TUGAS KECIL 2

IF2211 STRATEGI ALGORITMA

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

Oleh

13520153 – Vito Ghifari

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER 2 2021/2022

Algoritma Divide and Conquer pada Permasalahan Convex Hull

Algoritma yang digunakan untuk mencari solusi *convex hull* dengan pendekatan *divide and conquer* adalah sebagai berikut.

1. Pertama, dicari terlebih dahulu titik ekstrem dari suatu himpunan titik pada bidang planar. Dalam hal ini, titik ekstrem tersebut dapat berupa titik yang mempunyai absis minimum (p_1) dan absis maksimum (p_n).
2. Garis yang menghubungkan p_1 dan p_n akan membagi himpunan titik tersebut menjadi dua bagian, yaitu himpunan titik yang berada di atas garis (S1) dan di bawah garis (S2). Untuk mengecek jika suatu titik berada di atas garis, dapat digunakan rumus sebagai berikut dengan $p_1(x_1, y_1)$, $p_2(x_2, y_2)$ dan $p_{test}(x_3, y_3)$.

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

   Jika hasilnya adalah positif, p_{test} berada di atas garis. Sebaliknya, jika hasilnya negatif, p_{test} berada di bawah garis. Sementara itu, titik yang berada tepat di garis akan menghasilkan nilai nol.
3. Titik-titik yang berada pada S1 dapat membentuk *convex hull* bagian atas, sedangkan titik-titik pada S2 dapat membentuk *convex hull* bagian bawah.
4. Mulai dari sini, algoritma *divide and conquer* dapat diterapkan. Pada S1, cari suatu titik yang jaraknya paling besar dari garis p_1p_n. Titik ini dinamakan p_{max}. Untuk mencari p_{max}, diperlukan rumus untuk mencari jarak dari suatu titik ke garis yang dihubungkan oleh dua titik. Rumus tersebut adalah

$$distance = \frac{|(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$

5. Jika terdapat lebih dari satu titik dengan jarak paling besar, cari titik yang membentuk sudut $p_{max}p_1p_n$ terbesar.
6. Titik p_1 dan p_{max} dihubungkan oleh suatu garis. Garis ini akan membagi kembali titik-titik di atas garis dan di bawah garis. Cari kembali titik yang merupakan titik terjauh dari garis yang dihubungkan p_1 dan p_{max}.
7. Ulangi langkah sebelumnya hingga tidak ada lagi titik yang berada di atas garis. Semua titik sebelumnya yang terpilih menjadi jarak terjauh (p_{max}) akan menjadi garis terluar untuk *convex hull*.
8. Langkah-langkah ini juga diterapkan pada p_{max} dan p_n, yaitu mencari semua titik dengan jarak terbesar hingga tidak ada lagi titik yang berada di atas garis.
9. Ulangi dari langkah ke-4 hingga langkah ke-8 untuk S2, tetapi pencarian titik dengan jarak terbesar dilakukan untuk kumpulan titik yang berada di bawah garis.
10. Garis yang terbentuk dari algoritma *divide and conquer* pada *convex hull* bagian atas (S1) dihubungkan dengan garis bagian bawah (S2). Dengan ini, *convex hull* yang utuh akan terbentuk.

Kode Sumber Program

ConvexHull.ipynb

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

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

# Data wine
data_wine = datasets.load_wine()
df_wine = pd.DataFrame(data_wine.data,
columns=data_wine.feature_names)
df_wine['Target'] = pd.DataFrame(data_wine.target)

# data breast cancer
data_cancer = datasets.load_breast_cancer()
df_cancer = pd.DataFrame(data_cancer.data,
columns=data_cancer.feature_names)
df_cancer['Target'] = pd.DataFrame(data_cancer.target)
```

```
class myConvexHull:
    def __init__(self, frame: pd.DataFrame, col_x: str = None, col_y:
str = None):
        self.frame = frame
        self.col_x = col_x
        self.col_y = col_y
        self.simplices = self.solveConvexHull(frame, col_x, col_y)

    def det(self, x_1: float, y_1: float, x_2: float, y_2: float,
x_test: float, y_test: float) -> float:
        """
        Menghitung determinan dari titik (x_test, y_test) terhadap
garis
        yang dihubungkan dengan titik (x_1, y_1) dan (x_2, y_2)

        Args:
            x_1: absis titik kedua yang dilewati garis
            y_1: ordinat titik pertama yang dilewati garis
            x_1: absis titik kedua yang dilewati garis
            y_1: ordinat titik kedua yang dilewati garis
```

```
            x_test: absis titik yang akan dites
            y_test: ordinat titik yang akan dites

        Returns:
            Hasil determinan dengan tipe data float.
                Jika hasilnya positif, titik terdapat di atas garis.
                Jika hasilnya nol, titik terdapat pada garis
                Jika hasilnya negarif, titik terdapat di bawah garis.

        """
        return x_1*y_2 + x_test*y_1 + x_2*y_test - x_test*y_2 -
x_2*y_1 - x_1*y_test

    def distance(self, x_1: float, y_1: float, x_2: float, y_2:float,
x_test: float, y_test: float) -> float:
        """
        Menghitung jarak titik (x_test, y_test) dari garis yang
dihubungkan titik (x_1, y_1) dan (x_2, y_2).

        Args:
            x_1: absis titik kedua yang dilewati garis
            y_1: ordinat titik pertama yang dilewati garis
            x_1: absis titik kedua yang dilewati garis
            y_1: ordinat titik kedua yang dilewati garis
            x_test: absis titik yang akan dites
            y_test: ordinat titik yang akan dites

        Returns:
            Jarak titik (x_test, y_test) terhadap garis yang
dihubungkan titik (x_1, y_1) dan (x_2, y_2).
        """
        return abs((x_2-x_1)*(y_test-y_1) - (y_2-y_1)*(x_test-
x_1))/((x_2-x_1)**2 + (y_2-y_1)**2)**(1/2)

    def dnc_convexhull(self, frame: pd.DataFrame, p_min: int, p_n:
int, includes: list, up=True) -> list:
        """
        Implementasi algoritma divide and conquer pada convex hull.
        Menerima dataframe yang terdiria atas 2 kolom. Kolom pertama
sebagai absis dan kolom kedua sebagai ordinat.
        Mengembalikan kumpulan dua titik yang dihubungkan garis
convex hull.

        Args:
            frame: DataFrame yang digunakan
            p_min: index dari titik pada frame dengan absis lebih
kecil yang membentuk garis convex hull
```

```
            p_n: index dari titik pada frame dengan absis lebih besar
yang membentuk garis convex hull
            col_x: kolom yang digunakan sebagai absis
            col_y: kolom yang digunakan sebagai absis
        Returns:
            List berisi dua titik sebagai garis convex hull, yang
berada dalam list.
        """
        if len(includes) == 0:
            # Tidak ada titik antara p_min dan p_n, artinya titik
p_min dan p_n membentuk garis convex hull
            return [[p_min, p_n]]
        elif len(includes) == 1:
            # Hanya ada 1 titik antara p_min dan p_n, artinya garis
convex hull antara p_min dan p_n
            # dihubungkan oleh titik tersebut
            return [[p_min, includes[0]], [includes[0], p_n]]

        # Mengeluarkan p_min dan p_n jika masuk ke includes
        inc_list = list(set(includes) - {p_min, p_n})

        # kolom x dan y
        col_x = frame.columns[0]
        col_y = frame.columns[1]

        x_pmin = frame.loc[p_min][col_x]
        y_pmin = frame.loc[p_min][col_y]

        x_pn = frame.loc[p_n][col_x]
        y_pn = frame.loc[p_n][col_y]

        # Menghitung jarak maksimal titik dari titik p_min dan p_n
        p_max = self.distance(x_pmin, y_pmin, x_pn, y_pn,
frame.loc[inc_list][col_x], frame.loc[inc_list][col_y]).idxmax()
        x_pmax = frame.loc[p_max][col_x]
        y_pmax = frame.loc[p_max][col_y]

        inc_list = list(set(inc_list) - {p_max})

        # Mencari semua titik pada area S1 dan S2
        S1 = []
        S2 = []
        for i, row in frame[[col_x, col_y]].loc[inc_list].iterrows():
            if (x_pmin < x_pn):
                indicator_1 = self.det(x_pmin, y_pmin, x_pmax,
y_pmax, row[col_x], row[col_y])
            else:
```

```
                indicator_1 = self.det(x_pmax, y_pmax, x_pmin,
y_pmin, row[col_x], row[col_y])
            if (x_pn < x_pmax):
                indicator_2 = self.det(x_pn, y_pn, x_pmax, y_pmax,
row[col_x], row[col_y])
            else:
                indicator_2 = self.det(x_pmax, y_pmax, x_pn, y_pn,
row[col_x], row[col_y])

            if up:
                if indicator_1 > 0:
                    S1.append(i)
                if indicator_2 > 0:
                    S2.append(i)
            else:
                if indicator_1 < 0:
                    S1.append(i)
                if indicator_2 < 0:
                    S2.append(i)

        res_S1 = self.dnc_convexhull(frame, p_min, p_max, S1, up=up)
        res_S2 = self.dnc_convexhull(frame, p_max, p_n, S2, up=up)

        return res_S1 + res_S2


    def solveConvexHull(self, frame: pd.DataFrame, col_x=None,
col_y=None):
        """
        Eksekusi pembentukan convex hull dengan algoritma divide and
conquer.

        Args:
            frame: DataFrame yang digunakan
            col_x: kolom yang digunakan sebagai absis
            col_y: kolom yang digunakan sebagai ordinat
        Returns:
            List berisi dua titik sebagai garis convex hull, yang
berada dalam list.
        """
        if len(frame.columns) < 2:
            raise Exception("Jumlah kolom kurang dari 2")

        if col_x is None:
            col_x = frame.columns[0]
        if col_y is None:
            col_y = frame.columns[1]
```

```
        # Mencari titik dengan absis terkecil
        min_data = frame[frame[col_x] == frame[col_x].min()]
        min_x = list(min_data[col_x])[0]
        min_y = list(min_data[col_y])[0]

        # Mencari titik dengan absis terbesar
        max_data = frame[frame[col_x] == frame[col_x].max()]
        max_x = list(max_data[col_x])[0]
        max_y = list(max_data[col_y])[0]

        # Menghilangkan index min-data dan max_data pada pencarian
titik
        includes = list(set(frame.index) - {max_data.index[0],
min_data.index[0]})

        # Membagi area dengan S1 dan S2
        S1 = []
        S2 = []
        for i, row in frame.loc[includes].iterrows():
            res = self.det(min_x, min_y, max_x, max_y, row[col_x],
row[col_y])
            if res > 0:
                S1.append(i)
            elif res < 0:
                S2.append(i)

        res_S1 = self.dnc_convexhull(frame[[col_x, col_y]],
min_data.index[0], max_data.index[0], S1)
        res_S2 = self.dnc_convexhull(frame[[col_x, col_y]],
min_data.index[0], max_data.index[0], S2, up=False)
        return res_S1 + res_S2
```

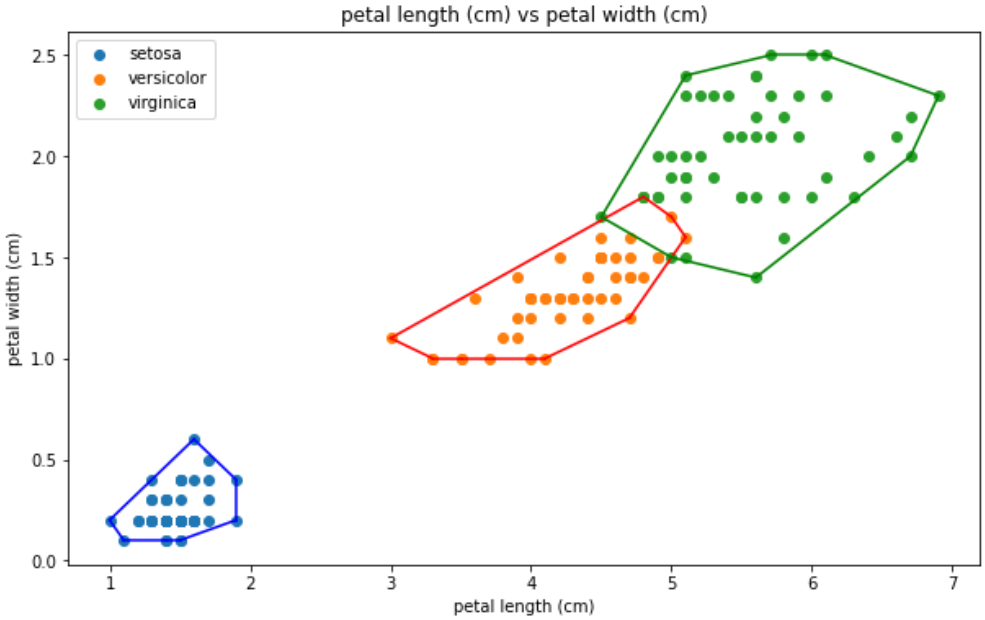
```
# Template untuk melakukan input dataframe menjadi visualisasi
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])
feature_0 = data.feature_names[0]
feature_1 = data.feature_names[1]
plt.title(feature_0 + " vs " + feature_1)
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    hull = myConvexHull(bucket, feature_0, feature_1)
    # ConvexHull Divide & Conquer
    plt.scatter(bucket.loc[:, feature_0], bucket.loc[:, feature_1],
label=data.target_names[i])
    for simplex in hull.simplices:
        print(simplex)
```

```
        plt.plot(bucket.loc[simplex, feature_0], bucket.loc[simplex,
feature_1], colors[i])
    print("----")
plt.legend()
```

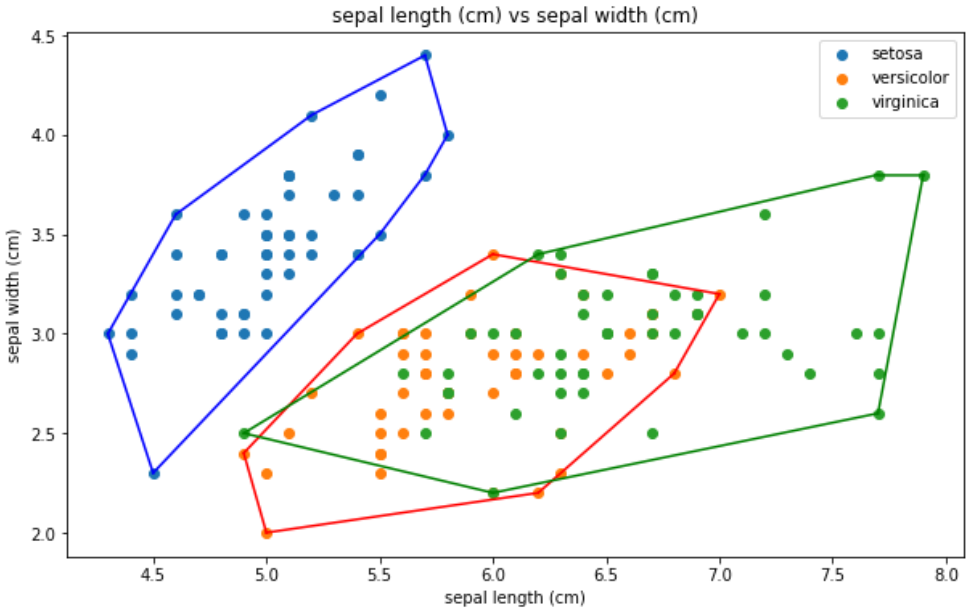
```
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.xlabel(data_wine.feature_names[3])
plt.ylabel(data_wine.feature_names[4])
feature_0 = data_wine.feature_names[3]
feature_1 = data_wine.feature_names[4]
plt.title(feature_0 + " vs " + feature_1)
for i in range(len(data_wine.target_names)):
    bucket = df_wine[df_wine['Target'] == i]
    hull = myConvexHull(bucket, feature_0, feature_1)
    # ConvexHull Divide & Conquer
    plt.scatter(bucket.loc[:, feature_0], bucket.loc[:, feature_1],
label=data_wine.target_names[i])
    for simplex in hull.simplices:
        print(simplex)
        plt.plot(bucket.loc[simplex, feature_0], bucket.loc[simplex,
feature_1], colors[i])
    print("----")
plt.legend()
```

Screenshot Hasil Eksekusi Program

I. Iris dataset

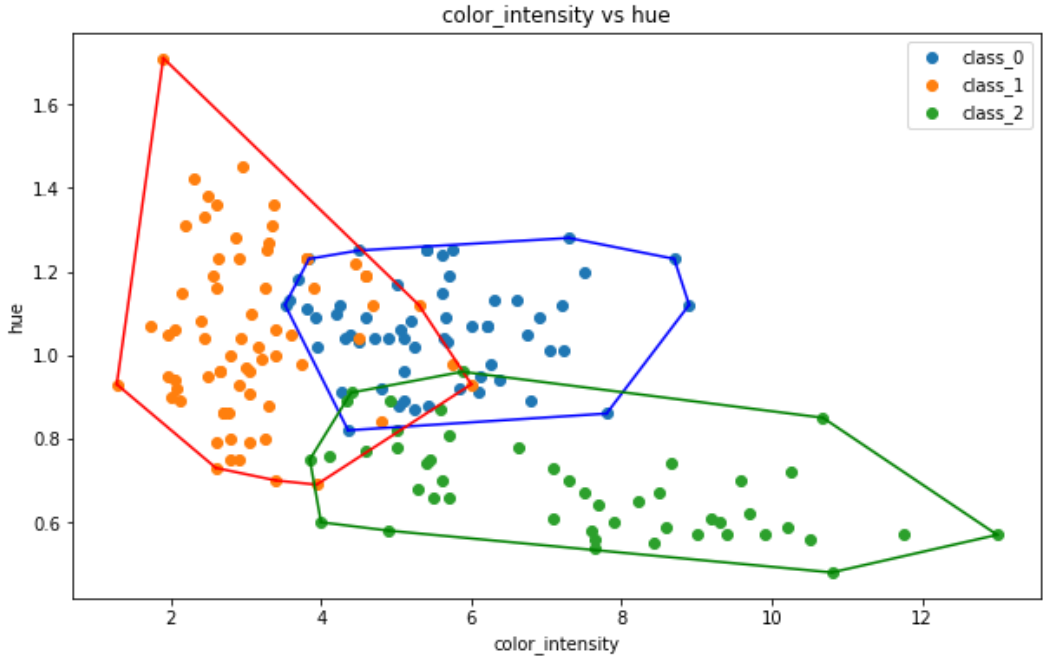


Gambar 1. Visualisasi *Convex Hull* pada petal length vs petal width

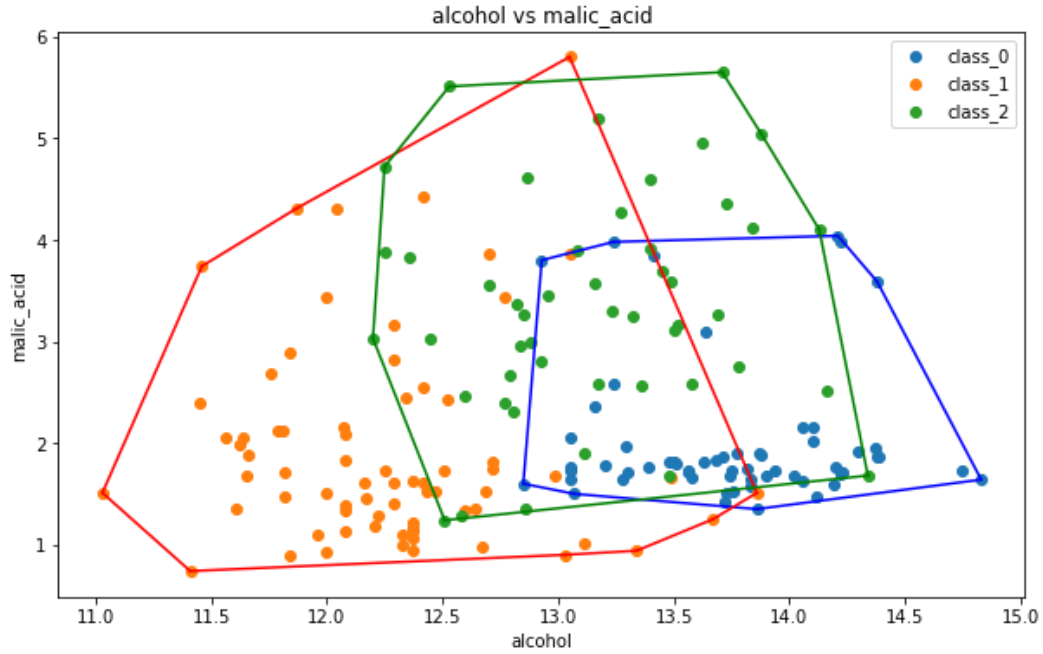


Gambar 2. Visualisasi *Convex Hull* pada sepal length vs sepal width

II. Wine Dataset

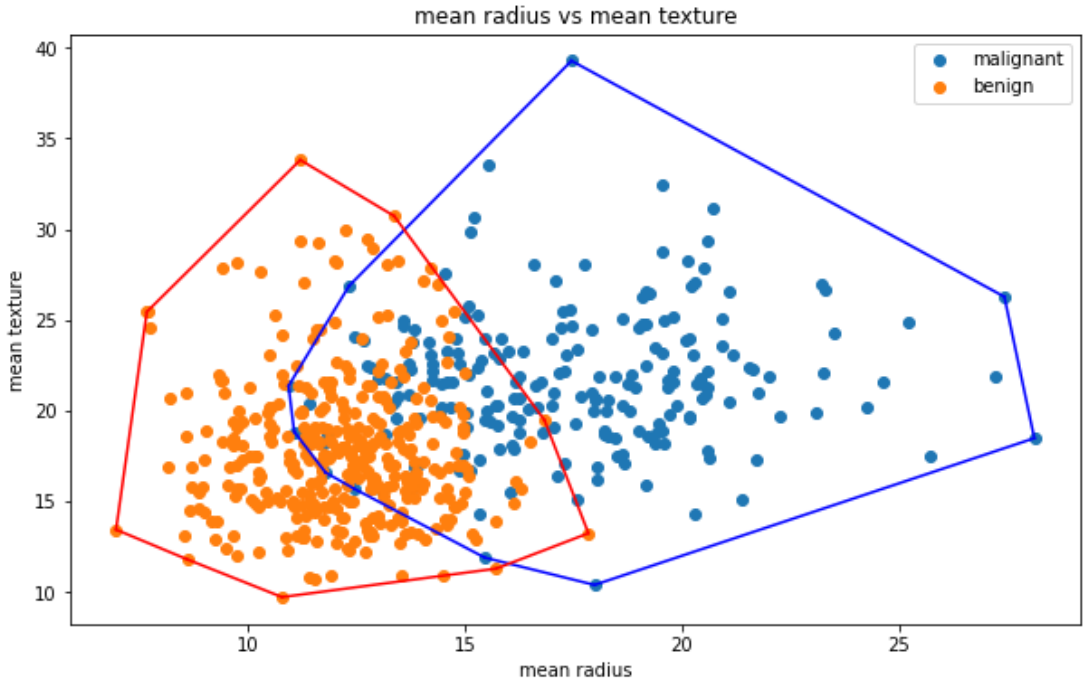


Gambar 3. Visualisasi *Convex Hull* pada color intensity vs hue

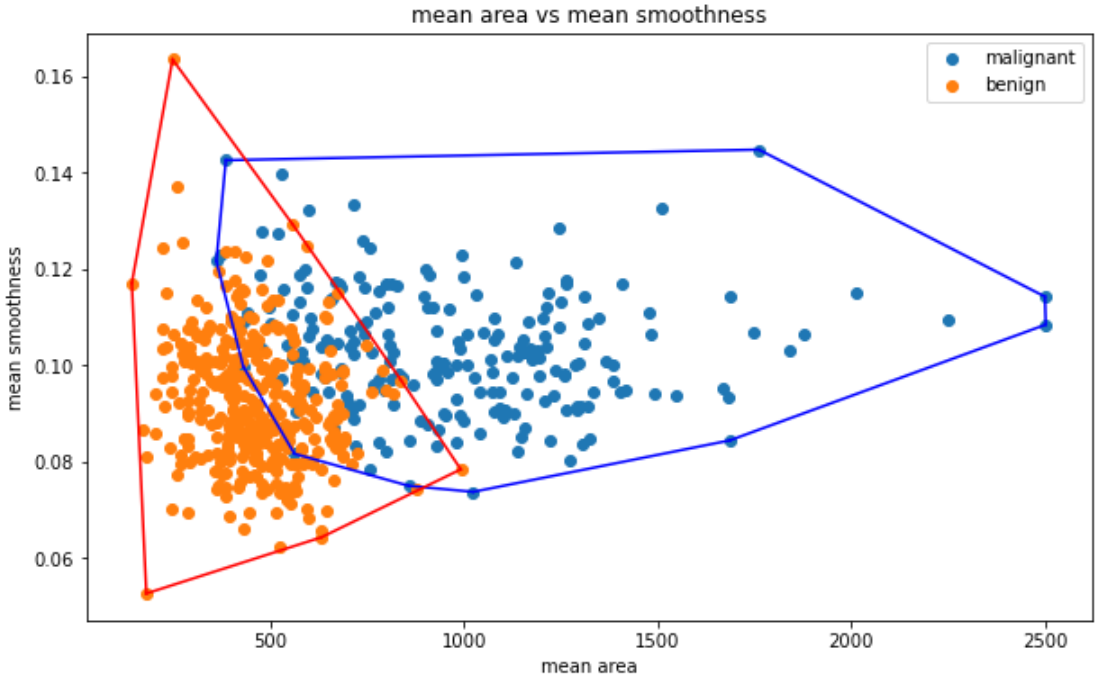


Gambar 4. Visualisasi *Convex Hull* pada alcohol vs malic acid

III. Breast Cancer Dataset



Gambar 5. Visualisasi *Convex Hull* pada mean radius vs mean texture



Gambar 5. Visualisasi *Convex Hull* pada mean area vs mean smoothness

Link Repository Tugas Kecil 2 Strategi Algoritma

https://github.com/VanillaMacchiato/tucil2-stima

Daftar Periksa Program

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	✓	