TUGAS KECIL 2

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

LAPORAN

Diajukan sebagai salah satu tugas mata kuliah IF2211 Strategi Algoritma pada Semester II

Tahun Akademik 2021-2022

Oleh

Fachry Dennis Heraldi 13520139

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

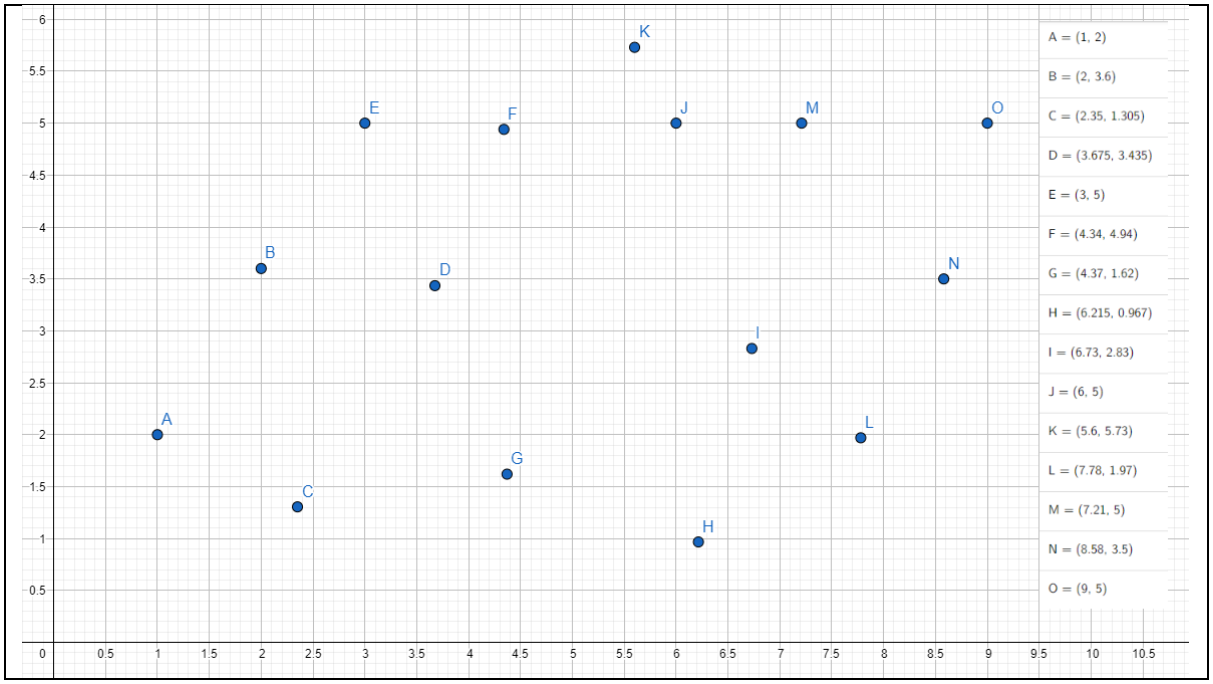
BANDUNG

2022

DAFTAR ISI

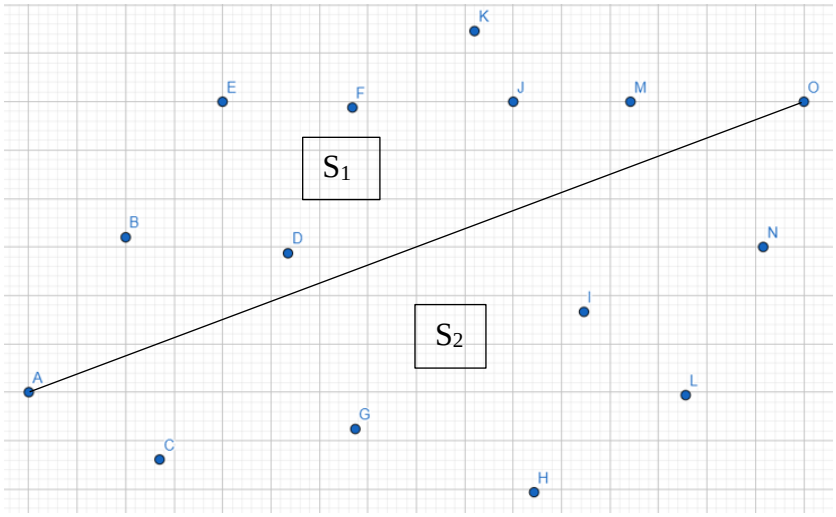
Algoritma *Divide and Conquer*

Pada tugas ini, algoritma *divide and conquer* untuk menentukan *convex hull* disadur dari *Bahan Kuliah IF2211 Stima - Algoritma Divide and Conquer Bagian 4 Oleh: Nur Ulfa Maulidevi & Rinaldi Munir*. Sebagai ilustrasi bagaimana algoritma ini bekerja, diberikan himpunan titik S pada bidang kartesian dua dimensi.



Langkah-langkah penentuan titik yang membangun *convex hull* adalah sebagai berikut.

1. Kumpulan titik diurutkan berdasarkan nilai absis yang menaik, dan jika ada nilai absis yang sama, maka diurutkan dengan nilai ordinat yang menaik. Sebelum diurutkan, urutan elemen titik pada himpunan disimpan terlebih dahulu, misalkan disimpan dalam himpunan originalS.
2. p_1 dan p_n adalah dua titik ekstrim yang akan membentuk *convex hull* untuk kumpulan titik tersebut. Karena himpunan S telah diurutkan, maka yang menjadi titik ekstrim adalah elemen pertama dan elemen terakhir di himpunan. Pada kumpulan titik di atas, yang menjadi p_1 adalah titik A(1,2) dan yang menjadi p_2 adalah titik O(9,5).
3. Selanjutnya hubungkan dengan garis dua titik ekstrim yang telah ditentukan sebagai berikut.



Garis yang menghubungkan titik A dan titik O membagi S menjadi dua bagian, bagian atas S_1 dan bagian bawah S_2. Untuk memeriksa apakah titik berada dibagian atas atau bawah garis, gunakan penentuan determinan dari tiga titik dan iterasi untuk semua titik kecual titik A dan titik O. Rumus penentuan determinan dari tiga titik adalah sebagai berikut.

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

Dengan titik A(x_1,y_1), titik O(x_2,y_2), dan titik lain, misal titik P(x_3,y_3). Apabila nilai determinan positif maka titik P berada di atas, daftarkan ke dalam himpunan S_1. Sebaliknya untuk nilai determinan negatif maka titik P berada di bawah, daftarkan ke dalam himpunan S_2.

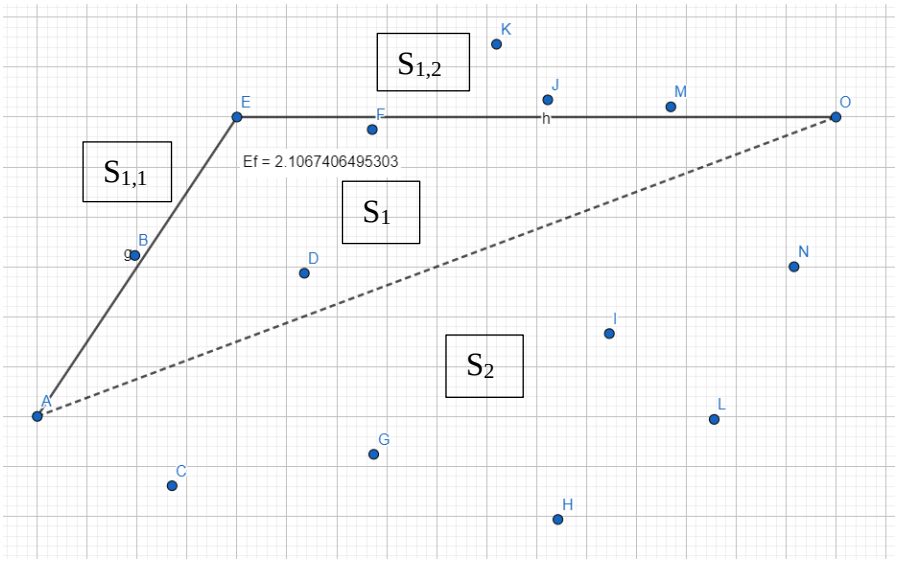
4. Selanjutnya akan dilakukan operasi pembentukan *convex hull* pada titik bagian atas (S_1) dan bagian bawah (S_2) secara rekursif. Sebagai contoh, akan dijelaskan untuk operasi bagian atas. Dibagian operasi rekursif ini penerapan *divide and conquer* diterapkan.
5. Akan dicari titik yang memiliki jarak paling jauh dari garis penghubung titik A dan titik O. Gunakan rumus jarak dari titik ke garis penghubung dua titik sebagai berikut.

$$\text{distance}(P_1, P_2, (x_0, y_0)) = \frac{|(x_2 - x_1)(y_1 - y_0) - (x_1 - x_0)(y_2 - y_1)|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}.$$

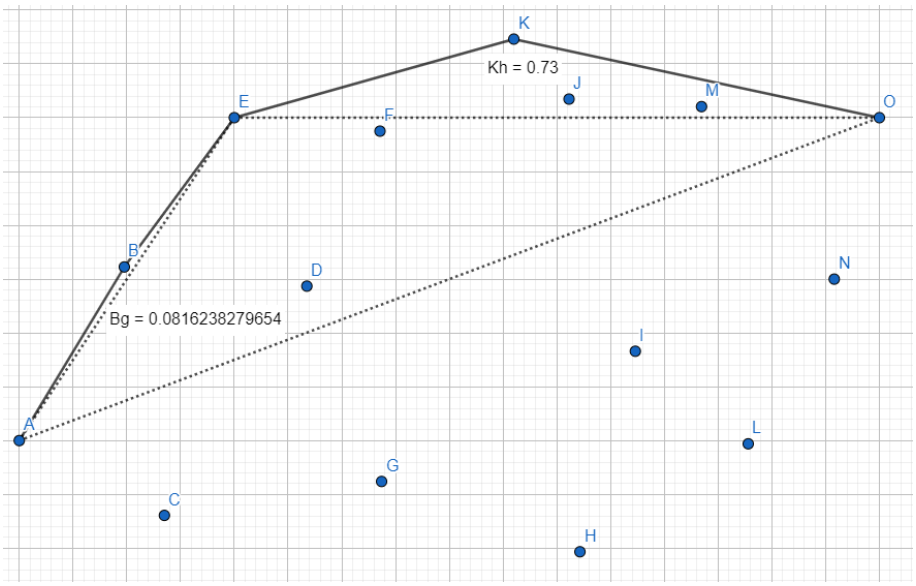
Dengan P_1 adalah titik A, P_2 adalah titik O, dan titik kandidat adalah (x_0,y_0). Lakukan iterasi untuk semua titik selain titik A dan titik O hingga mendapat titik dengan jarak

maksimum. Didapatkan titik E adalah titik dengan jarak terjauh dari garis penghubung titik A dan titik O.

6. Selanjutnya, serupa dengan langkah nomor 3, hubungkan titik A dengan titik E sebagai titik ekstrem yang baru dan hubungkan juga titik E dengan titik O. Penghubungan titik yang baru ini akan membentuk bagian titik yang baru, bagian kiri atas ($S_{1,1}$) dan bagian kanan atas ($S_{1,2}$). Ilustrasinya sebagai berikut.

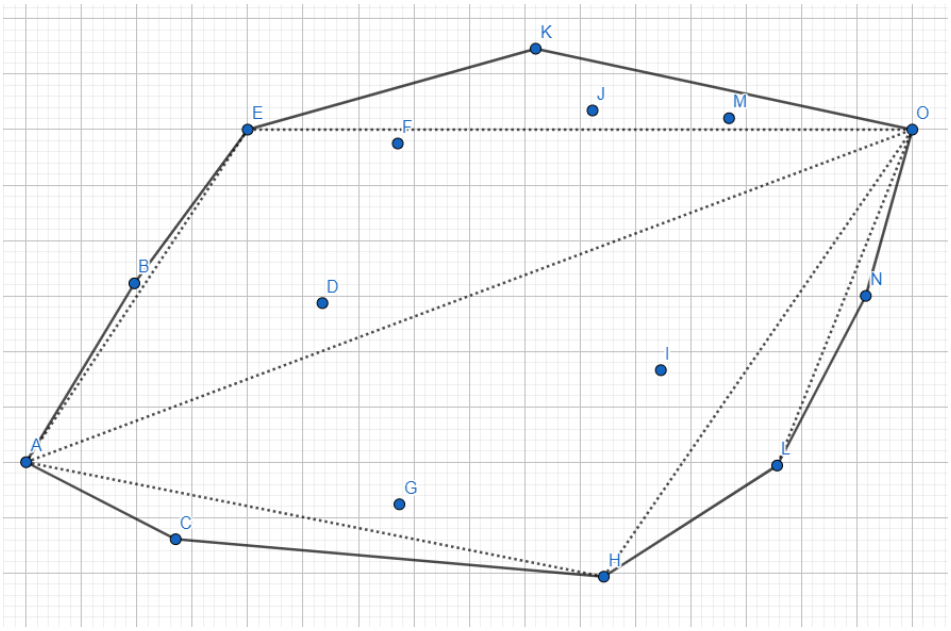


7. Pada bagian baru yang dibentuk, lakukan operasi yang serupa seperti langkah nomor 4. Operasi rekursif akan berhenti jika tidak ada bagian baru yang dibentuk. Apabila bagian baru tidak dapat dibentuk, maka garis penghubung titik dengan titik ekstrem yang baru menjadi pembentuk *convex hull*. Ilustrasinya sebagai berikut.



Daftarkan pasangan titik pembentuk *convex hull* ke dalam suatu himpunan baru.

8. Ulangi langkah yang sama untuk bagian bawah (S_2) hingga terbentuk *convex hull* yang sempurna. Prosesnya diilustrasikan sebagai berikut.



Garis putus-putus adalah garis pembagi bagian titik yang menghubungkan titik pembentuk *convex hull*.

Source Program

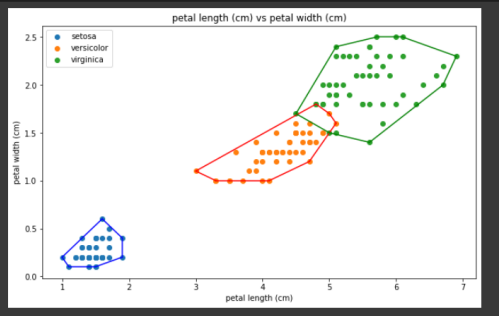
```
class myConvexHull:
  '''
  Convex hull dari himpunan titik S adalah himpunan convex terkecil
  (convex polygon) yang mengandung S
  Himpunan titik pada bidang planar disebut convex jika untuk sembarang dua titik
  pada bidang tersebut (misal p dan q), seluruh segmen garis yang berakhir di p dan
  q berada pada himpunan tersebut.
  Algoritma divide and conquer untuk menentukan convex hull
  disadur dari Bahan Kuliah IF2211 Stima - Algoritma Divide and Conquer Bagian 4
  Oleh: Nur Ulfa Maulidevi & Rinaldi Munir

  '''
  # ==== User defined-constructor dengan masukan arrayTitik =====
  # arrayTitik merupakan object ndarray 2 dimensi (bentukan dari library numpy)
  # atau dalam tipe data list (akan dilakukan validasi)
  def __init__(self, arrayTitik):
    # inisiasi atribut simplices dengan list kosong
    # simplices merupakan variabel yang menampung
    # pasangan indeks pada arrayTitik yang membentuk convex hull
    self.simplices = []
    # inisiasi atribut setOfConvexHull dengan list kosong
    # setOfConvexHull merupakan variabel yang menampung
    # pasangan titik yang membentuk convex hull
    self.setOfConvexHull = []
    # inisiasi atribut S, mengubah tipe data arrayTitik
    # dari object ndarray menjadi list
    self.S = self.fromArraytoList(arrayTitik)
    # inisiasi atribut originalS
    # menyimpan informasi kumpulan titik sebelum diurutkan
    self.originalS = self.S
    # memanggil method untuk mengurutkan kumpulan titik
    self.sortList()
    # validasi banyak titik
    # convex hull hanya dapat dibentuk oleh dua titik unik atau lebih
    if len(list(set(tuple(element) for element in self.S))) > 1:
      self.convexHullAwal()
    # else: len(self.S) <= 1
    # do nothing

  # ==== Kelompok Method Type Converter ====
  def fromArraytoList(self, array):
    # mengubah tipe object array menjadi tipe list
    if (type(array) == list):
        return array
    return np.ndarray.tolist(array)

  def sortList(self):
    # Kumpulan titik diurutkan berdasarkan nilai absis yang menaik
    # jika ada nilai absis yang sama
    # maka diurutkan dengan nilai ordinat yang menaik
    sortbyX = sorted(self.S, key=lambda x: x[1])
    sortbyY = sorted(sortbyX, key=lambda x: x[0])
    self.S = sortbyY

  # ==== Kelompok Method Getter ====
  def getElmtIndex(self, element):
    # mendapatkan indeks elemen pada list
    for i in range(len(self.originalS)):
      if self.originalS[i] == element :
        return i

  def getp1pn(self):
    # mendapatkan p1 dan pn
    # S merupakan kumpulan titik yang telah diurut
    # p1 = titik dengan absis paling minimum = elemen pertama S
    # pn = titik dengan absis paling maksimum = elemen terakhir S
    return self.S[0],self.S[len(self.S)-1]

  # ==== Kelompok Method Operasi Dasar ====
  def determinan(self,p1,p2,p3):
    # menghitung determinan matriks yang terbentuk dari tiga titik
    '''
    | x1 y1 1 |
    | x2 y2 1 |  = x1*y2 + x3*y1 + x2*y3 - x3*y2 - x2*y1 - x1*y3
    | x3 y3 1 |
    sumber : Bahan Kuliah IF2211 Stima - Algoritma Divide and Conquer Bagian 4
             Oleh: Nur Ulfa Maulidevi & Rinaldi Munir
    '''
    return p1[0] * p2[1] + p3[0] * p1[1] + p3[1] * p2[0] - p3[0] * p2[1] - p2[0] * p1[1] - p1[0] * p3[1]

  def jarakTitik(self,p1,p2,p3):
    # menghitung jarak dari titik ke garis yang dibentuk antara dua titik
    '''
    d(P1(x1,y1),P2(x2,y2),P3(x3,y3)) = abs((x2-x1)(y1-y3) - (x1-x3)(y2-y1)) / sqrt((x2-x1)^2 + (y2-y1)^2)
    sumber : https://en.wikipedia.org/wiki/Distance_from_a_point_to_a_line#Line_defined_by_two_points
    '''
    return abs((p2[0]-p1[0])*(p1[1]-p3[1]) - (p1[0]-p3[0])*(p2[1]-p1[1])) / ((p2[0]-p1[0])**2 + (p2[1]-p1[1])**2)**(0.5)
```

```
  # ==== Kelompok Method Operasi List ====
  def bagiSisi(self,p1,pn,S):
    # S merupakan himpunan titik
    # akan dibagi S ke dua bagian, bagian atas atau bagian bawah dari
    # garis yang dibentuk oleh titik p1 dan pn
    # S1 merupakan kumpulan titik bagian atas
    # S2 merupakan kumpulan titik bagian bawah
    S1 = []
    S2 = []
    # pembagian ditentukan melalui determinan yang terbentuk oleh tiga titik
    for p3 in S:
      if((p1 != p3) and (pn != p3)):
        detTitik = self.determinan(p1,pn, p3)
        if detTitik > 0 : # Jika determinan positif maka titik p3(x3,y3) berada dibagian atas
          S1.append(p3)
        elif detTitik < 0 : # Jika determinan negatif maka titik p3(x3,y3) berada dibagian bawah
          S2.append(p3)
        # else : determinan 0, abaikan titik (berada pada garis)
    return S1, S2

  def titikTerjauh(self,p1,pn,S):
    # mencari titik terjauh dari garis yang dibentuk oleh p1 dan pn
    jarakMaks = 0
    pmax = S[0]
    for p3 in S:
      tempJarak = self.jarakTitik(p1,pn,p3)
      if tempJarak > jarakMaks:
        jarakMaks = tempJarak
        pmax = p3
    return pmax
  # ==== Kelompok Method Operasi Utama ====
  def convexHullAwal(self):
    # mendapatkan p1 dan pn
    p1,pn = self.getp1pn()
    # membagi S menjadi dua bagian
    S1, S2 = self.bagiSisi(p1,pn,self.S)
    # Lakukan operasi convex hull untuk bagian atas
    self.convexHullAtas(p1,pn,S1)
    # Lakukan operasi convex hull untuk bagian bawah
    self.convexHullBawah(p1,pn,S2)

  def convexHullAtas(self,p1,pn,S):
    # membentuk convex hull di bagian atas secara rekursif
    # basis
    if len(S) == 0 :
      # jika tidak ada titik lagi pada S
      # maka p1 dan pn menjadi pembentuk convex hull
      self.appendPoints(p1,pn)
    # rekurens
    else:
      # mencari titik terjauh
      pmax = self.titikTerjauh(p1,pn,S)
      # mengambil sisi bagian kiri atas saja
      S11,_ = self.bagiSisi(p1,pmax,S)
      # mengambil sisi bagian kanan atas saja
      S12,_ = self.bagiSisi(pmax,pn,S)
      # Lakukan operasi convex hull untuk bagian kiri atas
      self.convexHullAtas(p1,pmax,S11)
      # Lakukan operasi convex hull untuk bagian kanan atas
      self.convexHullAtas(pmax,pn,S12)

  def convexHullBawah(self,p1,pn,S):
    # membentuk convex hull di bagian bawah secara rekursif
    # basis
    if len(S) == 0 :
      # jika tidak ada titik lagi pada S
      # maka p1 dan pn menjadi pembentuk convex hull
      self.appendPoints(p1,pn)
    # rekurens
    else:
      # mencari titik terjauh
      pmax = self.titikTerjauh(p1,pn,S)
      # mengambil sisi bagian kiri bawah saja
      _,S21 = self.bagiSisi(p1,pmax,S)
      # mengambil sisi bagian kanan bawah saja
      _,S22 = self.bagiSisi(pmax,pn,S)
      # Lakukan operasi convex hull untuk bagian kiri bawah
      self.convexHullBawah(p1,pmax,S21)
      # Lakukan operasi convex hull untuk bagian kanan bawah
      self.convexHullBawah(pmax,pn,S22)

  def appendPoints(self,p1,pn):
    # menambahkan pasangan (p1,pn) sebagai pembentuk convex hull
    # mencari indeks untuk titik p1 dan pn pada originalS
    indeksp1 = self.getElmtIndex(p1)
    indekspn = self.getElmtIndex(pn)
    self.simplices.append([indeksp1,indekspn])
    self.setOfConvexHull.append([p1,pn])
```

Screenshot Input dan *Output*

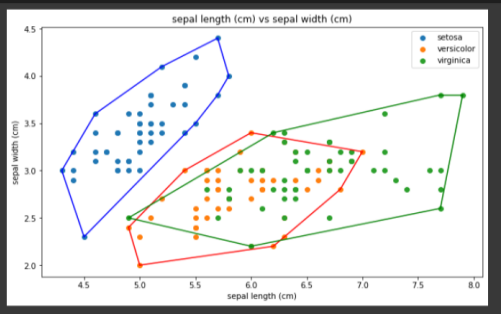
1. Dataset Iris : Petal Length vs Petal Width

```
data = datasets.load_iris()
df = createDataFrame(data)
visualisasiConvexHull(df,2,3)
```



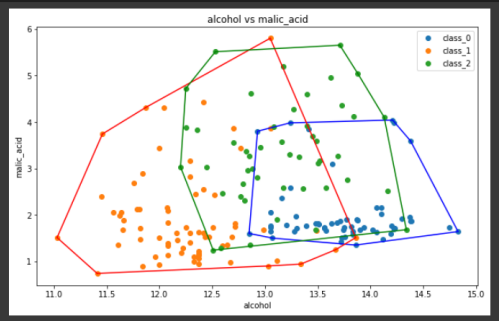
2. Dataset Iris : Sepal Length vs Sepal Width

```
data = datasets.load_iris()
df = createDataFrame(data)
visualisasiConvexHull(df,0,1)
```



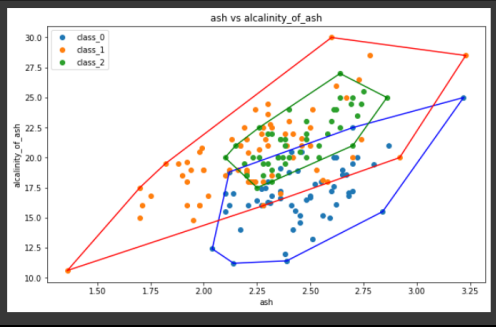
3. Dataset Wine : Alcohol vs Malic Acid

```
data = datasets.load_wine()
df = createDataFrame(data)
visualisasiConvexHull(df,0,1)
```



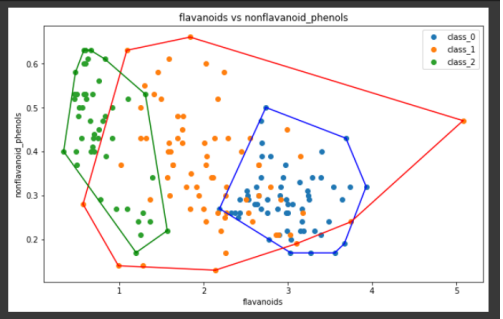
4. Dataset Wine : Ash vs Alcalinity of Ash

```
[7] data = datasets.load_wine()
    df = createDataFrame(data)
    visualisasiConvexHull(df,2,3)
```



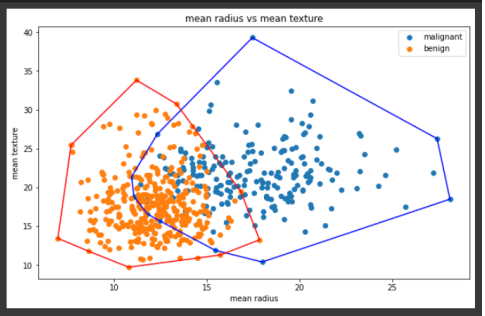
6. Dataset Wine: Flavanoids vs Non Flavanoid Phenols

```
[9] data = datasets.load_wine()
    df = createDataFrame(data)
    visualisasiConvexHull(df,6,7)
```



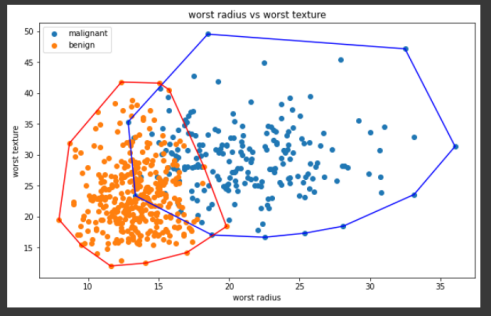
7. Dataset Breast Cancer: Mean Radius vs Mean Texture

```
data = datasets.load_breast_cancer()
df = createDataFrame(data)
visualisasiConvexHull(df,0,1)
```



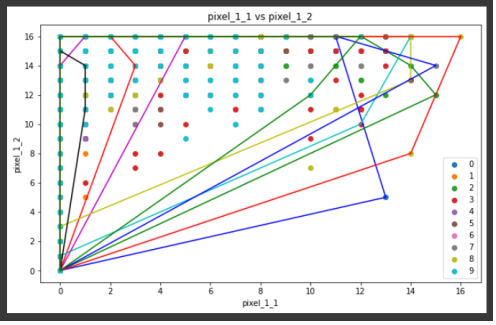
8. Dataset Breast Cancer: Worst Radius vs Worst Texture

```
data = datasets.load_breast_cancer()
df = createDataFrame(data)
visualisasiConvexHull(df,20,21)
```



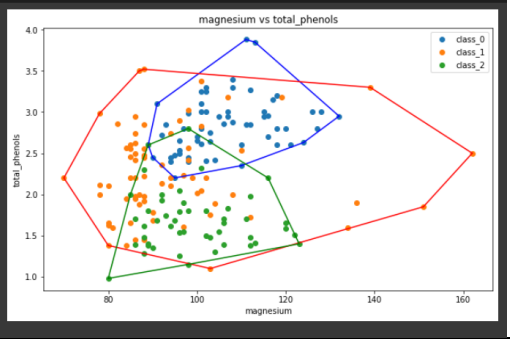
9. Dataset Digit: Pixel 1 1 vs Pixel 1 2

```
[12] data = datasets.load_digits()
     df = createDataFrame(data)
     visualisasiConvexHull(df,9,10)
```



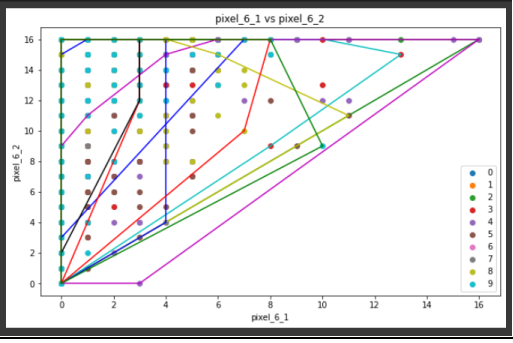
5. Dataset Wine : Magnesium vs Total Phenols

```
data = datasets.load_wine()
df = createDataFrame(data)
visualisasiConvexHull(df,4,5)
```



10. Dataset Digit: Pixel 6 1 vs Pixel 6 2

```
[13] data = datasets.load_digits()
     df = createDataFrame(data)
     visualisasiConvexHull(df,49,50)
```



Alamat *Repository* Kode Program

Github: https://github.com/dennisheraldi/Tucil2_13520139

Google Colab: https://colab.research.google.com/drive/1t3-ibSqPoemnNqudbzIqyqXd6nD-klyI?usp=sharing

agar lebih mudah, disarankan untuk mengakses Notebook secara langsung melalui Google Colab.

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.	✓	