LAPORAN TUGAS KECIL 2
IF2211 STRATEGI ALGORITMA

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

NIM	: 13520072
Nama	: Jova Andres Riski Sirait
Kelas	: K03

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2022

A. Algoritma Divide and Conquer

Algoritma Divide and Conquer adalah algoritma pemecahan masalah yang besar menjadi sub-masalah yang lebih kecil sehingga lebih mudah diselesaikan. Solusi dari sub-masalah tersebut kemudian digabungkan untuk menjadi solusi dari permasalahan utama.

Pada tugas kecil kali ini, implementasi Convex Hull untuk Visualisasi Tes Linear Separability Dataset dibuat dengan algoritma divide and conquer yakni dengan langkah sebagai berikut.

1. Pilih dua buah titik ekstrim yang terdapat pada kumpulan titik-titik pada dataset. Titik minimum (p1) merupakan titik dengan absis paling kecil, sedangkan titik maksimum (pn) merupakan titik dengab absis paling besar.
2. Pisahkan kumpul titik yang berada di atas dan di bawah garis p1pn sebagai sub-masalah yang lebih kecil. Untuk memeriksa apakah sebuah titik berada di atas suatu garis yang dibentuk dua titik, digunakan penentuan determinan sebagai berikut:

   $$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = x_1y_2 + x_3y_1 + x_2y_3 - x_3y_2 - x_2y_1 - x_1y_3$$

   Titik (x3, y3) berada di sebelah atas dari garis ((x1, y1), (x2, y2)) jika hasil determinannya positif, dan sebaliknya. Jika titik berada pada garis (determinannya 0), maka titik tersebut tidak mungkin membentuk convex hull.
3. Kumpulan titik pada bagian atas dapat membentuk convex hull atas dan kumpulan titik pada bagian bawah dapat membetuk convex hull bawah. Penerapan divide and conquer dapat dilihat langsung pada bagian ini dimana kedua sisi (atas dan bawah) masing-masing diterapkan algoritma divide and conquer untuk mencari solusi.
4. Untuk divide and conquer pada salah satu bagian, misalnya bagian atas, terdapat beberapa kemungkinan. Jika tidak terdapat titik lain, maka p1 dan pn menjadi pembentuk convex hull. Jika terdapat satu titik di atasnya, maka ketiga titik tersebut dapat juga dihubungkan menjadi pembentuk convex hull. Jika terdapat beberapa titik, maka titik yang dipilih adalah titik yang mempunyai jarak terjauh dari garis yang dibentuk oleh titik p1 dan pn. Untuk mencari titik terjauh, formula yang digunakan adalah sebagai berikut:

   $$\text{distance}(P_1, P_2, (x_0, y_0)) = \frac{|(x_2 - x_1)(y_1 - y_0) - (x_1 - x_0)(y_2 - y_1)|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}.$$

   Untuk beberapa titik dengan jarak yang sama, titik yang dipilih adalah titik yang membentuk sudut pmax-p1-pn yang paling maksimal. Untuk menghitung sudut yang dibentuk, dapat digunakan prinsip vector yakni membuat dua buah vector dari tiitk p1-pmax dan p1-pn, kemudian sudut diantara kedua vector tersebut dihitung dengan dot-product formula.

$$\theta = \arccos\left(\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}\right)$$

Kumpulan titik yang berada dalam segitiga p1-pmax-pn dapat diabaikan dari perhitungan.

5. Tentukan kumpulan titik yang berada di atas garis p1-pmax dan di atas garis pmax-pn untuk selanjutnya dilakukan langkah yang sama (langkah 4) sampai bagian atas dan bawah garis kosong.
6. Semua pasangan titik yang dihasilkan kemudian dihubungkan menjadi sebuah convex hull yang utuh.

B. Source Code Program

Terdapat 3 file utama pada program yaitu kelas ConvexHull (myConvexHull.py) sebagai implementasi utama, kelas Area (Area.py) sebagai Enum pembantu kelas ConvexHull dan main.py sebagai main program yang akan membuat visualisasi dari convex hull yang telah dihasilkan.

1. Area.py

```python
from enum import Enum


class Area(Enum):
    ABOVE = 0
    BELOW = 1
```

2. myConvexHull.py

```python
from math import sqrt, pow, acos

from Area import Area


class ConvexHull:
    def __init__(self, _dataset) -> None:
        self.dataset = _dataset
        self.simplices = []
        self._buildConvexHull()

    def _leftMost(self):
        minIndex = 0
        points = self.dataset
        x = 0

        for i in range(1, len(points)):
            if points[i][x] < points[minIndex][x]:
                minIndex = i

        return minIndex

    def _rightMost(self):
        maxIndex = 0
        points = self.dataset
        x = 0
```

```
        for i in range(1, len(points)):
            if points[i][x] > points[maxIndex][x]:
                maxIndex = i

        return maxIndex

    def _distance(self, p1, p2, p3):
        points = self.dataset
        x, y = 0, 1

        dividend = ((points[p2][x] - points[p1][x]) * (points[p1][y] - points[p3][y]))
- (
            (points[p1][x] - points[p3][x]) * (points[p2][y] - points[p1][y]))
        divisor = sqrt(pow(points[p2][x] - points[p1][x],
                       2) + pow(points[p2][y] - points[p1][y], 2))

        return abs(dividend) / divisor

    def _angle(self, p1, p2, p3):
        points = self.dataset
        x, y = 0, 1

        p12 = (points[p1][x] - points[p2][x], points[p1][y] - points[p2][y])
        p13 = (points[p1][x] - points[p3][x], points[p1][y] - points[p3][y])

        return acos((p12[x] * p13[x] + p12[y] * p13[y]) / (sqrt(pow(p12[x], 2) +
pow(p12[y], 2)) * sqrt(pow(p13[x], 2) + pow(p13[y], 2))))

    def _defineArea(self, p1, p2, p3):
        points = self.dataset
        x, y = 0, 1

        area = points[p1][x] * points[p2][y] + points[p3][x] * points[p1][y] +
points[p2][x] * points[p3][y] - \
            points[p3][x] * points[p2][y] - points[p2][x] * \
            points[p1][y] - points[p1][x] * points[p3][y]

        if area > 1e-10:
            return Area.ABOVE
        elif area < -1e-10:
            return Area.BELOW

    def _separate(self, pi, pn, points):
        aboveAreaDots = []
```

```
        belowAreaDots = []

        for i in range(0, len(points)):
            area = self._defineArea(pi, pn, points[i])
            if area == Area.BELOW:
                belowAreaDots.append(points[i])
            elif area == Area.ABOVE:
                aboveAreaDots.append(points[i])

        return aboveAreaDots, belowAreaDots

    def _dnc(self, pi, pn, parts, direction: Area):
        if len(parts) == 0:
            self.simplices.append([pi, pn])
        elif len(parts) == 1:
            self.simplices.append([pi, parts[0]])
            self.simplices.append([parts[0], pn])
        else:
            pmax = parts[0]
            maxDistance = self._distance(pi, pn, parts[0])

            for i in range(1, len(parts)):
                currentDistance = self._distance(pi, pn, parts[i])
                if currentDistance > maxDistance:
                    maxDistance = currentDistance
                    pmax = parts[i]
                elif currentDistance == maxDistance:
                    if self._angle(pi, pn, parts[i]) > self._angle(pi, pn, pmax):
                        pmax = parts[i]

            dataparts1 = self._separate(pi, pmax, parts)
            dataparts2 = self._separate(pmax, pn, parts)

            self._dnc(pi, pmax,
                      dataparts1[direction.value], direction)
            self._dnc(pmax, pn,
                      dataparts2[direction.value], direction)

    def _buildConvexHull(self):
        pi = self._leftMost()
        pn = self._rightMost()

        dataparts = self._separate(
            pi, pn, [i for i in range(0, len(self.dataset))])
```

```
        self._dnc(pi, pn, dataparts[0], Area.ABOVE)
        self._dnc(pi, pn, dataparts[1], Area.BELOW)
```

3. main.py

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

print("Convex Hull for Visualization of Dataset Linear Separability Tests")

print("""
    1. Iris
    2. Wine
    3. Breasts Cancer
    """)

datanum = int(input("Masukkan dataset yang ingin diuji: "))
if datanum == 1:
    data = datasets.load_iris()
elif datanum == 2:
    data = datasets.load_wine()
elif datanum == 3:
    data = datasets.load_breast_cancer()
else:
    print("Input salah!")

for i in range(len(data.feature_names)):
    print(f'{i + 1}. {data.feature_names[i]}')
print("\n")

x = int(input("Pilih variabel pada sumbu-x: ")) - 1
y = int(input("Pilih variabel pada sumbu-y: ")) - 1

df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)

plt.figure(figsize=(10, 6))
colors = ['b', 'r', 'g']

plt.title(
    f'{data.feature_names[x].capitalize()} vs {data.feature_names[y].capitalize()}')
```

```
plt.xlabel(data.feature_names[x])
plt.ylabel(data.feature_names[y])

for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:, [x, y]].values
    hull = ConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull.simplices:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])

plt.legend()
plt.show()

print("Thank you!!!")
```

C. Screenshot Input dan Output

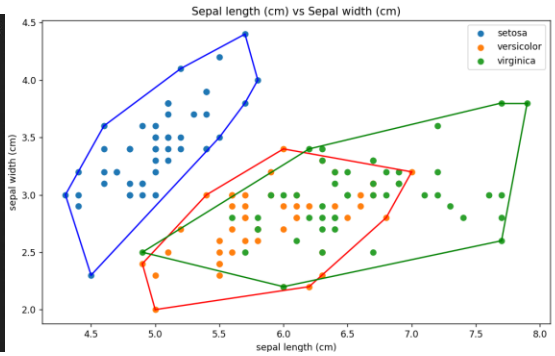
1. Dataset Iris (Sepal length vs Sepal width)

```
Convex Hull for Visualization of Dataset Linear Separability Tests

    1. Iris
    2. Wine
    3. Breasts Cancer

Masukkan dataset yang ingin diuji: 1
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)


Pilih variabel pada sumbu-x: 1
Pilih variabel pada sumbu-y: 2
```



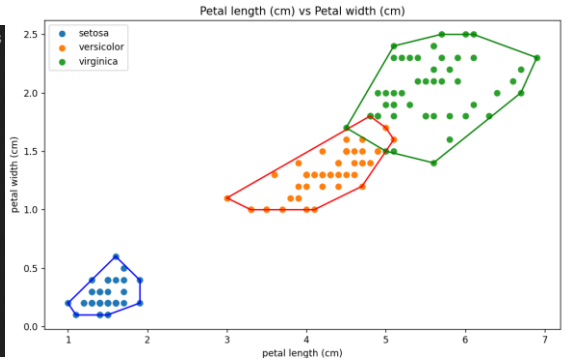
2. Dataset Iris (Petal length vs Petal width)

```
Convex Hull for Visualization of Dataset Linear Separability Tests

    1. Iris
    2. Wine
    3. Breasts Cancer

Masukkan dataset yang ingin diuji: 1
1. sepal length (cm)
2. sepal width (cm)
3. petal length (cm)
4. petal width (cm)


Pilih variabel pada sumbu-x: 3
Pilih variabel pada sumbu-y: 4
```



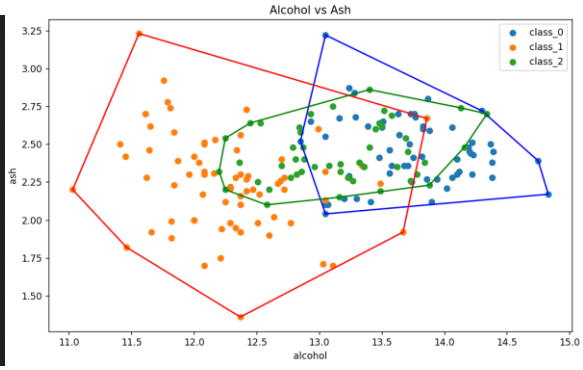
3. Dataset Wine (Alcohol vs Ash)

```
Convex Hull for Visualization of Dataset Linear Separability Tests

    1. Iris
    2. Wine
    3. Breasts Cancer

Masukkan dataset yang ingin diuji: 2
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


Pilih variabel pada sumbu-x: 1
Pilih variabel pada sumbu-y: 3
```



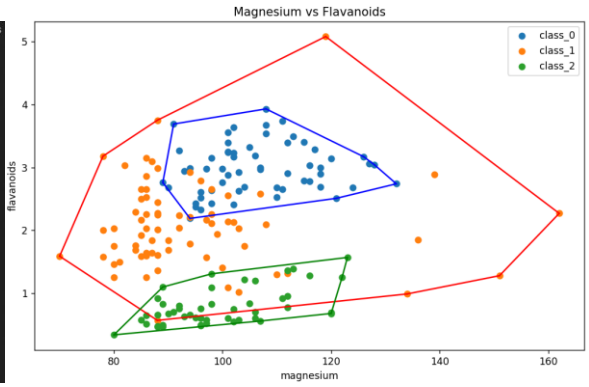
4. Dataset Wine (Magnesium vs Flavonoids)

```
Convex Hull for Visualization of Dataset Linear Separability Tests

    1. Iris
    2. Wine
    3. Breasts Cancer

Masukkan dataset yang ingin diuji: 2
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


Pilih variabel pada sumbu-x: 5
Pilih variabel pada sumbu-y: 7
```

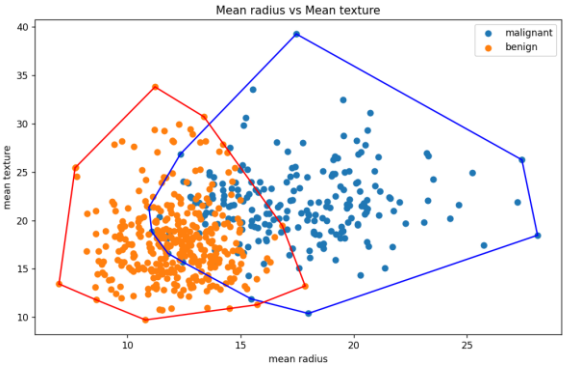


5. Dataset Breasts Cancer (Mean radius vs Mean texture)

```
Convex Hull for Visualization of Dataset Linear Separability Tests

    1. Iris
    2. Wine
    3. Breasts Cancer

Masukkan dataset yang ingin diuji: 3
1. mean radius
2. mean texture
3. mean perimeter
4. mean area
5. mean smoothness
6. mean compactness
7. mean concavity
8. mean concave points
9. mean symmetry
10. mean fractal dimension
11. radius error
12. texture error
13. perimeter error
14. area error
15. smoothness error
16. compactness error
17. concavity error
18. concave points error
```

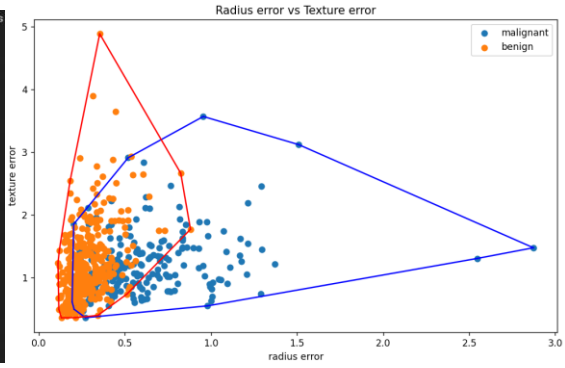


6. Dataset Breasts Cancer (Radius error vs Texture Error)

```
Convex Hull for Visualization of Dataset Linear Separability Tests

    1. Iris
    2. Wine
    3. Breasts Cancer

Masukkan dataset yang ingin diuji: 3
1. mean radius
2. mean texture
3. mean perimeter
4. mean area
5. mean smoothness
6. mean compactness
7. mean concavity
8. mean concave points
9. mean symmetry
10. mean fractal dimension
11. radius error
12. texture error
13. perimeter error
14. area error
15. smoothness error
16. compactness error
17. concavity error
18. concave points error
```



D. Alamat Drive

https://github.com/jovaandres/Convex-Hull-DnC

E. Alamat Drive

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.	✓	