Laporan

Tugas Kecil 2 IF2211 Strategi Algoritma

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

Disusun Oleh:

Kent Liusudarso	13520069

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

DAFTAR ISI

ALGORITMA DIVIDE AND CONQUER

Keterangan:

- S: himpunan titik sebanyak n, dengan n >1, yaitu titik p1(x1, y1) hingga pn(xn, yn) pada bidang kartesian dua dimensi
- Kumpulan titik diurutkan berdasarkan nilai absis yang menaik, dan jika ada nilai absis yang sama, maka diurutkan dengan nilai ordinat yang menaik
- p1 dan pn adalah dua titik ekstrim yang akan membentuk convex hull untuk kumpulan titik tersebut.

Algoritma:

1. Pertama, terdapat titik p1 dan pn yang membentuk garis (p1pn) yang membagi S menjadi dua bagian yaitu S1 (kumpulan titik di sebelah kiri atau atas garis p1pn) dan S2 (kumpulan titik di sebelah kanan atau bawah garis p1pn).
2. Untuk menentukan apakah sebuah titik berada di S1 atau S2, digunakanlah penentuan determinan:

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

Titik (x3,y3) berada di sebelah kiri dari garis ((x1,y1),(x2,y2)) jika hasil determinan positif

3. Semua titik yang berada pada garis p1pn atau menghasilkan determinan nol, tidak mungkin membentuk *convex hull*, sehingga dapat diabaikan.
4. Kumpulan titik S1 akan membentuk *convex hull* bagian atas, sedangkan S2 akan membentuk *convex hull* bagian bawah
5. Untuk sebuah bagian (misal S1) terdapat dua kemungkinan:
   a. Jika tidak ada titik selain S1, maka titik p1 dan pn menjadi pembentuk *convex hull* bagian S1
   b. Jika S1 tidak kosong, pilih sebuah titik yang memiliki jarak terjauh dari garis p1pn (misal pmax). Jika terdapat titik dengan jarak yang sama, pilih titik yang memaksimalkan sudut pmaxp1pn
   c. Semua titik yang berada di dalam daerah segitika pmaxpipn diabaikan karena tidak mungkin membentuk *convex hull*
6. Tentukan kumpulan titik yang berada di sebelah kiri garis p1pmax (bagian S1,1), dan sebelah akan garis pnpmax (bagian S1,2)
7. Ulangi tahap 5 dan 6 untuk S2 hingga bagian kiri dan kanan kosong
8. Terakhir, kembalikan titik yang dihasilkan

SOURCE CODE

- myConvexHull.py

```
import numpy as np
import math

vertices = []
simplices = []

def getP1Pn(bucket):
    p1 = bucket[0]
    pn = bucket[0]

    for i in range (len(bucket)):
        if(bucket[i] < p1):
            p1 = bucket[i]
        elif(bucket[i] > pn):
            pn = bucket[i]

    return p1,pn

def getLeftRightPoints(bucket,p1,pn):
    left = []
    right = []

    for i in range(len(bucket)):
            a = np.array([p1,pn,bucket[i]])
            b = np.array([[1], [1], [1]])
            m = np.append(a, b, axis=1)
            det = np.linalg.det(m)
            if (det > 0):
                left.append(bucket[i])
            if (det < 0):
                right.append(bucket[i])

    return left,right

def getFarthestPoint(bucket,p1,pn):
    dist = 0
    idx = 0

    for i in range(len(bucket)):
        p1dist = math.sqrt((bucket[i][0] - p1[0])**2 + (bucket[i][1] -
p1[1])**2)
        pndist = math.sqrt((bucket[i][0] - pn[0])**2 + (bucket[i][1] -
pn[1])**2)

        if ((p1dist + pndist) > dist):
            dist = p1dist + pndist
            idx = i

    return bucket[idx]

def recursiveLeft(bucket,p1,pn):
    if(len(bucket) == 0):
        return
    else:
        farthestPoint = getFarthestPoint(bucket,p1,pn)

        bucket.remove(farthestPoint)
        vertices.append(farthestPoint)

        left1, right1 = getLeftRightPoints(bucket,p1,farthestPoint)
        left2, right2 = getLeftRightPoints(bucket,farthestPoint,pn)

        recursiveLeft(left1,p1,farthestPoint)
        recursiveLeft(left2,farthestPoint,pn)

def recursiveRight(bucket,p1,pn):
    if(len(bucket) == 0):
        return
    else:
        farthestPoint = getFarthestPoint(bucket,p1,pn)

        bucket.remove(farthestPoint)
        vertices.append(farthestPoint)

        left1, right1 = getLeftRightPoints(bucket,p1,farthestPoint)
        left2, right2 = getLeftRightPoints(bucket,farthestPoint,pn)

        recursiveRight(right1,p1,farthestPoint)
        recursiveRight(right2,farthestPoint,pn)

def recursiveMain(bucket,p1,pn):
    vertices.clear()

    leftPoints, rightPoints = getLeftRightPoints(bucket,p1,pn)

    vertices.append(p1)
    vertices.append(pn)

    recursiveLeft(leftPoints,p1,pn)
    recursiveRight(rightPoints,p1,pn)

def getVertices(bucket):
    bucket = bucket.tolist()
    p1, pn = getP1Pn(bucket)

    recursiveMain(bucket,p1,pn)

    return vertices

def sortVertices(vertices):
    centerX = sum(point[0] for point in vertices) / len(vertices)
    centerY = sum(point[1] for point in vertices) / len(vertices)

    vertices.sort(key = lambda point: math.atan2(point[0] - centerX, point[1] - centerY))

    return vertices

def getSimplices(vertices):
    simplices.clear()

    for i in range(len(vertices)):
        if(i == len(vertices) - 1):
            simplices.append([vertices[i], vertices[0]])
        else:
            simplices.append([vertices[i], vertices[i+1]])

def myConvexHull(bucket):
    vertices = getVertices(bucket)
    SortedVertices = sortVertices(vertices)
    getSimplices(SortedVertices)

    return simplices
```

- main.ipynb (converted to .py)

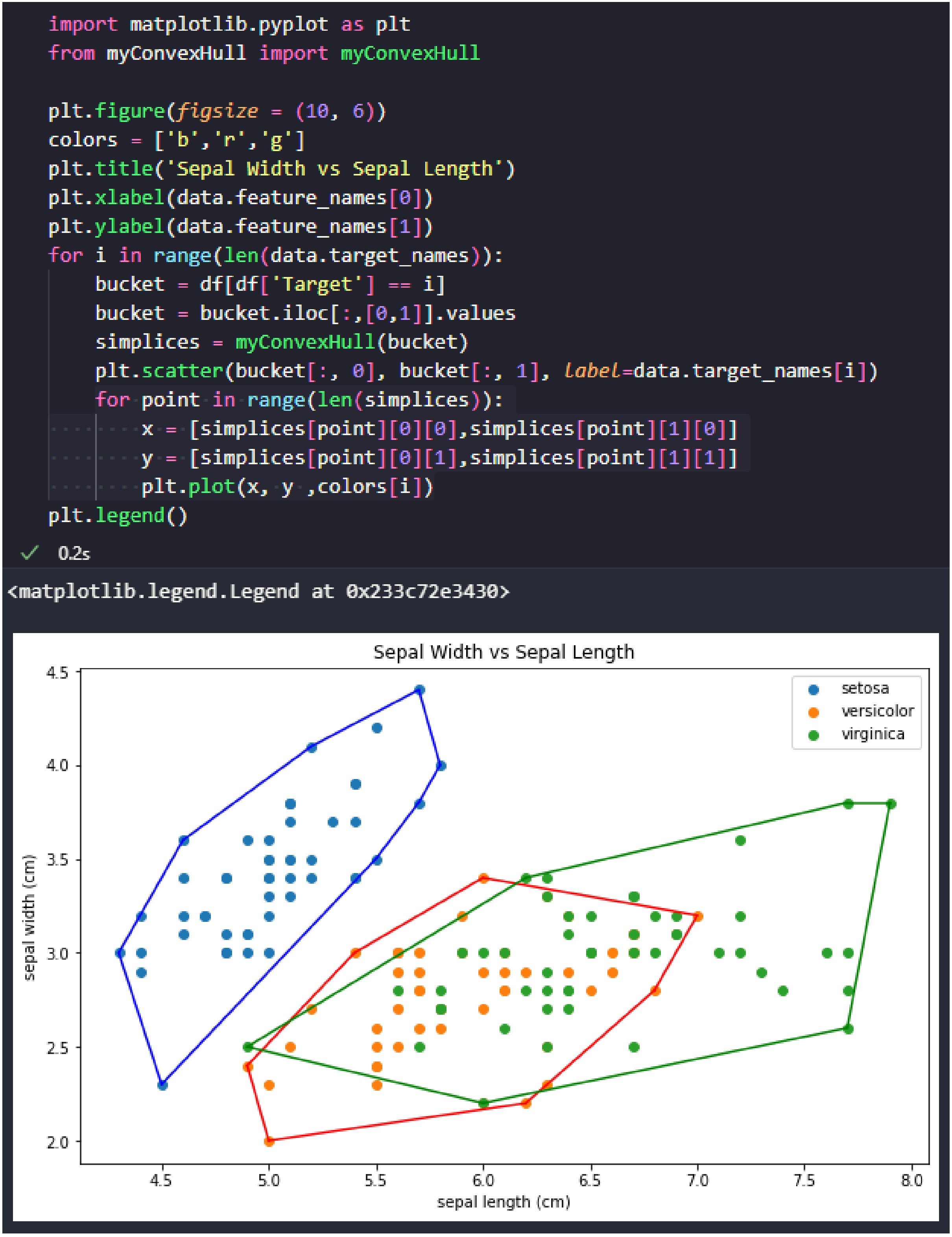
```
# %% [markdown]
# # Data Iris
#

# %%
import pandas as pd
import matplotlib.pyplot as plt

from sklearn import datasets
data = datasets.load_iris()

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

```
print(df.shape)
df.head()

# %% [markdown]
# ### Visualisasi Hasil _Convex Hull_

# %%
import matplotlib.pyplot as plt
from myConvexHull import myConvexHull

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
    simplices = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for point in range(len(simplices)):
        x = [simplices[point][0][0],simplices[point][1][0]]
        y = [simplices[point][0][1],simplices[point][1][1]]
        plt.plot(x, y ,colors[i])
plt.legend()

# %%
import matplotlib.pyplot as plt
from myConvexHull import myConvexHull

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
    simplices = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for point in range(len(simplices)):
        x = [simplices[point][0][0],simplices[point][1][0]]
        y = [simplices[point][0][1],simplices[point][1][1]]
        plt.plot(x, y ,colors[i])
plt.legend()

# %% [markdown]
# # Data Breast
```

```
#

# %%
import pandas as pd
import matplotlib.pyplot as plt

from sklearn import datasets
data = datasets.load_breast_cancer()

#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

# %% [markdown]
# ### Visualisasi Hasil _Convex Hull_

# %%
import matplotlib.pyplot as plt
from myConvexHull import myConvexHull

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Mean Radius vs Mean Texture')
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
    simplices = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for point in range(len(simplices)):
        x = [simplices[point][0][0],simplices[point][1][0]]
        y = [simplices[point][0][1],simplices[point][1][1]]
        plt.plot(x, y ,colors[i])
plt.legend()

# %%
import matplotlib.pyplot as plt
from myConvexHull import myConvexHull

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Mean Area vs Mean Compactness')
plt.xlabel(data.feature_names[3])
plt.ylabel(data.feature_names[5])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[3,5]].values
    simplices = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for point in range(len(simplices)):
        x = [simplices[point][0][0],simplices[point][1][0]]
        y = [simplices[point][0][1],simplices[point][1][1]]
        plt.plot(x, y ,colors[i])
plt.legend()

# %% [markdown]
# # Data Wine
#

# %%
import pandas as pd
import matplotlib.pyplot as plt

from sklearn import datasets
data = datasets.load_wine()

#create a DataFrame
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = pd.DataFrame(data.target)
print(df.shape)
df.head()

# %% [markdown]
# ### Visualisasi Hasil _Convex Hull_

# %%
import matplotlib.pyplot as plt
from myConvexHull import myConvexHull

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Alcohol vs Malic Acid')
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
    simplices = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for point in range(len(simplices)):
        x = [simplices[point][0][0],simplices[point][1][0]]
        y = [simplices[point][0][1],simplices[point][1][1]]
        plt.plot(x, y ,colors[i])
plt.legend()

# %%
import matplotlib.pyplot as plt
from myConvexHull import myConvexHull

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Ash vs Alcalinity of Ash')
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
    simplices = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for point in range(len(simplices)):
        x = [simplices[point][0][0],simplices[point][1][0]]
        y = [simplices[point][0][1],simplices[point][1][1]]
        plt.plot(x, y ,colors[i])
plt.legend()
```

SCREENSHOT INPUT OUTPUT

1. Data Iris
   a. Sepal Width – Sepal Length

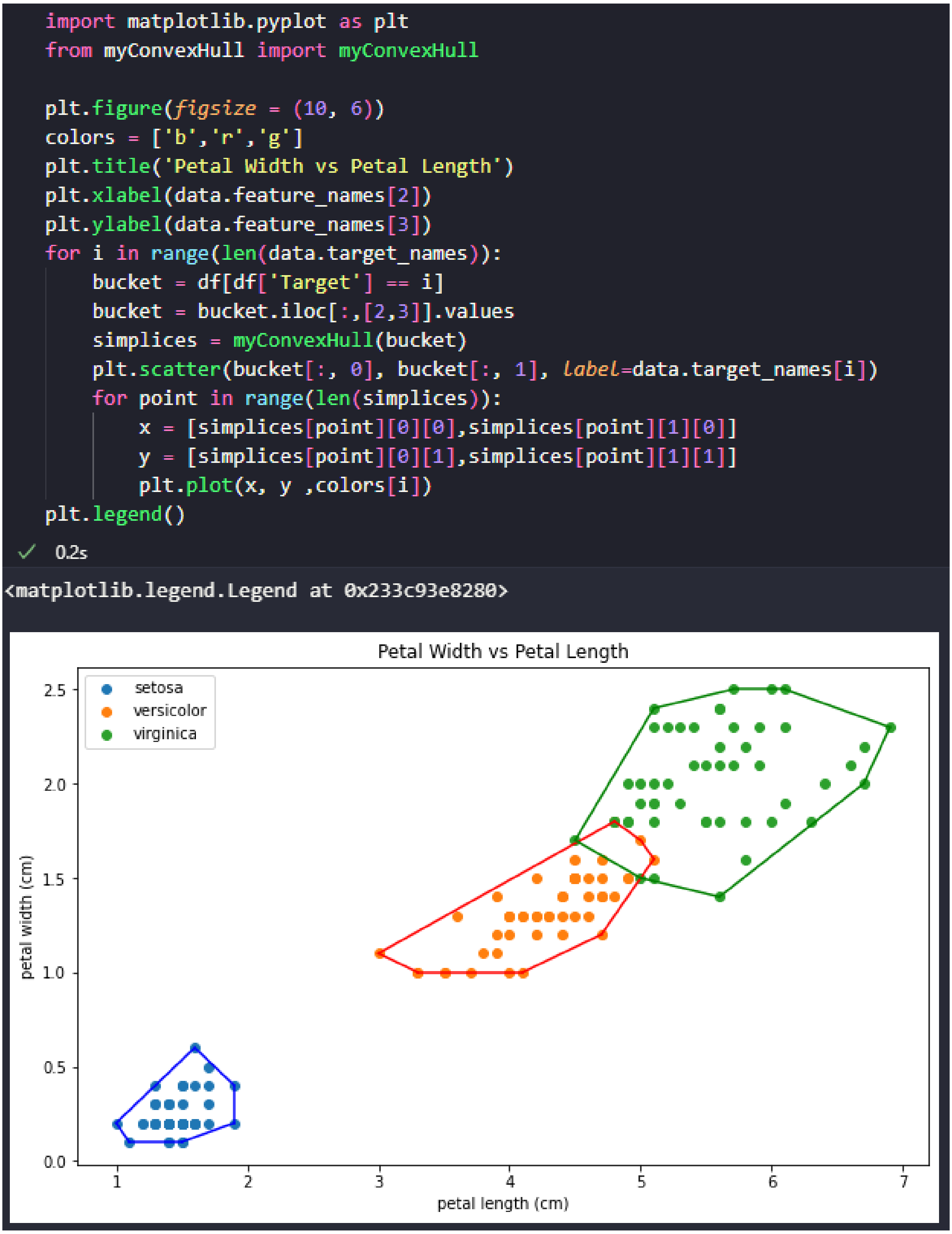


```python
import matplotlib.pyplot as plt
from myConvexHull import myConvexHull

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
    simplices = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for point in range(len(simplices)):
        x = [simplices[point][0][0],simplices[point][1][0]]
        y = [simplices[point][0][1],simplices[point][1][1]]
        plt.plot(x, y ,colors[i])
plt.legend()
```

✓ 0.2s

```
<matplotlib.legend.Legend at 0x233c72e3430>
```

   b. Petal Width – Petal Length



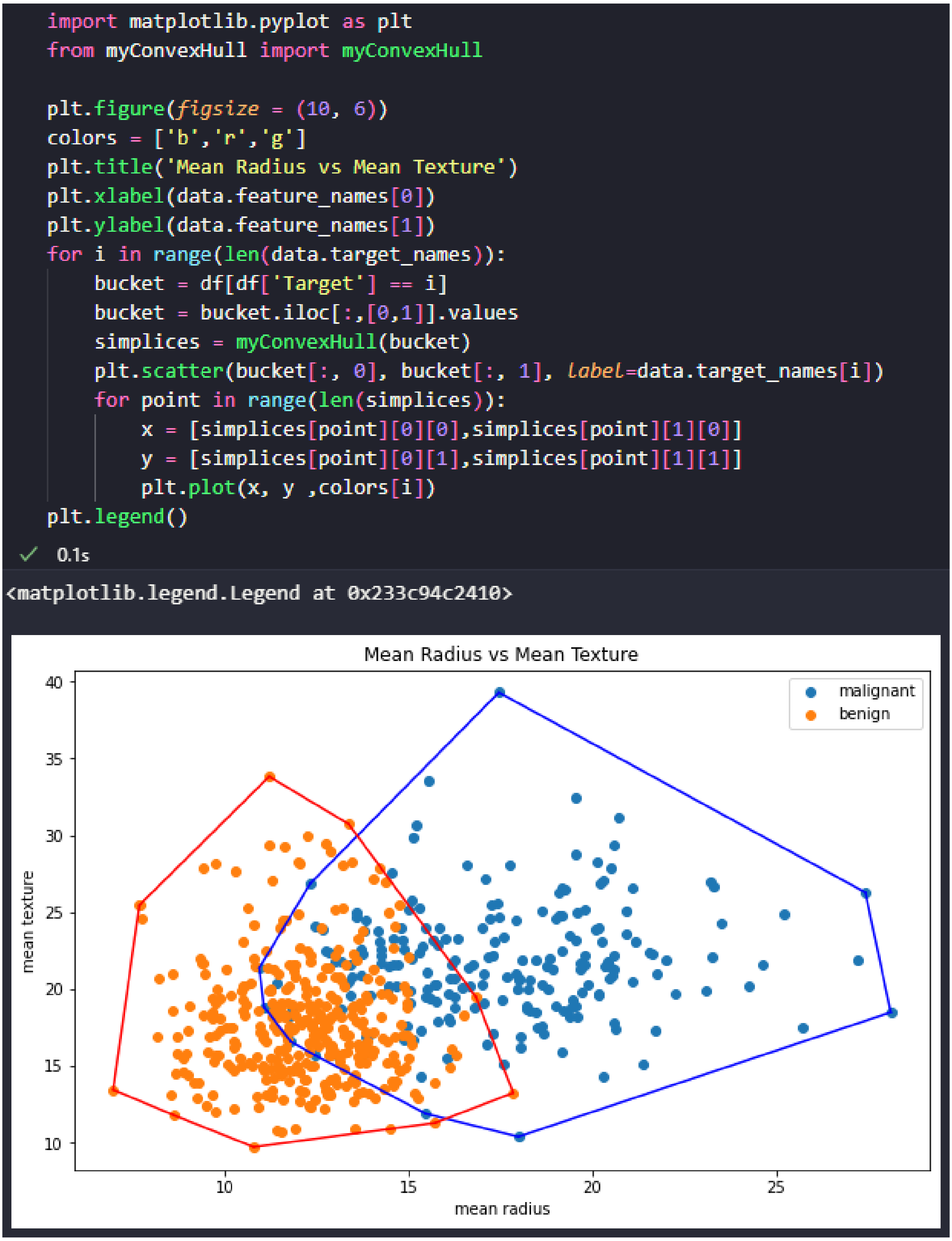
```
import matplotlib.pyplot as plt
from myConvexHull import myConvexHull

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
    simplices = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for point in range(len(simplices)):
        x = [simplices[point][0][0],simplices[point][1][0]]
        y = [simplices[point][0][1],simplices[point][1][1]]
        plt.plot(x, y ,colors[i])
plt.legend()
```

✓ 0.2s

```
<matplotlib.legend.Legend at 0x233c93e8280>
```

2. Data Breast

a. Mean Radius - Mean Texture

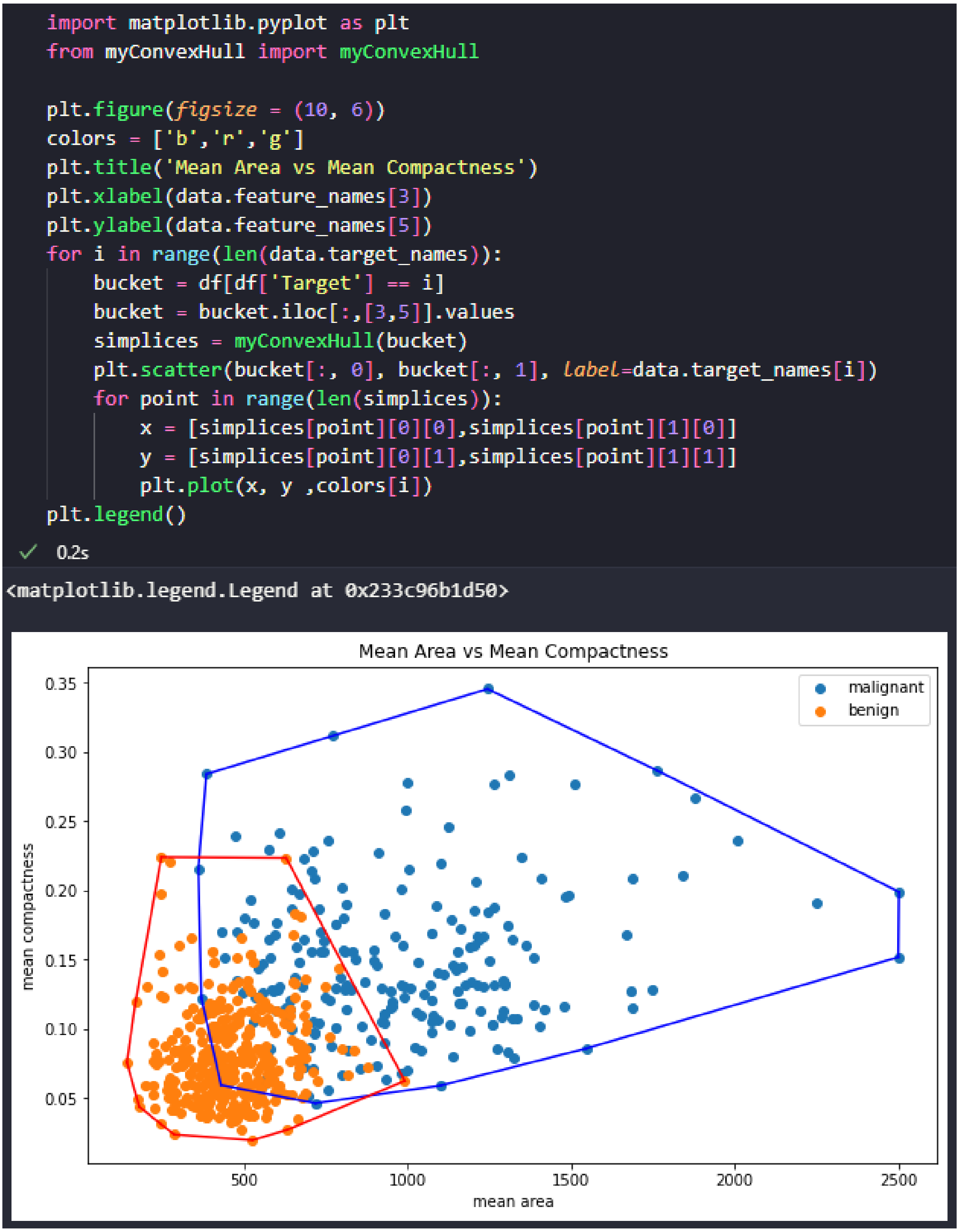


```
import matplotlib.pyplot as plt
from myConvexHull import myConvexHull

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Mean Radius vs Mean Texture')
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
    simplices = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for point in range(len(simplices)):
        x = [simplices[point][0][0],simplices[point][1][0]]
        y = [simplices[point][0][1],simplices[point][1][1]]
        plt.plot(x, y ,colors[i])
plt.legend()
```

```
✓ 0.1s
```

```
<matplotlib.legend.Legend at 0x233c94c2410>
```

b. Mean Area - Mean Compactness

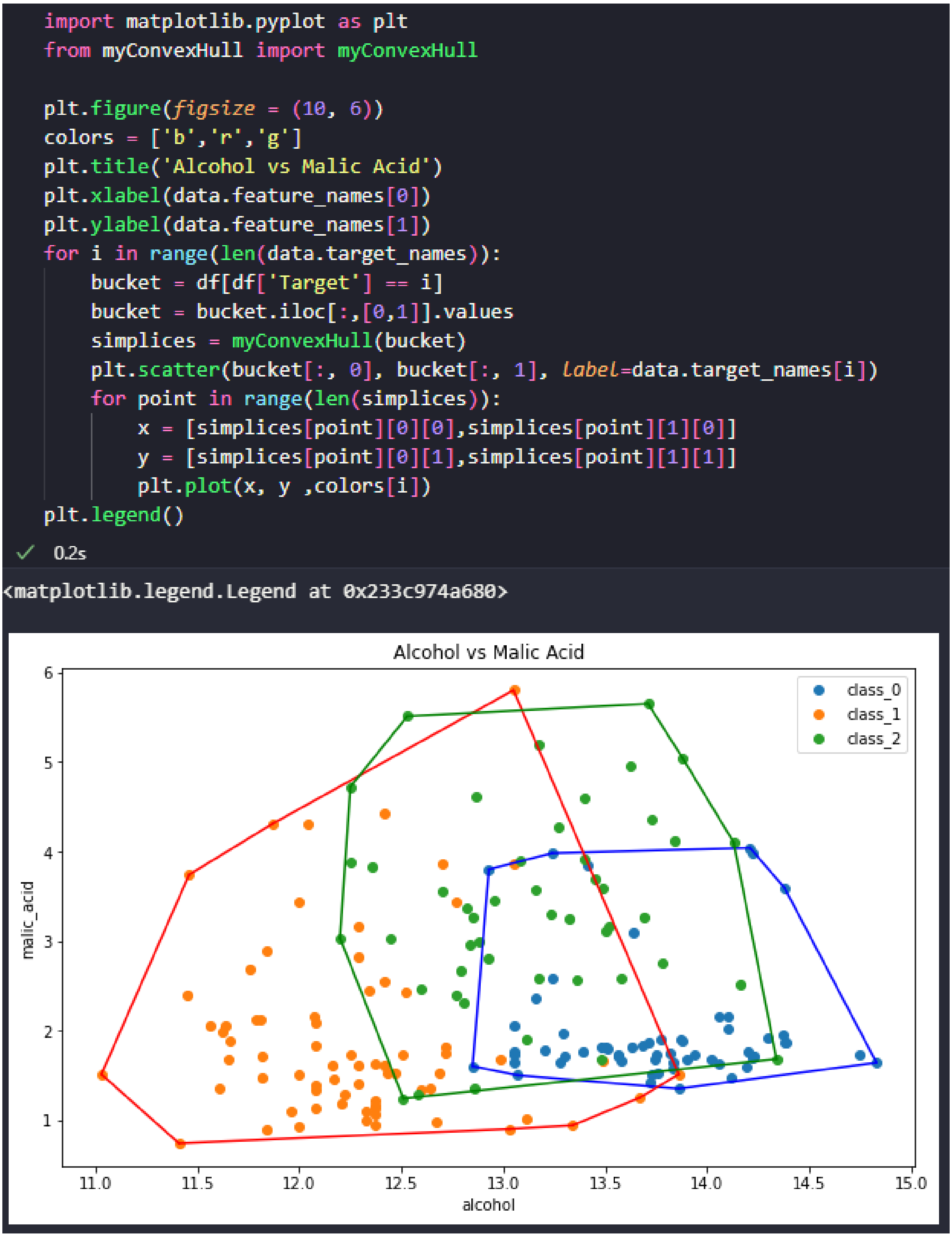


```python
import matplotlib.pyplot as plt
from myConvexHull import myConvexHull

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Mean Area vs Mean Compactness')
plt.xlabel(data.feature_names[3])
plt.ylabel(data.feature_names[5])
for i in range(len(data.target_names)):
    bucket = df[df['Target'] == i]
    bucket = bucket.iloc[:,[3,5]].values
    simplices = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for point in range(len(simplices)):
        x = [simplices[point][0][0],simplices[point][1][0]]
        y = [simplices[point][0][1],simplices[point][1][1]]
        plt.plot(x, y ,colors[i])
plt.legend()
```

✓ 0.2s

```
<matplotlib.legend.Legend at 0x233c96b1d50>
```

3. Data Wine
    a. Alcohol - Malic Acid

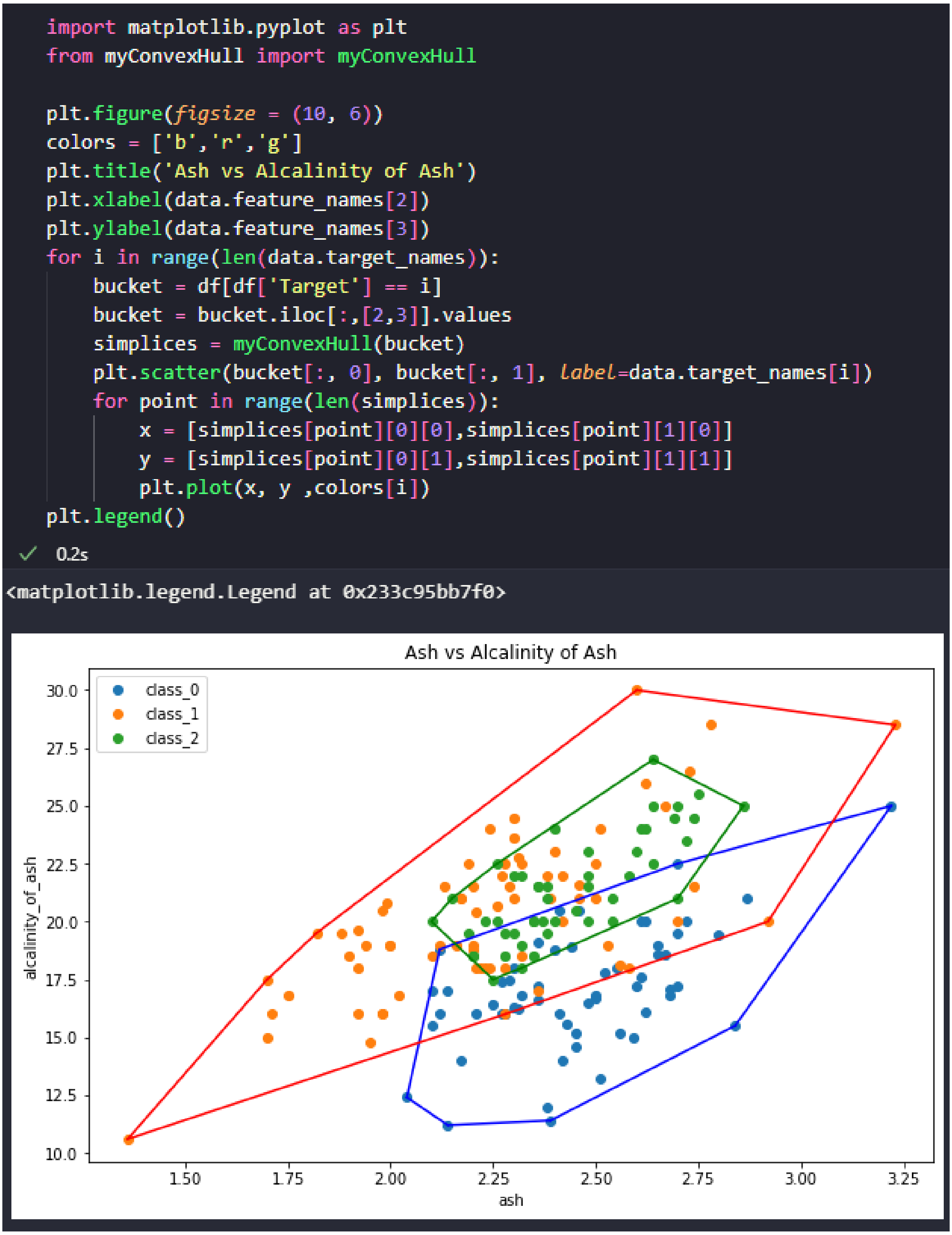


```python
import matplotlib.pyplot as plt
from myConvexHull import myConvexHull

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Alcohol vs Malic Acid')
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
    simplices = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for point in range(len(simplices)):
        x = [simplices[point][0][0],simplices[point][1][0]]
        y = [simplices[point][0][1],simplices[point][1][1]]
        plt.plot(x, y ,colors[i])
plt.legend()
```

✓ 0.2s

```
<matplotlib.legend.Legend at 0x233c974a680>
```

b. Ash - Alcalinity of Ash



```python
import matplotlib.pyplot as plt
from myConvexHull import myConvexHull

plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Ash vs Alcalinity of Ash')
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
    simplices = myConvexHull(bucket)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for point in range(len(simplices)):
        x = [simplices[point][0][0],simplices[point][1][0]]
        y = [simplices[point][0][1],simplices[point][1][1]]
        plt.plot(x, y ,colors[i])
plt.legend()
```

✓ 0.2s

```
<matplotlib.legend.Legend at 0x233c95bb7f0>
```

SOURCE CODE FILE

https://github.com/kentlius/Tucil2_13520069

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.	√	