Laporan Tugas Kecil 2 IF2211 Strategi Algoritma

Semester II Tahun Akademik 2021/2022

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

Disusun oleh :

Muhammad Gilang Ramadhan
13520137
K-02

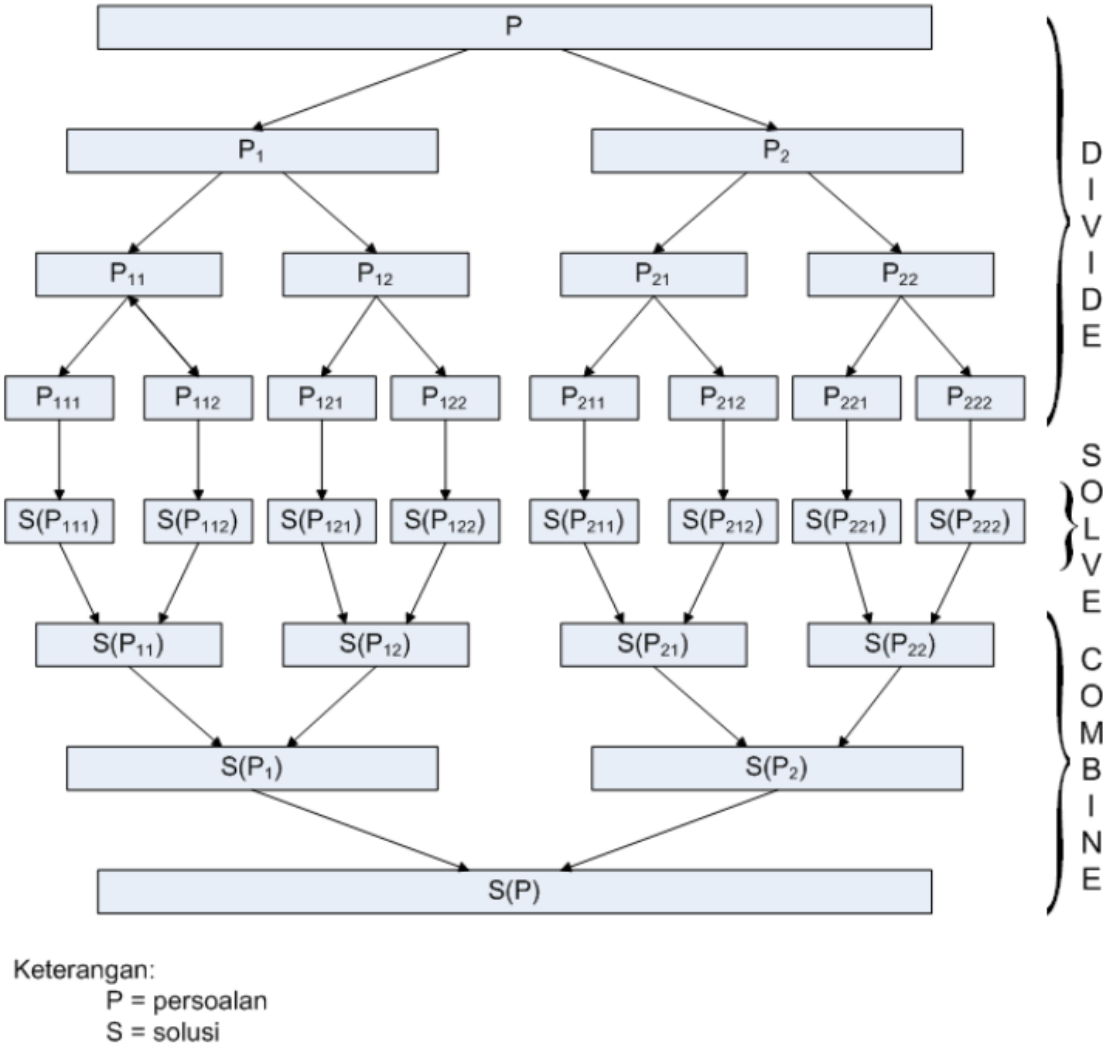
Program Studi S1 Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2022

BAB 1

Algoritma Divide and Conquer pada Convex Hull

Definisi Algoritma Divide and Conquer

Divide and Qouquer adalah suatu algoritma yang memecahkan masalah dengan cara membagi masalah tersebut menjadi beberapa upa-masalah yang lebih kecil dan sederhana, kemudian masing-masing upa-masalah tersebut diselesaikan secara independent kemudian solusi dari masing-masing upa-masalah tersebut digabung sehingga menghasilkan solusi dari permasalahan semula. Adapun pada gambar di bawah ini disajikan ilustrasi mengenai cara kerja *Algoritma Divide and Conquer*.

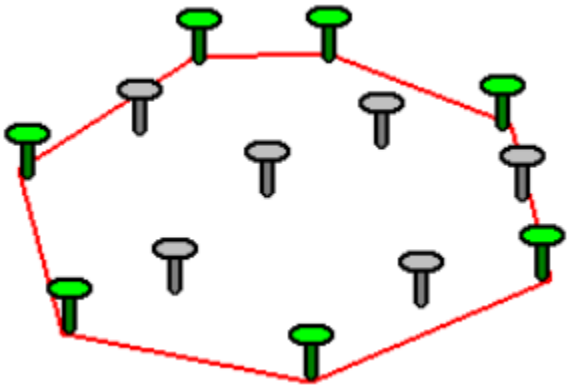


Gambar 1.1 Ilustrasi algoritma *Divide and Conquer*

(*sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Divide-and-Conquer-(2018)*)

Algoritma Divide and Conquer untuk menyelesaikan masalah *Convex Hull*

Convex Hull merupakan permasalahan yang klasik dalam bidang geometri komputasional. *Convex Hull* sendiri didefinisikan sebagai himpunan titik terkecil yang mengandung set of titik (koordinat) yang sifatnya *convex*. Suatu set of titik dikatakan *convex* apabila untuk sembarang 2 buah titik yang berada pada set tersebut, semua segmen garis yang berakhir pada dua titik tersebut terdapat pada set tersebut. Adapun contoh ilustrasi dari *Convex Hull*, yaitu sebagai berikut.



Gambar 1.2 Ilustrasi *Convex Hull*

(sumber : https://brilliant.org/wiki/convex-hull/)

Untuk menyelesaikan permasalahan Convex Hull sendiri, bisa dengan bermacam-macam algoritma, diantaranya dengan algoritma *Jarvis's March*, algoritma *Gift Wrapping*, algoritma *Graham Scan*, ataupun dengan algoritma *QuickHull*. Namun, pada Tugas Kecil 2 ini, algoritma yang digunakan adalah Algoritma *QuickHull*, yang mana algoritma ini menggunakan strategi *Divide and Qonquer*. Adapun berikut adalah langkah-langkah untuk mencari *Convex Hull* dengan Algoritma *QuickHull* :

1. Cari titik paling kanan (*Right most*) dan paling kiri (*Left most*) dari suatu himpunan titik.
2. Setelah didapat titik-titik yang saling terkait maka partisi himpunan titik-titik tersebut menjadi dua bagian, yaitu segmen atas dan segmen bawah dengan berdasarkan jarak antara sebuah titik dengan suatu garis.
3. Cari titik lain yang jaraknya terjauh dengan garis yang telah terbentuk tersebut agar nantinya himpunan titik *Convex Hull* tersebut dapat berisi titik-titik yang lain. Adapun cara mencari jarak dari titik ke garis tersebut dapat menggunakan *Cross Product* antara titik dengan garis. Dimana jika hasilnya positif menunjukkan titik berada di atas garis dan jika hasilnya negatif titik berada di bawah garis, tapi kalau hasilnya nol maka titik berada pada garis. Adapun titik ini akan membentuk segitiga dengan garis tersebut.
4. Titik-titik yang berada di dalam segitiga itu bukan merupakan merupakan *vertex* dari *Convex Hull*, sehingga dapat diabaikan pada langkah selanjutnya.
5. Ulangi dua langkah sebelumnya pada dua garis yang dibentuk oleh segitiga (bukan garis awal).
6. Lakukan terus langkah tersebut sampai semua titik telah diproses, rekursi selesai, dan titik-titik yang dipilih merupakan *vertex* dari *Convex Hull*.

Dalam keberjalannya mencari *Convex Hull* adapun beberapa fungsi yang berperan dalam pencarian tersebut, yaitu sebagai berikut.

FUNGSI	KEGUNAAN
_QuickHull(self)	Fungsi ini merupakan fungsi dari Class myConvexHull yang berperan untuk mereturn titik himpunan titik yang merupakan *Convex Hull*
_Mencarihull(self, array_of_koordinat, Vertex1, Vertex2)	Fungsi ini merupakan fungsi rekursif dari Class myConvexHull yang berperan untuk melakukan *Divide and Conquer* sampai semua titik telah diproses.
Menghitung_jarak(mulai, selesai, koordinat, eps=1e-8)	Fungsi ini mereturn jarak dari garis mulaiselesai ke koordinat dengan *cross product*
Pertisi_setOfkoordinat_menjadiDua(left_most, right_most, self.array_of_point)	Fungsi untuk mempartisi sebuah set of koordinat menjadi dua bagian dengan comparing *cross product*
Partisi_Segitiga(Array_of_koordinat, Vertex1, Vertex2, Vertex3)	Fungsi untuk mempartisi satu set of koordinat menjadi 2 buah set of koordinat yang berbentuk segitiga
CLOCK_SORT(koordinat)	Fungsi untuk mensorting *vertices* pada *clockwise* dengan berdasarkan sudutnya, yaitu antara sumbu x dan vektor yang menuju center dari koordinat

BAB 2

Source Code Program

2.1 myConvexHull.py

```python
# NAMA  : Muhammad Gilang Ramadhan
# NIM   : 13520137
# TUCIL 2 IF2211 Strategi Algoritma
# SOLVE CONVEX HULL PROBLEM WITH DIVIDE AND CONQUER STRATEGY

import numpy as np

# Fungsi yang mengembalikan cross product dari koordinat ke segmen garis
# Sekaligus menghitung jarak dari koordinat ke garis
def Menghitung_jarak(mulai, selesai, koordinat, eps=1e-8):
    jarak = np.cross(selesai-mulai,koordinat-mulai)/(np.linalg.norm(selesai-mulai)+eps)
    return jarak

# Fungsi untuk membagi Sebuah set of koordinat menjadi dua daerah dengan comparing cross product
def partisi_setOfkoordinat_menjadiDua(mulai, selesai, array_of_koordinat):
    # Jika koordinat is None atau < 1
    if array_of_koordinat is None or array_of_koordinat.shape[0] < 1:
        return None, None
    # Jika tidak kosong atau > 1
    else:
        temp1 = []
        temp2 = []
        for koordinat in array_of_koordinat:
            jarak = Menghitung_jarak(mulai, selesai, koordinat)
            if jarak > 0:
                temp1.append(koordinat)
            else:
                temp2.append(koordinat)

    # Mengubah dari bentuk koordinat dari temp1 dan temp2 menjadi bentuk array of numpy
    if len(temp1):
        temp1 = np.vstack(temp1)
    else:
        temp1 = None
    if len(temp2):
        temp2 = np.vstack(temp2)
    else:
        temp2 = None
    return temp1, temp2
```

Gambar 2.1.1 myConvexHull.py (line 1-39)

```python
# Fungsi untuk mempartisi satu set of koordinat menjadi 2 set of koordinat yang berbentuk segitiga
def Partisi_Segitiga(Array_of_koordinat, Vertex1, Vertex2, Vertex3):
    # Jika array of koordinat is None
    if Array_of_koordinat is None:
        return None, None
    # Jika array of koordinat tidak kosong
    else:
        # Mencari di sisi mana titik tersebut berada dengan cross product
        temp1 = []
        temp2 = []
        for koordinat in Array_of_koordinat:
            jarakV1V2 = Menghitung_jarak(Vertex1, Vertex2, koordinat)
            jarakV2V3 = Menghitung_jarak(Vertex2, Vertex3, koordinat)
            if jarakV1V2 > 0 and jarakV2V3 < 0:
                temp1.append(koordinat)
            elif jarakV1V2 < 0 and jarakV2V3 > 0:
                temp2.append(koordinat)

    # Mengubah dari bentuk koordinat dari temp1 dan temp2 menjadi bentuk array of numpy
    if len(temp1):
        temp1 = np.vstack(temp1)
    else:
        temp1 = None
    if len(temp2):
        temp2 = np.vstack(temp2)
    else:
        temp2 = None
    return temp1, temp2

# Fungsi untuk mensorting vertices pada clockwise dengan berdasarkan sudutnya
# Yaitu antara sumbu x dan vektor yang menuju center dari koordinat
def CLOCK_SORT(koordinat):
    absis = koordinat[:,0].mean()
    ordinat = koordinat[:,1].mean()
    theta = np.arctan2(koordinat[:,1] - ordinat, koordinat[:,0] - absis)
    indeks = np.argsort(theta)
    koordinat = koordinat[indeks]
    return koordinat
```

Gambar 2.1.2 myConvexHull.py (line 41-78)

```python
# Fungsi yang menghasilkan array of koordinat yang merupakan convex hull yang dihasilkan menggunakan
quick hull
# Quick hull merupakan salah satu cara untuk mencari convex hull dengan pendekatan divide and conquer
strategy
class myConvexHull:
    def __init__(self):
        self.array_of_koordinat = None
        self.convext_hull = []

    def reset(self):
        self.array_of_koordinat = None
        self.convext_hull = []

    def forward(self, koordinat_set):
        if koordinat_set is None or len(koordinat_set) < 3:
            print()
            print("Convex Hull yang ditemukan tidak valid! Mohon sediakan lebih dari 3 array of koordinat
yang unik")
            return None

        self.reset()
        # Menghilangkan elemen yang duplicate
        self.array_of_koordinat = np.unique(koordinat_set, axis=0)
        return self._QuickHull()

    def __call__(self, koordinat_set):
        return self.forward(koordinat_set)

    def _QuickHull(self):
        # sort the data by x-axis, then by y-axis
        self.array_of_koordinat =
self.array_of_koordinat[np.lexsort(np.transpose(self.array_of_koordinat)[::-1])]
        # Mencari left-most of koordinat dan right-most of koordinat
        left_most, right_most = self.array_of_koordinat[0], self.array_of_koordinat[-1]
        # Mendapatkan rest of array_of_koordinat
        self.array_of_koordinat = self.array_of_koordinat[1:-1]
        #Menambahkan left-most koordinat ke dalam output
        self.convext_hull.append(left_most)
        #Menambahkan right-most koordinat ke dalam output
        self.convext_hull.append(right_most)

        self.right_array_of_koordinat, self.left_array_of_koordinat =
partisi_setOfkoordinat_menjadiDua(left_most, right_most, self.array_of_koordinat)

        self._Mencarihull(self.right_array_of_koordinat, left_most, right_most)
        self._Mencarihull(self.left_array_of_koordinat, right_most, left_most)

        self.convext_hull = np.stack(self.convext_hull)
        self.convext_hull = CLOCK_SORT(self.convext_hull)
        if self.convext_hull.shape[0] >= 3:
            return self.convext_hull
        else:
            print()
            print("Array of koordinat yang ditemukan tidak cukup untuk convex hull. Mohon cek lagi input
anda!")
            return None

    def _Mencarihull(self, array_of_koordinat, Vertex1, Vertex2):
        if array_of_koordinat is None:
            return None
        else:
            jarak = 0.0
            koordinat_sekarang = None
            indeks = None
            for i, koordinat in enumerate(array_of_koordinat):
                jarak_sekarang = abs(Menghitung_jarak(Vertex1, Vertex2, koordinat))
                # Mencari jarak yang berjarak maksimum dari PQ yang berasal dari array_of_koordinat
                if jarak_sekarang > jarak:
                    koordinat_sekarang = koordinat
                    indeks = i
                    jarak = jarak_sekarang
            if koordinat_sekarang is None:
                print()
                print("input array of koordinat berada di line yang sama. Tidak ada convex hull yang
ditemukan!")
                return None
            else:
                self.convext_hull.append(koordinat_sekarang)
                # Delete koordinat sekarang dari array of koordinat yang original
                array_of_koordinat = np.delete(array_of_koordinat, indeks, axis=0)

            # Mereturn hasil partisi segitiga ke temp1 dan temp2
            temp1, temp2 = Partisi_Segitiga(array_of_koordinat, Vertex1, koordinat_sekarang, Vertex2)
            self._Mencarihull(temp1, Vertex1, koordinat_sekarang)
            self._Mencarihull(temp2, koordinat_sekarang, Vertex2)
```

Gambar 2.1.3 myConvexHull.py (line 80-157)

2.2 Visualisasi.py

```python
from myConvexHull import *
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
import random

# Buat Visualisasi

#inisialisasi variabel global
convex_hull = myConvexHull()

# Fungsi untuk pewarnaan
def Warna(n):
    warna = ['m', 'g', 'y', 'k', 'b', 'c', 'w', 'r']
    if n > len(warna):
        for i in (range(n-len(warna))):
            r = random.random()
            g = random.random()
            b = random.random()
            warna.append((r, g, b))
    return warna

# Fungsi untuk memilih dataset
def load_datasets(nomor_dataset):
    if nomor_dataset == 1:
        data = datasets.load_iris()
    elif nomor_dataset == 2:
        data = datasets.load_wine()
    elif nomor_dataset == 3:
        data = datasets.load_breast_cancer()
    elif nomor_dataset == 4:
        data = datasets.load_diabetes()
    elif nomor_dataset == 5:
        data = datasets.load_digits()
    df = pd.DataFrame(data.data, columns=data.feature_names)
    df['target'] = pd.DataFrame(data.target)
    return df, data

# Fungsi untuk menampilkan tabel dataset
def tampilkan_tabel_dataset(df):
    print(df.shape)
    print(df.head)

# Fungsi untuk menampilkan plot grafik
def tampilkan_plot_grafik(df, data):
    sumbu_x = int(input("Masukkan Kolom untuk sumbu-x : "))
    sumbu_y = int(input("Masukkan Kolom untuk sumbu-y : "))
    savefile = input("Masukkan nama file yang ingin disave : ")
    plt.figure(figsize = (10, 6))
    UkuranLabel = len(df['target'].unique())
    colors = Warna(UkuranLabel)
    plt.title(str(data.feature_names[sumbu_x])+" vs "+str(data.feature_names[sumbu_y]))
    plt.xlabel(data.feature_names[sumbu_x])
    plt.ylabel(data.feature_names[sumbu_y])
    print("Vertex dari convex hull adalah : ")
    for i in range(UkuranLabel):
        bucket = df[df['target'] == i]
        bucket = bucket.iloc[:,[sumbu_x,sumbu_y]].values
        convexhull = convex_hull(bucket)
        plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i], color=colors[i])
        print(convexhull)
        convexhull = np.vstack([convexhull, convexhull[0]])
        plt.plot(convexhull[:,0], convexhull[:,1], colors[i])
    plt.legend()
    plt.savefig('output/' + savefile)
    plt.show()
```

Gambar 2.2.1 Visualisasi.py (line 1-66)

2.3 MainProgram.py

```
from myConvexHull import *
from Visualisasi import *

# Print Logo
def printlogo():
    print("[]=============================================================================================
[]")
    print("||    _ _             _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _      _ _ _ _ _ _ _ _ _
||")
    print("||   /   |           |  \    __ _|   |    _ _ _ _|    \    \    /    |    __ _ _|
||")
    print("||  /   /    _ __     \  \  /_ __    |   |  /   _ _ _   \    \_ _/     |   /_ _ __
||")
    print("||  \   \    /    \    /   /        |   |   | /   /    \   \            |          |
||")
    print("||   \   \ /  /\  \  /   /   _ _ __|   |   | \   \_ _ _/   / |\      /|   |    _ _ _ _|
||")
    print("||    \   \/  /__\  \/   /    \_ _ _    |_ _|_ \_ __       /  | \_ _ _/ |   |    \_ _ __
||")
    print("||     \_ _ _ _ _ _ _ __/_ _ _ _ _ _|_ _ _ _ _|_ _ _ _|_ _ _/_ _|        |_ _|__ _ _ _ _ |
||")
    print("||                                _ _ _ _ _ ____ _ _ _ _
||")
    print("||                              |_ __    _ __|       \
||")
    print("||                                   |  |  /    _ _    \
||")
    print("||                                   |  | /    /   \    \
||")
    print("||                                   |  | \    \_ _/    /
||")
    print("||                                   |  |  \           /
||")
    print("||                                   |__|   \ _ _ _ __/
||")
    print("||   _ _ _ _ _ _ _ _ _ _      _ _ _          _ _ _ _ _ _ _        _ _ _
||")
    print("||  |    _ _ _|      \    \    |  | \        /    /  _ _ _ |\     /    /
||")
    print("||  |   |   /   _ _   \    \   |  |  \      /    /  /_ _ \__  \_ _/    /
||")
    print("||  |   | /   /   \   \    \ |  |   \    /    /          |  _ _    /
||")
    print("||  |   |  \   \_ _/   /     \ |   |\    \_ /    /     _ _ __| /   \   \
||")
    print("||  |   |_ _\_        /       \|   | \          /     \_ _/__ /     \   \
||")
    print("||  |_ _ _ _ _|_ _ _ /|_ _|\_ _ _ _|  \_ _ _ __/_ _ _ _ _ _ _|       \_ _\
||")
    print("||                                              _ __      _ _ _ _       _ _ _ _ _ _
||")
    print("||                                            |    |    |   |   |     |   |   |   |
||")
    print("||                                            |    |    |   |   |     |   |   |   |
||")
    print("||                                            |    |_ __|   |   |     |   |   |   |
||")
    print("||                                            |     _ __     |   |     |   |   |   |
||")
    print("||                                            |    |    |    |   |_ _ _|   |   |   |
||")
    print("||                                            |    |    |    |             |   |_ _|_ _ _
||")
    print("||                                            |_ __|    |_ _|_ _ _ _ _ _ _|_ _ _ _ _|_ _|
||")
    print("||                 _ _ __ __ _ _ _ _ _ __ _ _          _ _ _ _ _ __ _ _ _ _ _
||")
    print("||               /    _ _ _|      \    |  \      /    /  _ _ _|    _ _    |
||")
    print("||              /    /   /    _ _    \   |   \    /    /  /_ _ _    |_ _|   |
||")
    print("||             \    \  /    /    \    \  |\    \   /    /         |       /
||")
    print("||              \    \ \    \_ _/    /  | \    \_/    /    _ _ __|    |\    \
||")
    print("||           __ _\    \ \          /    |_ \_ __     /     \_ _ _    | \    \
||")
    print("||          |_ _ _ _ _/  \_ _ _ _ /_ _ _ _ _ _ _|_ _ /_ _ _ _ _ _ _ |_ _|   \_ _\
||")
    print("||          print("||
||")
    print("[]=============================================================================================
[]")
    print("||>>>>>>>>>>>>>>>>>>>> Dibuat oleh Muhammad Gilang Ramadhan (13520137)
<<<<<<<<<<<<<<<<<<<<||")
    print("||>>>>>>>>>>>>>>>>>>>>>>>> Tugas Kecil 2 IF2211 Strategi Algoritma
<<<<<<<<<<<<<<<<<<<<<<<<||")
    print("[]=============================================================================================
[]")
```

Gambar 2.3.1 MainProgram.py (line 1-47)

```python
def logo_penutup():
    print("[]==========================================================================================
[]")
    print("||              _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _         _ __ _ _ _ _ _ __
||")
    print("||             |_ _   _ _|  _ _ _|  _ _  |   |     \       /    |     _ _    |
||")
    print("||                 |  |    /_ _ _  |_ _| |   |      \     /     |    /_ _\   |
||")
    print("||                 |  |          |      /|   |       \_ _/      |    _ __    |
||")
    print("||                 |  |     _ _ _| |\   \|   |   |\       /|    |    |    |   |
||")
    print("||                 |  |     \ _ __ | \   \   |   | \ _ _ _/ |   |    |    |   |
||")
    print("||                 |__|_ _ _ _ _ _||  \_ _\ _|_ _|          |_ _|_ _|    |_ _|
||")
    print("||                     _ _   _ __ _ _ __ __  _ _ _ _ _ _ _ _       _ _
||")
    print("||                    |   | /   /   _ _    |/     _ _ _||   |     |   |
||")
    print("||                    |   |/   /   /_ _\   |\    /  |   |   |_ _ _|   |
||")
    print("||                    |       /    _ __    | \   \  |   |             |
||")
    print("||                    |       \   |    |   |  \   \ |   |    _ _ _    |
||")
    print("||                    |   |\   \  |    |   |_ _\   \|   |   |     |   |
||")
    print("||                    |_ _| \_ _\_|    |_ _|_ _ _ _/|_ _|_ _|     |_ _|
||")
    print("||
||")
    print("[]==========================================================================================
[]")

def MainProgram():
    printlogo()
    print()
    print("--------------------------------------------------------------------------------------------
")
    print()
    print("[]======================================== Menu ==========================================
[]")
    print("||
||")
    print("||                                 Berikut Dataset yang tersedia :
||")
    print("||                                    1. Dataset Iris
||")
    print("||                                    2. Dataset Wine
||")
    print("||                                    3. Dataset Breast_Cancer
||")
    print("||                                    4. Dataset Diabetes
||")
    print("||                                    5. Dataset Digits
||")
    print("||
||")
    print("[]==========================================================================================
[]")
    print("||                              >> Pilih 0 jika ingin keluar program
||")
    print("[]==========================================================================================
[]")
    command = int(input("Masukkan Command yang diinginkan : "))
    while command != 0:
        if command == 1 or command == 2 or command == 3 or command == 4 or command == 5:
            df, data = load_datasets(command)
            tampilkan_tabel_dataset(df)
            tampilkan_plot_grafik(df, data)
        else:
            print("Maaf masukkan tidak dikenali")
        print("--------------------------------------------------------------------------------------------
---")
        command = int(input("Masukkan Kembali Command Anda : "))
    logo_penutup()
    exit()
if __name__ == "__main__":
    MainProgram()
```

Gambar 2.3.2 MainProgram.py (line 49-98)

BAB 3

Screenshot Program

1. Laman Pembuka

Gambar 3.1.1 Laman Pembuka

2. Program meminta input dataset

```
[]========================================== Menu ==========================================[]
||                                                                                          ||
||                                Berikut Dataset yang tersedia :                           ||
||                                   1. Dataset Iris                                        ||
||                                   2. Dataset Wine                                        ||
||                                   3. Dataset Breast_Cancer                               ||
||                                   4. Dataset Diabetes                                    ||
||                                   5. Dataset Digits                                      ||
||                                                                                          ||
[]==========================================================================================[]
||                            >> Pilih 0 jika ingin keluar program                          ||
[]==========================================================================================[]
Masukkan Command yang diinginkan :
```

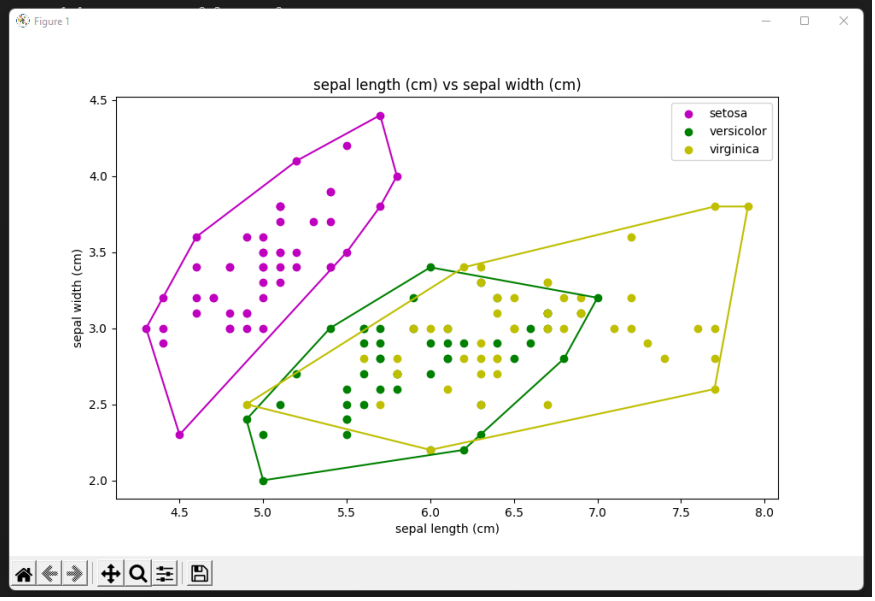
Gambar 3.2.1 Program meminta input dataset pada menu utama

```
------------------------------------------------------------------------------------------
Masukkan Kembali Command Anda :
```

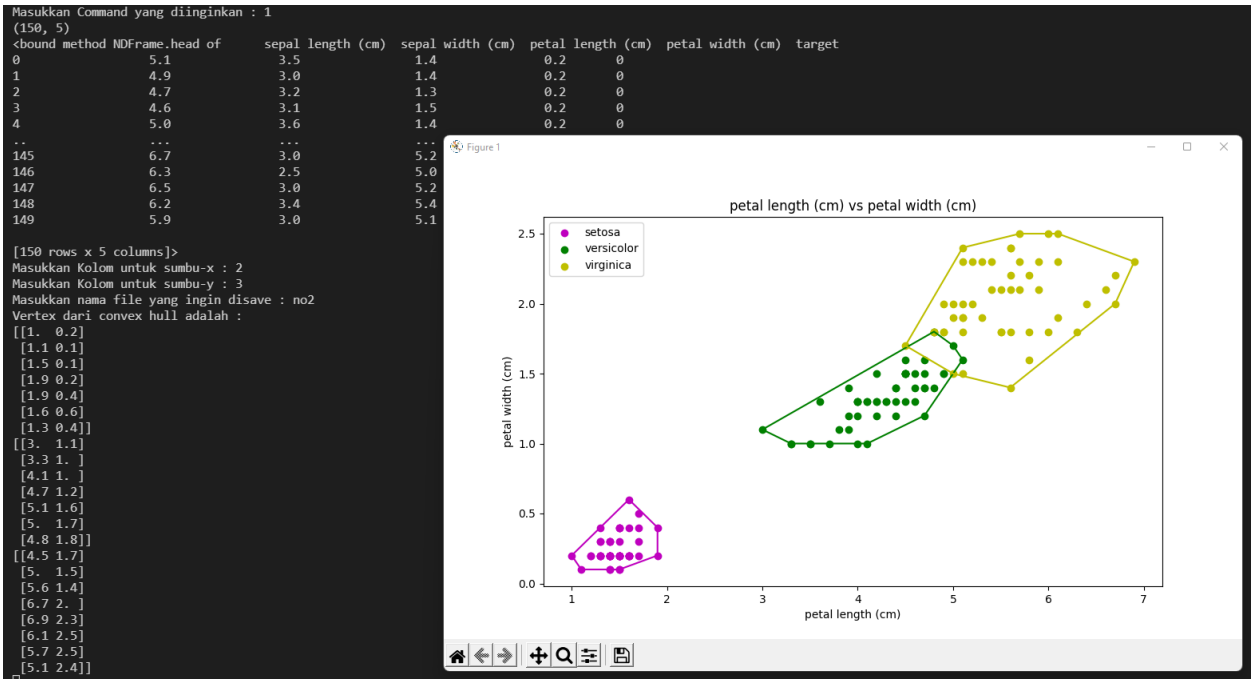
Gambar 3.2.2 Program meminta input Kembali ketika selesai memvisualisasikan dataset

3. Hasil Output

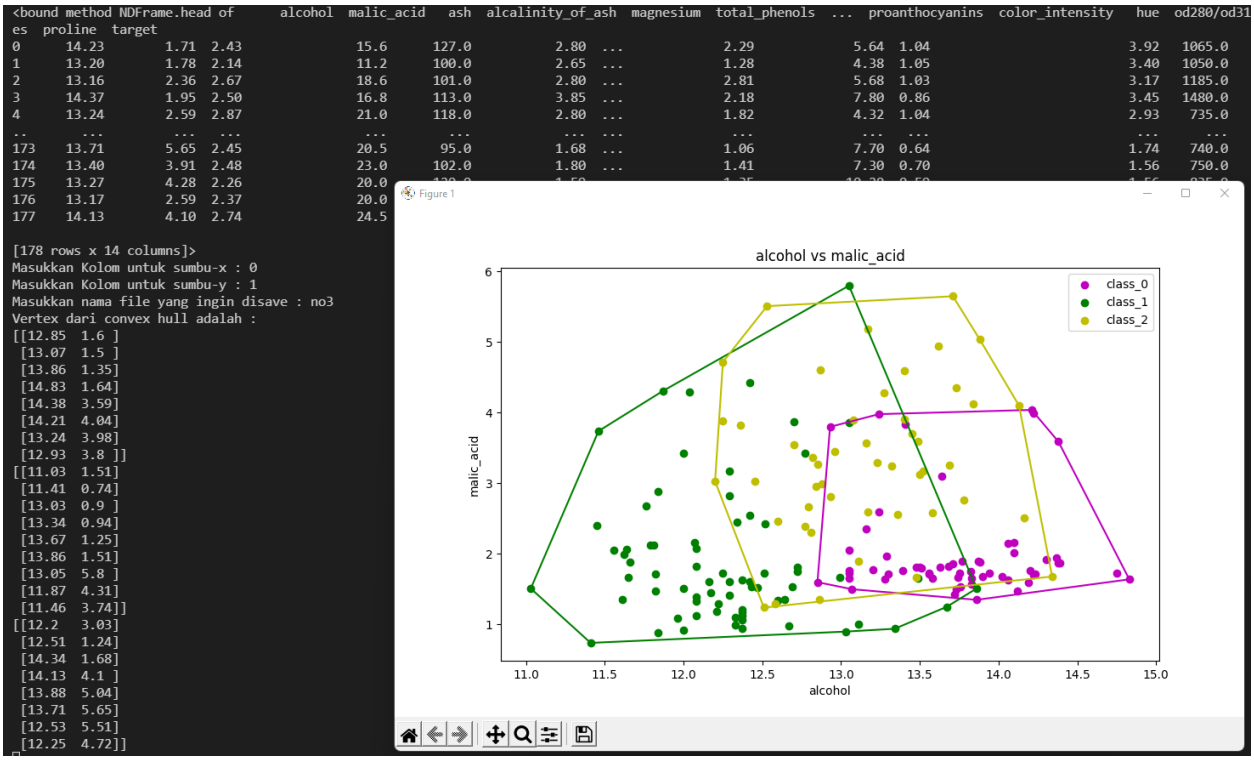
```
Masukkan Command yang diinginkan : 1
(150, 5)
<bound method NDFrame.head of      sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  target
0                  5.1               3.5                1.4               0.2       0
1                  4.9               3.0                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
..                 ...               ...
145                6.7               3.0
146                6.3               2.5
147                6.5               3.0
148                6.2               3.4
149                5.9               3.0

[150 rows x 5 columns]>
Masukkan Kolom untuk sumbu-x : 0
Masukkan Kolom untuk sumbu-y : 1
Masukkan nama file yang ingin disave : no1
Vertex dari convex hull adalah :
[[4.3 3. ]
 [4.5 2.3]
 [5.5 3.5]
 [5.7 3.8]
 [5.8 4. ]
 [5.7 4.4]
 [5.2 4.1]
 [4.6 3.6]]
[[4.9 2.4]
 [5.  2. ]
 [6.2 2.2]
 [6.8 2.8]
 [7.  3.2]
 [6.  3.4]
 [5.4 3. ]]
[[4.9 2.5]
 [6.  2.2]
 [7.7 2.6]
 [7.9 3.8]
 [7.7 3.8]
 [6.2 3.4]]
```



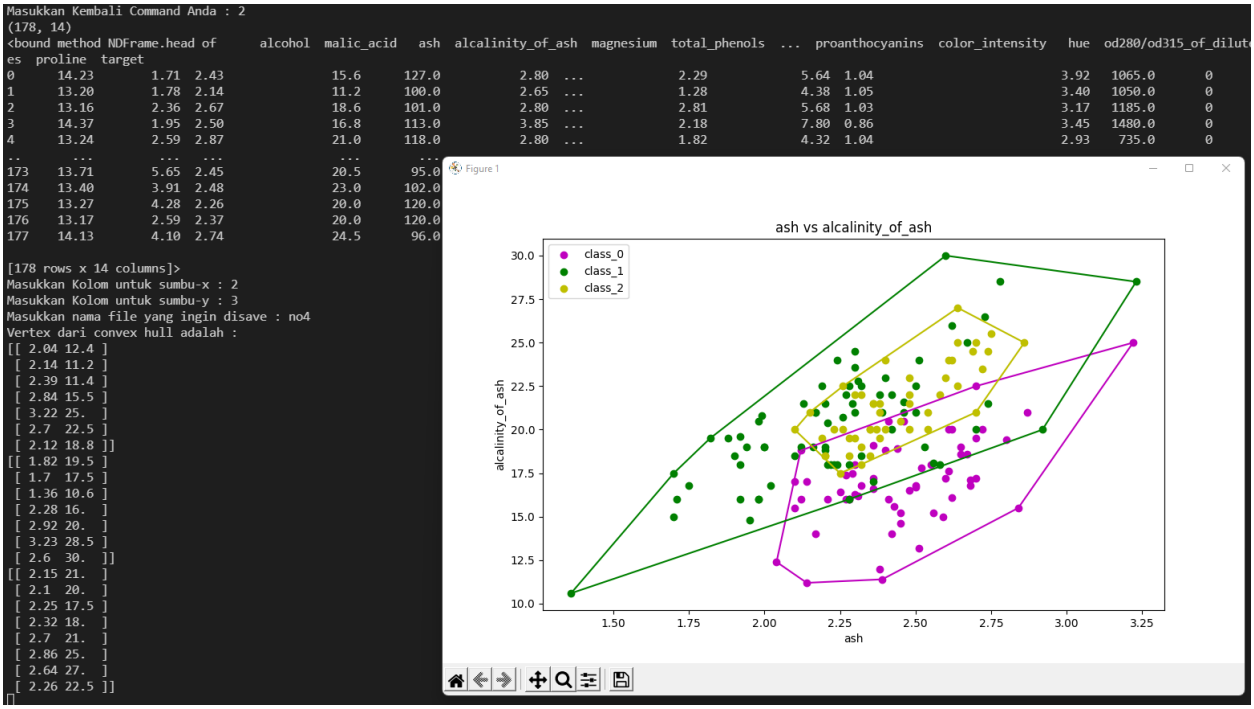
Gambar 3.3.1 Output dataset Iris : sepal length (cm) vs sepal width (cm)



Gambar 3.3.2 Output dataset Iris : petal length (cm) vs petal width (cm)

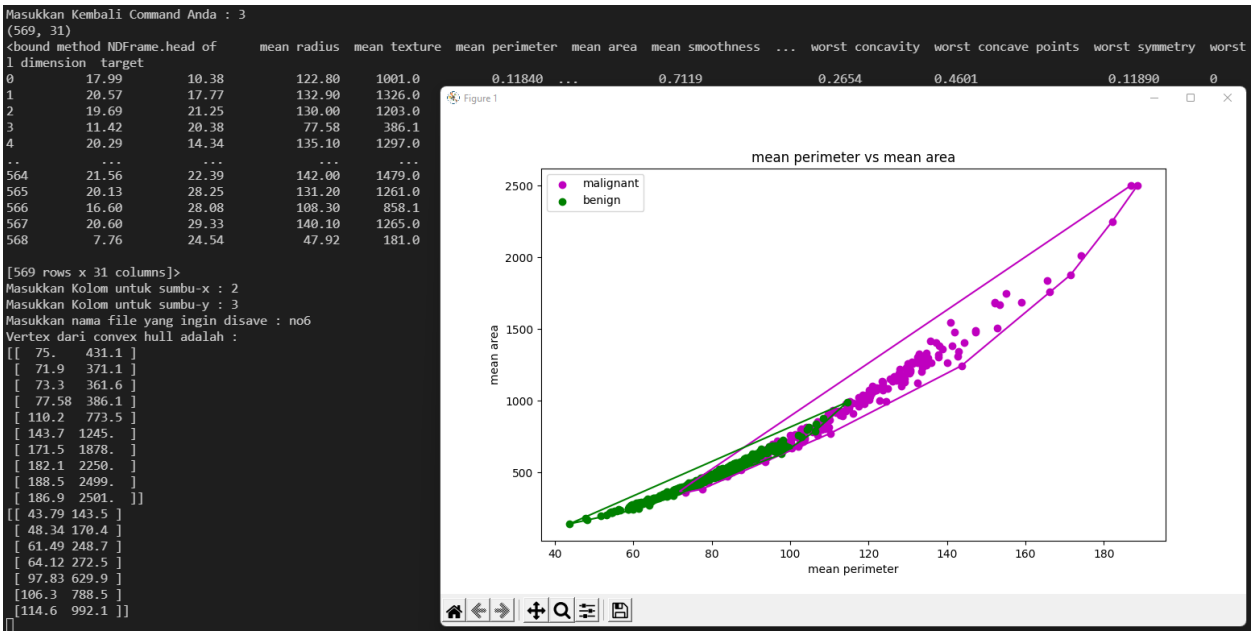


Gambar 3.3.3 Output dataset Iris : alcohol vs malic_acid



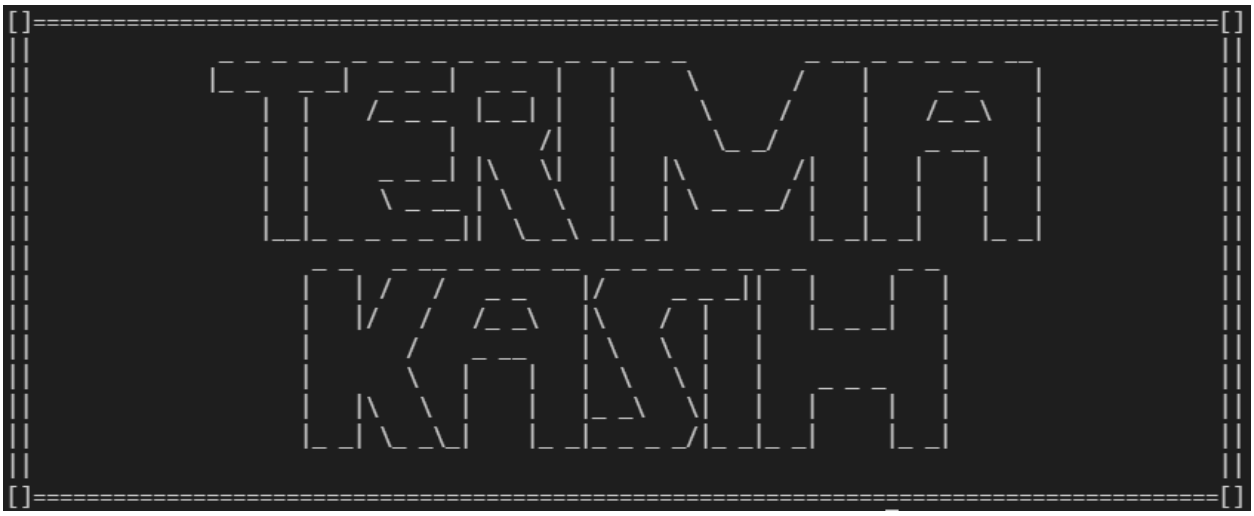
Gambar 3.3.4 Output dataset Iris : ash vs alcalinity_of_ash

Gambar 3.3.5 Output dataset Breast_Cancer : mean radius vs mean texture



Gambar 3.3.6 Output dataset Breast_Cancer : mean perimeter vs mean area

4. Laman Penutup



Gambar 3.4.1 Laman penutup

LAMPIRAN

Lampiran 1

Checklist penilaian :

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.	✓	

Lampiran 2

Link *Repository* github : https://github.com/gilanglahat22/Tucil2_13520137