Laporan Tugas Kecil 2

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

Mata Kuliah IF2211 - Strategi Algoritma

Oleh :

Nama : Jundan Haris

NIM : 13520155

Kelas : 02

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021/2022

DAFTAR ISI

BAB I
ALGORITMA *DEVIDE AND CONQUER*

Algoritma *Devide and Conquer*

Pada awalnya, *devide and conquer* merupakan strategi militer yang dikenal dengan *devide ut imperes.* Strategi tersebut lah yang menjadi dasar dari algoritma *devide and conquer* yang digunakan dalam ilmu komputer saat ini. Definisinya sendiri yaitu :

1. *Devide* : membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil.
2. *Conquer (solve)*: menyelesaikan masing-masing upa-persoalan (secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar).
3. *Combine*: mengabungkan solusi masing-masing upa-persoalan sehingga membentuk solusi persoalan semula.

Persoalan yang dapat diselesaikan menggunakan algoritma *devide and conquer* diantaranya adalah persoalan minmaks, menghitung perpangkatan, algoritma sorting, perkalian matriks, *convex hull* dan lain-lain. Pada kali ini, persoalan yang akan diselesaikan menggunakan algoritma *devide and conquer* adalah *convex hull* untuk memvisualisasikan tes *linier separability dataset*.

Algoritma *Devide and Conquer* Untuk Visualisasi Tes *Linier Separability Dataset*

Program yang dibuat menggunakan algoritma *devide and conquer*. Untuk langkah-langkah algoritmanya secara umum yaitu :

1. Data berupa array of point yang diterima diurutkan terlebih dahulu berdasarkan absisnya dari yang terkecil ke terbesar.
2. Simpan titik dengan absis terkecil dan terbesar. Apabila terdapat titik dengan absis sama, ambil titik dengan ordinat lebih besar.
3. Dari dua titik yang sudah didapatkan, ditarik garis lurus sehingga data terbagi menjadi dua. Simpan masing-masing kumpulan titik ke sebuah array.
4. Simpan garis antara titik pertama (A) dan titik kedua (B) ke dalam array solusi.
5. Dari kumpulan titik yang sudah disimpan, cari titik dengan jarak titik-garis paling jauh. Misalkan titik tersebut dinamakan C.
6. Hapus garis AB dari array solusi. Masukkan garis AC dan CB.
7. Dengan adanya garis AC dan AB, akan muncul bagian baru lagi.

8. Ambil bagian luar dari garis AC lalu simpan di dalam array. Ulangi langkah 5-7 menggunakan array yang baru hingga tidak ada titik di luar garis.
9. Ambil bagian luar dari garis AB lalu simpan di dalam array. Ulangi langkah 5-7 pada array baru ini hingga tidak ada titik di luar garis.
10. Ulangi terus hingga tidak ada lagi titik di luar garis, maka akan didapatkan Convex Hull.

BAB II
SOURCE PROGRAM DALAM BAHASA PYTHON

Pada pengerjaannya, program ditulis dalam bahasa python. Program secara lengkap dapat diakses pada pranala https://github.com/jundanha/Tucil2_13520155 . Pada bab ini akan ditampilkan beberapa bagian potongan program guna memenuhi spesifikasi dari Tugas Kecil 2 IF2211 Strategi Algoritma.

myConvexHull.ipynb

```
def distance(A, B, C):
    #mencari jarak titik C dengan garis AB
    x1, y1 = A
    x2, y2 = B
    x3, y3 = C
    a = y2 - y1
    b = x1 - x2
    c = x2*y1 - x1*y2
    dist = (abs(a*x3 + b*y3 + c))/math.sqrt(a**2 + b**2)
    return dist

def det3(A, B, C):
    #mencari determinan dari 3 titik
    x1, y1 = A
    x2, y2 = B
    x3, y3 = C
    det = x1*y2 + x3*y1 + x2*y3 - x3*y2 - x2*y1 - x1*y3
    return det

def myConvexHull(bucket):
    #inisialisasi
    kananAB = []    #himpunan titik di sebelah kanan garis AB
    kiriAB = []     #himpunan titik di sebelah kiri garis AB

    #mengosongkan hasil terlebih dahulu
    hasil.clear()

    #sort
    bucket = sorted(bucket, key=lambda k: (k[0], k[1]))

    #ubah tipe ke list
    for i in range(len(bucket)):
        bucket[i] = bucket[i].tolist()

    #ambil titik min dan maks
    P1 = bucket[0]
    P2 = bucket[len(bucket)-1]
```

```
    #tentukan semua titik apakah masuk ke sebelah
    #kanan atau kiri dari garis yang terbentuk
    for point in bucket:
        det = det3(P1, P2, point)
        if (det > 0):
            kiriAB.append(point)
        elif (det < 0):
            kananAB.append(point)

    #masukkan solusi
    hasil.append((P1,P2))
    hasil.append((P2,P1))

    #cari di kedua bagian
    myFindHull(kananAB, P1, P2)
    myFindHull(kiriAB, P2, P1)

    return hasil
```

```
def myFindHull(Set, P1, P2):
    if len(Set) == 0:
        #ketika Set kosong, tidak akan diproses
        return
    else :
        #inisialisasi
        Selected = []       #menyimpan titik terjauh
        tempdistance = -1   #menyimpan jarak sementara
        maxdistance = -1    #menyimpan jarak terjauh

        #cari point terjauh
        for point in Set:
            tempdistance = distance(P1, P2, point)
            if tempdistance > maxdistance:
                maxdistance = tempdistance
                Selected = point

        #ganti solusinya
        hasil.remove((P1, P2))
        hasil.append((P1, Selected))
        hasil.append((Selected, P2))
```

```
    #hapus selected dari Set
    Set.remove(Selected)

    #buat partisi baru
    Setbaru1 = []
    Setbaru2 = []
    for point in Set:
        det1 = det3(P1, Selected, point)
        det2 = det3(Selected, P2, point)
        if det1 < 0:
            Setbaru1.append(point)
        if det2 < 0:
            Setbaru2.append(point)

    #rekurens
    myFindHull(Setbaru1, P1, Selected)
    myFindHull(Setbaru2, Selected, P2)
```

```
print()
print("==============Convex Hull untuk Visualisasi Tes Linier Separability Dataset==============")
print()


# inisialisasi
hasil = []  #menyimpan hasil dari convex hull
colors = [ 'c', 'm', 'y','r', 'g', 'b', 'k']

print("Silahkan pilih dataset yang akan digunakan")
print()
print("1. iris")
print("2. wine")
print("3. digits")
print()
print("Masukkan pilihan : ")
a = int(input())

if (a == 1):
    data = datasets.load_iris()
    #create a DataFrame
    df = pd.DataFrame(data.data, columns=data.feature_names)
    df['Target'] = pd.DataFrame(data.target)
elif (a == 2):
    data = datasets.load_wine()
    #create a DataFrame
    df = pd.DataFrame(data.data, columns=data.feature_names)
    df['Target'] = pd.DataFrame(data.target)
elif (a == 3):
    data = datasets.load_digits()
    #create a DataFrame
    df = pd.DataFrame(data.data, columns=data.feature_names)
    df['Target'] = pd.DataFrame(data.target)
else:
    print("input salah!")
```

```
if (a != 0):
    #sumbu x
    print()
    print()
    print("Daftar pilihan feature")
    for i in range(len(data.feature_names)):
        print(str(i+1) + ". " + data.feature_names[i])
    print()
    print("Silahkan pilih sumbu x : ")
    idx1 = int(input())
    while (idx1 < 1 or idx1 >len(data.feature_names)):
        print("input salah!")
        print()
        print("Silahkan pilih sumbu x : ")
        idx1 = int(input())

    #sumbu y
    print()
    print("Daftar pilihan feature")
    for i in range(len(data.feature_names)):
        print(str(i+1) + ". " + data.feature_names[i])
    print()
    print("Silahkan pilih sumbu y : ")
    idx2 = int(input())
    while (idx2 < 1 or idx2 >len(data.feature_names)):
        print("input salah!")
        print()
        print("Silahkan pilih sumbu y : ")
        idx2 = int(input())

    #visualisasi
    plt.figure(figsize = (10, 6))
    plt.title(data.feature_names[idx1-1] + " vs " + data.feature_names[idx2-1])
    plt.xlabel(data.feature_names[idx1-1])
    plt.ylabel(data.feature_names[idx2-1])
    for i in range(len(data.target_names)):
        bucket = df[df['Target'] == i]
        bucket = bucket.iloc[:,[idx1-1,idx2-1]].values
        plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
        hull = myConvexHull(bucket)
        for j in range(len(hull)):
            x = [hull[j][0][0], hull[j][1][0]]
            y = [hull[j][0][1], hull[j][1][1]]
            plt.plot(x,y, colors[i%7])
    plt.legend()
    plt.plot()
```

Daftar Fungsi

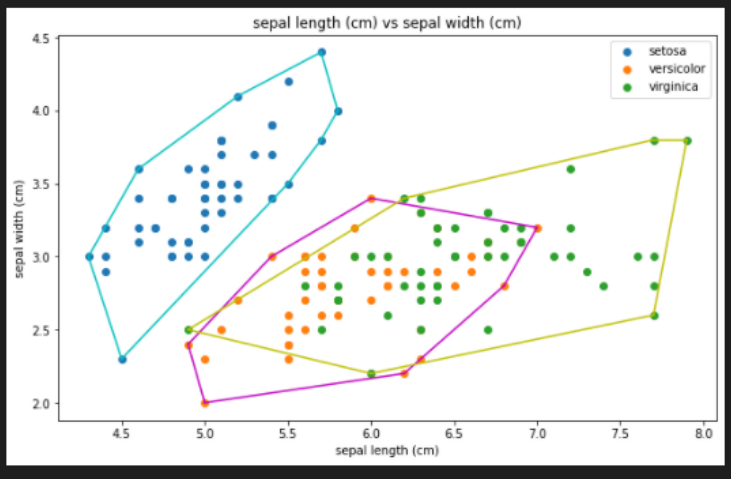
No`	Fungsi	Parameter	Keterangan
1	distance	A, B, C	Menghitung jarak antara titik C dengan garis yang terbentuk dari titik A dan titik B
2	det3	A, B, C	Menghitung determinan dari tiga titik
3	myConvexHull	bucket	Fungsi utama dari Convex Hull
4	myFindHull	Set, A, B	Fungsi rekursif untuk mengimplementasikan algoritma devide and conquer

BAB III
CONTOH *INPUT* DAN *OUTPUT* PROGRAM

Dataset Iris

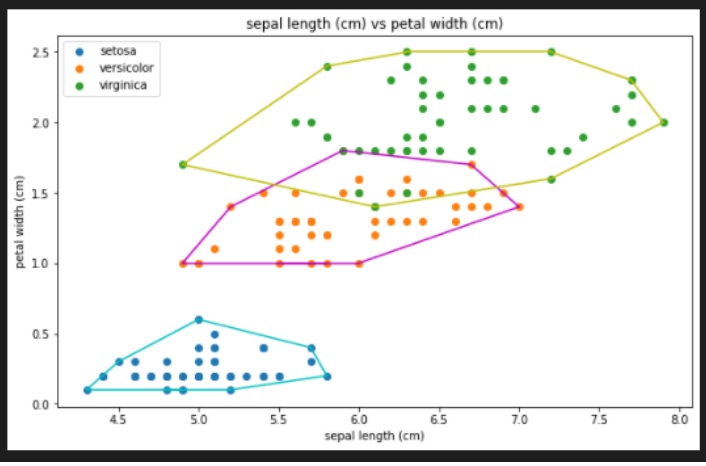
1. Sepal Length VS Sepal Width

```
==============Convex Hull untuk Visualisasi Tes Linier Separability Dataset==============


Silahkan pilih dataset yang akan digunakan

1. iris
2. wine

Masukkan pilihan :


Daftar pilihan feature
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)

Silahkan pilih sumbu x :

Daftar pilihan feature
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)

Silahkan pilih sumbu y :
```



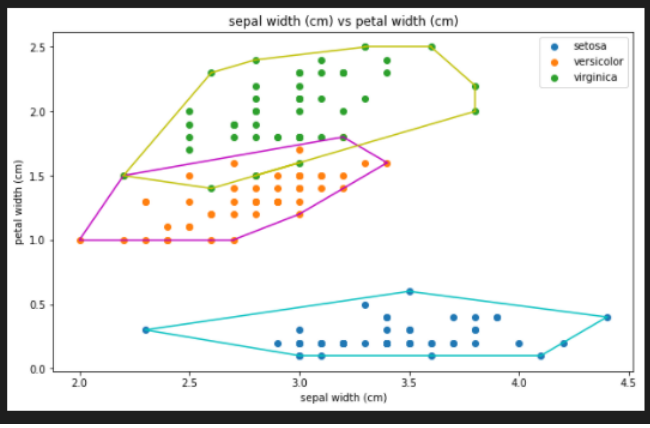
2. Sepal Length VS Petal Width

```
==============Convex Hull untuk Visualisasi Tes Linier Separability Dataset==============


Silahkan pilih dataset yang akan digunakan

1. iris
2. wine

Masukkan pilihan :


Daftar pilihan feature
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)

Silahkan pilih sumbu x :

Daftar pilihan feature
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)

Silahkan pilih sumbu y :
```



3. Sepal Width VS Petal Width

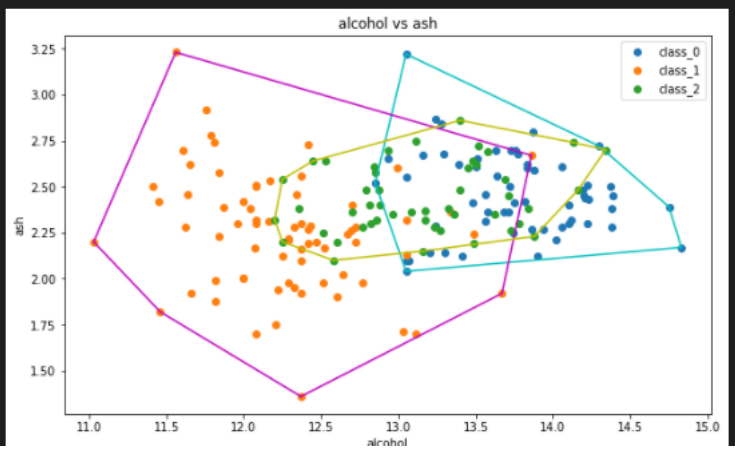
```
==============Convex Hull untuk Visualisasi Tes Linier Separability Dataset==============


Silahkan pilih dataset yang akan digunakan

1. iris
2. wine

Masukkan pilihan :


Daftar pilihan feature
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)

Silahkan pilih sumbu x :

Daftar pilihan feature
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)

Silahkan pilih sumbu y :
```



Dataset Wine

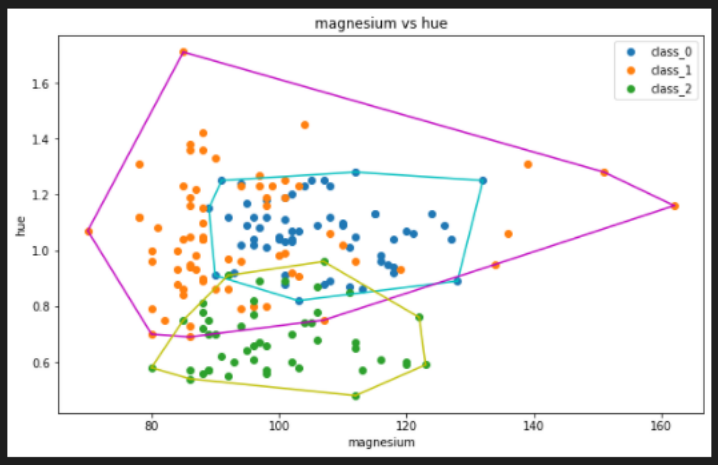
1. Alcohol VS Ash

```
==========Convex Hull untuk Visualisasi Tes Linier Separability Dataset==========

Silahkan pilih dataset yang akan digunakan

1. iris
2. wine

Masukkan pilihan :


Daftar pilihan feature
1. alcohol
2. malic_acid
3. ash
4. alcalinity_of_ash
5. magnesium
6. total_phenols
7. flavanoids
8. nonflavanoid_phenols
9. proanthocyanins
10. color_intensity
11. hue
12. od280/od315_of_diluted_wines
13. proline
...
11. hue
12. od280/od315_of_diluted_wines
13. proline

Silahkan pilih sumbu y :
```



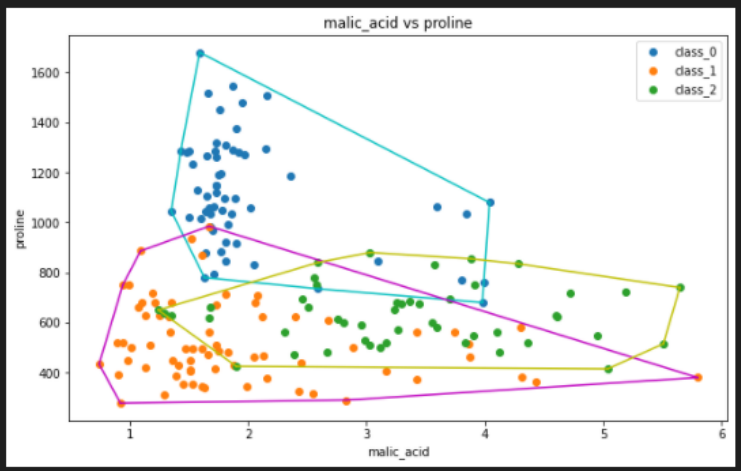
2. Magnesium VS hue

```
Silahkan pilih dataset yang akan digunakan

1. iris
2. wine

Masukkan pilihan :


Daftar pilihan feature
1. alcohol
2. malic_acid
3. ash
4. alcalinity_of_ash
5. magnesium
6. total_phenols
7. flavanoids
8. nonflavanoid_phenols
9. proanthocyanins
10. color_intensity
11. hue
12. od280/od315_of_diluted_wines
13. proline
...
11. hue
12. od280/od315_of_diluted_wines
13. proline

Silahkan pilih sumbu y :
```



3. Malic_acid VS Proline

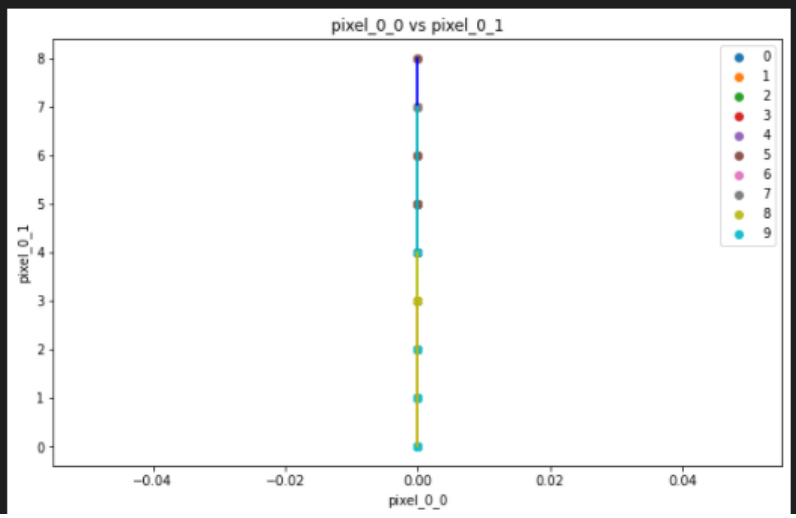
```
Silahkan pilih dataset yang akan digunakan

1. iris
2. wine

Masukkan pilihan :


Daftar pilihan feature
1. alcohol
2. malic_acid
3. ash
4. alcalinity_of_ash
5. magnesium
6. total_phenols
7. flavanoids
8. nonflavanoid_phenols
9. proanthocyanins
10. color_intensity
11. hue
12. od280/od315_of_diluted_wines
13. proline
...
11. hue
12. od280/od315_of_diluted_wines
13. proline

Silahkan pilih sumbu y :
```



Dataset Digits

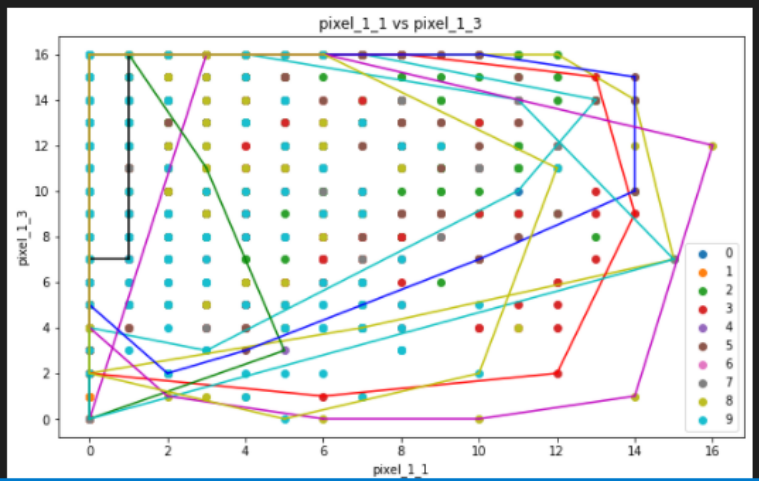
1. Pixel_0_0 VS pixel_0_1

```
Silahkan pilih dataset yang akan digunakan

1. iris
2. wine
3. digits

Masukkan pilihan :


Daftar pilihan feature
1. pixel_0_0
2. pixel_0_1
3. pixel_0_2
4. pixel_0_3
5. pixel_0_4
6. pixel_0_5
7. pixel_0_6
8. pixel_0_7
9. pixel_1_0
10. pixel_1_1
11. pixel_1_2
12. pixel_1_3
...
62. pixel_7_5
63. pixel_7_6
64. pixel_7_7

Silahkan pilih sumbu y :
```



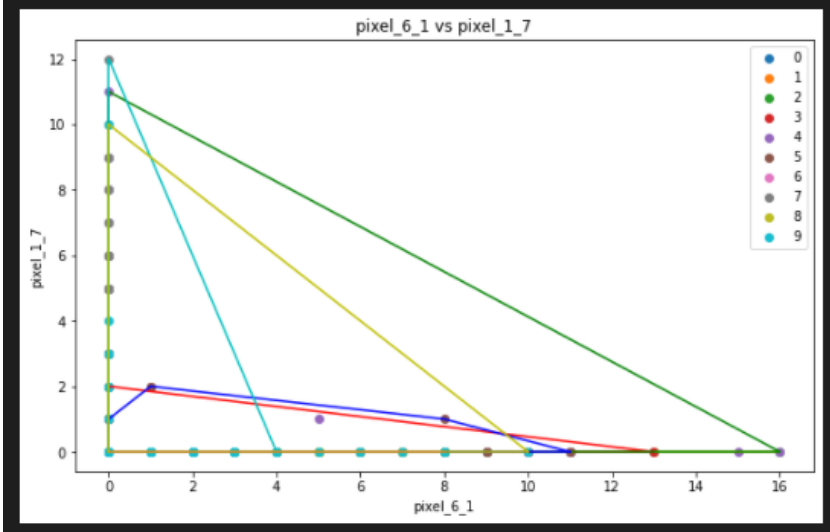
2. Pixel_1_1 VS pixel_1_3

```
Silahkan pilih dataset yang akan digunakan

1. iris
2. wine
3. digits

Masukkan pilihan :


Daftar pilihan feature
1. pixel_0_0
2. pixel_0_1
3. pixel_0_2
4. pixel_0_3
5. pixel_0_4
6. pixel_0_5
7. pixel_0_6
8. pixel_0_7
9. pixel_1_0
10. pixel_1_1
11. pixel_1_2
12. pixel_1_3
...
62. pixel_7_5
63. pixel_7_6
64. pixel_7_7

Silahkan pilih sumbu y :
```



3. Pixel_6_1 VS pixel_1_7

```
3. digits

Masukkan pilihan :


Daftar pilihan feature
1. pixel_0_0
2. pixel_0_1
3. pixel_0_2
4. pixel_0_3
5. pixel_0_4
6. pixel_0_5
7. pixel_0_6
8. pixel_0_7
9. pixel_1_0
10. pixel_1_1
11. pixel_1_2
12. pixel_1_3
...
62. pixel_7_5
63. pixel_7_6
64. pixel_7_7

Silahkan pilih sumbu y :
```



BAB IV
KESIMPULAN DAN SARAN

Kesimpulan

Program Implementasi Convex Hull untuk Visualisasi Tes *Linier Separability Dataset* dengan Algoritma *Devide and Conquer* yang telah penulis buat dapat berjalan dengan baik karena dapat memenuhi semua spesifikasi tugas yang diberikan. Namun mungkin saja masih ada beberapa kesalahan yang tidak saya sadari.

Berikut adalah tabel yang diperlukan untuk mempermudah 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	✓	

Saran

Tentu saja program yang penulis buat masih sangat sederhana dan jauh dari kata sempurna. Masih banyak sekali hal yang bisa dilakukan untuk memperbaiki dan memperbagus program yang penulis buat, misalnya dengan menambahkan antarmuka GUI untuk mempermudah pengguna atau fitur-fitur lainnya seperti bisa menerima dataset lain selain dataset yang disediakan.

REFERENSI

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/stima21-22.htm

https://codecrucks.com/convex-hull-using-divide-and-conquer/

https://scikit-learn.org/stable/datasets/toy_dataset.html

https://www.programiz.com/python-programming/methods/list/remove

https://www.w3schools.com/python/ref_func_sorted.asp

https://numpy.org/doc/1.22/

https://www.w3schools.com/python/ref_list_append.asp

https://www.geeksforgeeks.org/graph-plotting-in-python-set-1/

https://stackoverflow.com/questions/36470343/how-to-draw-a-line-with-matplotlib

https://matplotlib.org/stable/gallery/color/named_colors.html