TUGAS KECIL IF2211
STRATEGI ALGORITMA

IMPLEMENTASI CONVEX HULL UNTUK VISUALISASI TES
LINEAR SEPARABILITY DATASET
DENGAN ALGORITMA DIVIDE AND CONQUER

Disusun oleh
Farrel Farandieka Fibriyanto - 13520054

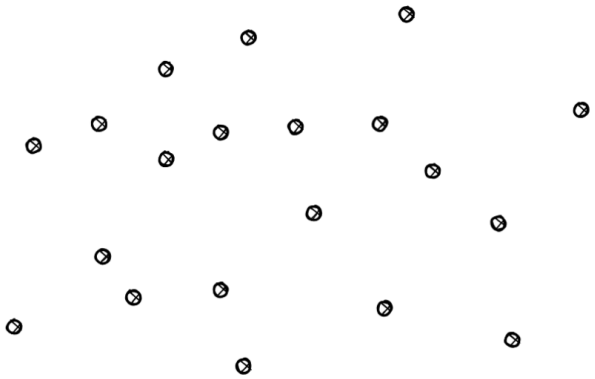
PROGRAM STUDI TEKNIK INFORMATIKA SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG

Semester II Tahun 2021/2022

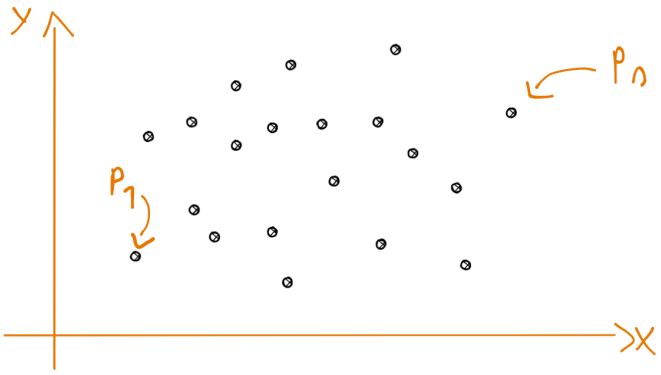
Daftar Isi

Algoritma Divide & Conquer

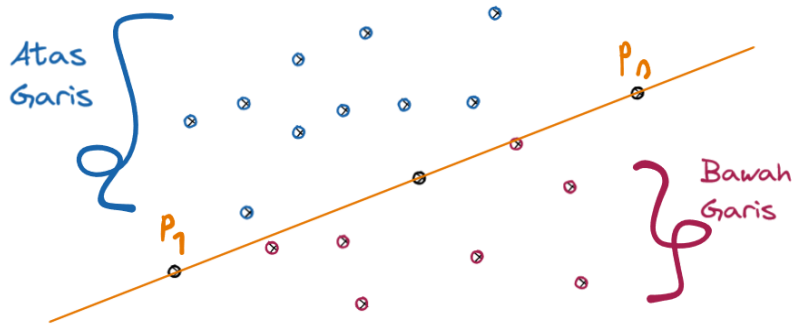
Agar mempermudah visualiasi, dataset yang ingin dibuat convex hullnya akan terlebih dahulu di-*plot* dalam dua dimensi:



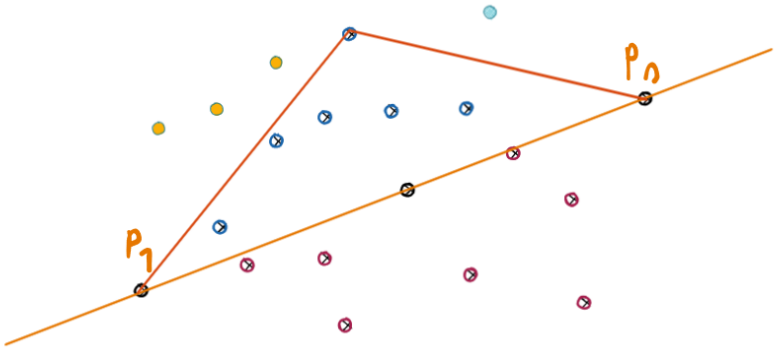
1. Cari titik yang memiliki x minimum dan maksimum, tetapkan kedua titik tersebut sebagai p_1 dan p_n



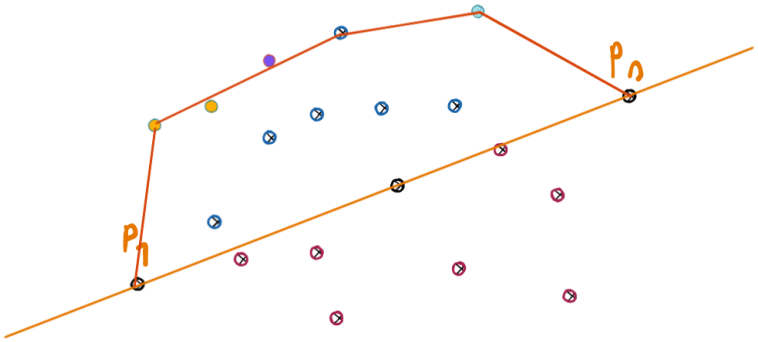
2. Tarik garis bayangan melalui kedua titik tersebut dan bagi dua himpunan titik menjadi titik yang berada di bawah garis tersebut dan yang berada di atas garis tersebut dan abaikan titik yang berada pada garis



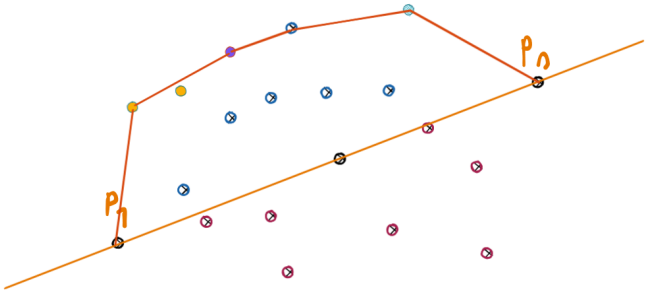
3. Kita mulai dari himpunan titik yang berada di atas garis, cari titik yang memiliki jarak terjauh dari garis dan hubungkan titik tersebut dengan p_1 dan p_n menggunakan garis bayangan. Carilah titik yang berada di atas kedua garis yang terbentuk (kita tidak perlu memperdulikan titik yang berada di bawah karena tidak mungkin membentuk *convex hull*)



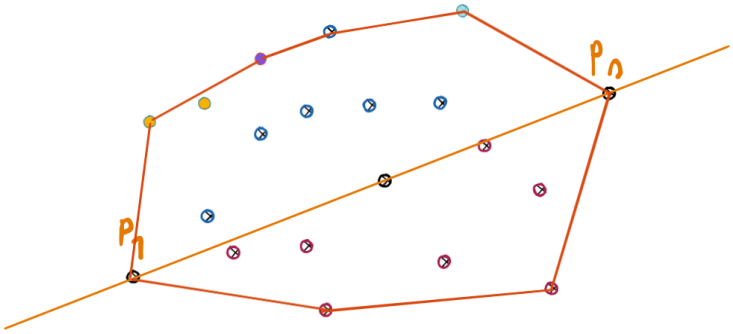
4. Ulangi tahap 2 sampai 3 menggunakan kedua garis yang terbentuk



5. Ulangi tahap 4 sampai tidak ada lagi titik yang berada di atas suatu garis yang sedang dicek



6. Ulangi tahap 3 sampai 5 menggunakan himpunan titik yang dibawah, dengan merubah kata kunci “di atas” menjadi “di bawah” dan sebaliknya.



Source Code Program

```python
def detFunc(p1, p2, p3):
    """
    does the determinant formula thing from Rin-sensei to check if a given point p3 is above the line made between point p1 and p2
    """

    x1 = p1[0]
    y1 = p1[1]
    x2 = p2[0]
    y2 = p2[1]
    x3 = p3[0]
    y3 = p3[1]
    determinant = x1 * y2 + x3 * y1 + x2 * y3 - x3 * y2 - x2 * y1 - x1 * y3
    return determinant

def heron(len1, len2, len3):
    """
    does the heron formula for calculating a given triangle area
    """
    s = (len1 + len2 + len3)/2
    return (s * (s-len1) * (s-len2) * (s-len3))**0.5

def findDistance(p1, p2, p3):
    """
    returns the distance of point p3 relative to the line made between point p1 and point p2
    """
    from math import dist
    x1 = p1[0]
    y1 = p1[1]
    x2 = p2[0]
    y2 = p2[1]
    # print(p1, p2, p3)
    pembilang = heron(dist(p1, p2), dist(p2, p3), dist(p3, p1))
    penyebut = ((x2-x1)**2 + (y2-y1)**2)**0.5
    return pembilang/penyebut


def findPointsBelowLine(p1, p2, idxArr, baseArr):
    """
    finds the points BELOW the line connecting p1 and p2 in points whose index exists in idxArr,

    returns the index of those points based on baseArr
    """
    temp = []
    for idx in idxArr:
        if (detFunc(p1, p2, baseArr[idx]) < 0):
            temp.append(idx)
    return temp
```

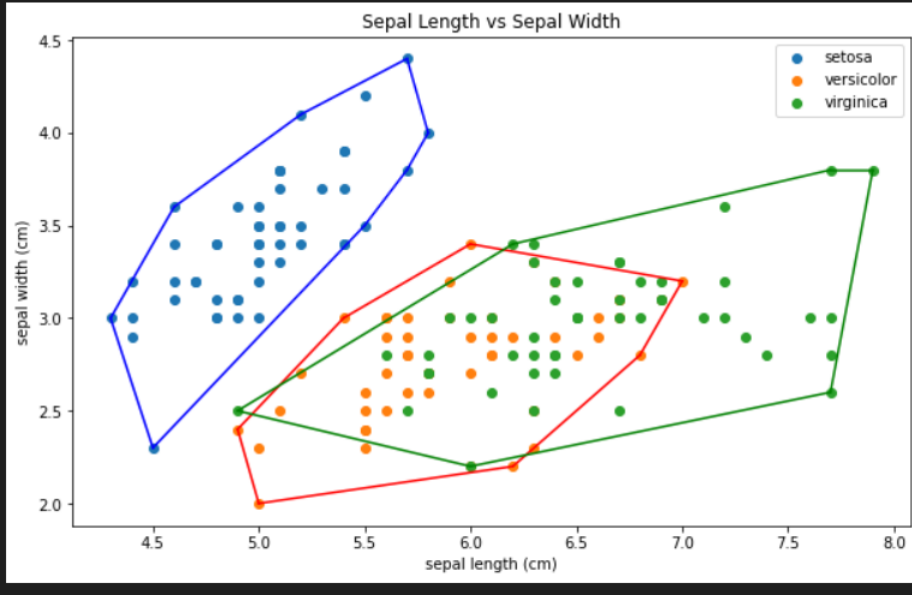
```
def findExtremesBelow(idxPoint1, idxPoint2, idxArr, baseArr):
    """
    finds the points that makes the bottom of a convex hull

    returns the index of those points in the array baseArr
    """
    maxDistLocal = 0
    tempIdx = -1  # basic error code if not found
    for i in idxArr:
        if (maxDistLocal < findDistance(baseArr[idxPoint1], baseArr[idxPoint2], baseArr[i])):
            maxDistLocal = findDistance(baseArr[idxPoint1], baseArr[idxPoint2], baseArr[i])
            tempIdx = i
    if (tempIdx == -1):
        return [idxPoint1, idxPoint2]
    else:
        return findExtremesBelow(idxPoint1, tempIdx, findPointsBelowLine(baseArr[idxPoint1], baseArr[tempIdx], idxArr,
baseArr), baseArr) + findExtremesBelow(tempIdx, idxPoint2, findPointsBelowLine(baseArr[tempIdx], baseArr[idxPoint2],
idxArr, baseArr), baseArr)

def findPointsAboveLine(p1, p2, idxArr, baseArr):
    """
    finds the points ABOVE the line connecting p1 and p2 in points whose index exists in idxArr,

    returns the index of those points based on baseArr
    """
    temp = []
    for idx in idxArr:
        if (detFunc(p1, p2, baseArr[idx]) > 0):
            temp.append(idx)
    return temp

def findExtremesAbove(idxPoint1, idxPoint2, idxArr, baseArr):
    """
    finds the points that makes the top of a convex hull

    returns the index of those points in the array baseArr
    """
    maxDistLocal = 0
    tempIdx = -1  # basic error code if not found
    for i in idxArr:
        if (maxDistLocal < findDistance(baseArr[idxPoint1], baseArr[idxPoint2], baseArr[i])):
            maxDistLocal = findDistance(baseArr[idxPoint1], baseArr[idxPoint2], baseArr[i])
            tempIdx = i
    if (tempIdx == -1):
        return [idxPoint1, idxPoint2]
    else:
        return findExtremesAbove(idxPoint1, tempIdx, findPointsAboveLine(baseArr[idxPoint1], baseArr[tempIdx], idxArr,
baseArr), baseArr) + findExtremesAbove(tempIdx, idxPoint2, findPointsAboveLine(baseArr[tempIdx], baseArr[idxPoint2],
idxArr, baseArr), baseArr)


def connectTwo(idx1, idx2, arr):
    """
    connects two indexes and adds it into arr

    Ex: (1, 2, []), the array becomes [[1,2]]
    """
    arr.append([idx1, idx2])
```

```python
def connectAllPoints(idxArr):
    """
    connects all points given in the array idxArr and returns it

    Ex: [1, 2, 3] becomes [[1,2] [2,3]]
    """
    temp = []
    for i in range(len(idxArr) - 1):
        if (idxArr[i] != idxArr[i + 1]):   # making sure theres no such thing as a point connecting to itself
            connectTwo(idxArr[i], idxArr[i+1], temp)
    return temp

def myConvexHull(inp):
    """
    QuickHull :D

    returns an array of index of points in the input array that makes a convex hull
    """
    bucket = inp

    # first, find the maximum and minimum Points in the X axis (ignores height, or Y)
    nEffective = len(bucket)
    x_idxMax = 0
    x_idxMin = 0
    for i in range(nEffective):
        if (bucket[i][0] > bucket[x_idxMax][0]):
            x_idxMax = i
        if (bucket[i][0] < bucket[x_idxMin][0]):
            x_idxMin = i

    # makes the base index array wooooooaaaaaaa
    baseIndexArr = []
    for i in range(len(bucket)):
        baseIndexArr.append(i)

    # finally, create the connection index array
    connIdx = connectAllPoints(findExtremesAbove(x_idxMin, x_idxMax, findPointsAboveLine(bucket[x_idxMin],
bucket[x_idxMax], baseIndexArr, bucket), bucket))
    connIdx += connectAllPoints(findExtremesBelow(x_idxMin, x_idxMax, findPointsBelowLine(bucket[x_idxMin],
bucket[x_idxMax], baseIndexArr, bucket), bucket))

    # returns the array
    return connIdx
```

Screenshot

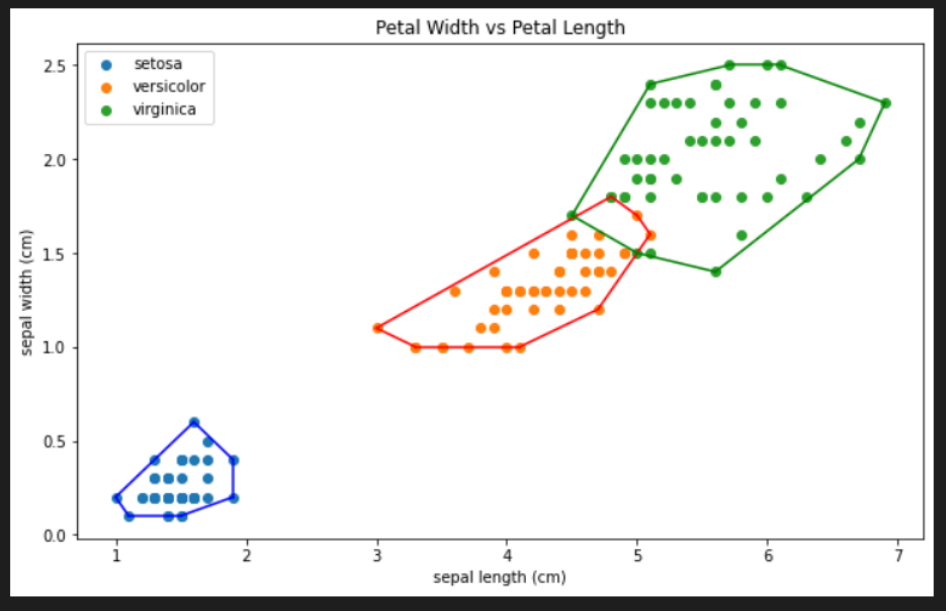
Sepal Length vs Sepal Width

```
<matplotlib.legend.Legend at 0xf07e68>
```



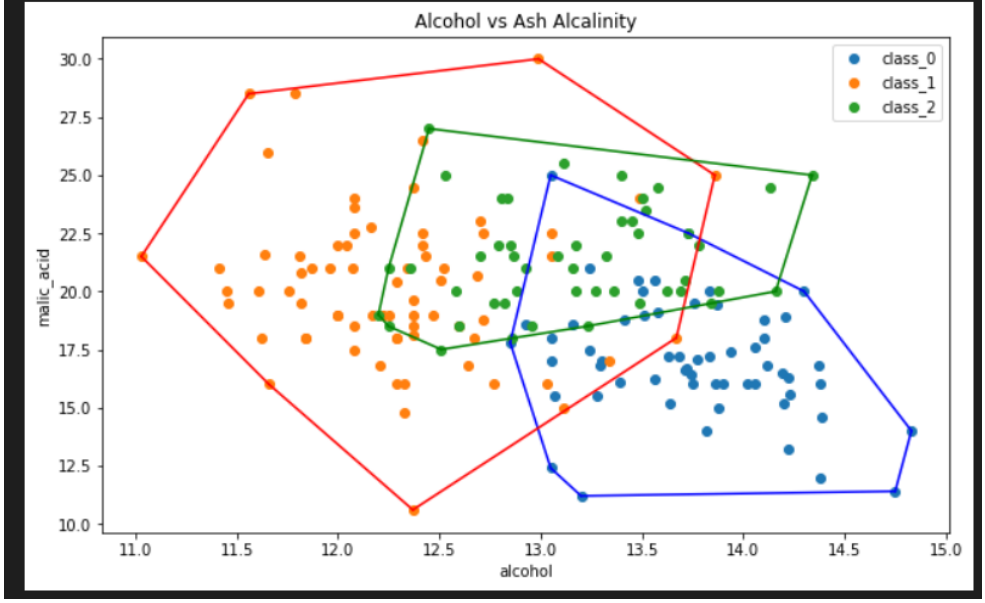
Petal Width vs Petal Length

```
<matplotlib.legend.Legend at 0xeb29b8>
```



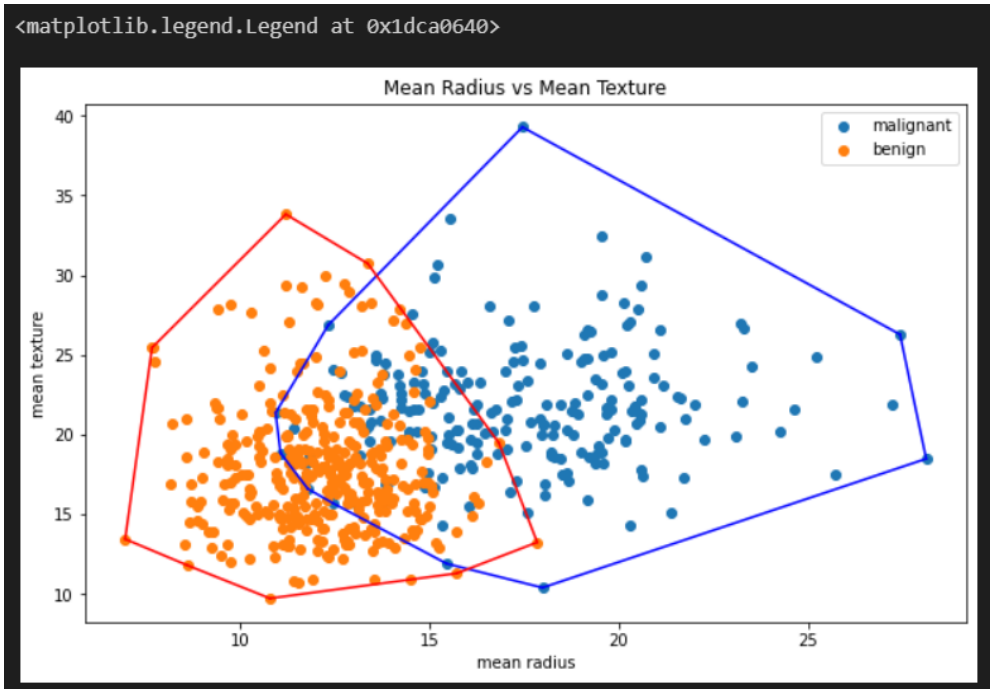
Alcohol vs Ash Alcalinity (wine dataset)

```
<matplotlib.legend.Legend at 0x1f88f8e0>
```



Mean Radius vs Mean Texture

```
<matplotlib.legend.Legend at 0x1dca0640>
```



Checklist

No.	Poin	Keberhasilan Poin
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	☑

Link Penting

Drive Source Code:
https://drive.google.com/drive/folders/1A7khD3xv-vTW5hvwQRuoHIWDvMhM8pBw?usp=sharing

Repository Github:
https://drive.google.com/drive/folders/1A7khD3xv-vTW5hvwQRuoHIWDvMhM8pBw?usp=sharing