LAPORAN

TUGAS KECIL 2

IF2121 STRATEGI ALGORITMA

Dosen: Dr. Masayu Leylia Khodra, S.T., M.T.

Oleh:

Maharani Ayu Putri Irawan / 13520019

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

BAB I
ALGORITMA DIVIDE AND CONQUER

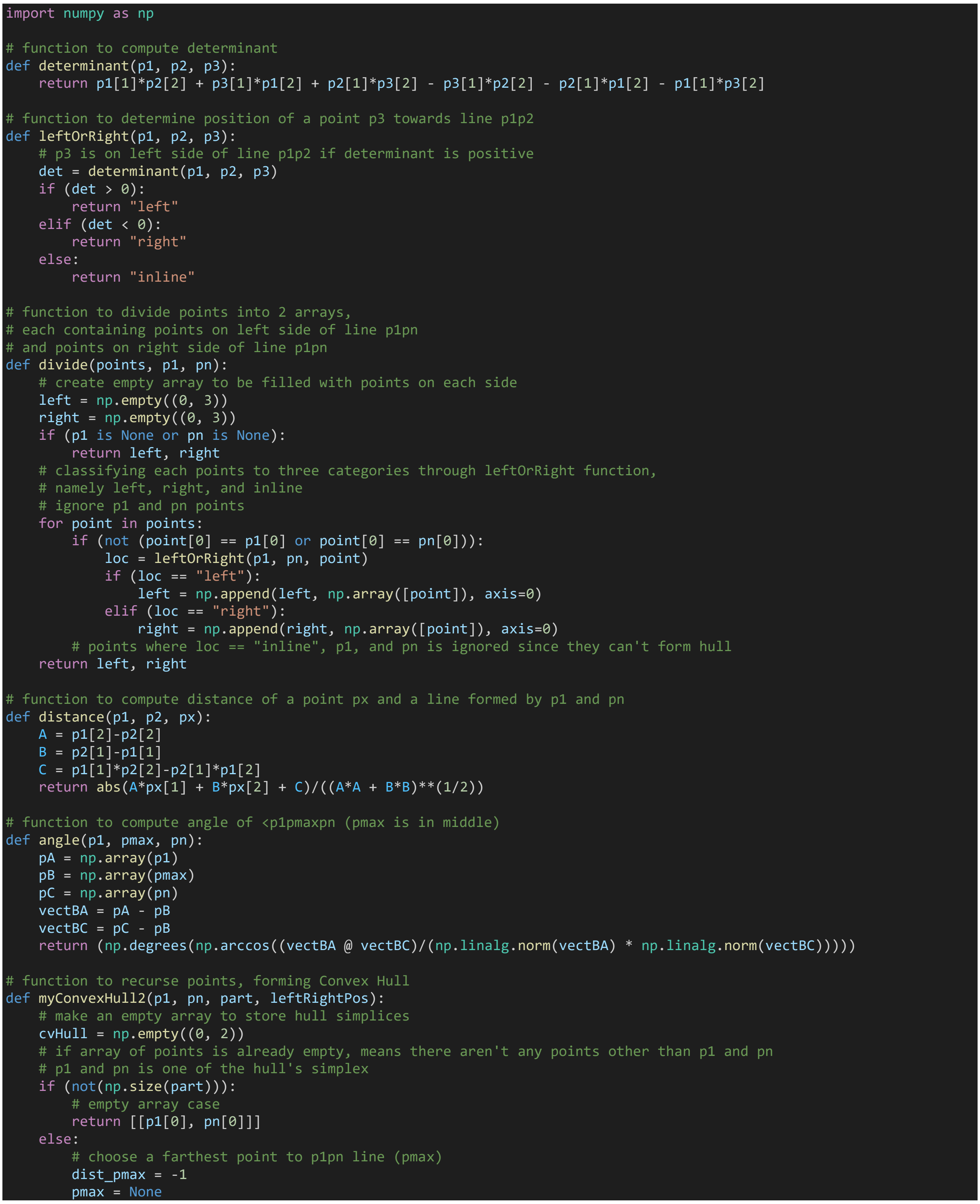
Pada tugas kecil ini, digunakan algoritma *divide and conquer* dengan deskripsi sebagai berikut:

1. Persiapan data
   a. Mulai dengan menomori seluruh titik yang hendak diproses.
   b. Lalu, urutkan seluruh titik yang diproses, menaik berdasarkan sumbu-X. Bila ada nilai pada sumbu-X yang sama, urutkan menaik berdasarkan sumbu-Y.
2. Pemrosesan
   a. Pilih 2 titik, masing-masing p1 dan pn, mewakili dua titik terujung, minimum dan maksimum, berdasarkan sumbu-X.
   b. Jika titik yang diproses kosong, kembalikan p1 dan pn sebagai pembentuk *convex hull*.
   c. Jika ada titik yang dapat diproses, dilakukan pengelompokan titik-titik ini berdasarkan area yang dipisahkan garis p1pn, kiri dan kanan.
   d. Memanggil fungsi rekursif untuk memproses masing-masing titik yang berada di kiri dan kanan
   e. Mengembalikan kumpulan pasangan titik yang membentuk garis-garis *convex hull*.
3. Rekursi
   a. Bila titik yang akan diproses sudah habis, mengembalikan p1 dan pn sebagai pembentuk *convex hull.*
   b. Mencari titik terjauh dari garis p1pn, diberi nama pmax. Bila terdapat dua titik yang sama jauhnya, dipilih yang membentuk sudut p1pmaxpn terbesar
   c. Bila pmax ditemukan, bagi area yang dipisahkan garis p1pmax dan pmaxpn menjadi kiri dan kanan masing-masing
   d. Titik yang berada pada area di dalam segitiga p1pmaxpn diabaikan sehingga untuk bagian kiri, hanya diambil kumpulan titik yang berada pada sisi luar, yakni kiri. Sebaliknya, untuk bagian kanan, diambil kumpulan titik yang berada pada sisi kanan.
   e. Ulangi rekursi untuk kumpulan titik yang terpilih untuk masing-masing sisi, hingga kumpulan titik kiri dan kanan habis.
   f. Mengembalikan kumpulan pasangan titik yang membentuk garis-garis *convex hull*.
4. Keluaran
   a. Hasil pemrosesan pustaka berupa 2D numpy array yang merupakan *simplices*.

BAB II

KODE SUMBER

Kode ditulis dalam Bahasa Python. Berikut merupakan kode sumber yang terdapat di dalam file myConvexHull.py. Kode sumber juga dapat diakses melalui Github https://github.com/rannnayy/stima-convexhull atau Google Drive pengumpulan.

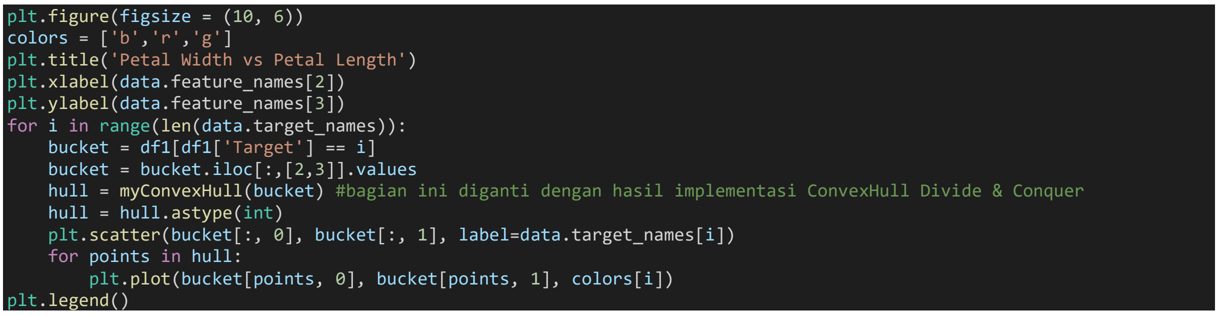


```python
import numpy as np

# function to compute determinant
def determinant(p1, p2, p3):
    return p1[1]*p2[2] + p3[1]*p1[2] + p2[1]*p3[2] - p3[1]*p2[2] - p2[1]*p1[2] - p1[1]*p3[2]

# function to determine position of a point p3 towards line p1p2
def leftOrRight(p1, p2, p3):
    # p3 is on left side of line p1p2 if determinant is positive
    det = determinant(p1, p2, p3)
    if (det > 0):
        return "left"
    elif (det < 0):
        return "right"
    else:
        return "inline"

# function to divide points into 2 arrays,
# each containing points on left side of line p1pn
# and points on right side of line p1pn
def divide(points, p1, pn):
    # create empty array to be filled with points on each side
    left = np.empty((0, 3))
    right = np.empty((0, 3))
    if (p1 is None or pn is None):
        return left, right
    # classifying each points to three categories through leftOrRight function,
    # namely left, right, and inline
    # ignore p1 and pn points
    for point in points:
        if (not (point[0] == p1[0] or point[0] == pn[0])):
            loc = leftOrRight(p1, pn, point)
            if (loc == "left"):
                left = np.append(left, np.array([point]), axis=0)
            elif (loc == "right"):
                right = np.append(right, np.array([point]), axis=0)
        # points where loc == "inline", p1, and pn is ignored since they can't form hull
    return left, right

# function to compute distance of a point px and a line formed by p1 and pn
def distance(p1, p2, px):
    A = p1[2]-p2[2]
    B = p2[1]-p1[1]
    C = p1[1]*p2[2]-p2[1]*p1[2]
    return abs(A*px[1] + B*px[2] + C)/((A*A + B*B)**(1/2))

# function to compute angle of <p1pmaxpn (pmax is in middle)
def angle(p1, pmax, pn):
    pA = np.array(p1)
    pB = np.array(pmax)
    pC = np.array(pn)
    vectBA = pA - pB
    vectBC = pC - pB
    return (np.degrees(np.arccos((vectBA @ vectBC)/(np.linalg.norm(vectBA) * np.linalg.norm(vectBC)))))

# function to recurse points, forming Convex Hull
def myConvexHull2(p1, pn, part, leftRightPos):
    # make an empty array to store hull simplices
    cvHull = np.empty((0, 2))
    # if array of points is already empty, means there aren't any points other than p1 and pn
    # p1 and pn is one of the hull's simplex
    if (not(np.size(part))):
        # empty array case
        return [[p1[0], pn[0]]]
    else:
        # choose a farthest point to p1pn line (pmax)
        dist_pmax = -1
        pmax = None
        idx_pmax = 0
        ctr = 0
        for point in part:
            temp_dist = distance(p1, pn, point)
            # choose farthest by distance
            if (temp_dist > dist_pmax):
                dist_pmax = temp_dist
                pmax = point
                idx_pmax = ctr
            # if there are two/more points with same distance, choose by maximum angle gotten
            elif (temp_dist == dist_pmax and not(pmax is None)):
                if (angle(p1, point, pn) > angle(p1, pmax, pn)):
                    dist_pmax = temp_dist
                    pmax = point
                    idx_pmax = ctr
            ctr += 1
        if (not(pmax is None)):
            # maximum is found
            part = np.delete(part, idx_pmax, axis=0)
            # divide to two parts, only take the outer points
            p1pmaxleft, p1pmaxright = divide(part, p1, pmax)
            pmaxpnleft, pmaxpnright = divide(part, pmax, pn)
            # for points on left side of p1pn, take only left parts
            if (leftRightPos == "left"):
                cvHull = np.append(cvHull, np.array(myConvexHull2(p1, pmax, p1pmaxleft, "left")), axis=0)
                cvHull = np.append(cvHull, np.array(myConvexHull2(pmax, pn, pmaxpnleft, "left")), axis=0)
            # for points on right side of p1pn, take only right parts
            elif (leftRightPos == "right"):
                cvHull = np.append(cvHull, np.array(myConvexHull2(p1, pmax, p1pmaxright, "right")), axis=0)
                cvHull = np.append(cvHull, np.array(myConvexHull2(pmax, pn, pmaxpnright, "right")), axis=0)
        return cvHull

# function to label each points by an identifier number
def numTitik(points):
    # create new array to store numbered points
    tempPoints = np.empty((0, 3))
    # iterate for each point, add an identifier number
    for i in range(len(points)):
        tempPoints = np.append(tempPoints, np.array([[i, points[i][0], points[i][1]]]), axis=0)
    return tempPoints

# function to compute convex hull, helped by myConvexHull2
def myConvexHull(points):
    # first, number all points
    points = numTitik(points)
    # make an empty array for storing simplices computed
    cvHull = np.empty((0, 2))
    # sort points
    points = points[np.lexsort((points[:,2], points[:,1]))]
    # p1 and pn, leftmost and rightmost points respectively
    p1 = points[0]
    pn = points[-1]

    # if there's no other points other than p1 and pn, p1 and pn forms the convex hull
    if (not(np.size(points))):
        # empty case
        return [[p1[0], pn[0]]]
    else:
        # divide points to two parts separated by line p1pn
        left, right = divide(points, p1, pn)
        # call recursive functions
        cvHull = np.append(cvHull, np.array(myConvexHull2(p1, pn, left, "left")), axis=0)
        cvHull = np.append(cvHull, np.array(myConvexHull2(p1, pn, right, "right")), axis=0)
        return cvHull
```

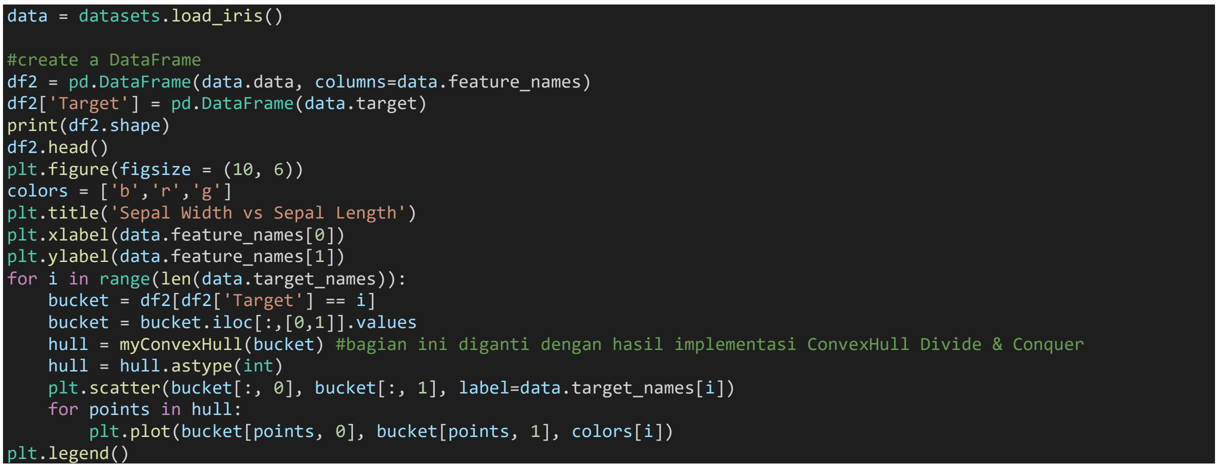
Kode berikut merupakan kode untuk memvisualisasikan hasil penggunakan Pustaka yang telah dibuat. Kode berikut terdapat di dalam file myConvexHull.ipynb.

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

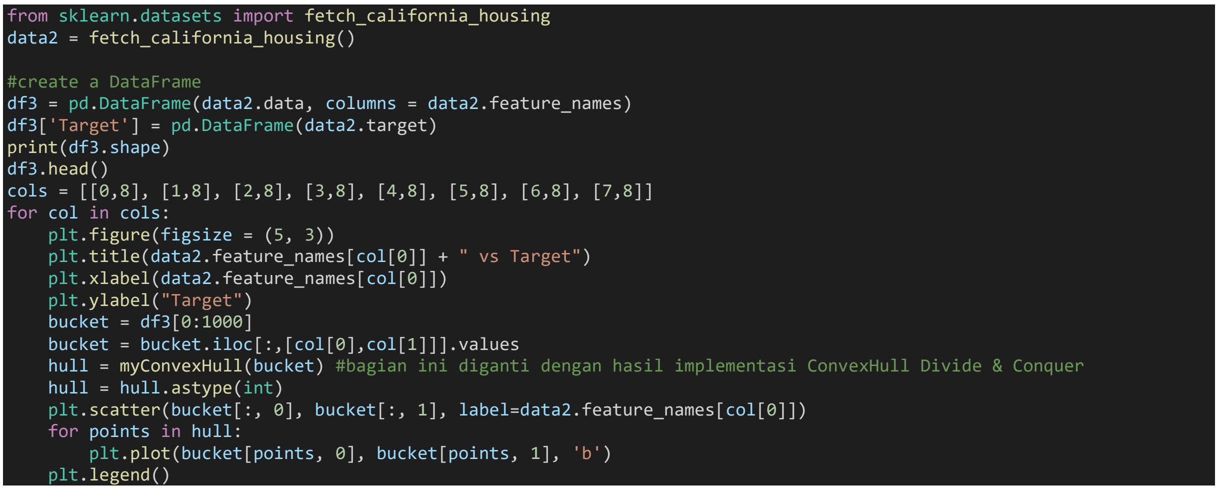
```
data = datasets.load_iris()

#create a DataFrame
df1 = pd.DataFrame(data.data, columns=data.feature_names)
df1['Target'] = pd.DataFrame(data.target)
print(df1.shape)
df1.head()
```



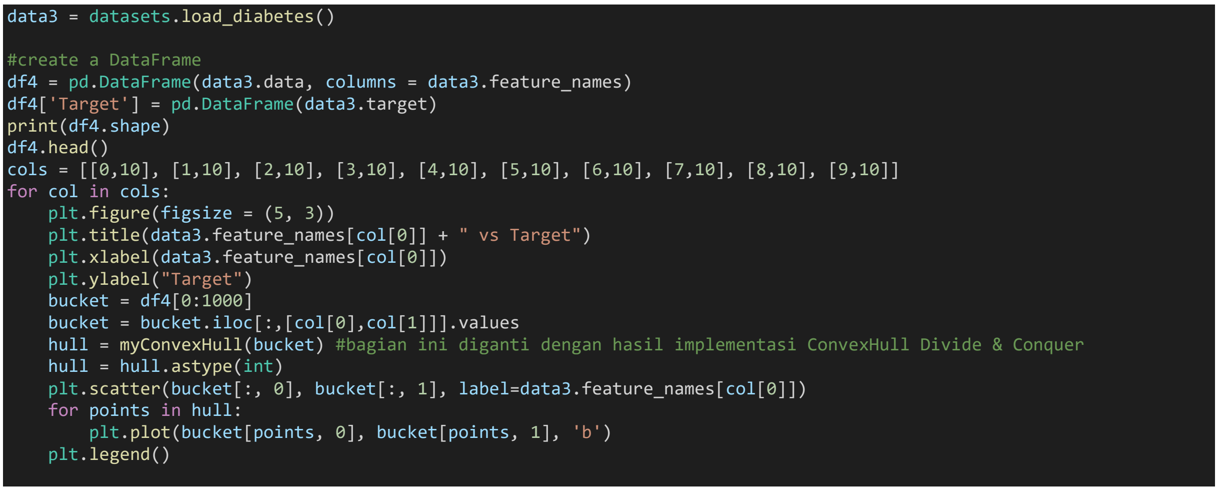
```
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal Length')
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])
for i in range(len(data.target_names)):
    bucket = df1[df1['Target'] == i]
    bucket = bucket.iloc[:,[2,3]].values
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull = hull.astype(int)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for points in hull:
        plt.plot(bucket[points, 0], bucket[points, 1], colors[i])
plt.legend()
```



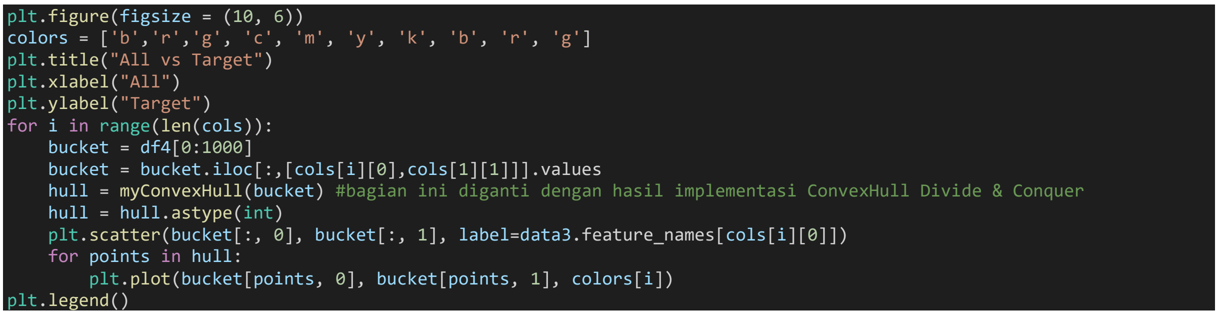
```
data = datasets.load_iris()

#create a DataFrame
df2 = pd.DataFrame(data.data, columns=data.feature_names)
df2['Target'] = pd.DataFrame(data.target)
print(df2.shape)
df2.head()
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 = df2[df2['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull = hull.astype(int)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for points in hull:
        plt.plot(bucket[points, 0], bucket[points, 1], colors[i])
plt.legend()
```



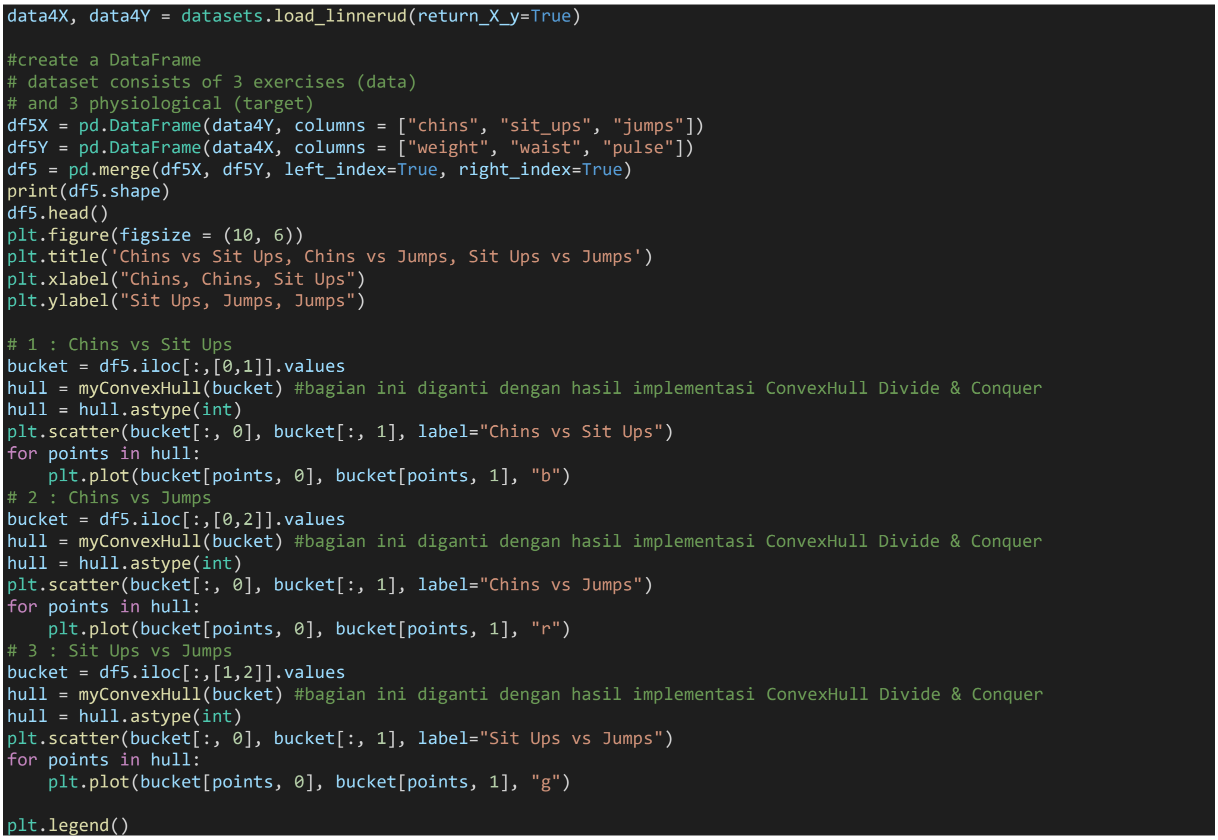
```
from sklearn.datasets import fetch_california_housing
data2 = fetch_california_housing()

#create a DataFrame
df3 = pd.DataFrame(data2.data, columns = data2.feature_names)
df3['Target'] = pd.DataFrame(data2.target)
print(df3.shape)
df3.head()
cols = [[0,8], [1,8], [2,8], [3,8], [4,8], [5,8], [6,8], [7,8]]
for col in cols:
    plt.figure(figsize = (5, 3))
    plt.title(data2.feature_names[col[0]] + " vs Target")
    plt.xlabel(data2.feature_names[col[0]])
    plt.ylabel("Target")
    bucket = df3[0:1000]
    bucket = bucket.iloc[:,[col[0],col[1]]].values
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull = hull.astype(int)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data2.feature_names[col[0]])
    for points in hull:
        plt.plot(bucket[points, 0], bucket[points, 1], 'b')
    plt.legend()
```



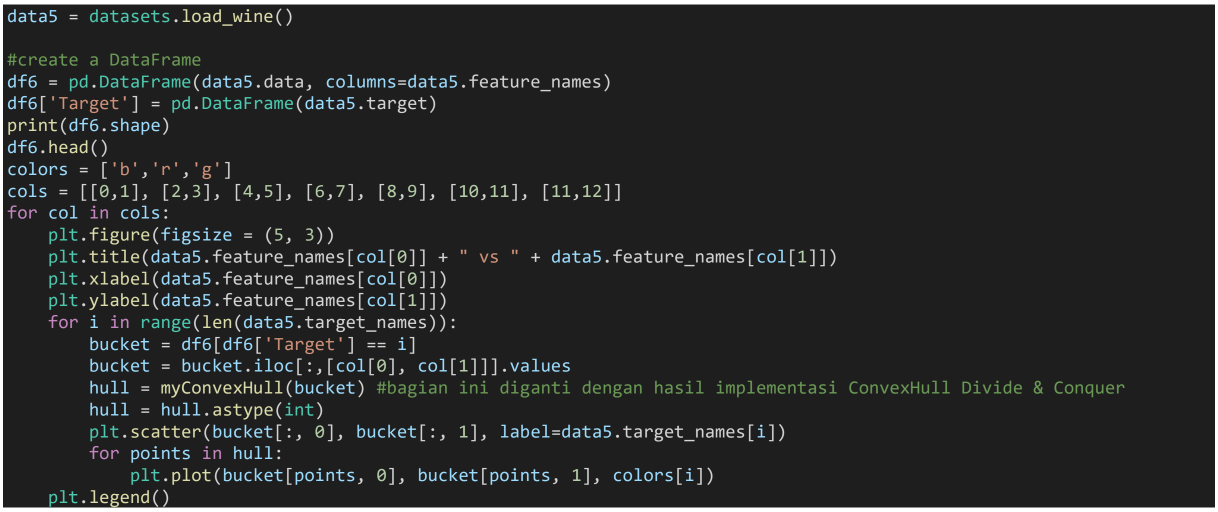
```
data3 = datasets.load_diabetes()

#create a DataFrame
df4 = pd.DataFrame(data3.data, columns = data3.feature_names)
df4['Target'] = pd.DataFrame(data3.target)
print(df4.shape)
df4.head()
cols = [[0,10], [1,10], [2,10], [3,10], [4,10], [5,10], [6,10], [7,10], [8,10], [9,10]]
for col in cols:
    plt.figure(figsize = (5, 3))
    plt.title(data3.feature_names[col[0]] + " vs Target")
    plt.xlabel(data3.feature_names[col[0]])
    plt.ylabel("Target")
    bucket = df4[0:1000]
    bucket = bucket.iloc[:,[col[0],col[1]]].values
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull = hull.astype(int)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data3.feature_names[col[0]])
    for points in hull:
        plt.plot(bucket[points, 0], bucket[points, 1], 'b')
    plt.legend()
```



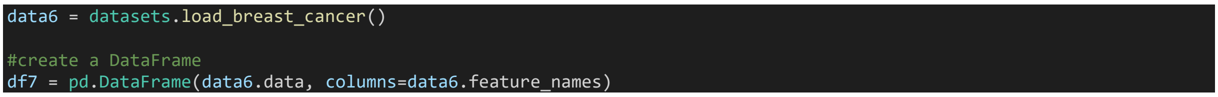
```
plt.figure(figsize = (10, 6))
colors = ['b','r','g', 'c', 'm', 'y', 'k', 'b', 'r', 'g']
plt.title("All vs Target")
plt.xlabel("All")
plt.ylabel("Target")
for i in range(len(cols)):
    bucket = df4[0:1000]
    bucket = bucket.iloc[:,[cols[i][0],cols[1][1]]].values
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull = hull.astype(int)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data3.feature_names[cols[i][0]])
    for points in hull:
        plt.plot(bucket[points, 0], bucket[points, 1], colors[i])
plt.legend()
```



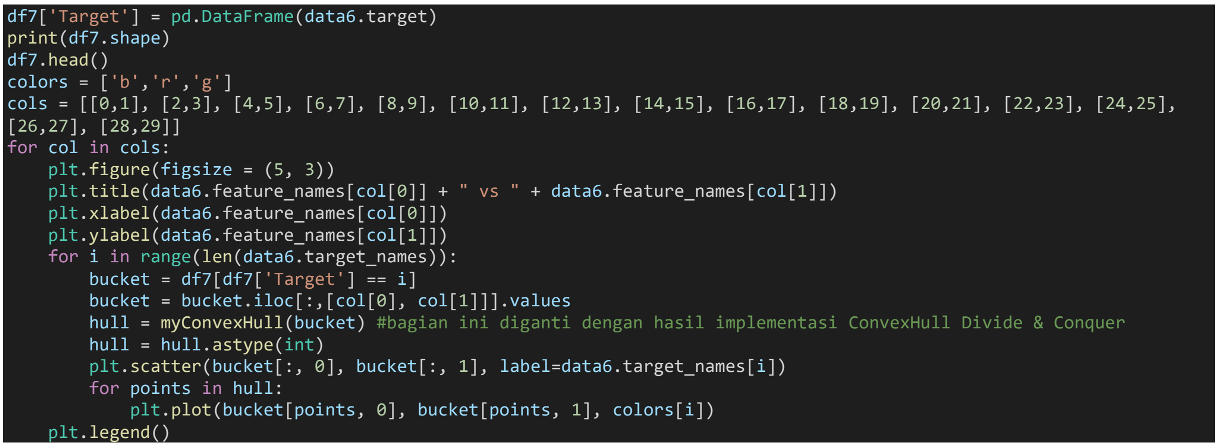
```
data4X, data4Y = datasets.load_linnerud(return_X_y=True)

#create a DataFrame
# dataset consists of 3 exercises (data)
# and 3 physiological (target)
df5X = pd.DataFrame(data4Y, columns = ["chins", "sit_ups", "jumps"])
df5Y = pd.DataFrame(data4X, columns = ["weight", "waist", "pulse"])
df5 = pd.merge(df5X, df5Y, left_index=True, right_index=True)
print(df5.shape)
df5.head()
plt.figure(figsize = (10, 6))
plt.title('Chins vs Sit Ups, Chins vs Jumps, Sit Ups vs Jumps')
plt.xlabel("Chins, Chins, Sit Ups")
plt.ylabel("Sit Ups, Jumps, Jumps")

# 1 : Chins vs Sit Ups
bucket = df5.iloc[:,[0,1]].values
hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
hull = hull.astype(int)
plt.scatter(bucket[:, 0], bucket[:, 1], label="Chins vs Sit Ups")
for points in hull:
    plt.plot(bucket[points, 0], bucket[points, 1], "b")
# 2 : Chins vs Jumps
bucket = df5.iloc[:,[0,2]].values
hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
hull = hull.astype(int)
plt.scatter(bucket[:, 0], bucket[:, 1], label="Chins vs Jumps")
for points in hull:
    plt.plot(bucket[points, 0], bucket[points, 1], "r")
# 3 : Sit Ups vs Jumps
bucket = df5.iloc[:,[1,2]].values
hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
hull = hull.astype(int)
plt.scatter(bucket[:, 0], bucket[:, 1], label="Sit Ups vs Jumps")
for points in hull:
    plt.plot(bucket[points, 0], bucket[points, 1], "g")

plt.legend()
```



```
data5 = datasets.load_wine()

#create a DataFrame
df6 = pd.DataFrame(data5.data, columns=data5.feature_names)
df6['Target'] = pd.DataFrame(data5.target)
print(df6.shape)
df6.head()
colors = ['b','r','g']
cols = [[0,1], [2,3], [4,5], [6,7], [8,9], [10,11], [11,12]]
for col in cols:
    plt.figure(figsize = (5, 3))
    plt.title(data5.feature_names[col[0]] + " vs " + data5.feature_names[col[1]])
    plt.xlabel(data5.feature_names[col[0]])
    plt.ylabel(data5.feature_names[col[1]])
    for i in range(len(data5.target_names)):
        bucket = df6[df6['Target'] == i]
        bucket = bucket.iloc[:,[col[0], col[1]]].values
        hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
        hull = hull.astype(int)
        plt.scatter(bucket[:, 0], bucket[:, 1], label=data5.target_names[i])
        for points in hull:
            plt.plot(bucket[points, 0], bucket[points, 1], colors[i])
    plt.legend()
```



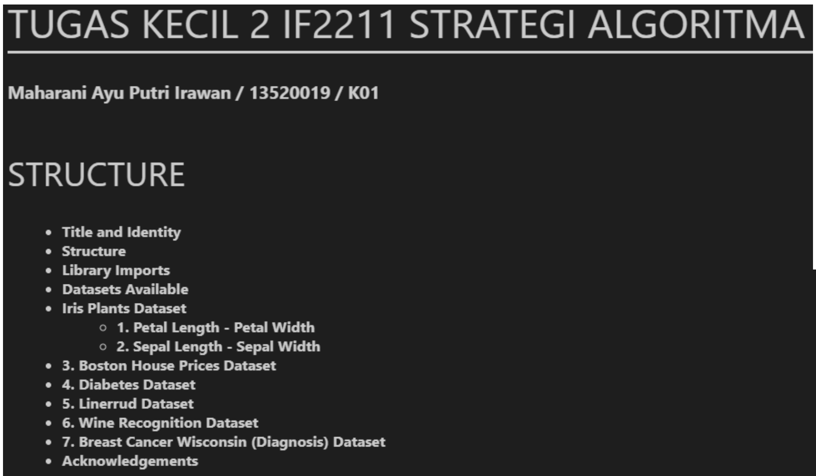
```
data6 = datasets.load_breast_cancer()

#create a DataFrame
df7 = pd.DataFrame(data6.data, columns=data6.feature_names)
```



```
df7['Target'] = pd.DataFrame(data6.target)
print(df7.shape)
df7.head()
colors = ['b','r','g']
cols = [[0,1], [2,3], [4,5], [6,7], [8,9], [10,11], [12,13], [14,15], [16,17], [18,19], [20,21], [22,23], [24,25],
[26,27], [28,29]]
for col in cols:
    plt.figure(figsize = (5, 3))
    plt.title(data6.feature_names[col[0]] + " vs " + data6.feature_names[col[1]])
    plt.xlabel(data6.feature_names[col[0]])
    plt.ylabel(data6.feature_names[col[1]])
    for i in range(len(data6.target_names)):
        bucket = df7[df7['Target'] == i]
        bucket = bucket.iloc[:,[col[0], col[1]]].values
        hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
        hull = hull.astype(int)
        plt.scatter(bucket[:, 0], bucket[:, 1], label=data6.target_names[i])
        for points in hull:
            plt.plot(bucket[points, 0], bucket[points, 1], colors[i])
    plt.legend()
```

BAB III

EKSPERIMEN

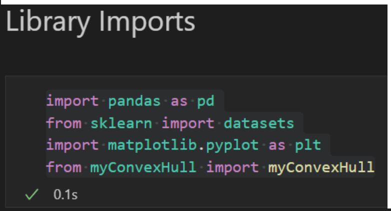


TUGAS KECIL 2 IF2211 STRATEGI ALGORITMA

Maharani Ayu Putri Irawan / 13520019 / K01

STRUCTURE

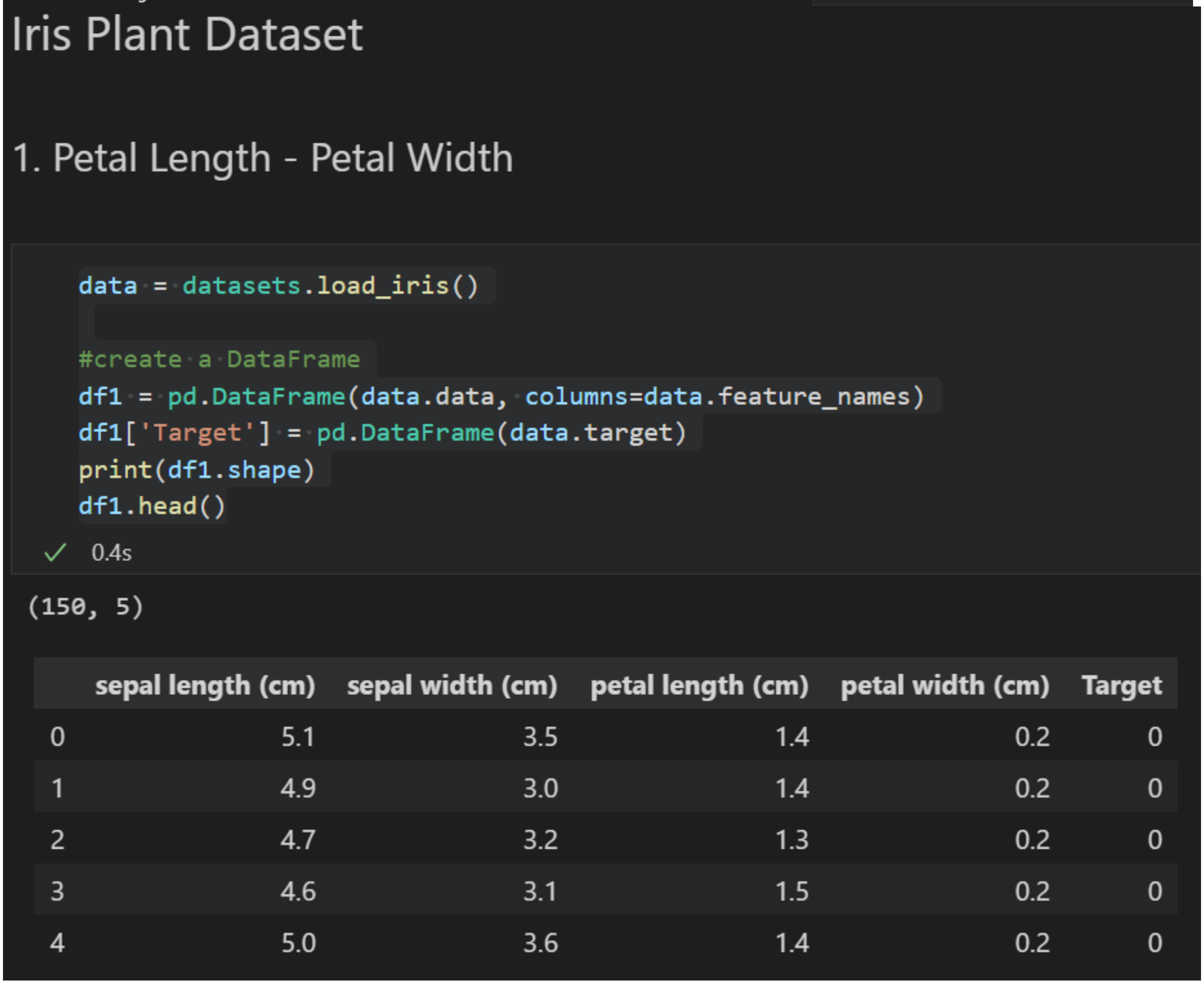
- Title and Identity
- Structure
- Library Imports
- Datasets Available
- Iris Plants Dataset
    - 1. Petal Length - Petal Width
    - 2. Sepal Length - Sepal Width
- 3. Boston House Prices Dataset
- 4. Diabetes Dataset
- 5. Linerrud Dataset
- 6. Wine Recognition Dataset
- 7. Breast Cancer Wisconsin (Diagnosis) Dataset
- Acknowledgements



Library Imports

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

✓ 0.1s



Iris Plant Dataset

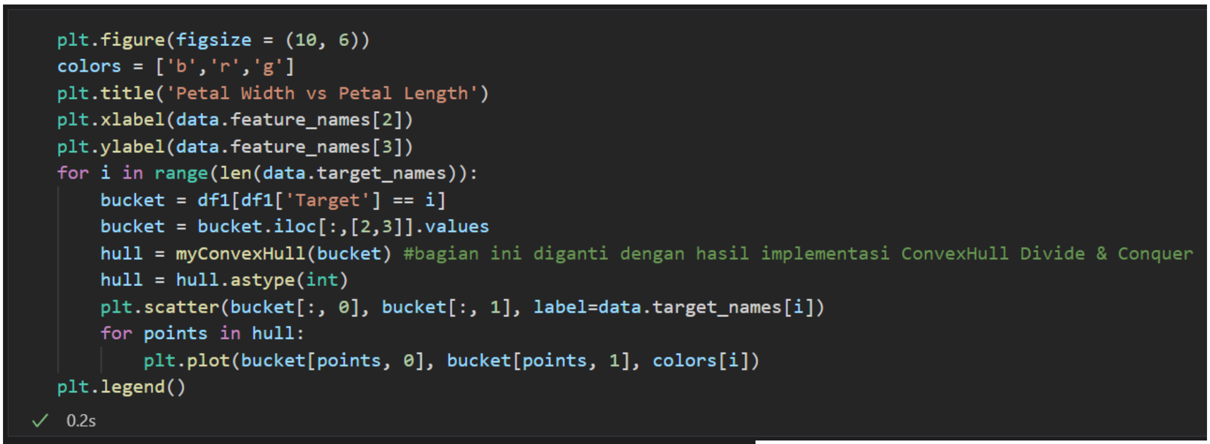
1. Petal Length - Petal Width

```
data = datasets.load_iris()

#create a DataFrame
df1 = pd.DataFrame(data.data, columns=data.feature_names)
df1['Target'] = pd.DataFrame(data.target)
print(df1.shape)
df1.head()
```

✓ 0.4s

```
(150, 5)
```

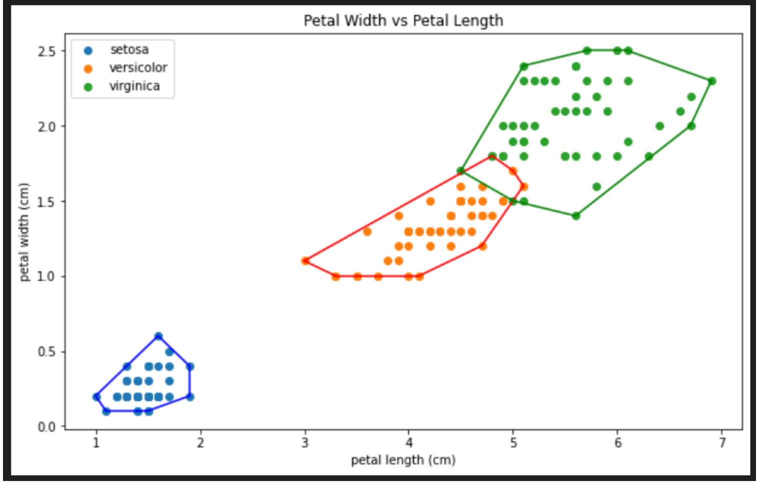
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

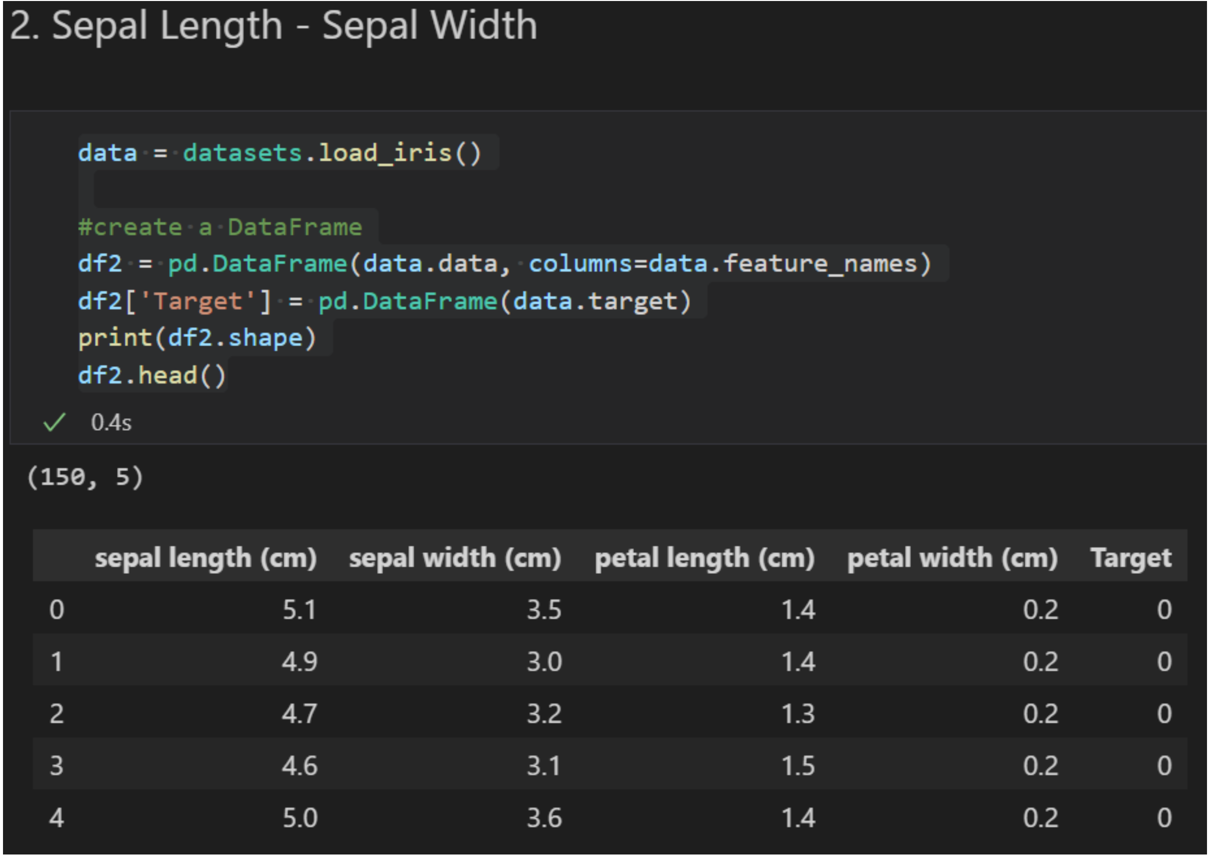


```
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Petal Width vs Petal Length')
plt.xlabel(data.feature_names[2])
plt.ylabel(data.feature_names[3])
for i in range(len(data.target_names)):
    bucket = df1[df1['Target'] == i]
    bucket = bucket.iloc[:,[2,3]].values
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull = hull.astype(int)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for points in hull:
        plt.plot(bucket[points, 0], bucket[points, 1], colors[i])
plt.legend()
```

✓ 0.2s

```
<matplotlib.legend.Legend at 0x1f72e1f5960>
```





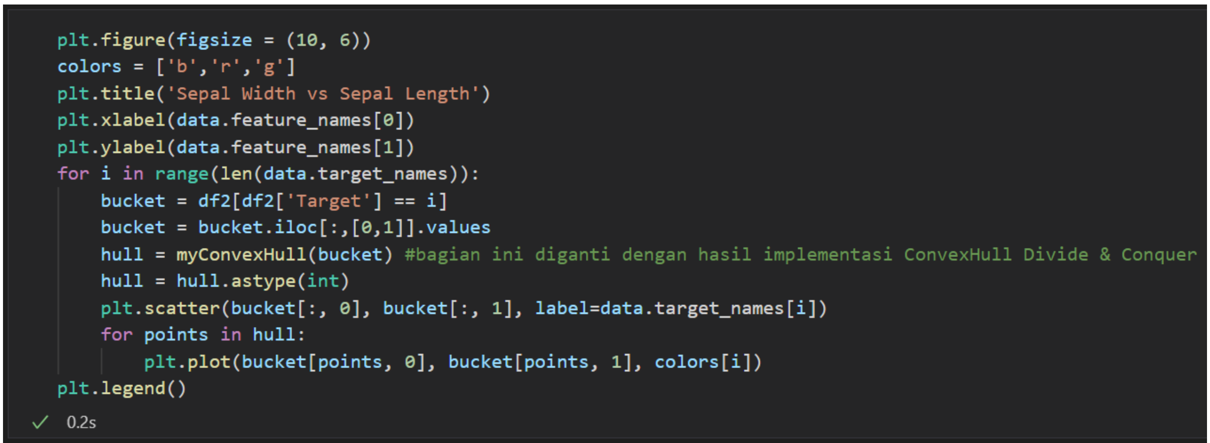
2. Sepal Length - Sepal Width

```
data = datasets.load_iris()

#create a DataFrame
df2 = pd.DataFrame(data.data, columns=data.feature_names)
df2['Target'] = pd.DataFrame(data.target)
print(df2.shape)
df2.head()
```

✓ 0.4s

```
(150, 5)
```

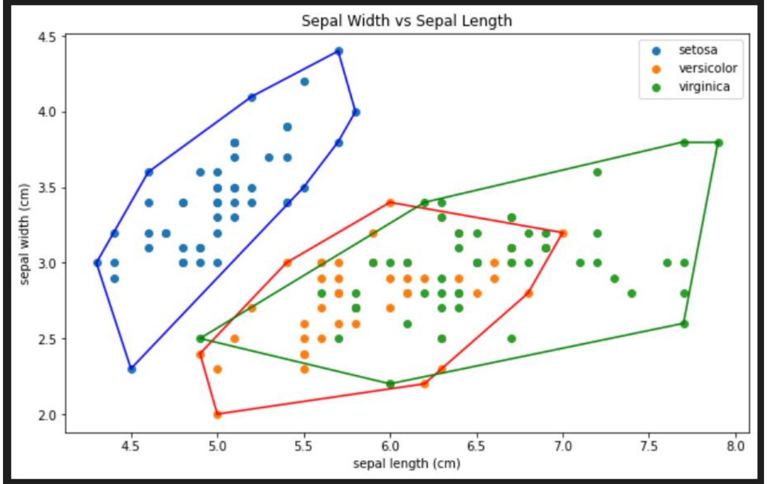
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

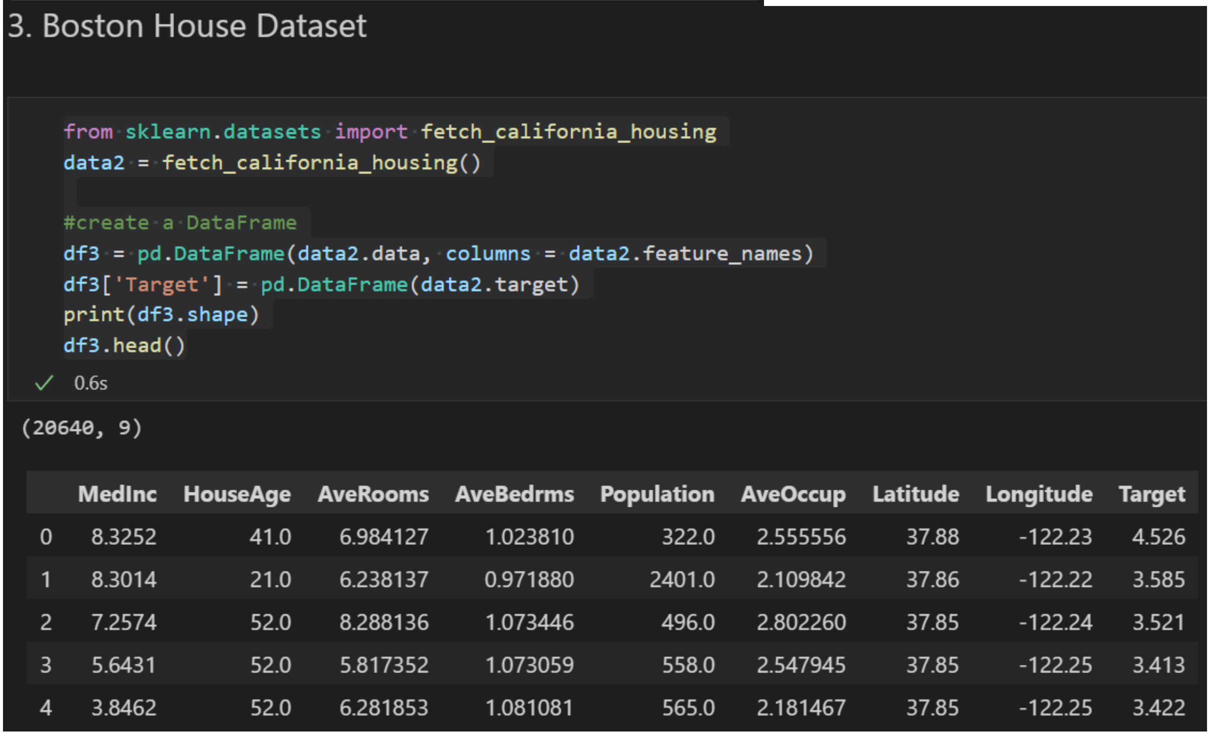


```
plt.figure(figsize = (10, 6))
colors = ['b','r','g']
plt.title('Sepal Width vs Sepal Length')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])
for i in range(len(data.target_names)):
    bucket = df2[df2['Target'] == i]
    bucket = bucket.iloc[:,[0,1]].values
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull = hull.astype(int)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target_names[i])
    for points in hull:
        plt.plot(bucket[points, 0], bucket[points, 1], colors[i])
plt.legend()
```

✓ 0.2s

```
<matplotlib.legend.Legend at 0x1f72e3fafe0>
```





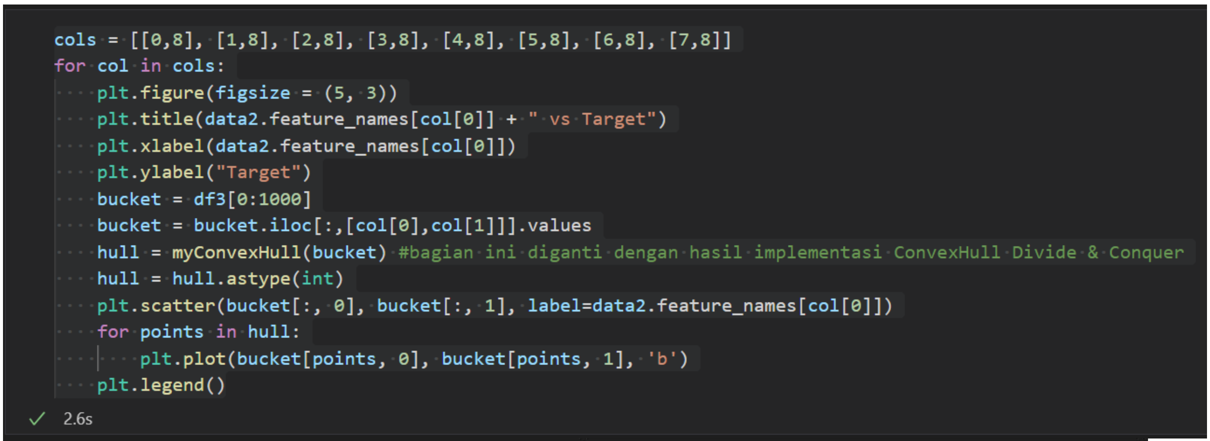
3. Boston House Dataset

```
from sklearn.datasets import fetch_california_housing
data2 = fetch_california_housing()

#create a DataFrame
df3 = pd.DataFrame(data2.data, columns = data2.feature_names)
df3['Target'] = pd.DataFrame(data2.target)
print(df3.shape)
df3.head()
```

✓ 0.6s

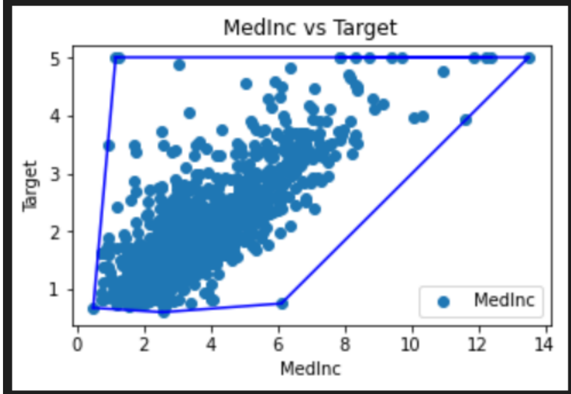
```
(20640, 9)
```

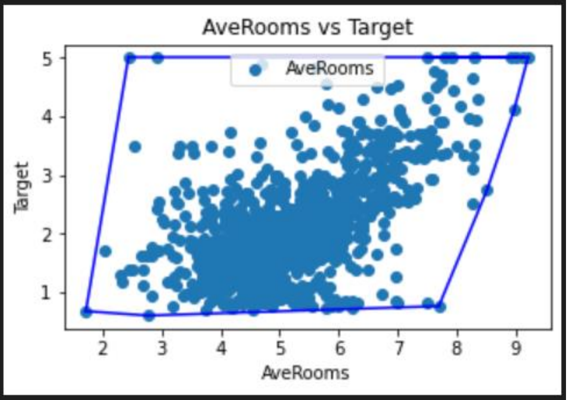
	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	Target
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

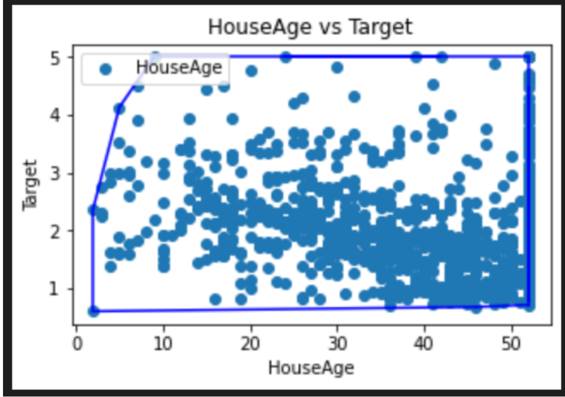


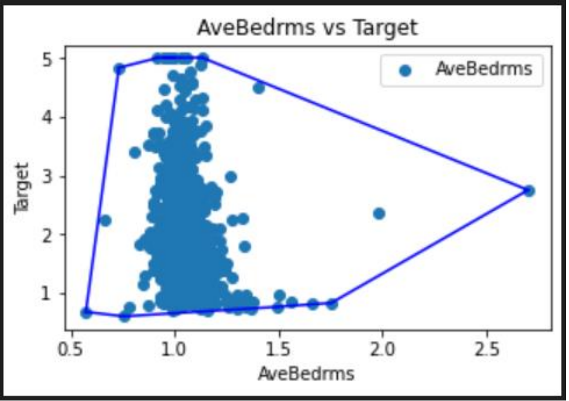
```
cols = [[0,8], [1,8], [2,8], [3,8], [4,8], [5,8], [6,8], [7,8]]
for col in cols:
    plt.figure(figsize = (5, 3))
    plt.title(data2.feature_names[col[0]] + " vs Target")
    plt.xlabel(data2.feature_names[col[0]])
    plt.ylabel("Target")
    bucket = df3[0:1000]
    bucket = bucket.iloc[:,[col[0],col[1]]].values
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull = hull.astype(int)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data2.feature_names[col[0]])
    for points in hull:
        plt.plot(bucket[points, 0], bucket[points, 1], 'b')
    plt.legend()
```

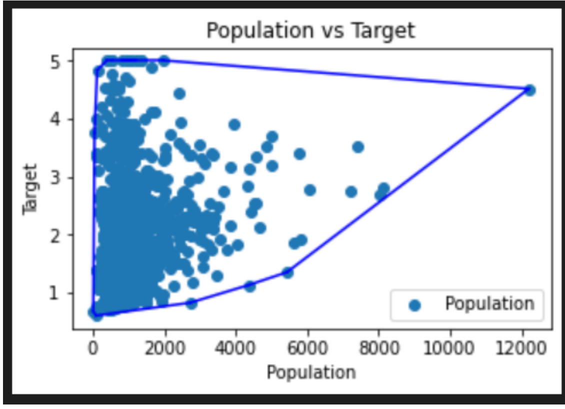
✓ 2.6s

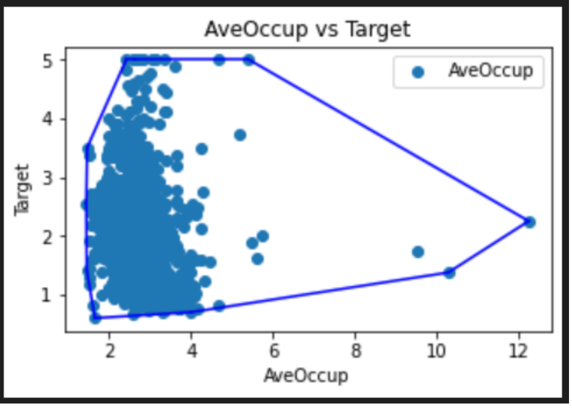


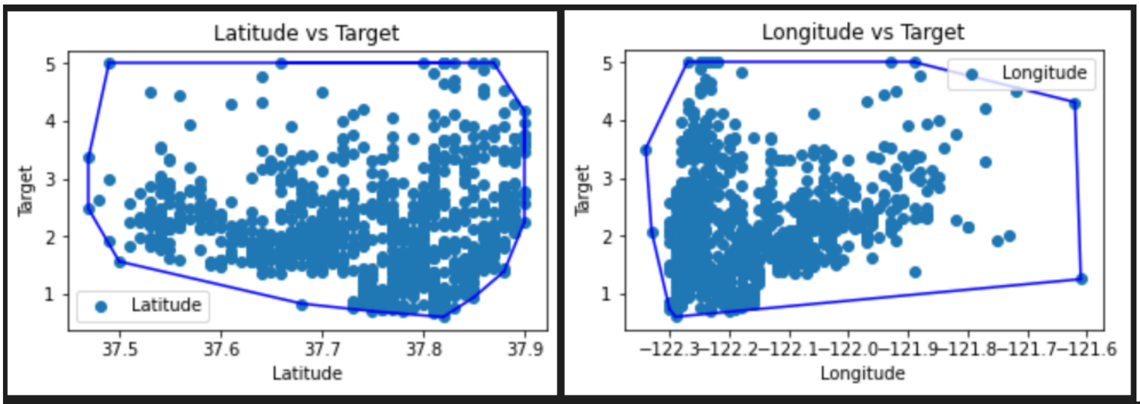


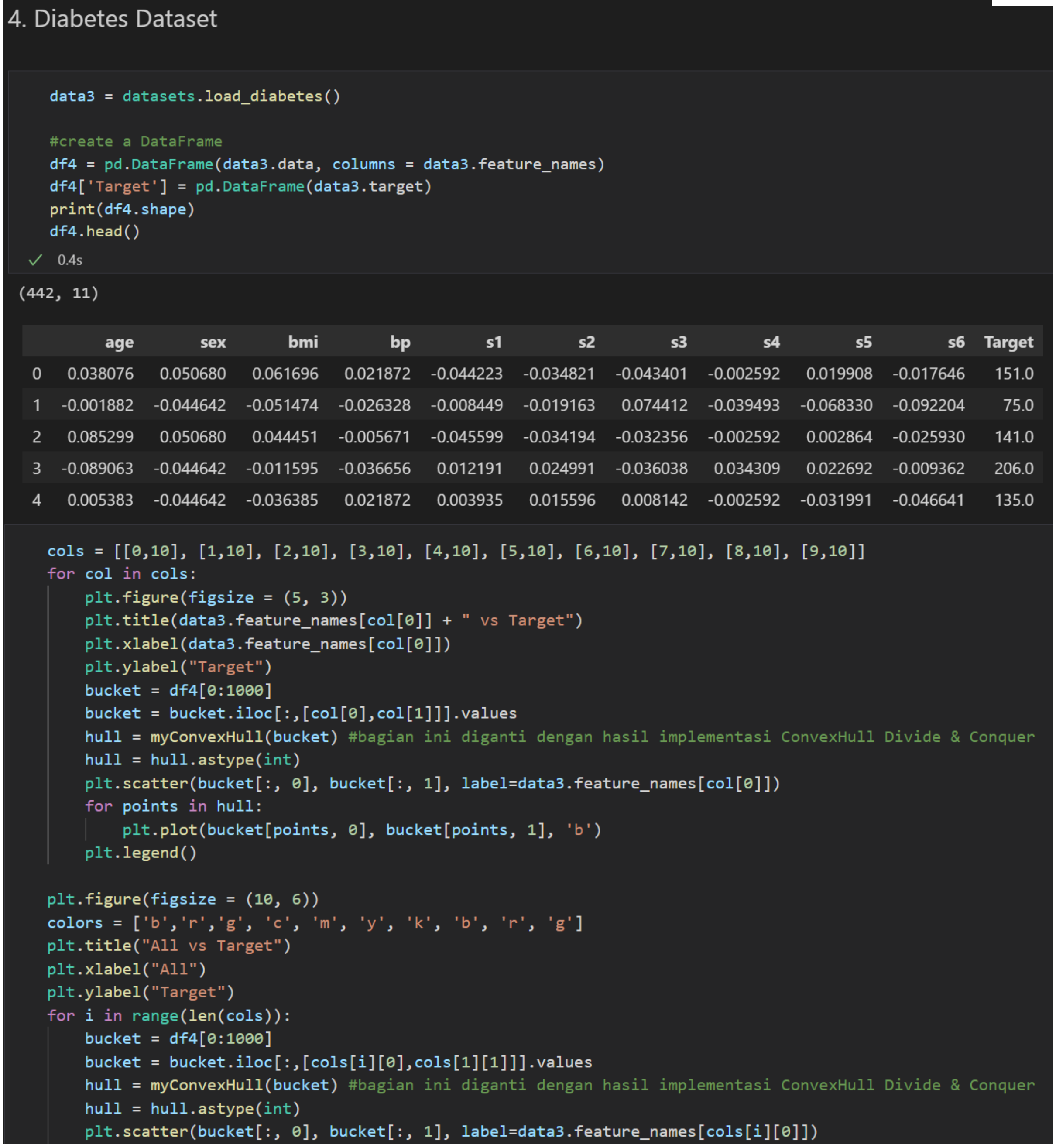












4. Diabetes Dataset

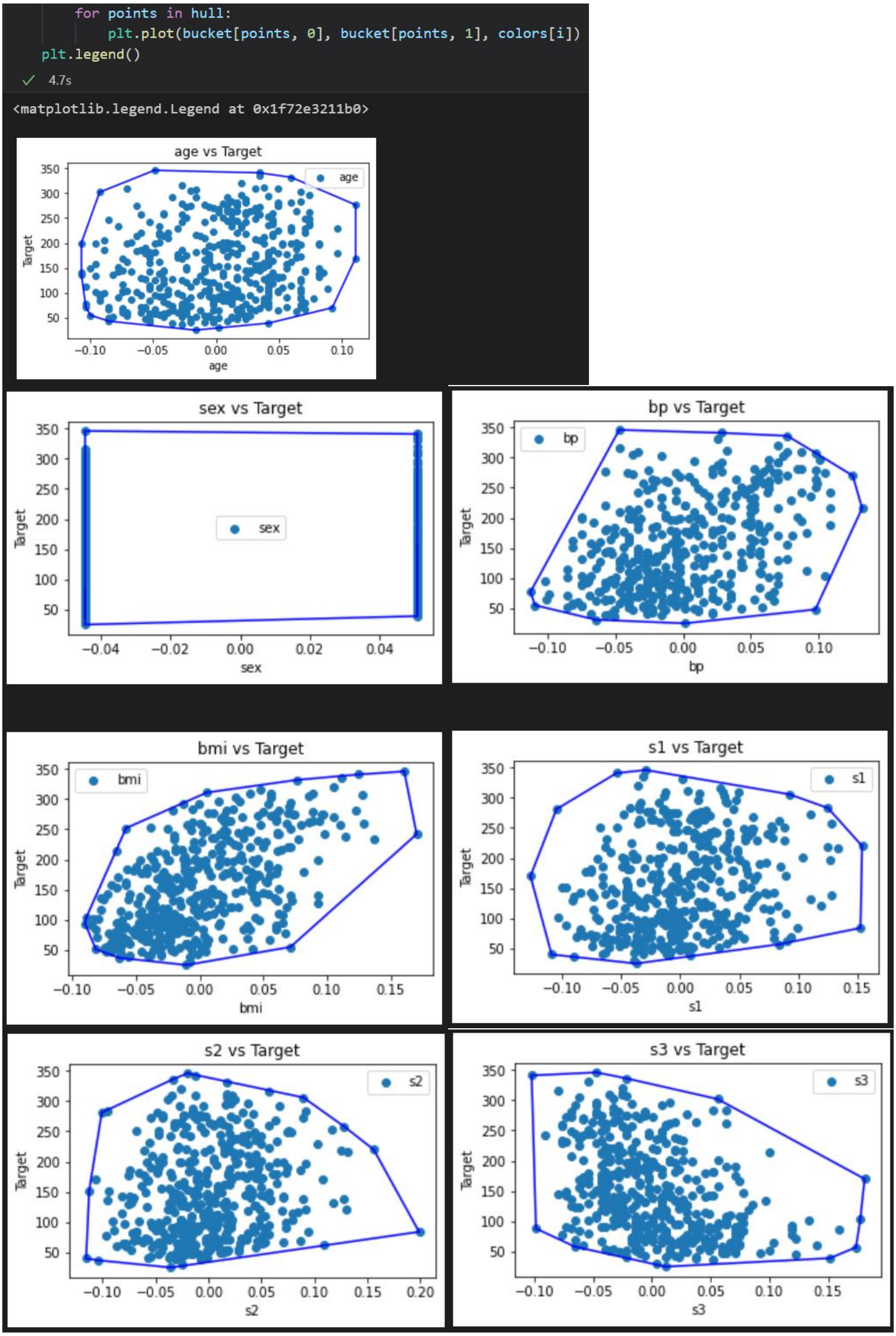
```python
data3 = datasets.load_diabetes()

#create a DataFrame
df4 = pd.DataFrame(data3.data, columns = data3.feature_names)
df4['Target'] = pd.DataFrame(data3.target)
print(df4.shape)
df4.head()
```

✓ 0.4s

```
(442, 11)
```

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	Target
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019908	-0.017646	151.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068330	-0.092204	75.0
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.002864	-0.025930	141.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022692	-0.009362	206.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031991	-0.046641	135.0

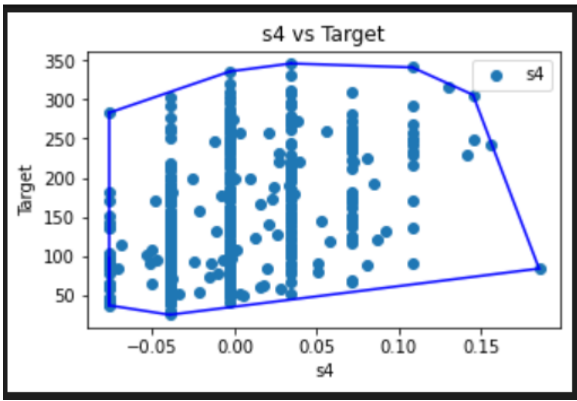
```python
cols = [[0,10], [1,10], [2,10], [3,10], [4,10], [5,10], [6,10], [7,10], [8,10], [9,10]]
for col in cols:
    plt.figure(figsize = (5, 3))
    plt.title(data3.feature_names[col[0]] + " vs Target")
    plt.xlabel(data3.feature_names[col[0]])
    plt.ylabel("Target")
    bucket = df4[0:1000]
    bucket = bucket.iloc[:,[col[0],col[1]]].values
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull = hull.astype(int)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data3.feature_names[col[0]])
    for points in hull:
        plt.plot(bucket[points, 0], bucket[points, 1], 'b')
    plt.legend()

plt.figure(figsize = (10, 6))
colors = ['b','r','g', 'c', 'm', 'y', 'k', 'b', 'r', 'g']
plt.title("All vs Target")
plt.xlabel("All")
plt.ylabel("Target")
for i in range(len(cols)):
    bucket = df4[0:1000]
    bucket = bucket.iloc[:,[cols[i][0],cols[1][1]]].values
    hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
    hull = hull.astype(int)
    plt.scatter(bucket[:, 0], bucket[:, 1], label=data3.feature_names[cols[i][0]])
```

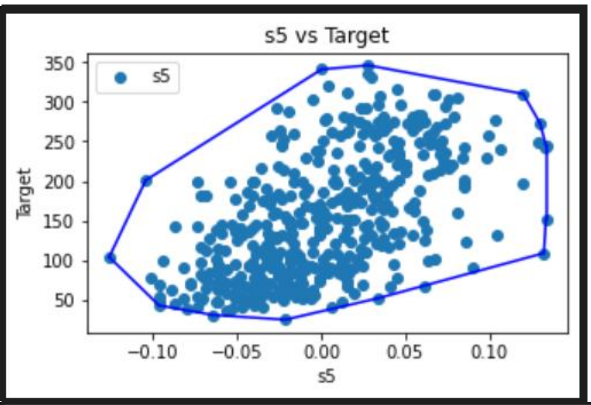


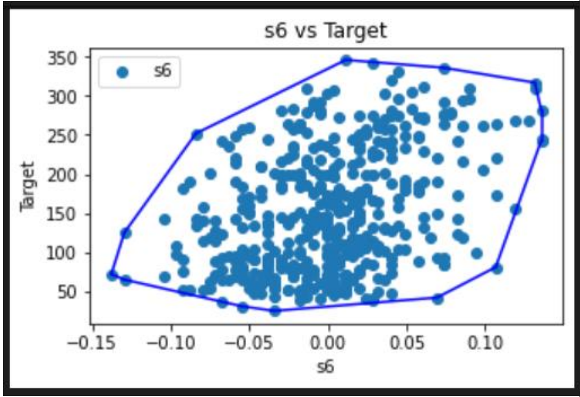
```
    for points in hull:
        plt.plot(bucket[points, 0], bucket[points, 1], colors[i])
plt.legend()
```

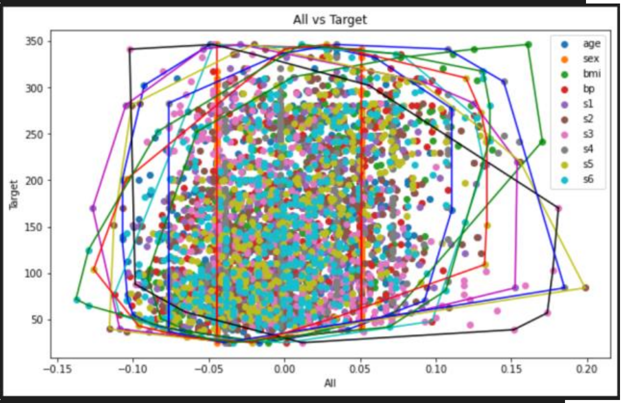
✓ 4.7s

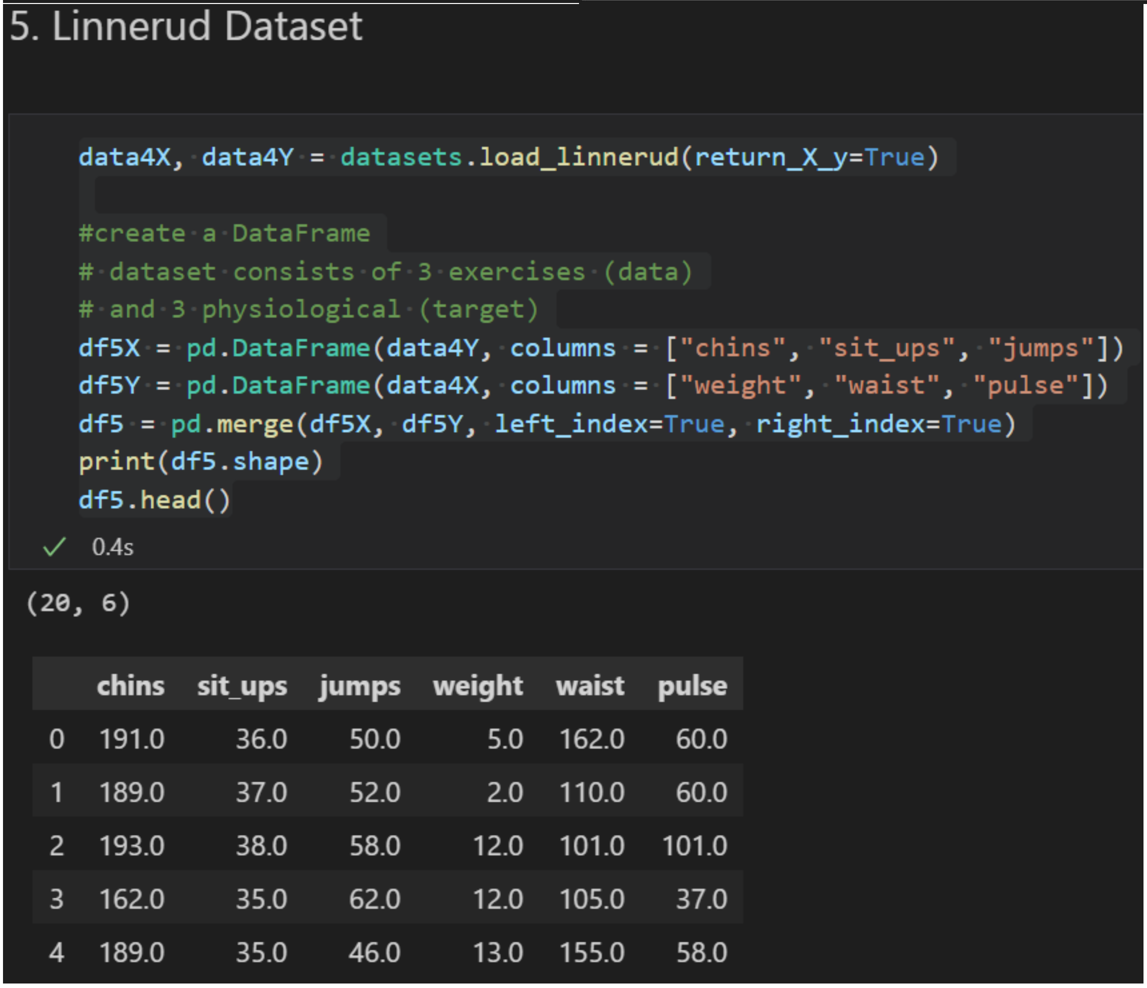
```
<matplotlib.legend.Legend at 0x1f72e3211b0>
```











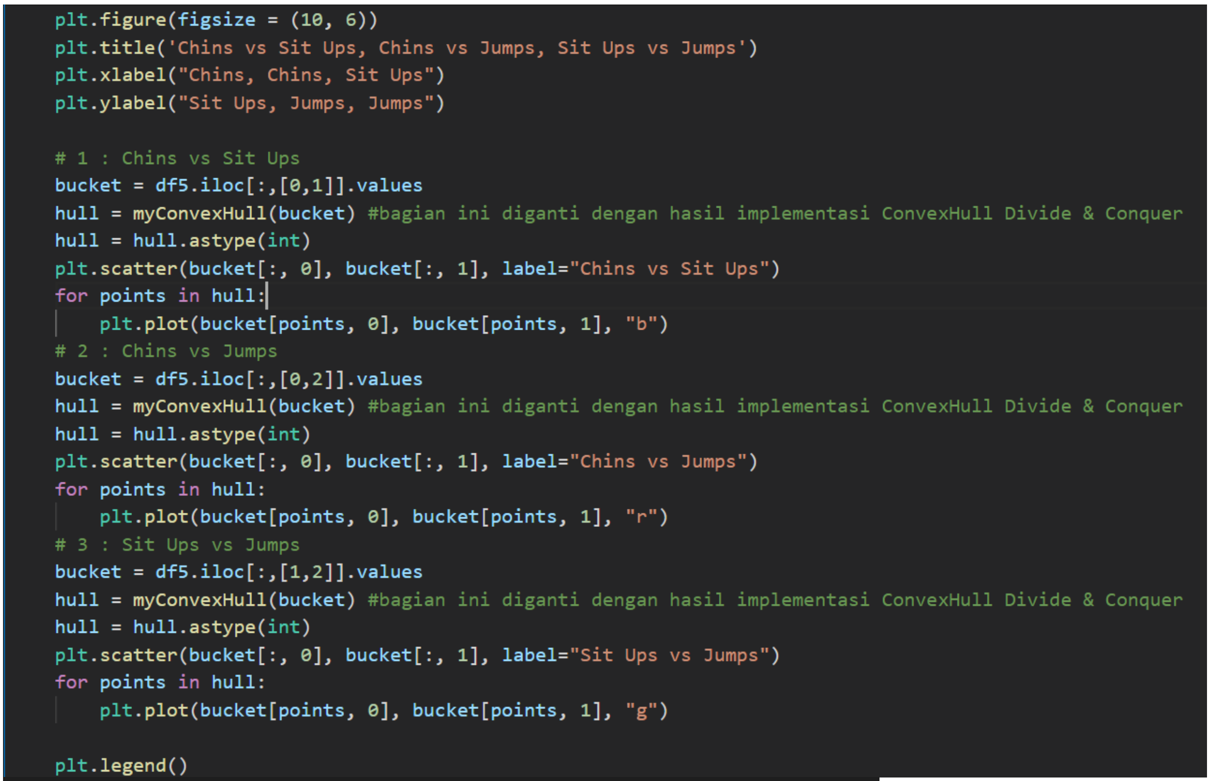
5. Linnerud Dataset

```
data4X, data4Y = datasets.load_linnerud(return_X_y=True)

#create a DataFrame
# dataset consists of 3 exercises (data)
# and 3 physiological (target)
df5X = pd.DataFrame(data4Y, columns = ["chins", "sit_ups", "jumps"])
df5Y = pd.DataFrame(data4X, columns = ["weight", "waist", "pulse"])
df5 = pd.merge(df5X, df5Y, left_index=True, right_index=True)
print(df5.shape)
df5.head()
```

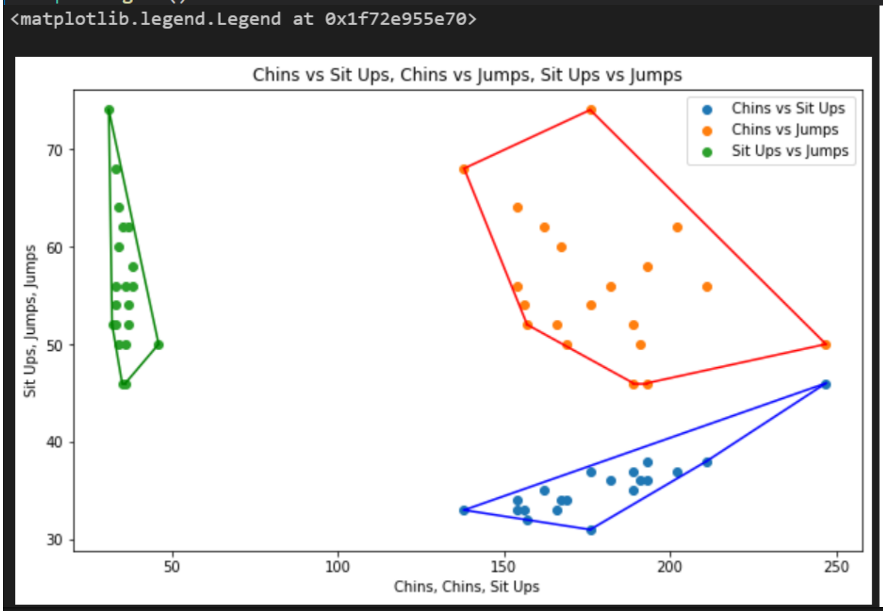
✓ 0.4s

```
(20, 6)
```

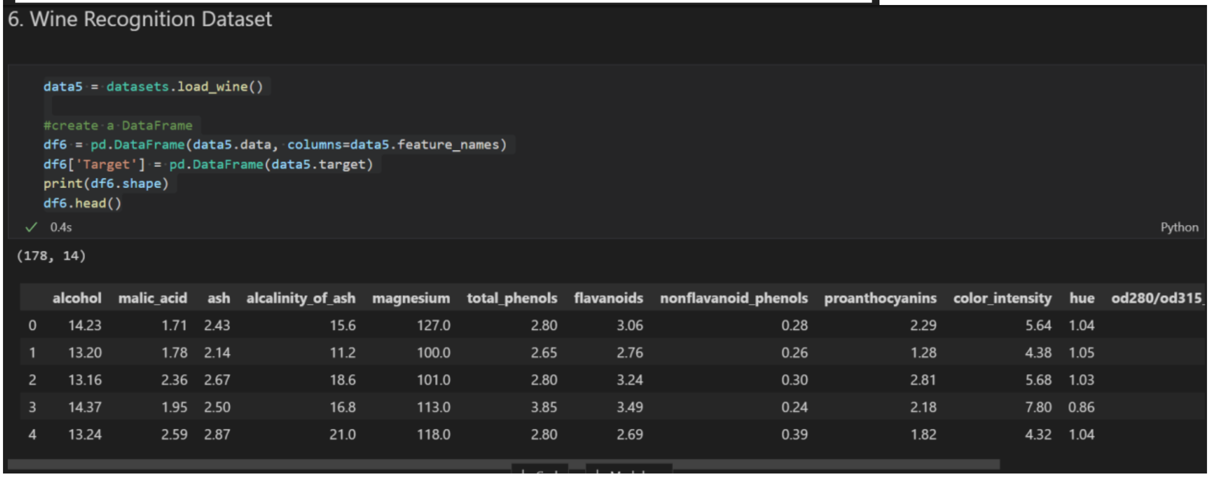
	chins	sit_ups	jumps	weight	waist	pulse
0	191.0	36.0	50.0	5.0	162.0	60.0
1	189.0	37.0	52.0	2.0	110.0	60.0
2	193.0	38.0	58.0	12.0	101.0	101.0
3	162.0	35.0	62.0	12.0	105.0	37.0
4	189.0	35.0	46.0	13.0	155.0	58.0



```python
plt.figure(figsize = (10, 6))
plt.title('Chins vs Sit Ups, Chins vs Jumps, Sit Ups vs Jumps')
plt.xlabel("Chins, Chins, Sit Ups")
plt.ylabel("Sit Ups, Jumps, Jumps")

# 1 : Chins vs Sit Ups
bucket = df5.iloc[:,[0,1]].values
hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
hull = hull.astype(int)
plt.scatter(bucket[:, 0], bucket[:, 1], label="Chins vs Sit Ups")
for points in hull:
    plt.plot(bucket[points, 0], bucket[points, 1], "b")
# 2 : Chins vs Jumps
bucket = df5.iloc[:,[0,2]].values
hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
hull = hull.astype(int)
plt.scatter(bucket[:, 0], bucket[:, 1], label="Chins vs Jumps")
for points in hull:
    plt.plot(bucket[points, 0], bucket[points, 1], "r")
# 3 : Sit Ups vs Jumps
bucket = df5.iloc[:,[1,2]].values
hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
hull = hull.astype(int)
plt.scatter(bucket[:, 0], bucket[:, 1], label="Sit Ups vs Jumps")
for points in hull:
    plt.plot(bucket[points, 0], bucket[points, 1], "g")

plt.legend()
```



```
<matplotlib.legend.Legend at 0x1f72e955e70>
```

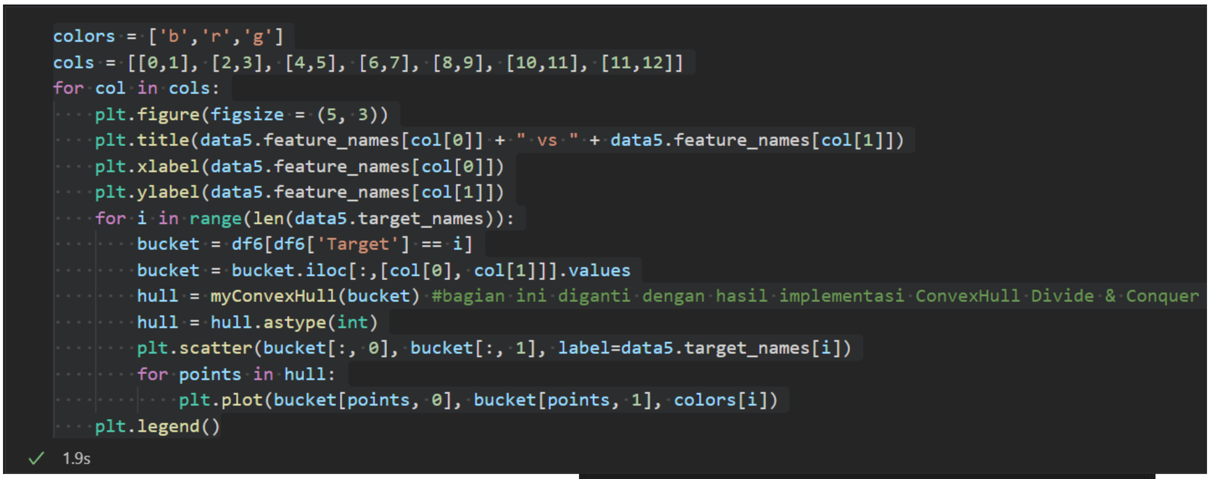


6. Wine Recognition Dataset

```python
data5 = datasets.load_wine()

#create a DataFrame
df6 = pd.DataFrame(data5.data, columns=data5.feature_names)
df6['Target'] = pd.DataFrame(data5.target)
print(df6.shape)
df6.head()
```

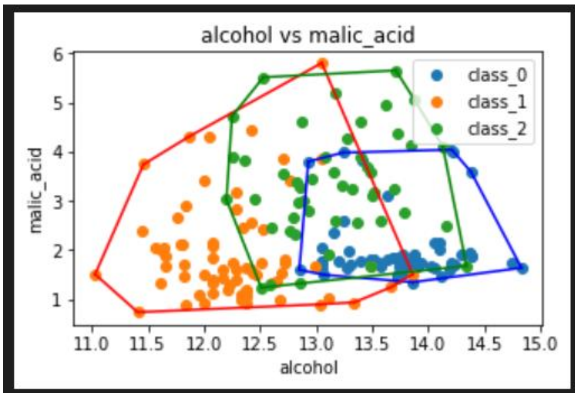
```
(178, 14)
```

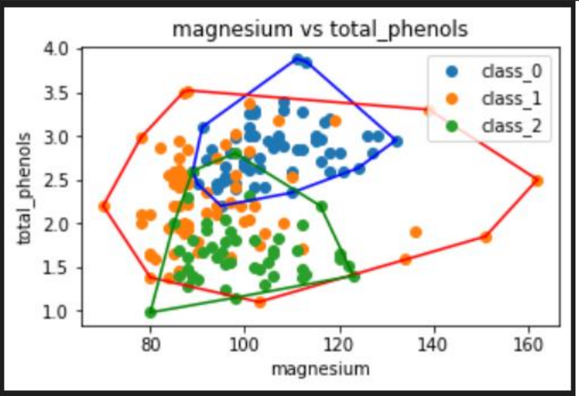
	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	

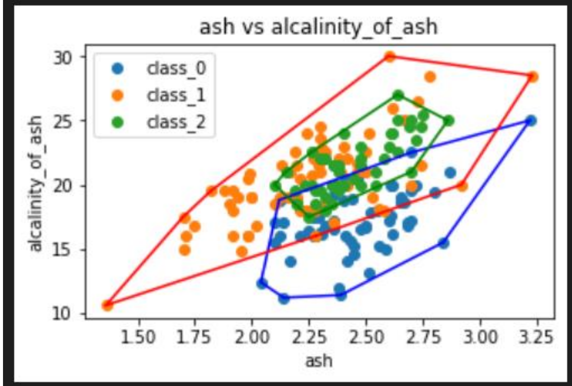


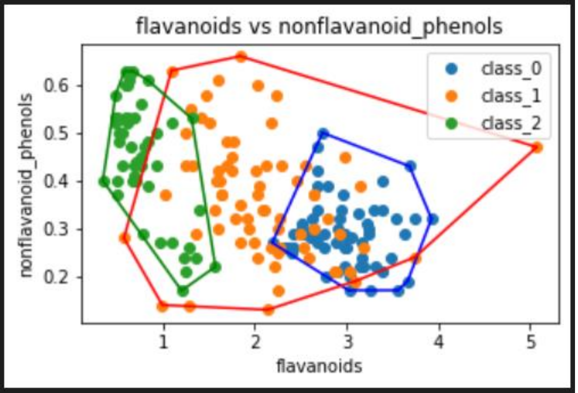
```python
colors = ['b','r','g']
cols = [[0,1], [2,3], [4,5], [6,7], [8,9], [10,11], [11,12]]
for col in cols:
    plt.figure(figsize = (5, 3))
    plt.title(data5.feature_names[col[0]] + " vs " + data5.feature_names[col[1]])
    plt.xlabel(data5.feature_names[col[0]])
    plt.ylabel(data5.feature_names[col[1]])
    for i in range(len(data5.target_names)):
        bucket = df6[df6['Target'] == i]
        bucket = bucket.iloc[:,[col[0], col[1]]].values
        hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
        hull = hull.astype(int)
        plt.scatter(bucket[:, 0], bucket[:, 1], label=data5.target_names[i])
        for points in hull:
            plt.plot(bucket[points, 0], bucket[points, 1], colors[i])
    plt.legend()
```

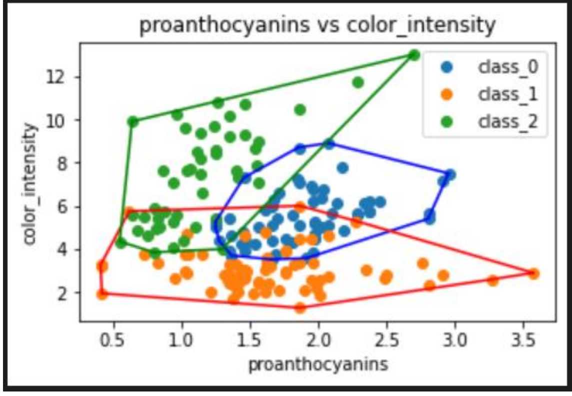
✓ 1.9s

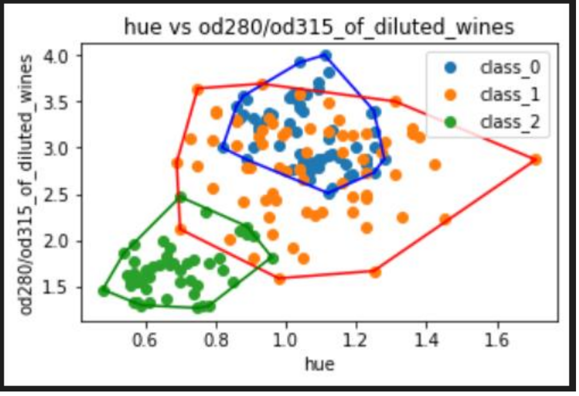


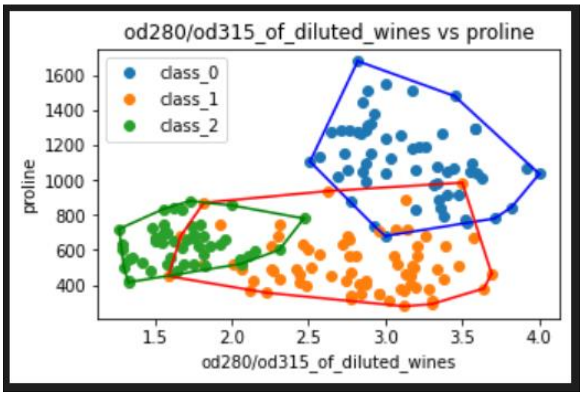


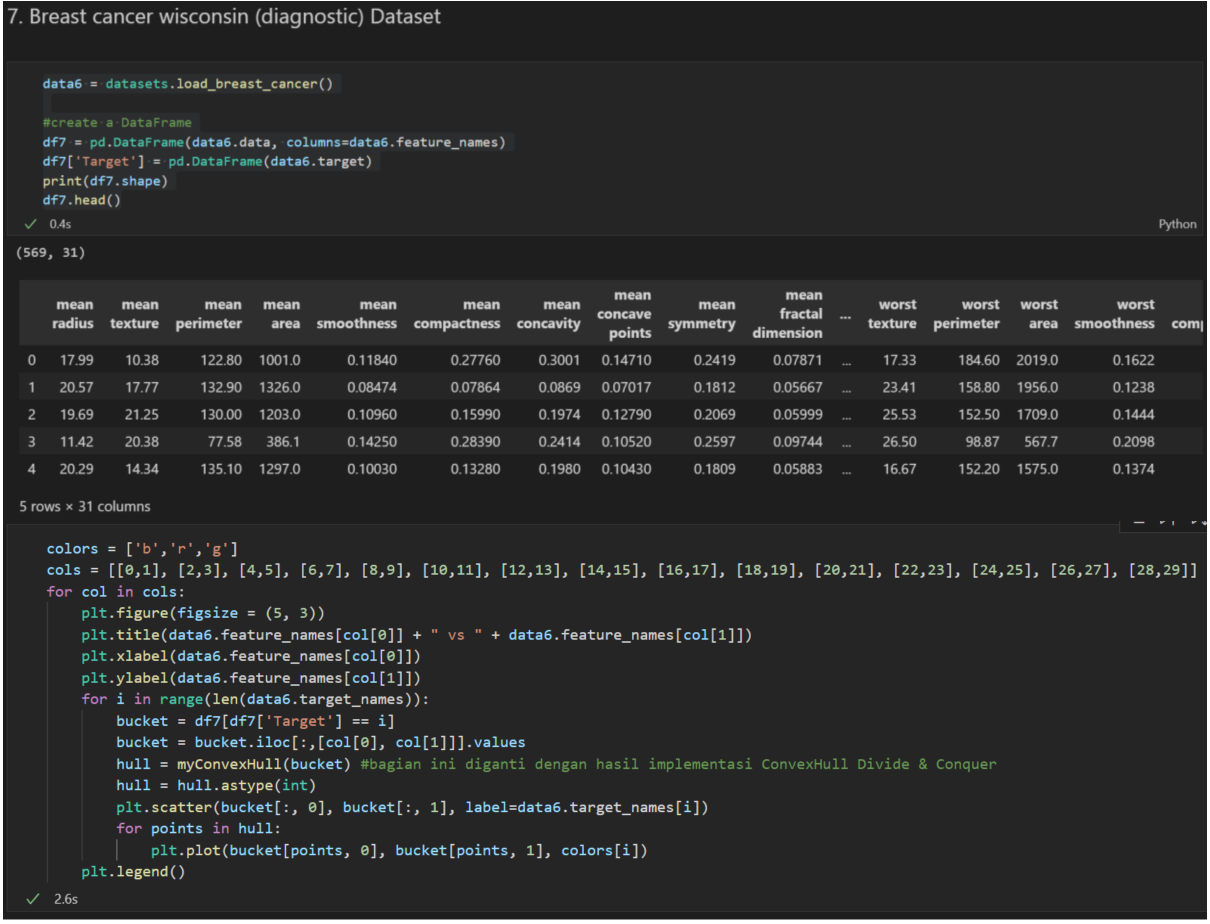












7. Breast cancer wisconsin (diagnostic) Dataset

```
data6 = datasets.load_breast_cancer()

#create a DataFrame
df7 = pd.DataFrame(data6.data, columns=data6.feature_names)
df7['Target'] = pd.DataFrame(data6.target)
print(df7.shape)
df7.head()
```

✓ 0.4s Python

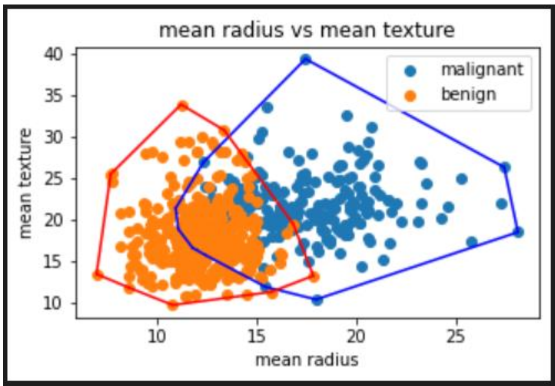
```
(569, 31)
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	com
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	

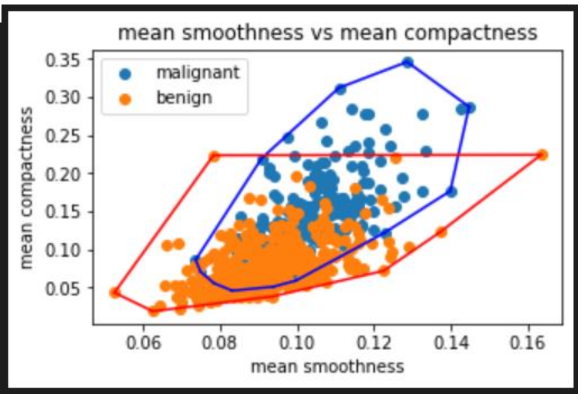
5 rows × 31 columns

```
colors = ['b','r','g']
cols = [[0,1], [2,3], [4,5], [6,7], [8,9], [10,11], [12,13], [14,15], [16,17], [18,19], [20,21], [22,23], [24,25], [26,27], [28,29]]
for col in cols:
    plt.figure(figsize = (5, 3))
    plt.title(data6.feature_names[col[0]] + " vs " + data6.feature_names[col[1]])
    plt.xlabel(data6.feature_names[col[0]])
    plt.ylabel(data6.feature_names[col[1]])
    for i in range(len(data6.target_names)):
        bucket = df7[df7['Target'] == i]
        bucket = bucket.iloc[:,[col[0], col[1]]].values
        hull = myConvexHull(bucket) #bagian ini diganti dengan hasil implementasi ConvexHull Divide & Conquer
        hull = hull.astype(int)
        plt.scatter(bucket[:, 0], bucket[:, 1], label=data6.target_names[i])
        for points in hull:
            plt.plot(bucket[points, 0], bucket[points, 1], colors[i])
    plt.legend()
```

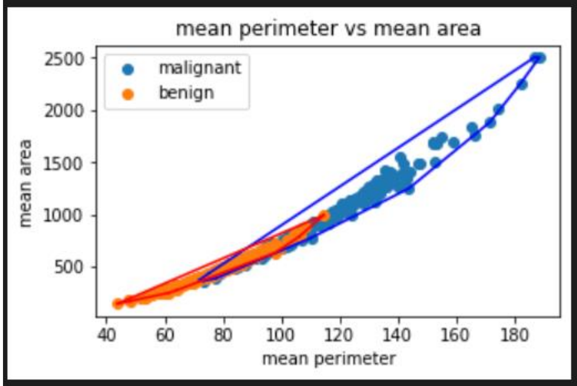
✓ 2.6s



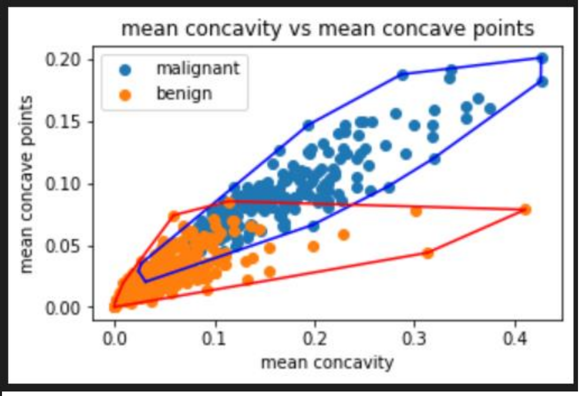
mean radius vs mean texture



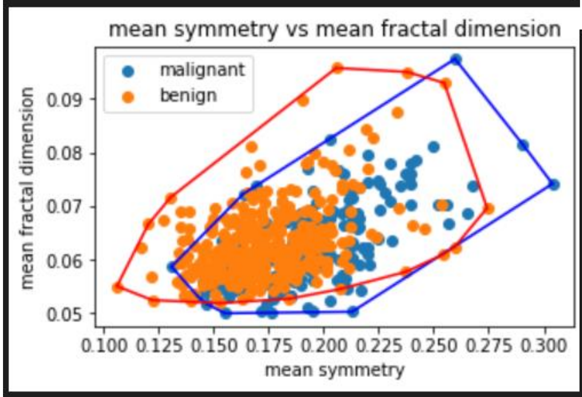
mean smoothness vs mean compactness



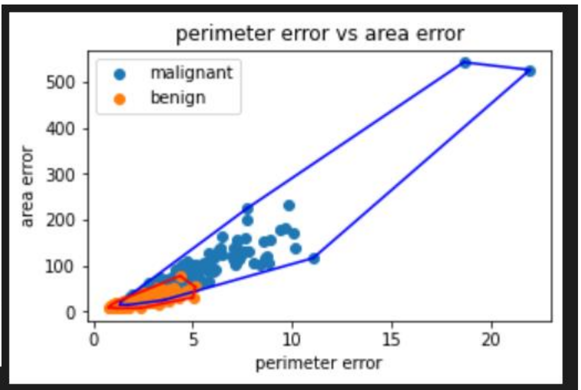
mean perimeter vs mean area



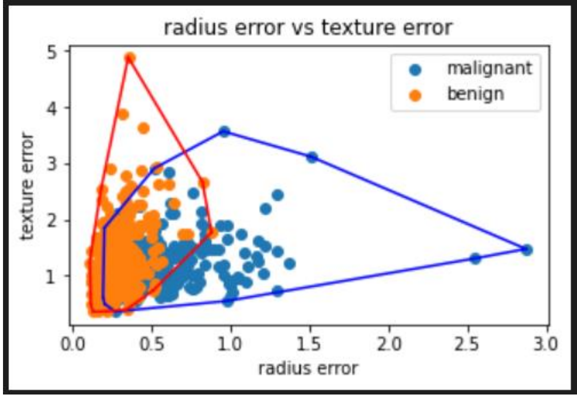
mean concavity vs mean concave points



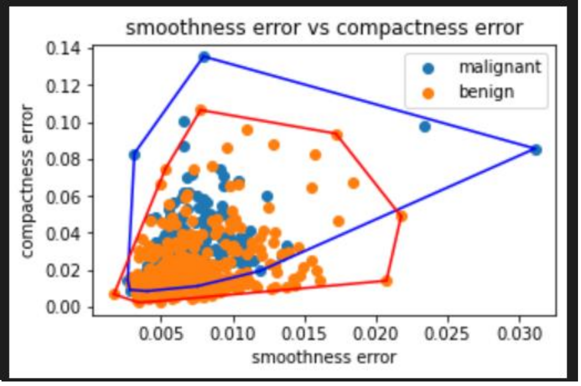
mean symmetry vs mean fractal dimension



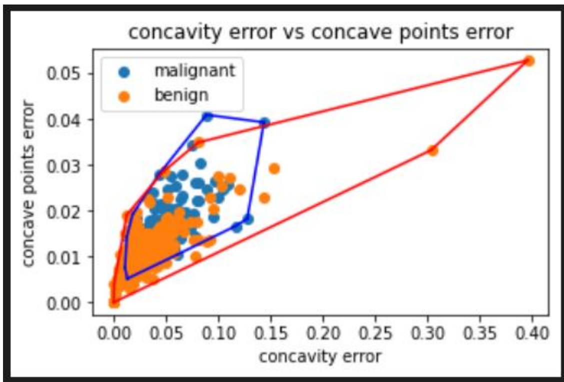
perimeter error vs area error

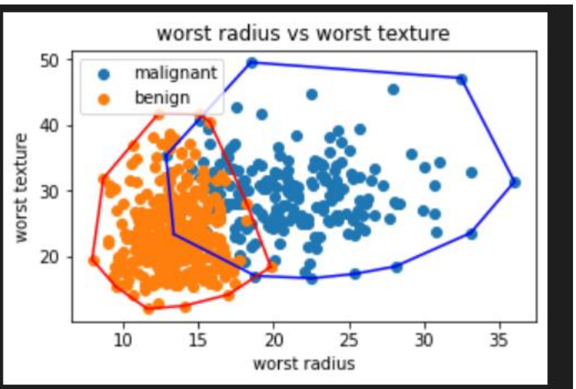


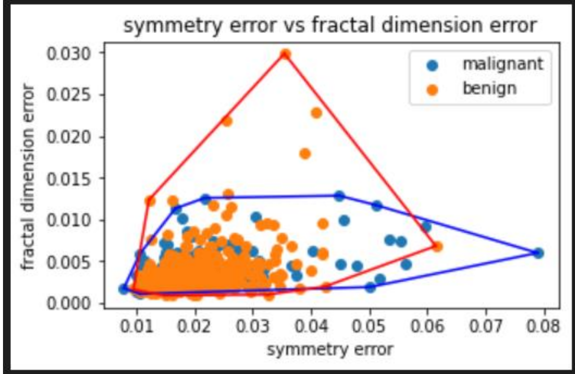
radius error vs texture error

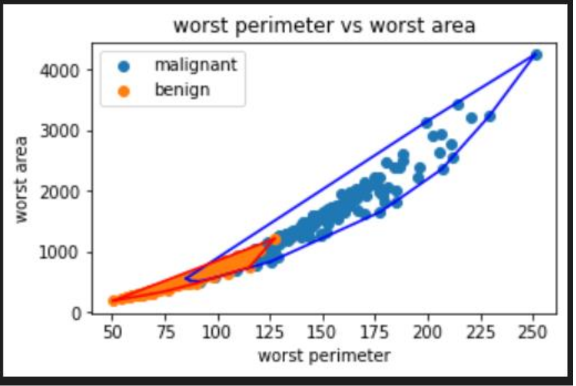


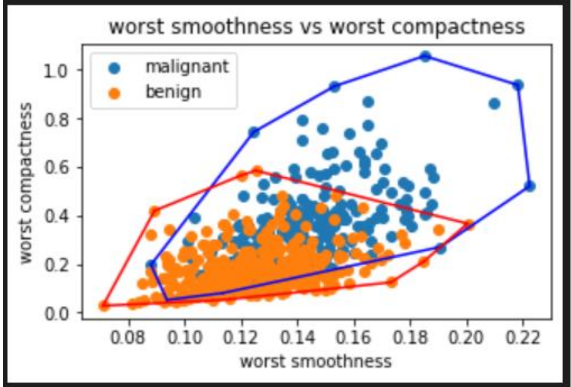
smoothness error vs compactness error

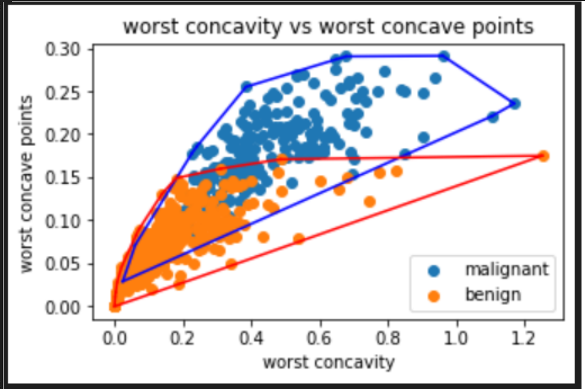


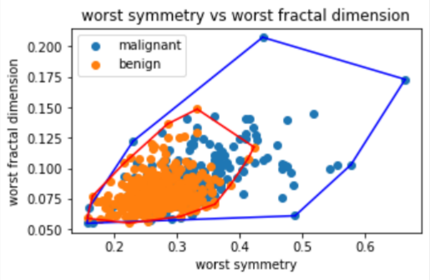












Acknowledgements

References:

- **Munir, R., Maulidevi, N. U. 2022. *Algoritma Divide and Conquer (Bagian 4)*. Bahan Kuliah IF2211 Strategi Algoritma.**
- **Anany V. Levitin. 2002. *Introduction to the Design and Analysis of Algorithms*. Addison-Wesley Longman Publishing Co., Inc., USA.**
- **Scikit-Learn Toy Datasets Webpage. Accessed from: https://scikit-learn.org/stable/datasets/toy_dataset.html#linnerrud-dataset on 27 February 2022.**

BAB IV

CHECKLIST

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.	✓	