**LAPORAN TUGAS KECIL 2**

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

Ditujukan untuk memenuhi salah satu tugas kecil mata kuliah IF2211 Strategi Algoritma (Stima) pada Semester II Tahun Akademik 2021/2022

Disusun oleh:

**Saul Sayers (K1)**  **13520094**



**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**

**2021**

1. **Algoritma *Divide and Conquer***

Algoritma *Divide and Conquer* merupakan pendekatan penyelesaian suatu permasalahan yang besar dengan cara memecah atau membaginya menjadi beberapa bagian kecil sehingga lebih mudah untuk diselesaikan. Kelebihan dari algoritma ini adalah mampu menyelesaikan masalah yang sulit/rumit yang masih cukup sulit untuk dipecahkan oleh algoritma biasa seperti *brute force*. Selain itu, algoritma ini juga lebih efisien untuk beberapa kasus tertentu dan dapat bekerja secara parallel untuk memaksimalkan penggunaan cache memory. Langkah – langkah dalam algoritma *Divide and Conquer* adalah sebagai berikut :

* Divide : Membagi sebuah masalah menjadi beberapa upa-masalah atau masalah yang lebih kecil
* Conquer : Menyelesaikan masing – masing upa-masalah (yang berupa basis dari rekursif) dan mendapatkan solusinya
* Combine : Menggabungkan solusi dari tiap masing – masing upa masalah agar mendapatkan solusi total dari solusi masalah yang awal.

Dalam Tugas Kecil ini, digunakan algoritma *Divide and Conquer* dalam implementasi penyelesaian Convex Hull untuk visualisasi dataset linear terpisah. Sebelum menerapkan algoritma *Divide and Conquer*, program akan mengambil dataset dari library scikit-learn (sklearn). Dataset sklearn yang dimuat dalam program ini adalah Iris, Wine, dan Breast Cancer. Dataset tersebut merupakan dataset berlabel yang ditandai oleh atribut target untuk tiap tabelnya. Program akan memanfaatkan library pandas untuk menyimpan dataset tersebut dalam sebuah dataframe. Kemudian, dari dataset tersebut kita akan mengambil dua atribut saja untuk dijadikan absis dan ordinat sehingga membentuk scatter plot untuk tiap labelnya. Program kemudian akan menentukan Convex Hull untuk tiap labelnya. Berikut adalah langkah algoritma *Divide and Conquer* yang telah saya buat untuk mencari Convex Hull scatter plotnya.

Pertama, program akan melakukan sorting terlebih dahulu terhadap absis secara menaik, kemudian secara ordinat secara menaik juga apabila memiliki absis yang sama agar mempermudah pemilihan koordinat. Kemudian, program memilih koordinat pertama pada dataset (koordinat paling kiri) P1 (x1, y1) serta koordinat terakhir pada dataset (koordinat paling kanan) Pn (xn, yn). Koordinat P1 dan Pn  dicatat oleh program sebagai salah dua koordinat Convex Hull karena berada di paling luar. Kemudian, program akan membagi semua koordinat yang berada pada dataset menjadi 2 bagian yakni bagian pertama S1 yang berada di kiri garis P1Pn  dan bagian kedua S2 yang berada di kanan garis P1Pn. Untuk titik yang berada tepat di garis P1Pn dapat diabaikan karena tidak termasuk sebagai koordinat Convex Hull.

Kedua, program akan menerapkan algoritma *Divide and Conquer* untuk masing – masing S1 dan S2 pada tahap ini. Program akan mencari titik terjauh Pmax terhadap garis P1Pn untuk masing – masing ruang kemudian mencatatnya sebagai salah satu koordinat Convex Hull. Kemudian, program akan mempartisi lagi tiap ruang menjadi segitiga P1PnPmax. Titik terjauh dihitung dengan cara mencari koordinat tersebut yang memaksimalkan sudut P1PnPmax. Pencarian titik terjauh dilakukan secara traversal dan perhitungan sudut tiap koordinatnya memanfaatkan library numpy. Untuk koordinat yang berada dalam segitiga tersebut akan diabaikan oleh program karena sudah pasti berada di dalam dan tidak mungkin menjadi koordinat Convex Hull. Di sini, program akan melakukan *divide* lagi secara rekursif untuk koordinat yang berada di luar sisi P1Pmax dan PmaxPn dan mencari titik terjauh terhadap garis tersebut. Divide akan dilakukan secara terus menerus hingga sisi tersebut tidak bisa dilakukan *divide* lagi yakni apabila sudah tidak ada lagi koordinat yang berada di luar sisi tersebut. Kondisi ini menjadi basis dari rekursi dan pada tahap ini ruang sudah di*conquer.*

Ketiga, program akan melakukan *combine* dari semua solusi yang didapat tadi. Semua koordinat yang telah tercatat sebagai titik terjauh terhadap garis awal merupakan koordinat Convex Hull. Semua koordinat Convex Hull untuk tiap partisi dan tiap ruang tersebut akan disimpan pada suatu array dan akan dikembalikan oleh program.

Keempat, program akan menampilkan scatter plot dari semua koordinat, di mana koordinat Convex hull akan ditandai dengan sebuah garis berwarna sama yang menghubungkan mereka. Program memanfaatkan library matplotlib untuk menampilkan plot tersebut. Garis yang menghubungkan koordinat Convex Hull dibentuk dengan cara melakukan sorting kembali array Convex Hull terhadap absis kemudian mempartisi koordinat Convex Hull menjadi 2 bagian yakni sebelah atas garis P1Pn  dan sebelah bawah P1Pn, lalu untuk masing – masing ruangan tersebut akan ditarik garis secara urut berdasarkan absis koordinatnya.

1. ***Source Program***

Source code program ditulis dalam bahasa pemrograman Python dengan menggunakan library numpy, pandas, matplotlib, dan sklearn. Source code program terbungkus menjadi 2 file utama, yakni myConvexHull.py dan main.py

1. **myConvexHull.py** merupakan library / modul untuk algoritma Convex Hull yang telah saya buat. Note bahwa penjelasan tiap fungsi sudah disertakan pada source code dalam bentuk komentar

|  |
| --- |
| # NAMA  : SAUL SAYERS  # NIM   : 13520094  # KELAS : K-01 STRATEGI ALGORITMA  # MERUPAKAN FILE LIBRARY MYCONVEXHULL UNTUK TUCIL 2 STRATEGI ALGORITMA  import numpy as np  from numpy.linalg import norm  # Fungsi berikut untuk menentukan lokasi dari sebuah titik p3 dengan cara menghitung determinan dari matriks yang tersusun.  # Apabila positif maka di sebelah kanan garis, negatif di sebelah kiri garis, dan 0 tepat pada garis sehingga diabaikan  def periksaLokasi(p1,p2,p3):      x1 = p1[0]      x2 = p2[0]      x3 = p3[0]      y1 = p1[1]      y2 = p2[1]      y3 = p3[1]      hasil = x1\*y2 + x3\*y1 + x2\*y3 - x3\*y2 - x2\*y1 - x1\*y3 # Untuk menghitung determinannya secara manual      return hasil  # Fungsi berikut untuk menghitung jarak sebuah titik p3 terhadap garis yang terbentuk dari p1 dan p2  def hitungJarak(p1, p2, p3):      a = np.array(p1)      b = np.array(p2) # Konversi dalam bentuk numpy      c = np.array(p3)      jarak = np.abs(np.cross(b-a, a-c)) / norm(b-a)        return jarak  # Fungsi berikut untuk menghitung sudut yang diapit, yakni sudut P3-P1-P2  def hitungSudut(p1,p2,p3):      if ((p1[0] == p3[0] and p1[1] == p3[1]) or (p2[0] == p3[0] and p2[1] == p3[1]) ) : # KASUS TITIK YANG DIPERIKSA ADALAH TITIK ITU SENDIRI          return 0      else:          a = np.array(p2)          b = np.array(p3) # Konversi dalam bentuk numpy          c = np.array(p1)            ca = a - c # Mendapatkan vektornya          cb = b - c            hasilkalidot = np.dot(ca,cb) # Melakukan perkalian dot          panjangCA = norm(ca)          panjangCB = norm(cb) # Mendapatkan panjang vectornya            sudutDalamCos = hasilkalidot/(panjangCA\*panjangCB) # Perkalian dot untuk mendapatkan sudutnya          sudutDalamRadian = np.arccos(sudutDalamCos) # Mengkonversi dari cos menjadi radian          return sudutDalamRadian  # Fungsi berikut untuk menentukan sebuah titik terjauh dari dalam variabel bucket terhadap titik p1 dan p2  # Titik terjauh dihitung menggunakan fungsi hitungSudut.  def cariTitikTerjauh(bucket,p1,p2) :      if len(bucket) == 1 :          return bucket[0] # Apabila bucket hanya berisi satu titik, maka itu adalah titik terjauh      else:          sudutterjauh = hitungSudut(p1,p2,bucket[0])          pterjauh = bucket[0]          for i in range (1,len(bucket)): # Pencarian sudut terbesar secara traversal dari awal hingga akhir              temp = hitungSudut(p1,p2,bucket[i])              if (temp > sudutterjauh) :                  sudutterjauh = temp                  pterjauh = bucket[i]          return pterjauh  # Fungsi berikut untuk melakukan sorting dari koordinat x secara menaik, jika sama maka dari koordinat y  def sortKoordinat(arr):      for i in range (len(arr)-1) :          for j in range (len(arr)-i-1):              if (arr[j][0] > arr[j+1][0] or (arr[j][0] == arr[j+1][0] and arr[j][1] > arr[j+1][1])):                  tempx = arr[j][0]                  arr[j][0] = arr[j+1][0]                  arr[j+1][0] = tempx                    tempy = arr[j][1]                  arr[j][1] = arr[j+1][1]                  arr[j+1][1] = tempy  # Fungsi berikut untuk menambahkan sebuah koordinat ke dalam array hasilakhir HullVertices  # Perlu diperhatikan bahwa hullVertices memiliki elemen unik sehingga dihandle agar menambahkan apabila belum ada saja  def tambahSolusi(bucket,p):      sudahada = False      for i in range (len(bucket)):          if (bucket[i][0] == p[0] and bucket[i][1] == p[1]): # Mengecek apakah koordinat tersebut sudah ada apa belum              sudahada = True      if (not sudahada) :          bucket.append(p) # Jika belum, langsung tambahkan saja    # KEDUA FUNGSI DI BAWAH UNTUK MELAKUKAN PARTISI SEBELAH KIRI GARIS DAN SEBELAH KANAN GARIS  def partisiKiri(bucket,pawal,pakhir):      arrHasil = []      for i in range (len(bucket)):          determinan = periksaLokasi(pawal,pakhir,bucket[i])          if (determinan) > 10\*\*-6 : # Untuk ngehandle rounding error yang kurang akurat              tambahSolusi(arrHasil, bucket[i])      return arrHasil  def partisiKanan(bucket,pawal,pakhir):      arrHasil = []      for i in range (len(bucket)):          determinan = periksaLokasi(pawal,pakhir,bucket[i])          if (determinan) < -1\*(10\*\*-6): # Untuk ngehandle rounding error yang kurang akurat              tambahSolusi(arrHasil, bucket[i])      return arrHasil  # Fungsi ini untuk menerapkan tahap rekursif dari algoritma convexHull. Fungsi akan melakukan divide and conquer pada tahap ini  # Basis dari fungsi ini adalah:  # 1. Titik pada sebuah ruangan Kosong, maka do nothing  # 2. Hanya ada satu titik pada ruangan sehingga titik tersebut juga termasuk convexHull sehingga tambahkan pada arrayhasil  # Tahap rekursi dari fungsi ini adalah :  # Tiap ruangan akan dicari terlebih dahulu titik terjauh yang merupakan titik convexhull juga sehingga masukkan ke array  # kemudian mempartisi nya menjadi dua ruangan dan conquer masing masing hingga mencapai basis  def convexHullRecursive(bucket,pawal,pakhir,arrHasil):      # if len = 0 then do nothing, basis pertama      if len(bucket) == 1:          tambahSolusi(arrHasil, bucket[0]) # Panjang array = 1, basis kedua      elif len(bucket) > 1 :          terjauh = cariTitikTerjauh(bucket,pawal,pakhir) # Untuk mencari titik terjauh          tambahSolusi(arrHasil, terjauh) # Untuk menambahkan titik terjauh tadi ke array hasil akhir          arrKiri = partisiKiri(bucket,pawal,terjauh)          arrKanan = partisiKiri(bucket,terjauh,pakhir) # Untuk mempartisikan array kiri dan kanan          convexHullRecursive(arrKiri,pawal,terjauh,arrHasil)          convexHullRecursive(arrKanan,terjauh,pakhir,arrHasil) # Untuk melanjutkan array kiri dan kanan ke tahap rekursi  # Fungsi berikut merupakan tahap pertama dari algoritma convexHull  # Fungsi ini akan melakukan sorting terlebih dahulu terhadap absis, kemudian terhadap ordinat  # Tarik garis dari koordinat paling kiri dan paling kanan untuk membagi menjadi 2 ruangan. Kedua titik tersebut termasuk Hull vertices  def myConvexHull(bucket) :      sortKoordinat(bucket) # Untuk melakukan sorting      pawal = bucket[0]      pakhir = bucket[len(bucket)-1] # Mencatat koordnat paling kiri dan kanan      arrKiri = partisiKiri(bucket,pawal,pakhir)      arrKanan = partisiKanan(bucket,pawal,pakhir) # Melakukan partisi ruangan kiri dan kanan      hullVertices = [pawal,pakhir] # Memasukkan koordinat terkiri dan terkanan tadi termasuk Hull Vertices      convexHullRecursive(arrKiri,pawal,pakhir,hullVertices)      convexHullRecursive(arrKanan,pakhir,pawal,hullVertices) # Untuk memasukkan tiap ruangan ke tahap rekursif untuk diconquer      sortKoordinat(hullVertices) # Untuk melakukan sorting agar hasil akhir tetap terurut agar mempermudah plotting      return hullVertices  # Fungsi tersebut untuk mensplit koordinat dari Hull Vertices menjadi array absis dan array ordinat  # Fungsi ini ditujukan untuk mempermudah plotting dari hull vertices  def splitXY(bucket):      arrKiri = partisiKiri(bucket,bucket[0],bucket[-1])      arrKanan = partisiKanan(bucket,bucket[0],bucket[-1]) # Untuk melakukan partisi tiap array      sumbuxkiri = [bucket[0][0]]      sumbuykiri = [bucket[0][1]] # Untuk mengisi elemen pertama dari hull vertices      sumbuxkanan = [bucket[0][0]]      sumbuykanan = [bucket[0][1]]      for koordinat in arrKiri :          sumbuxkiri.append(koordinat[0])          sumbuykiri.append(koordinat[1]) # Untuk menambahkan tiap absis/ordinat ke array      for koordinat in arrKanan :          sumbuxkanan.append(koordinat[0])          sumbuykanan.append(koordinat[1])      sumbuxkiri.append(bucket[-1][0])      sumbuxkanan.append(bucket[-1][0]) # Untuk mengisi elemen terakhir dari hull vertices      sumbuykiri.append(bucket[-1][1])      sumbuykanan.append(bucket[-1][1])      hasilakhir = [sumbuxkiri,sumbuykiri,sumbuxkanan,sumbuykanan] # Untuk membungkus semua array menjadi satu array      return hasilakhir |

1. **main.py** merupakan file yang dijalankan untuk mendapatkan plot Convex Hull.

|  |
| --- |
| # NAMA  : SAUL SAYERS  # NIM   : 13520094  # KELAS : K-01 STRATEGI ALGORITMA  # MERUPAKAN FILE MAIN MYCONVEXHULL UNTUK TUCIL 2 STRATEGI ALGORITMA  import pandas as pd  import matplotlib.pyplot as plt  from sklearn import datasets  import myConvexHull as mch  print("SELAMAT DATANG DI PROGRAM CONVEXHULL SAUL SAYERS :D")  print("--------------------------------------------------------")  kelar = False  while (not kelar) :      # Proses pemilihan dataset      print("Silahkan pilih dataset yang ingin digunakan")      print("Note: pemilihan atribut cukup dengan mengetikkan angkanya")      print()      print("Daftar dataset: ")      print("1. Dataset Iris")      print("2. Dataset Wine")      print("3. Dataset Breast Cancer")      print()      pilihdataset = int(input("Silahkan pilih dataset: "))      while (pilihdataset < 1) or (pilihdataset > 3):          pilihdataset = int(input("input tidak valid, silahkan input ulang dataset: "))      if pilihdataset == 1 :          data = datasets.load\_iris()      elif pilihdataset == 2:          data = datasets.load\_wine()      elif pilihdataset == 3:          data = datasets.load\_breast\_cancer()      # Pembuatan dataframe      df = pd.DataFrame(data.data, columns=data.feature\_names)      df['Target'] = pd.DataFrame(data.target)      df.head()      # Proses pemilihan atribut      print("Daftar atribut: ")      for i in range(len(data.feature\_names)):          print(str(i + 1) + ".", data.feature\_names[i])      print()      atribut1 = int(input("Silahkan pilih atribut yang ingin dipilih sebagai sumbu X: "))      while (atribut1 < 1) or (atribut1 > len(data.feature\_names)):          atribut1 = int(input("input tidak valid, silahkan input ulang atribut sumbu X: "))      print()        atribut2 = int(input("Silahkan pilih atribut yang ingin dipilih sebagai sumbu Y: "))      while (atribut2 < 1) or (atribut2 > len(data.feature\_names)) or (atribut1 == atribut2):          if (atribut1 == atribut2):              atribut2 = int(input("Atribut sumbu y tidak boleh sama, silahkan input ulang: "))          else:              atribut2 = int(input("input tidak valid, silahkan input ulang atribut sumbu Y: "))      # Visualisasi hasil ConvexHull      plt.figure(figsize = (10, 6))      colors = ['b','r','g']      plt.title(data.feature\_names[atribut1 -1] + " vs " + data.feature\_names[atribut2 -1])      plt.xlabel(data.feature\_names[atribut1 -1])      plt.ylabel(data.feature\_names[atribut2 -1])      for i in range(len(data.target\_names)) :          bucket = df[df['Target'] == i]          bucket = bucket.iloc[:,[atribut1 -1,atribut2 -1]].values          hull = mch.myConvexHull(bucket)          plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target\_names[i])          kumpulansisi = mch.splitXY(hull)          plt.plot(kumpulansisi[0], kumpulansisi[1], colors[i])          plt.plot(kumpulansisi[2],kumpulansisi[3], colors[i])      print()      print("Berikut adalah grafik Convex Hull nya: ")      print()      plt.show()        #Opsi save dataset      print("Apakah anda ingin save graf ConvexHull tersebut? ")      inginsave = input("Ketik y untuk iya atau n untuk tidak (defaultnya n): ")      if inginsave == "y" :          namafile = input("Silahkan input namafile (tanpa .png): ")          plt.title(data.feature\_names[atribut1 -1] + " vs " + data.feature\_names[atribut2 -1])          plt.xlabel(data.feature\_names[atribut1 -1])          plt.ylabel(data.feature\_names[atribut2 -1])          for i in range(len(data.target\_names)) :              bucket = df[df['Target'] == i]              bucket = bucket.iloc[:,[atribut1 -1,atribut2 -1]].values              hull = mch.myConvexHull(bucket)              plt.scatter(bucket[:, 0], bucket[:, 1], label=data.target\_names[i])              kumpulansisi = mch.splitXY(hull)              plt.plot(kumpulansisi[0], kumpulansisi[1], colors[i])              plt.plot(kumpulansisi[2],kumpulansisi[3], colors[i])          plt.savefig("./test/"+namafile+".png")          plt.clf()        # Opsi mengakhiri program atau tidak      print()      print("Apakah anda ingin solve Convex Hull dataset lain? ")      selesai = input("Ketik y untuk iya atau n untuk tidak (defaultnya n): ")      if (selesai != "y") :          kelar = True    print("Terimakasih telah menggunakan program saya :D") |

1. **Screenshots *input* dan *output***
2. **Dataset Iris**

Pemilihan dataset :

Text

Description automatically generated

Gambar 3.1.1 Pemilihan dataset Iris

Pemilihan atribut :

* sepal length (cm) vs sepal width (cm)

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Gambar 3.1.2 atribut sepal length vs sepal width

* petal length (cm) vs petal width (cm)

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Gambar 3.1.3 atribut petal length vs petal width

* sepal length (cm) vs petal length (cm)

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Gambar 3.1.4 atribut sepal length vs petal length

* sepal width (cm) vs petal width (cm)

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Gambar 3.1.5 atribut sepal width vs petal width

1. **Dataset Iris**

Pemilihan dataset :

Text

Description automatically generated

Gambar 3.2.1 Pemilihan dataset Wine

Pemilihan atribut :

* Alcohol vs malic\_acid

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Gambar 3.2.2 atribut alcohol vs malic\_acid

* Ash vs alcalinity\_of\_ash

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Gambar 3.2.3 atribut ash vs alcalinity of ash

* Magnesium vs total\_phenols

Text

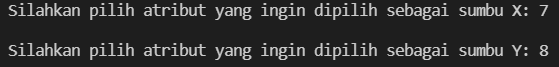
Description automatically generated

Chart, radar chart

Description automatically generated

Gambar 3.2.4 atribut magnesium vs total phenols

* flavanoids vs nonflavanoid\_phenols

 Chart, radar chart

Description automatically generated

Gambar 3.2.5 atribut flavanoids vs nonflavanoid phenols

* color\_intensity vs hue

Text

Description automatically generated

Chart

Description automatically generated

Gambar 3.2.5 atribut flavanoids vs nonflavanoid phenols

* proanthocyanins vs proline

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Gambar 3.2.6 atribut proanthocyanin vs proline

1. **Dataset Breast Cancer**

Pemilihan dataset :

Text

Description automatically generated

Gambar 3.3.1 Pemilihan dataset Breast Cancer

Pemilihan atribut :

* Mean radius vs radius error

Text

Description automatically generated

Chart, scatter chart

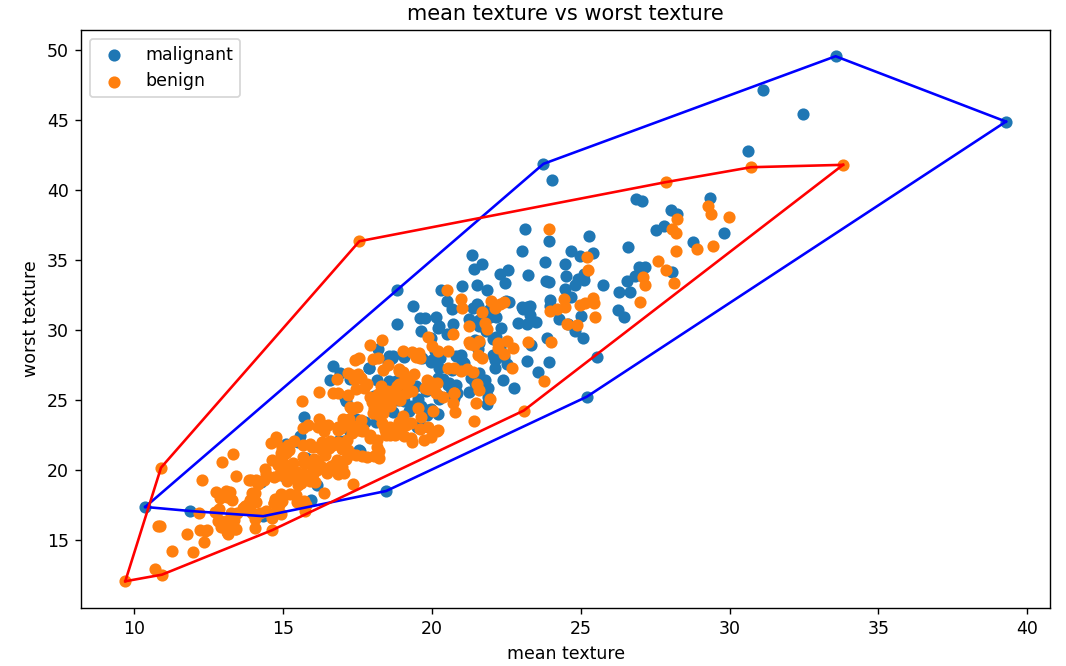
Description automatically generated

Gambar 3.3.2 Atribut mean radius vs radius error

* Mean texture vs worst texture

Text

Description automatically generated



Gambar 3.3.3 Atribut mean texture vs worst texture

* Perimeter error vs worst perimeter

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Gambar 3.3.4 atribut perimeter error vs worst perimeter

* Area error vs mean area

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Gambar 3.3.5 atribut area error vs mean area

* Worst smoothness vs mean smoothness

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Gambar 3.3.6 atribut worst smoothness vs mean smoothness

* Worst compactness vs compactness error

Text

Description automatically generated

Chart, scatter chart

Description automatically generated

Gambar 3.3.7 atribut worst compactness vs compactness error

1. **Link to Repository (Drive Source Code)**

<https://github.com/saulsayerz/Tucil2_13520094>

1. **Tabel Checklist**

Table

Description automatically generated