Nama = Luthfi Afrizal Ardhani

NIM = 175150218113022

Kelas = FILKOM 5A

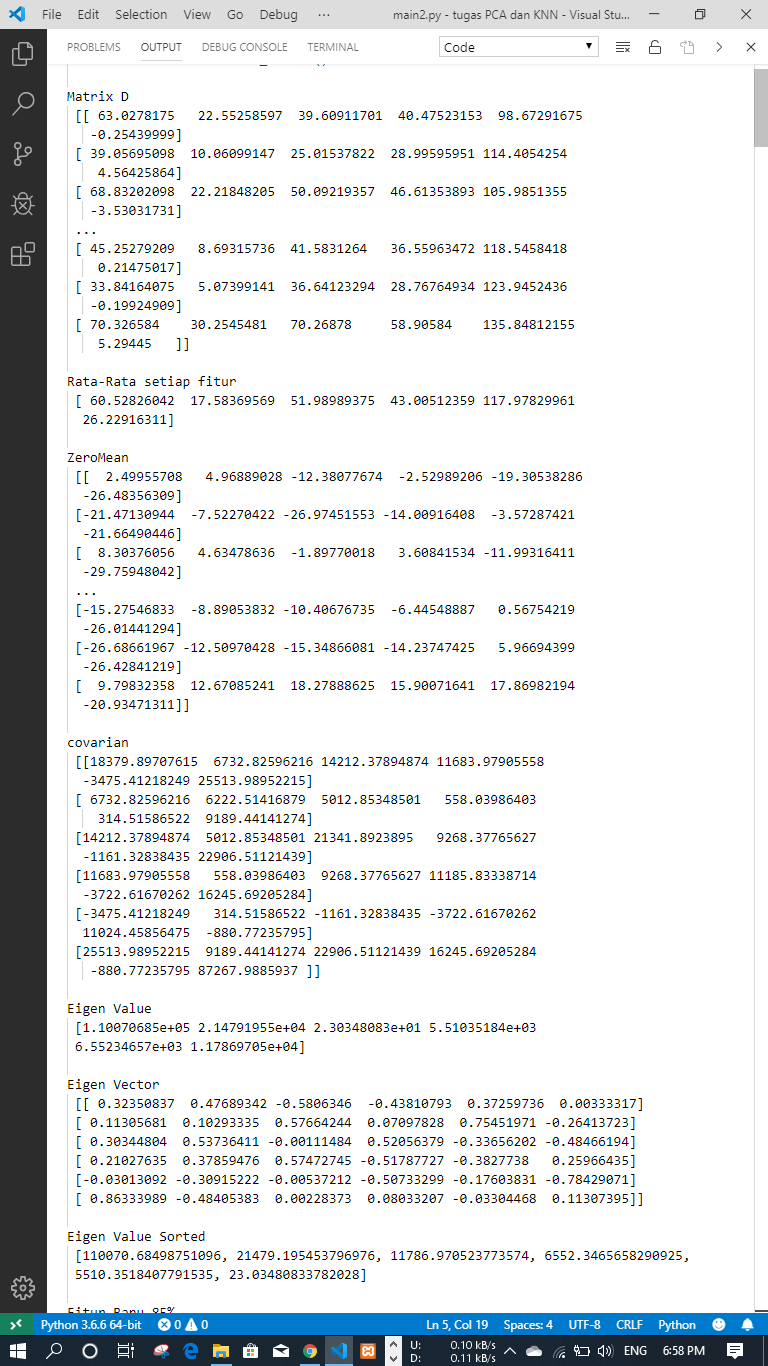
**Penjelasan Dataset**

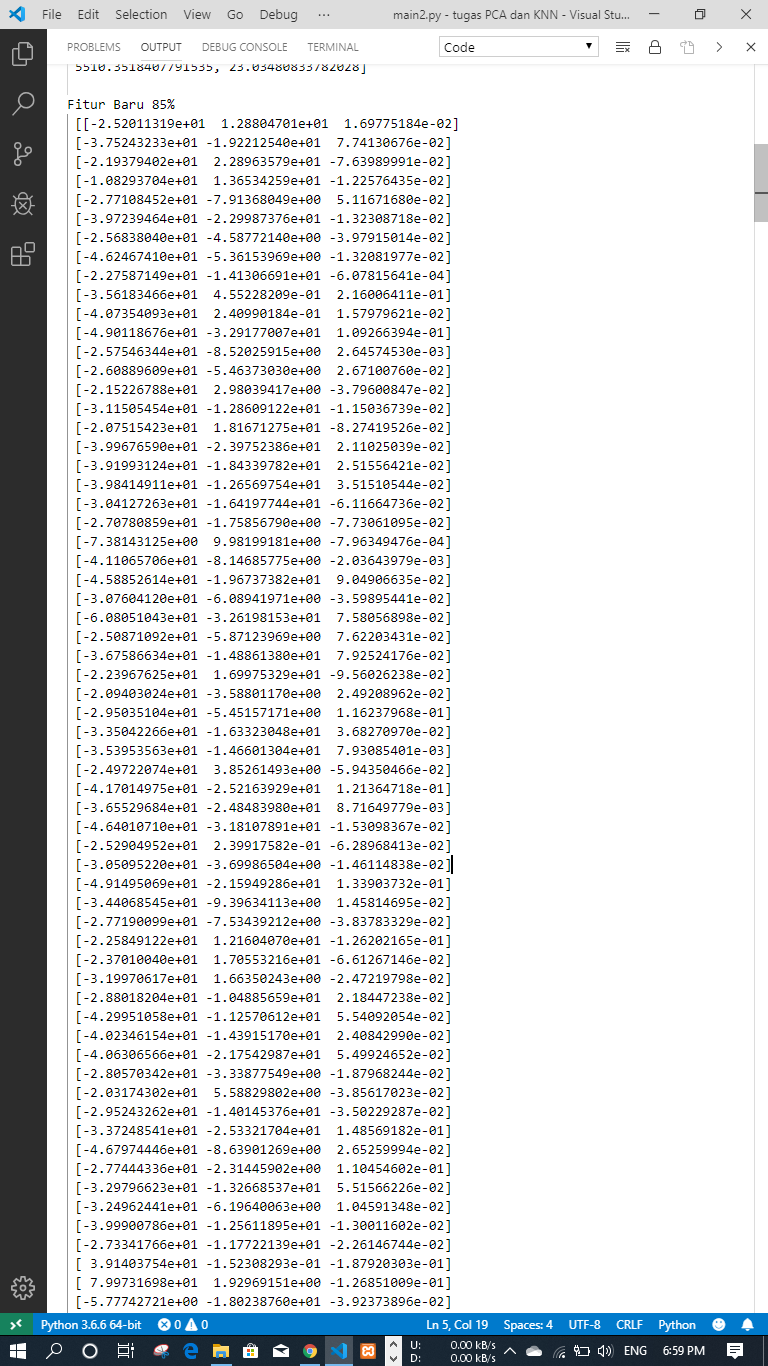
Dataset tersebut digunakan untuk mendiagnosis seseorang apakah mengidap penyakit hernia, spondylolisthesis, atau normal. Diagnosis tersebut bisa dijadikan nilai klasifikasi / kelas dalam menentukan data testing yang akan dilakukan diagnosis. Terdapat 6 fitur yaitu pelvic\_incindence, pelvic\_tilt, lumbar\_lordiosis\_angle, sacral\_slope, pelvic\_radius, degree\_spondylolisthesis. Dan terdapat 1 kolom class yang berisi hasil klasifikasi setiap data. Dan di baris terakhir dari data set terdapat data testing yang belum memiliki kelas, dimana kelas tersebut yang akan dicari pada program ini.

**Penjelasan Program**

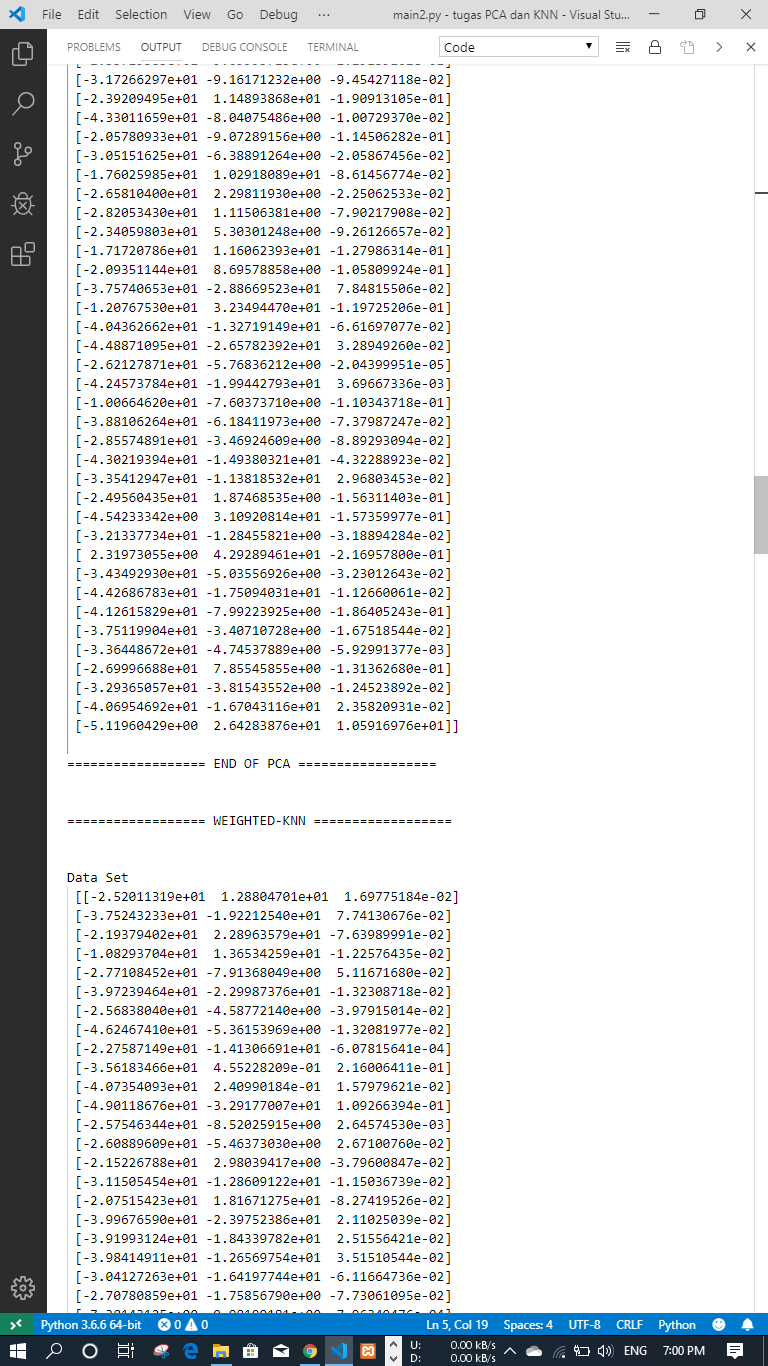
|  |  |
| --- | --- |
|  |  |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148 | """  Code by Luthfi A. A.  """  import pandas as pd  import numpy as np  from scipy import stats  from sklearn.metrics import classification\_report  #=========== Membaca Excel ==========#  data = pd.read\_excel (r'data2.xlsx')  #=========== Membaca Setiap Baris ==========#  PelvicIncidence = pd.DataFrame(data, columns= ['pelvic\_incidence'])  PelvicTilt = pd.DataFrame(data, columns= ['pelvic\_tilt'])  LumbarLordosisAngle = pd.DataFrame(data, columns= ['lumbar\_lordosis\_angle'])  SacralSlope = pd.DataFrame(data, columns= ['sacral\_slope'])  PelvicRadius = pd.DataFrame(data, columns= ['pelvic\_radius'])  DegreeSpondylolisthesis = pd.DataFrame(data, columns= ['degree\_spondylolisthesis'])  Class = pd.DataFrame(data, columns= ['class'])  matrixClassa = Class.as\_matrix()  #=========== Value bawah sendiri tidak dipanggil karena berupa data testing dan  kelasnya dicari ==========#  matrixClass = matrixClassa[0:len(Class)-1]  #=========== Mendapatkan D ==========#  matrixD = np.hstack((PelvicIncidence, PelvicTilt, LumbarLordosisAngle,  SacralSlope, PelvicRadius, DegreeSpondylolisthesis))  print ("\nMatrix D \n", matrixD)  #=========== Mencari Mean Setiap Fitur ==========#  meanPelvicIncidence = float(PelvicIncidence.mean())  meanPelvicTilt = float(PelvicTilt.mean())  meanLumbarLordosisAngle = float(LumbarLordosisAngle.mean())  meanSacralSlope = float(SacralSlope.mean())  meanPelvicRadius = float(PelvicRadius.mean())  meanDegreeSpondylolisthesis = float(DegreeSpondylolisthesis.mean())  average = np.hstack((meanPelvicIncidence, meanPelvicTilt, meanLumbarLordosisAngle,  meanSacralSlope, meanPelvicRadius, meanDegreeSpondylolisthesis))  print ("\nRata-Rata setiap fitur\n", average)  #=========== Mencari ZeroMean ==========#  zeroMean = np.subtract(matrixD, average)  print("\nZeroMean\n",zeroMean)  #=========== Menghitung Covarian ==========#  n = len(zeroMean[0])  covarian = 1/(n-1)\*(np.transpose(zeroMean).dot(zeroMean))  print("\ncovarian\n",covarian)  #=========== Menghitung Nilai Eigen dan Eigen Vector ==========#  w, v = np.linalg.eig(covarian)  print("\nEigen Value\n",w) # w = eigen value  print("\nEigen Vector\n",v) #v = eigen vector  #=========== Mengurutkan Eigen Value ==========#  wSort = sorted(w, reverse = True)  print("\nEigen Value Sorted\n",wSort)  #=========== Mempertahankan 85% data ==========#  i=0  lamb=0  for i in range(len(w)):      lamb += wSort[i]      i += 1  keep = (85/100) \* lamb  i2 = 0  j2 = 0  for i2 in range(len(w)):      j2 += wSort[i2]      i2 += 1      if j2 > keep:          break  newEgVec = v[0:len(v),0:i2]  fiturBaru = np.transpose(np.transpose(newEgVec).dot(np.transpose(zeroMean)))  print("\nFitur Baru 85%\n",fiturBaru)  print("\n================== END OF PCA ==================\n")  print("\n================== WEIGHTED-KNN ==================\n")  #=========== Memisahkan antara data testing dan data set dari data PCA ==========#  dataSet = fiturBaru[0:len(fiturBaru)-1,:]  dataTesting = fiturBaru[len(fiturBaru)-1:,:]  print("\nData Set\n",dataSet)  print("\nData Testing\n",dataTesting)  #=========== Perhitungan Eucledian ==========#  hitungSetTes  = np.subtract(dataTesting, dataSet)  powerSetTes = np.power(hitungSetTes, 2)  splitSetTes = np.hsplit(powerSetTes, len(powerSetTes[0]))  addSetTes = 0  for i in range(len(powerSetTes[0])):      addSetTes =+ splitSetTes[i]  sqrtSetTes = np.sqrt(addSetTes)  balikan = np.hstack((sqrtSetTes, matrixClass))  print("\nJarak dari data testing\n",balikan)  #=========== Weighted-KNN ==========#  hernia = 0  spondylolisthesis = 0  normal = 0  for i in range(len(balikan)):      if balikan[i,len(balikan[0])-1] == 'Hernia':          hernia =+ (1/np.power(balikan[i,0],2))      elif balikan[i,len(balikan[0])-1] == 'Spondylolisthesis':          spondylolisthesis =+ (1/np.power(balikan[i,0],2))      else:          normal =+ (1/np.power(balikan[i,0],2))  print("\nVote Hernia            = ", hernia)  print("Vote Spondylolisthesis = ", spondylolisthesis)  print("Vote Normal            = ", normal)  #=========== Voting untuk menentukan kelas ==========#  hasil = max(hernia, spondylolisthesis, normal)  if hasil == hernia:      predicted = ["hernia"]  elif hasil == spondylolisthesis:      predicted = ["spondylolisthesis"]  else:      predicted = ["normal"]  print("\nHASIL VOTE = ", predicted)  print("\n================== END OF WEIGHTED-KNN ==================\n")  print("\n================== CONFUSION MATRIX ==================\n")  #=========== Mencari Confusion Matrix ==========#  dataUji = 20 #Baris ke-n dari data excel#  if balikan[dataUji,len(balikan[0])-1] == 'Hernia':      expected = ["hernia"]  elif balikan[dataUji,len(balikan[0])-1] == 'Spondylolisthesis':      expected = ["spondylolisthesis"]  elif balikan[dataUji,len(balikan[0])-1] == 'Normal':      expected = ["normal"]  confussionMatrix = classification\_report(expected, predicted)  print("\nexpected = ", expected, "\npredicted = ", predicted)  print(confussionMatrix) |

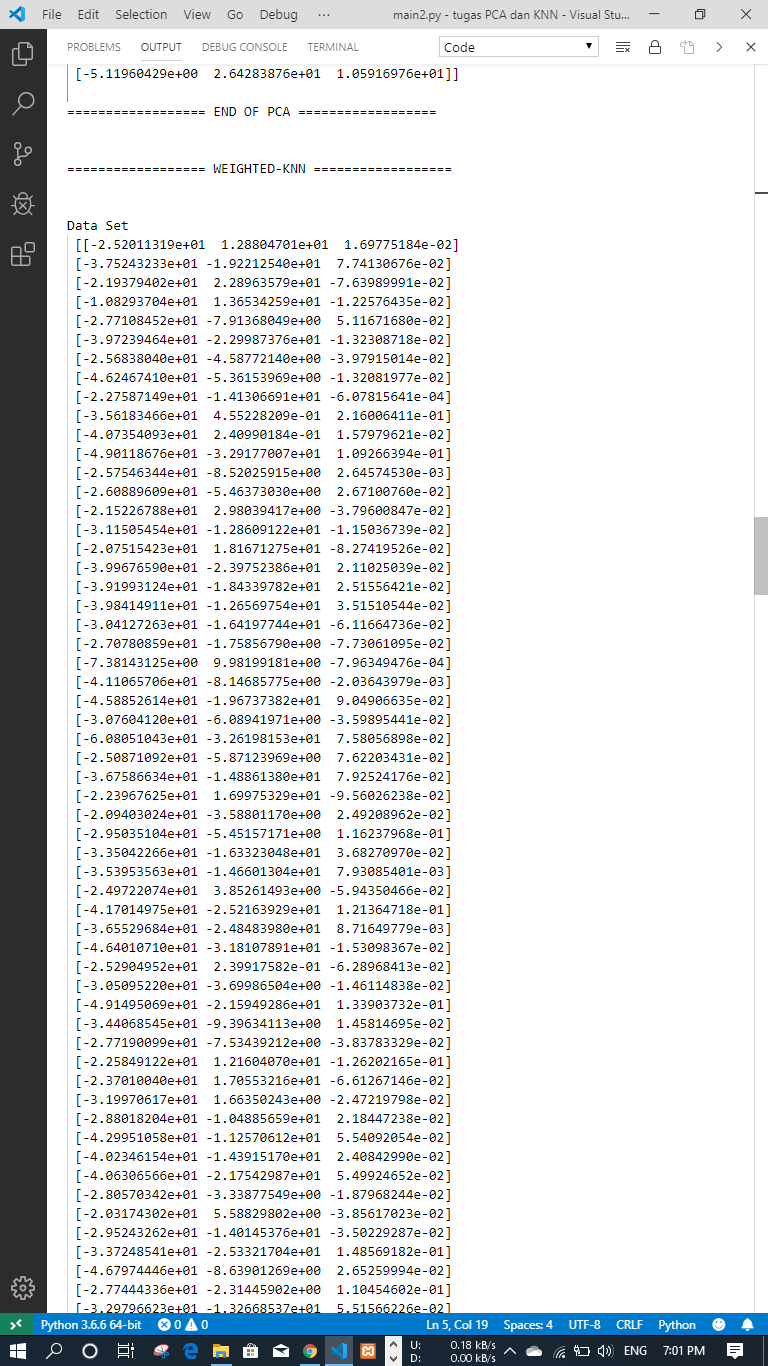
**Output Program**



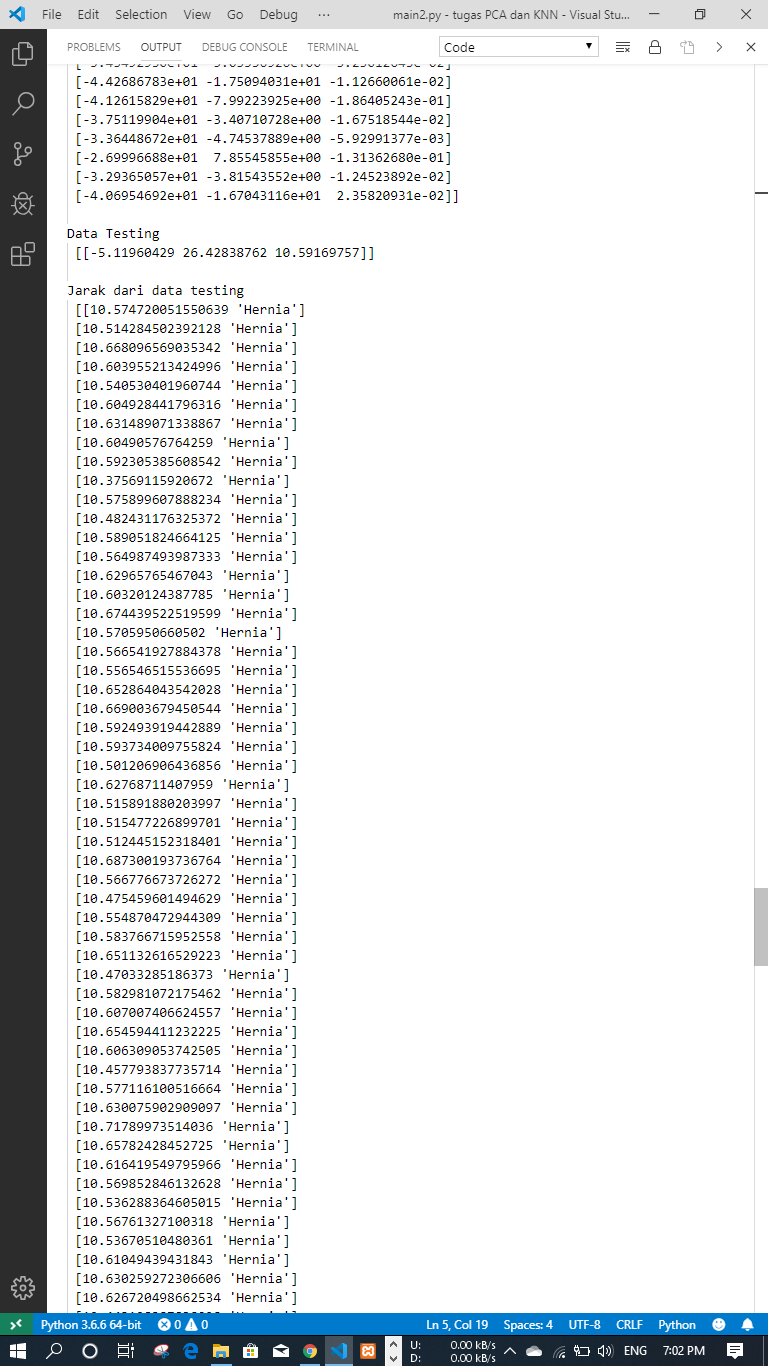


Sampai





sampai



Sampai

