol vs malic_acid')
plt.xlabel(data2.feature_names[0])
plt.ylabel(data2.feature_names[1])
```

```
for i in range(len(data2.target_names)):
    bucket = df2[df2['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    result = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1],
label=data2.target_names[i])
    for simplex in result:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1],
colors[i])
plt.legend()
```

I.5 file_myConvexHull.py

```
import numpy as np
from file_membuat_area import membuat_area
from file_membuat_tipe_numpy import membuat_tipe_numpy
from file_convex_hull_atas import fungsi_convex_hull_atas
from file_convex_hull_bawah import
fungsi_convex_hull_bawah
from file_titik_menjadi_indeks import titik_menjadi_indeks


def myConvexHull(array_titik):
    if len(array_titik) <= 2:
        return array_titik


  # membuat array kosong untuk convex hull (menyimpan
dalam bentuk titik)
    hasil_convex_hull = []
```

```
    urutan_x = sorted(array_titik, key=lambda x: x[0])
    titik1 = urutan_x[0] # Titik minimum
    titik2 = urutan_x[-1] # Titik maksimum

    hasil_convex_hull = hasil_convex_hull + [titik1,
titik2]

  # menghapus dari list setelah dimasukkan ke dalam proses
convex hull
    urutan_x.pop(0)
    urutan_x.pop(-1)

  #  dnc pertama, membagi 2 area menjadi atas_garis dan
bawah_garis
    atas_garis, bawah_garis = membuat_area(titik1, titik2,
urutan_x)

  # membagi dua hasil convex_hull, yaitu bagian atas dan
bagian bawah
    hasil_convex_hull_atas = []
    hasil_convex_hull_bawah = []
    hasil_convex_hull_atas = hasil_convex_hull +
fungsi_convex_hull_atas(titik1, titik2, atas_garis)
    hasil_convex_hull_bawah = hasil_convex_hull +
fungsi_convex_hull_bawah(titik1, titik2, bawah_garis)

    hasil_convex_hull_atas =
sorted(hasil_convex_hull_atas, key=lambda x: (x[0],
-x[1]))
```

```
    hasil_convex_hull_bawah =
sorted(hasil_convex_hull_bawah, key=lambda x: (x[0],
x[1]))


    hasil_convex_hull_atas_as_numpy =
membuat_tipe_numpy(hasil_convex_hull_atas)
    hasil_convex_hull_bawah_as_numpy =
membuat_tipe_numpy(hasil_convex_hull_bawah)


    return_value_atas = []
    return_value_bawah = []


    for i in range(len(hasil_convex_hull_atas_as_numpy) -
1):
        return_value_atas = return_value_atas +
[[titik_menjadi_indeks(hasil_convex_hull_atas_as_numpy[i],
array_titik),titik_menjadi_indeks(hasil_convex_hull_atas_a
s_numpy[i+1], array_titik)]]


    for i in range(len(hasil_convex_hull_bawah_as_numpy) -
1):
        return_value_bawah = return_value_bawah +
[[titik_menjadi_indeks(hasil_convex_hull_bawah_as_numpy[i]
,
array_titik),titik_menjadi_indeks(hasil_convex_hull_bawah_
as_numpy[i+1], array_titik)]]


    return return_value_atas + return_value_bawah
```

Terdapat beberapa proses yang terjadi pada *file* file_myConvexHull.py tersebut. Proses pertama adalah membuat suatu array kosong yang bernama hasil_convex_hull. Variabel array_titik yang terdapat di fungsi ini adalah suatu

array yang berisi kumpulan *list* titik-titik yang terdapat di dalam *datasets* tersebut. Variabel urutan_x menyimpan urutan dari array_titik tersebut sebagai awalan dari keberjalanan program Convex Hull ini, yaitu semua titik diurutkan berdasarkan absisnya, tetapi jika ada absis yang sama, titik tersebut diurutkan berdasarkan ordinatnya. Setelah itu, terdapat variabel titik1 yang merupakan titik dengan urutan paling awal dan titik2 yang merupakan titik dengan urutan paling akhir. Kedua titik tersebut kemudian dimasukkan ke dalam array hasil_convex_hull. Titik yang telah dimasukkan kemudian dihapus dari urutan_x.

Proses *divide and conquer* pada *file* file_myConvexHull.py ini diawali dengan membagi dua area menjadi area atas_garis dan area bawah_garis. Area atas_garis adalah area yang berada di atas garis yang dihubungkan oleh titik1 dan titik2, sedangkan area bawah_garis adalah area yang berada di bawah garis yang dihubungkan oleh titik1 dan titik2.

Selanjutnya, dibentuk dua array kosong, yaitu hasil_convex_hull_atas yang akan diisi oleh himpunan penyelesaian titik berdasarkan proses convex hull yang berada di atas garis, serta hasil_convex_hull_bawah yang akan diisi oleh himpunan penyelesaian titik berdasarkan proses convex hull yang berada di bawah garis. Kedua array tersebut diisikan oleh fungsi yang bernama fungsi_convex_hull_atas dan fungsi_convex_hull_bawah untuk menghasilkan titik-titik yang menjadi himpunan penyelesaian dalam kedua array tersebut.

Selanjutnya, array hasil_convex_hull_atas dan array hasil_convex_hull_bawah tersebut diurutkan kembali sesuai absisnya. Jika ada dua buah titik dengan absis sama, untuk hasil_convex_hull_atas dipilih ordinat yang paling besar, sedangkan untuk hasil_convex_hull_bawah dipilih ordinat yang paling kecil.

Setelah itu, dibuat array return_value_atas dan return_value_bawah untuk menyimpan pasangan titik-titik yang sudah ada tersebut dalam bentuk pasangan indeks. Titik yang sudah ada diubah menjadi indeks dengan fungsi titik_menjadi_indeks. Fungsi myConvexHull ini akan me-*return* nilai

return_value_bawah digabung dengan return_value_atas yang berisi kumpulan titik dalam bentuk pasangan indeks yang dihubungkan oleh garis dan merupakan hasil operasi convex hull.

I.6 file_convex_hull_atas.py

```
import numpy as np
from file_membuat_area import membuat_area
from file_jarak_titik_ke_garis import jarak_titik_ke_garis


def fungsi_convex_hull_atas(titik1, titik2, atas_garis):


    # mengecek apakah salah satu dari ketiga elemen
tersebut kosong atau tidak
    if (atas_garis == [] or titik1 is None or titik2 is
None):
        return atas_garis


    convex_hull_atas = []


    # menghitung jarak setiap titik dari garis dan mencari
titik dengan jarak terjauh
    jarak_terjauh = -1
    titik_terjauh = None


    indeks_titik = 0
    indeks_titik_terjauh = -1


    for titik in atas_garis:
```

```
        jarak_titik_i_ke_garis =
jarak_titik_ke_garis(titik1, titik2, titik)
        if (jarak_titik_i_ke_garis > jarak_terjauh):
            jarak_terjauh = jarak_titik_i_ke_garis
            titik_terjauh = titik
            indeks_titik_terjauh = indeks_titik
        indeks_titik += 1

    convex_hull_atas = convex_hull_atas + [titik_terjauh]

    # menghapus titik terjauh yang sudah terdata
    np.delete(atas_garis, indeks_titik_terjauh)

    # membuat area
    titik1atas, titik1bawah = membuat_area(titik1,
titik_terjauh, atas_garis)
    titik2atas, titik2bawah = membuat_area(titik_terjauh,
titik2, atas_garis)

    convex_hull_atas = convex_hull_atas +
fungsi_convex_hull_atas(titik1, titik_terjauh, titik1atas)
    convex_hull_atas = convex_hull_atas +
fungsi_convex_hull_atas(titik_terjauh, titik2, titik2atas)

    return convex_hull_atas
```

File file_convex_hull_atas.py ini berisi fungsi_convex_hull_atas yang mengembalikan hasil berupa himpunan titik-titik yang berada di atas garis utama dan merupakan penyelesaian setelah dilakukan proses convex hull. Di dalamnya

pun terdapat pemanggilan secara rekursif sampai tidak ditemukan lagi titik yang berada di atas garis tersebut.

Pada setiap proses, dilakukan pengecekan titik apa yang memiliki jarak terjauh dan berada di atas garis yang sedang dianalisis. Untuk setiap titik pada setiap titik yang berada di atas garis tersebut, dilakukan perhitungan jarak hingga ditemukan titik apa yang terjauh. Setelah itu, titik terjauh dimasukkan ke dalam array convex_hull_atas. Titik terjauh kemudian dihapus dan dilakukan proses selanjutnya.

Setelah itu, dibuat dua buah area, yaitu area titik1atas dan titik2atas. Titik1atas berisi titik-titik yang berada di atas garis yang terbentuk dari titik1 dan titikterjauh. Titik2atas berisi titik-titik yang berada di bawah garis yang terbentuk dari titikterjauh dan titik2. Karena fungsi_convex_hull_atas ini bersifat rekurens (sesuai definisi *divide and conquer*), dilakukan pemanggilan kembali fungsi_convex_hull_atas dan hasilnya ditambahkan ke dalam output convex_hull_atas.

I.7 file_convex_hull_bawah.py

```
import numpy as np
from file_membuat_area import membuat_area
from file_jarak_titik_ke_garis import jarak_titik_ke_garis


def fungsi_convex_hull_bawah(titik1, titik2, bawah_garis):


    # mengecek apakah salah satu dari ketiga elemen
tersebut kosong atau tidak
    if (bawah_garis == [] or titik1 is None or titik2 is
None):
        return bawah_garis
```

```
    convex_hull_bawah = []

    # menghitung jarak setiap titik dari garis dan mencari
titik dengan jarak terjauh
    jarak_terjauh = -1
    titik_terjauh = None

    indeks_titik = 0
    indeks_titik_terjauh = -1

    for titik in bawah_garis:
        jarak_titik_i_ke_garis =
jarak_titik_ke_garis(titik1, titik2, titik)
        if (jarak_titik_i_ke_garis > jarak_terjauh):
            jarak_terjauh = jarak_titik_i_ke_garis
            titik_terjauh = titik
            indeks_titik_terjauh = indeks_titik
        indeks_titik += 1

    convex_hull_bawah = convex_hull_bawah +
[titik_terjauh]

    # menghapus titik terjauh yang sudah terdata
    np.delete(bawah_garis, indeks_titik_terjauh)

    # membuat area
    titik1atas, titik1bawah = membuat_area(titik1,
titik_terjauh, bawah_garis)
```

```
    titik2atas, titik2bawah = membuat_area(titik_terjauh,
titik2, bawah_garis)


    convex_hull_bawah = convex_hull_bawah +
fungsi_convex_hull_bawah(titik1, titik_terjauh,
titik1bawah)
    convex_hull_bawah = convex_hull_bawah +
fungsi_convex_hull_bawah(titik_terjauh, titik2,
titik2bawah)


    return convex_hull_bawah
```

File file_convex_hull_bawah.py ini berisi fungsi_convex_hull_bawah yang mengembalikan hasil berupa himpunan titik-titik yang berada di bawah garis utama dan merupakan penyelesaian setelah dilakukan proses convex hull. Di dalamnya pun terdapat pemanggilan secara rekursif sampai tidak ditemukan lagi titik yang berada di bawah garis tersebut.

Pada setiap proses, dilakukan pengecekan titik apa yang memiliki jarak terjauh dan berada di bawah garis yang sedang dianalisis. Untuk setiap titik pada setiap titik yang berada di bawah garis tersebut, dilakukan perhitungan jarak hingga ditemukan titik apa yang terjauh. Setelah itu, titik terjauh dimasukkan ke dalam array convex_hull_bawah. Titik terjauh kemudian dihapus dan dilakukan proses selanjutnya.

Setelah itu, dibuat dua buah area, yaitu area titik1bawah dan titik2bawah. Titik1bawah berisi titik-titik yang berada di bawah garis yang terbentuk dari titik1 dan titikterjauh. Titik2bawah berisi titik-titik yang berada di bawah garis yang terbentuk dari titikterjauh dan titik2. Karena fungsi_convex_hull_bawah ini bersifat rekurens (sesuai definisi *divide and conquer*), dilakukan pemanggilan kembali fungsi_convex_hull_bawah dan hasilnya ditambahkan ke dalam output convex_hull_bawah.

I.8 file_membuat_area.py

```
def membuat_area(titik1, titik2, array_titik):


    area_atas = []
    area_bawah = []


    # kemungkinan 1: garis vertikal sehingga di atas dan
di bawah garis tidak ada titik
    if (titik1[0] == titik2[0]):
        return area_atas, area_bawah


    # mencari kemiringan dan titik potong sumbu-y dengan y
= mx + c
    # m = (y2 - y1)/(x2 - x1)
    # m = (titik2[1] - titik1[1]) /(titik2[0] - titik1[0])
    # c = - mx + y
    # (y - y1) / (x2 - x1)
    # c = -1 * m * titik1[0] + titik1[1]


    for sumbu in array_titik:
        y = sumbu[1]
        x = sumbu[0]
        y2 = titik2[1]
        y1 = titik1[1]
        x2 = titik2[0]
        x1 = titik1[0]
```

```
        # deteksi letak suatu titik
        if (y - y1) * (x2 - x1) > (x - x1) * (y2 - y1):
            area_atas.append(sumbu)
        elif (y - y1) * (x2 - x1) < (x - x1) * (y2 - y1):
            area_bawah.append(sumbu)

    return area_atas, area_bawah
```

File file_membuat_area.py ini memiliki dua parameter *output,* yaitu area atas dan area bawah. Pencarian area ini memanfaatkan materi matematika, yaitu persamaan garis lurus. Untuk (y - y1) * (x2 - x1) > (x - x1) * (y2 - y1), titik (x,y) berada di atas garis yang tersusun atas titik (x1,y1) dan(x2,y2). Untuk (y - y1) * (x2 - x1) < (x - x1) * (y2 - y1), titik (x,y) berada di bawah garis yang tersusun atas titik (x1,y1) dan (x2,y2). Titik-titik tersebut di-*append* sesuai dengan letaknya terhadap garis.

I.9 file_jarak_titik_ke_garis.py

```
import numpy as np

def jarak_titik_ke_garis(titik1, titik2, titik3):

    titik_1_as_numpy = np.array(titik1)
    titik_2_as_numpy = np.array(titik2)
    titik_3_as_numpy = np.array(titik3)

    vektor_a = titik_3_as_numpy - titik_1_as_numpy
    vektor_b = titik_2_as_numpy - titik_1_as_numpy
```

```
            vektor_c   =   np.sum(vektor_a   *   vektor_b)   /
np.sum(vektor_b * vektor_b) * vektor_b - vektor_a


    return np.sum(vektor_c * vektor_c)
```

Dalam program ini, proses pencarian jarak titik ke garis memanfaatkan rumus vektor untuk mempermudah perhitungan. Dengan memanfaatkan vektor satuan, *dot product,* dan besar vektor, dapat diketahui besar vektor c yang menjadi jarak titik ke garis.

I.10 file_jarak_titik_ke_titik.py

```
import numpy as np


def jarak_titik_ke_titik(titik1, titik2):
        return  np.sqrt((titik1[0]  -  titik2[0])  **  2  +
(titik1[1] - titik2[1]) ** 2)
```

Dalam program ini, jarak titik ke titik dicari seperti biasa, yaitu menggunakan rumus $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

I.11 file_titik_menjadi_indeks.py

```
def titik_menjadi_indeks(titik, array_of_titik):
    for i in range(len(array_of_titik)):
          if(titik[0] == array_of_titik[i][0] and titik[1]
== array_of_titik[i][1]):
            return i
```

File file_titik_menjadi_indeks.py berisi fungsi titik_menjadi_indeks yang merubah suatu titik menjadi indeksnya di dalam array_of_titik

I.12 file_membuat_tipe_numpy.py

```
import numpy as np


def membuat_tipe_numpy(elemen):
    hasil = np.array(elemen)
    return hasil
```

File *file_membuat_tipe_numpy.py* ini dibuat untuk memudahkan *typecasting* antara *list* pada python menjadi *numpy.*

SCREENSHOT INPUT DAN *OUTPUT* PROGRAM

II.1 Pasangan Atribut (Petal-Length, Petal-Width) pada *Datasets* Iris

II.1.1 Input

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

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()
```

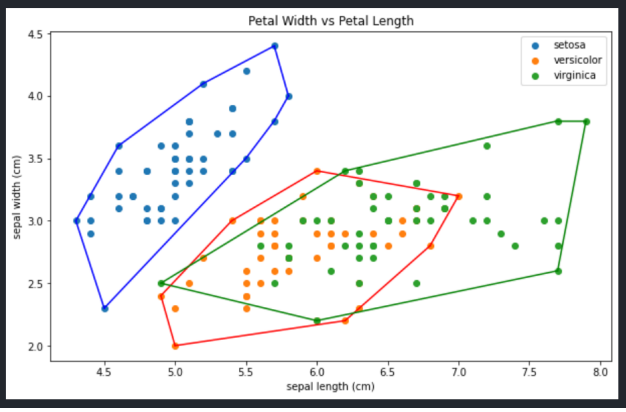
✓ 1.1s Python

```python
import matplotlib.pyplot as plt
from file_myConvexHull import myConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal 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
    result = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in result:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

✓ 0.4s Python

II.1.2 Output

```
<matplotlib.legend.Legend at 0x7f6e19996a90>
```



II.2 Pasangan Atribut (Sepal-Length, Sepal-Width) pada *Datasets* Iris

II.2.1 Input

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

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()
```

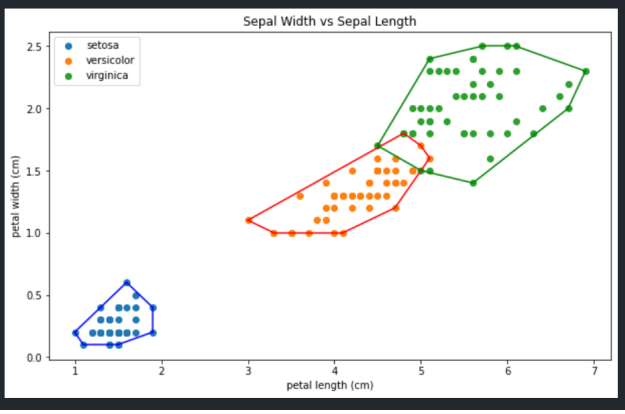
✓ 1.1s Python

```python
import matplotlib.pyplot as plt
from file_myConvexHull import myConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Sepal Width vs Sepal 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
    result = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in result:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

[3] ✓ 0.2s Python

II.2.2 Output

```
<matplotlib.legend.Legend at 0x7f6e1788c1f0>
```



II.3 Pasangan Atribut (Alcohol vs Malic_Acid) pada *Datasets* Wine

II.3.1 Input

```
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()
```

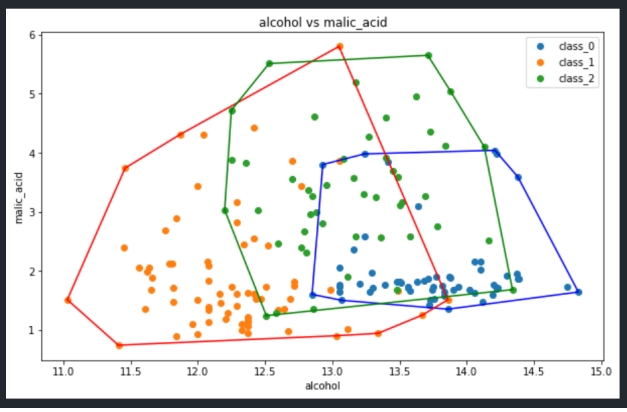
[4] ✓ 0.1s Python

```
import matplotlib.pyplot as plt
from file_myConvexHull import myConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('alcohol vs malic_acid')
plt.xlabel(data2.feature_names[0])
plt.ylabel(data2.feature_names[1])
for i in range(len(data2.target_names)):
    bucket = df2[df2['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    result = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data2.target_names[i])
    for simplex in result:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

[5] ✓ 0.2s Python

II.3.2 Output

```
... <matplotlib.legend.Legend at 0x7f6e177c6730>
```



II.4 Pasangan Atribut (Mean Radius vs Mean Area) pada *Datasets* Breast_Cancer

II.4.1 Input

```
data3 = datasets.load_breast_cancer()
#create a DataFrame
df3 = pd.DataFrame(data3.data, columns=data3.feature_names)
df3['Target'] = pd.DataFrame(data3.target)
print(df3.shape)
df3.head()
```

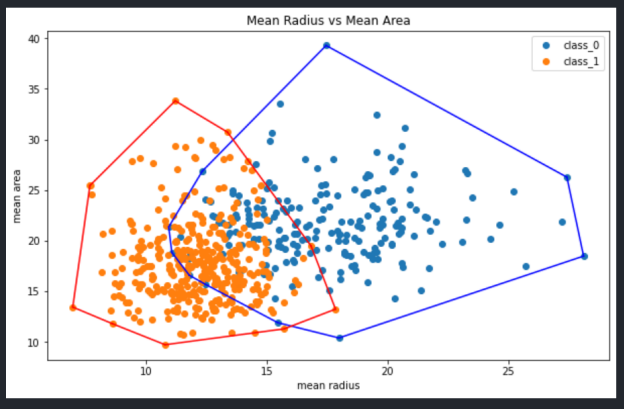
[6] ✓ 0.6s Python

```
import matplotlib.pyplot as plt
from file_myConvexHull import myConvexHull
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Mean Radius vs Mean Area')
plt.xlabel(data3.feature_names[0])
plt.ylabel(data3.feature_names[3])
for i in range(len(data3.target_names)):
    bucket = df3[df3['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    result = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data2.target_names[i])
    for simplex in result:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
plt.legend()
```

[7] ✓ 0.2s Python

II.4.2 Output

```
... <matplotlib.legend.Legend at 0x7fa7b5935610>
```



TABEL KELENGKAPAN KOMPONEN TUGAS

Poin	Ya	Tidak
1. Pustaka myConvexHull berhasil dibuat dan tidak ada kesalahan	✔	
2. *ConvexHull* 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	✔	

LINK REPOSITORY GITHUB

https://github.com/hanafathiyah/Tugas-Kecil-Stima-2