LAPORAN TUGAS KECIL 2

IF2211 STRATEGI ALGORITMA

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

Disusun oleh:

Muhammad Akmal Arifin 13520037

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2022

A. Algoritma *Divide and Conquer*

Algoritma merupakan sekumpulan instruksi yang terstruktur dan terbatas yang diimplementasikan ke dalam bentuk program komputer untuk menyelesaikan suatu masalah komputasi tertentu. Algoritma dalam pemrograman berarti menjelaskan langkah-langkah yang diperlukan dalam penyelesaian suatu permasalahan. Strategi algoritma merupakan berbagai algoritma yang dilakukan secara umum untuk memecahkan persoalan secara algoritmis, sehingga dapat diterapkan pada bermacam-macam persoalan. Salah satu strategi algoritma adalah algoritma *Divide and Conquer.*

Algoritma *Divide and Conquer* merupakan salah satu strategi algoritma yang cara pengerjaannya menggunakan konsep rekursif, yaitu pengulangan hingga suatu basis. Pengulangan yang dilakukan adalah dengan cara membagi persoalan menjadi persoalan yang lebih kecil hingga mencapai nilai basis. Karena persoalan yang dilakukan dibagi menjadi kecil, maka di akhir penyelesaian perlu dilakukan *combine* yaitu menyatukan solusi-solusi dari bagian-bagian kecil persoalan.

Secara garis besar, algoritma *divide and conquer* yang digunakan untuk penyelesaian permasalahan *convex hull* pada program ini adalah

1. Mencari titik paling kiri dan titik paling kanan pada diagram lalu membuat garis di antara kedua titik tersebut.
2. Melakukan rekursif untuk sisi atas dan sisi bawah.
3. Rekursif dilakukan dengan cara mencari titik terjauh dari garis yang telah terbentuk (disesuaikan untuk sisi atas dan sisi bawah) kemudian membuat garis dari titik paling kiri menuju titik terjauh dan titik terjauh menuju titik paling kanan.
4. Setelah itu mencari titik-titik yang berada di sisi atas atau sisi bawah garis yang baru terbentuk kemudian memanggil rekursif kembali.
5. Rekursif dilakukan hingga mencapai basis, yaitu tidak ada titik yang berada di sisi atas atau sisi bawah garis.

B. *SOURCE CODE* PROGRAM

Program penyelesaian *Convex Hull* dalam tugas ini menggunakan bahasa pemrograman Python. Berikut hasil tangkapan layer *source code* program.

MyConvexHull

```python
import numpy as np

def myConvexHull(dataFrame):
    convexHullSet = []

    farLeft = dataFrame[0]
    farRight = dataFrame[0]

    for point in dataFrame:
        if(point[0] < farLeft[0]):
            farLeft = point
        if(point[0] > farRight[0]):
            farRight = point

    # Rekursif untuk sisi atas
    convexHullSet.append(farLeft)
    dataFrameAbove = findPointAbove(dataFrame, farLeft, farRight)
    convexHullReccursiveAbove(dataFrameAbove, farLeft, farRight, convexHullSet)

    # Rekursive untuk sisi bawah
    convexHullSet.append(farRight)
    dataFrameBelow = findPointBelow(dataFrame, farLeft, farRight)
    convexHullReccursiveBelow(dataFrameBelow, farLeft, farRight, convexHullSet)


    answerSets = convertConvexHull(dataFrame, convexHullSet)
    arrayAnswer = np.array(answerSets)
    return arrayAnswer

```

ConvexHullRecursiveAbove

```python
# Fungsi rekursif untuk menyelesaikan convexHull
# Basis : jika sebelah kiri atau sebelah kanan garis sudah tidak ada lagi titik
def convexHullReccursiveAbove(dataFrame, leftPoint, rightPoint, convexHullSet):
    if dataFrame:
        farthestPoint = findFarAbove(dataFrame, leftPoint, rightPoint)

        # Rekursif untuk sisi atas sebelah kiri
        dataFrameAboveLeft = findPointAbove(
            dataFrame, leftPoint, farthestPoint)
        convexHullReccursiveAbove(
            dataFrameAboveLeft, leftPoint, farthestPoint, convexHullSet)

        convexHullSet.append(farthestPoint)

        # Rekursif untuk sisi atas sebelah kanan
        dataFrameAboveRight = findPointAbove(
            dataFrame, farthestPoint, rightPoint)
        convexHullReccursiveAbove(
            dataFrameAboveRight, farthestPoint, rightPoint, convexHullSet)

```

ConvexHullRecursiveBelow

```
51  def convexHullReccursiveBelow(dataFrame, leftPoint, rightPoint, convexHullSet):
52      if dataFrame:
53          farthestPoint = findFarBelow(dataFrame, leftPoint, rightPoint)
54
55          # Rekursif untuk sisi atas sebelah kanan
56          dataFrameBelowRight = findPointBelow(
57              dataFrame, farthestPoint, rightPoint)
58          convexHullReccursiveBelow(
59              dataFrameBelowRight, farthestPoint, rightPoint, convexHullSet)
60
61          convexHullSet.append(farthestPoint)
62
63          # Rekursif untuk sisi atas sebelah kiri
64          dataFrameBelowLeft = findPointBelow(
65              dataFrame, leftPoint, farthestPoint)
66          convexHullReccursiveBelow(
67              dataFrameBelowLeft, leftPoint, farthestPoint, convexHullSet)
68
```

FindFarAbove

```
69  # Fungsi mengembailkan point yang merupakan titik terjatuh pada sisi atas antara
70  # dua titik yang membentuk garis
71  def findFarAbove(dataFrame, leftPoint, rightPoint):
72      distancePoint = (dataFrame[0][0] - leftPoint[0])*(rightPoint[1] - leftPoint[1]) - \
73          (dataFrame[0][1] - leftPoint[1])*(rightPoint[0] - leftPoint[0])
74      point = dataFrame[0]
75      for data in dataFrame:
76          d = (data[0] - leftPoint[0])*(rightPoint[1] - leftPoint[1]) - \
77              (data[1] - leftPoint[1])*(rightPoint[0] - leftPoint[0])
78          if (d < distancePoint):
79              point = data
80              distancePoint = d
81      return point
82
```

FindFarBelow

```
83  # Fungsi mengembailkan point yang merupakan titik terjatuh pada sisi bawah antara
84  # dua titik yang membentuk garis
85  def findFarBelow(dataFrame, leftPoint, rightPoint):
86      distancePoint = (dataFrame[0][0] - leftPoint[0])*(rightPoint[1] - leftPoint[1]) - \
87          (dataFrame[0][1] - leftPoint[1])*(rightPoint[0] - leftPoint[0])
88      point = dataFrame[0]
89      for data in dataFrame:
90          d = (data[0] - leftPoint[0])*(rightPoint[1] - leftPoint[1]) - \
91              (data[1] - leftPoint[1])*(rightPoint[0] - leftPoint[0])
92          if (d > distancePoint):
93              point = data
94              distancePoint = d
95      return point
96
```

FindPointAbove

```python
# Fungsi mengembalikan sets of point yang merupakan titik-titik yang berada di sisi atas
# antara dua titik yang memebentuk garis
def findPointAbove(dataFrame, leftPoint, rightPoint):
    sets = []
    for data in dataFrame:
        d = (data[0] - leftPoint[0])*(rightPoint[1] - leftPoint[1]) - \
            (data[1] - leftPoint[1])*(rightPoint[0] - leftPoint[0])
        if (d < 0):
            sets.append(data)
    return sets

```

FindPointBelow

```python
# Fungsi mengembalikan sets of point yang merupakan titik-titik yang berada di sisi bawah
# antara dua titik yang memebentuk garis
def findPointBelow(dataFrame, leftPoint, rightPoint):
    sets = []
    for data in dataFrame:
        d = (data[0] - leftPoint[0])*(rightPoint[1] - leftPoint[1]) - \
            (data[1] - leftPoint[1])*(rightPoint[0] - leftPoint[0])
        if (d > 0):
            sets.append(data)
    return sets

```

ConvertConvexHull

```python
# Fungsi mengembalikan numpy array of index dari dataFrame yang di input di awal
# Berdasarkan jawaban yang telah dicari sebelumnya
def convertConvexHull(dataFrame, convexHullSet):
    answerSets = []
    for i in range(len(convexHullSet)-1):
        for j in range(len(dataFrame)):
            if (dataFrame[j][0] == convexHullSet[i][0] and dataFrame[j][1] == convexHullSet[i][1]):
                leftSet = j
            if (dataFrame[j][0] == convexHullSet[i+1][0] and dataFrame[j][1] == convexHullSet[i+1][1]):
                rightSet = j
        answerSets.append([leftSet, rightSet])
    for j in range(len(dataFrame)):
        if (dataFrame[j][0] == convexHullSet[len(convexHullSet)-1][0] and dataFrame[j][1] == convexHullSet[len(convexHullSet)-1][1]):
            leftSet = j
        if (dataFrame[j][0] == convexHullSet[0][0] and dataFrame[j][1] == convexHullSet[0][1]):
            rightSet = j
    answerSets.append([leftSet, rightSet])
    return answerSets

```

dataVisualitationIris.py

```
# visualisasi data iris
data = datasets.load_iris()
# buat DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)

```

Petal Width vs Petal Length

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

plt.show()

```

Sepal Width vs Sepal Length

```
# visualisasi Sepal Width vs Sepal Length
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
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
        plt.legend()

plt.show()
```

dataVisualitationWine.py

```
# visualisasi data wine
data = datasets.load_wine()
# buat DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)

```

Malic Acid vs Alcohol

```
# visualisasi Malic Acid vs Alcohol
plt.figure(figsize=(10, 6))
colors = ['b', 'r', 'g']
plt.title('Malic Acid vs Alcohol')
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
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
        plt.legend()

plt.show()

```

Total Phenols vs Magnesium

```
# visualisasi Total Phenols vs Magnesium
plt.figure(figsize=(10, 6))
colors = ['b', 'r', 'g']
plt.title('Total Phenols vs Magnesium')
plt.xlabel(data.feature_names[4])
plt.ylabel(data.feature_names[5])

for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:, [4, 5]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
        plt.legend()

plt.show()

```

dataVisualitationBreastCancer.py

```
# visualisasi data iris
data = datasets.load_breast_cancer()
# buat DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)

```

Area vs Perimeter

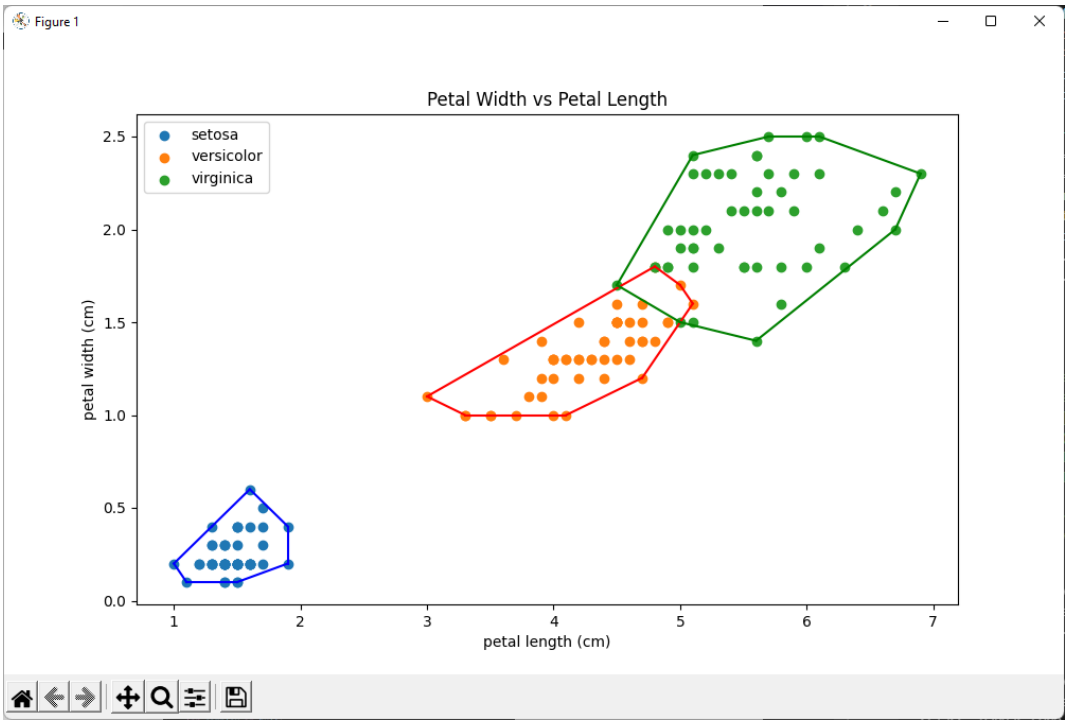
```
# visualisasi Area vs Perimeter
plt.figure(figsize=(10, 6))
colors = ['b', 'r', 'g']
plt.title('Area vs Perimeter')
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])

for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:, [2, 3]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
        plt.legend()

plt.show()

```

Compactness vs Smoothness

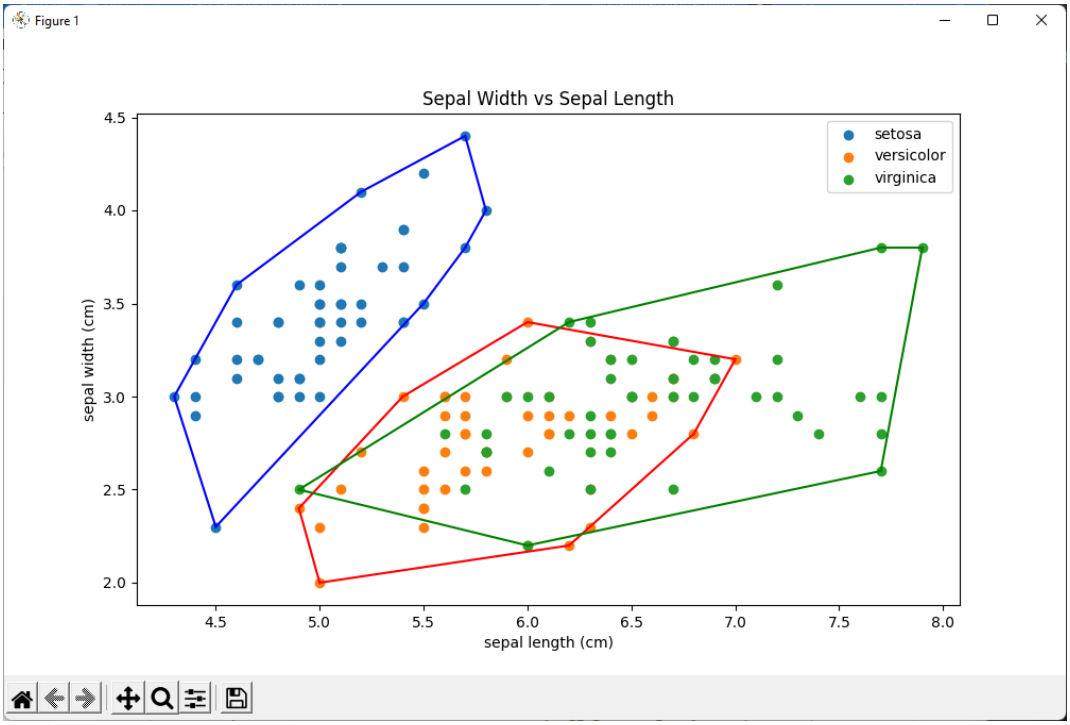
```
# visualisasi Compactness vs Smoothness
plt.figure(figsize=(10, 6))
colors = ['b', 'r', 'g']
plt.title('Compactness vs Smoothness')
plt.xlabel(data.feature_names[4])
plt.ylabel(data.feature_names[5])

for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:, [4, 5]].values
    hull = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for simplex in hull:
        plt.plot(bucket[simplex, 0], bucket[simplex, 1], colors[i])
        plt.legend()

plt.show()

```

C. *SCREENSHOT INPUT* DAN *OUTPUT*

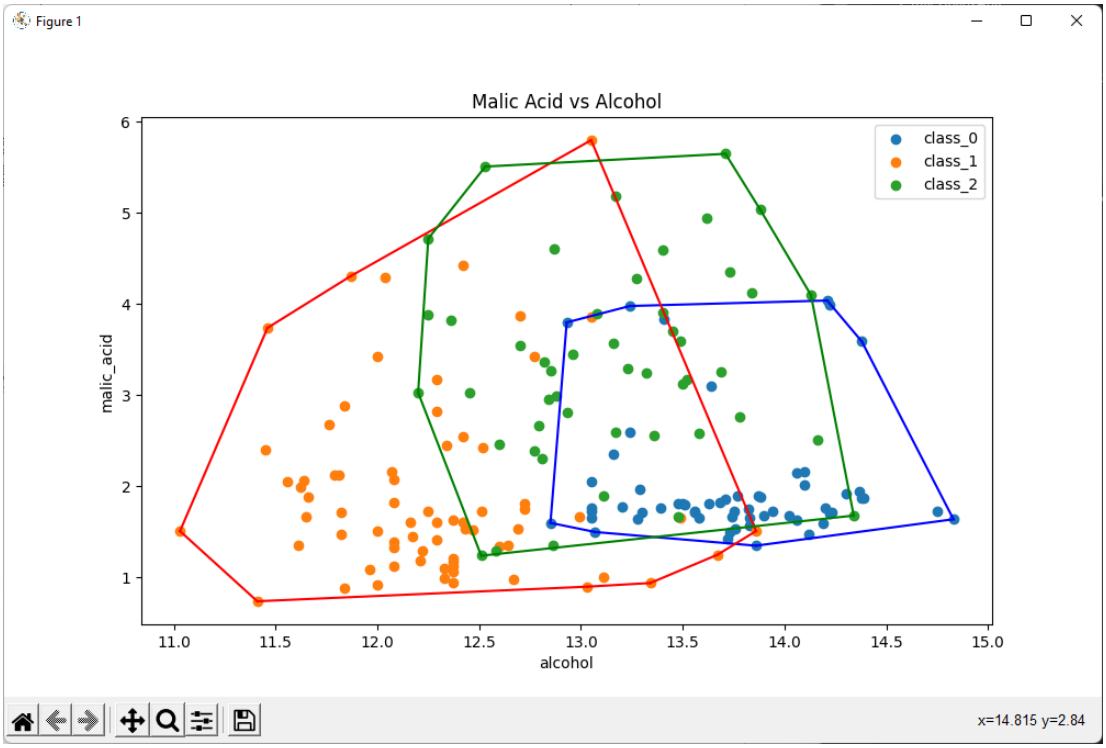
Dataset Iris : Petal Width vs Petal Length



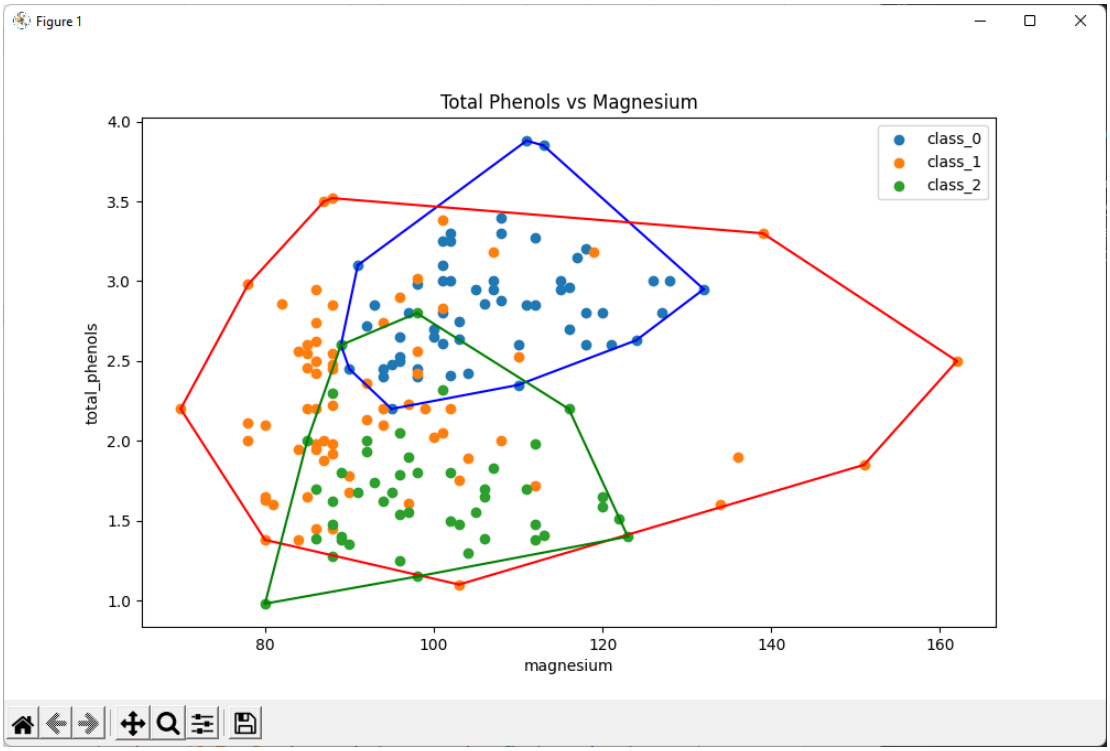
Dataset Iris: Sepal Width vs Sepal Length



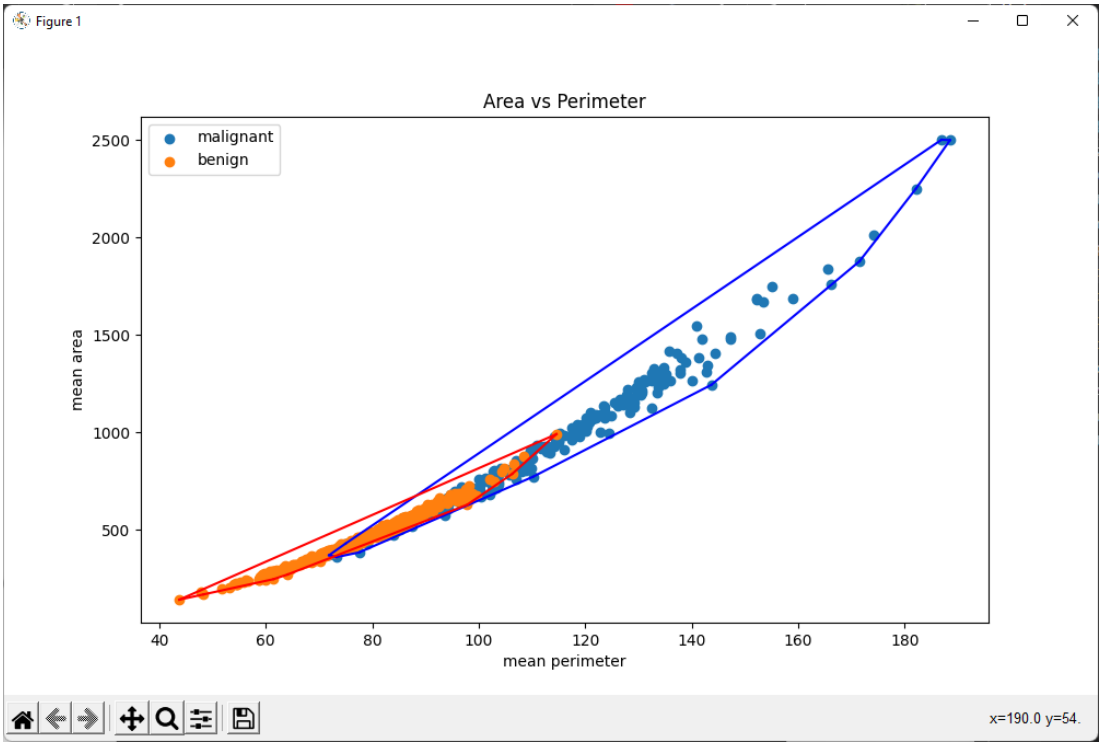
Dataset Wine: Malic Acid vs Alcohol



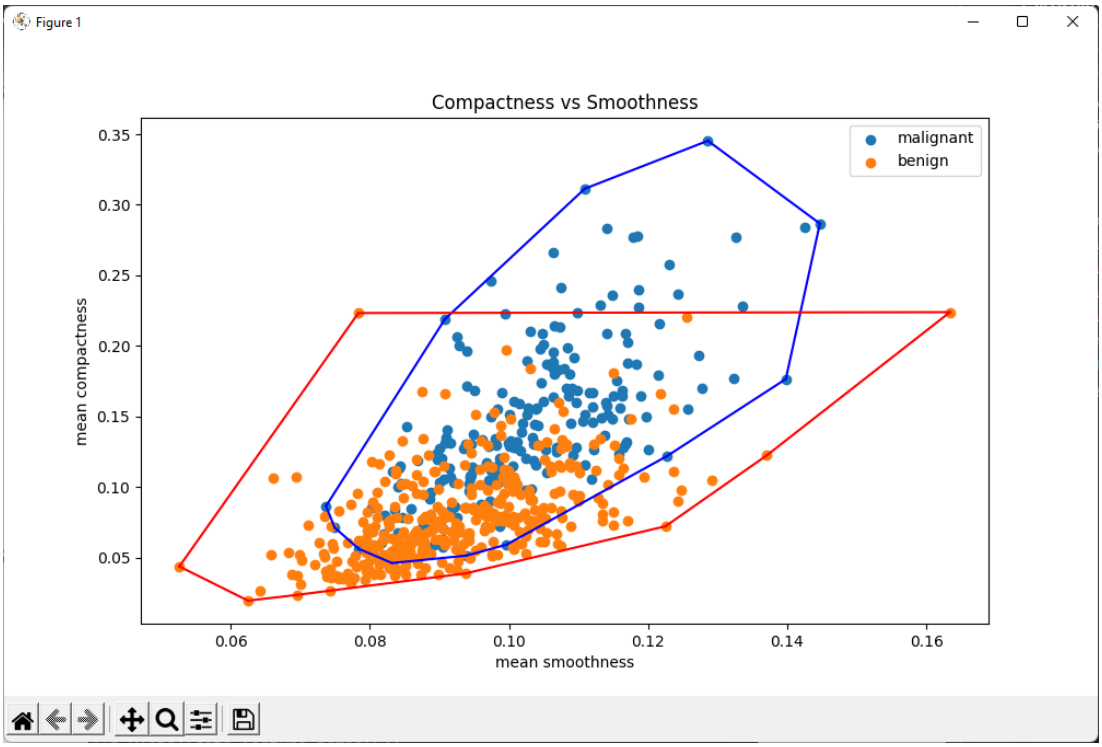
Dataset Wine: Total Phenols vs Magnesium



Dataset Breast Cancer: Area vs Perimeter



Dataset Breast Cancer: Compactness vs Smoothness



D. ALAMAT *DRIVE* KODE PROGRAM

Alamat drive untuk kode program ini adalah :

https://github.com/AkmalArifin/IF2211-Tugas-Kecil-2-Convex-Hull

E. CEKLIST

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.	√	