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.

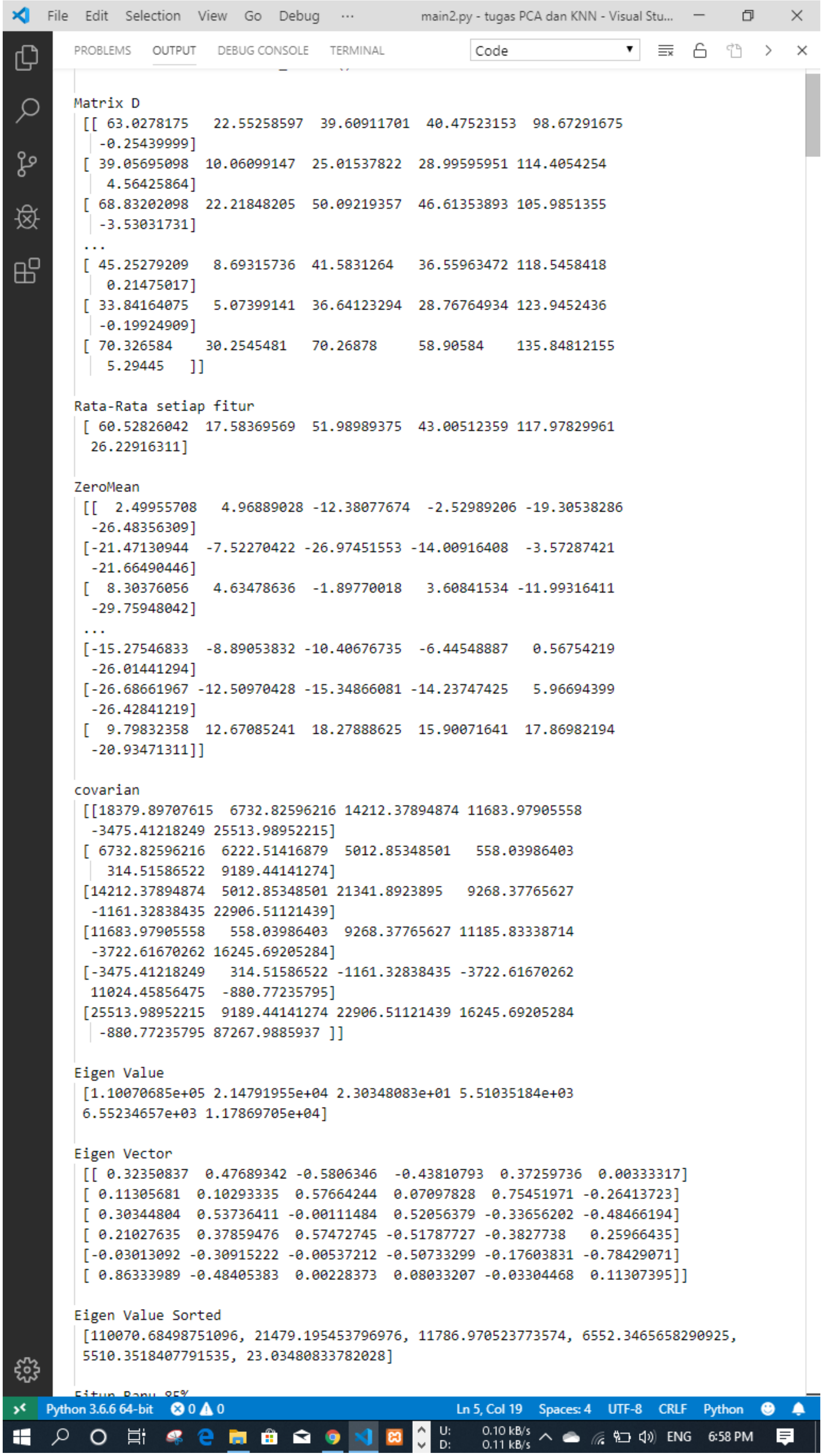
Penjelasan Program

```
"""
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]
```

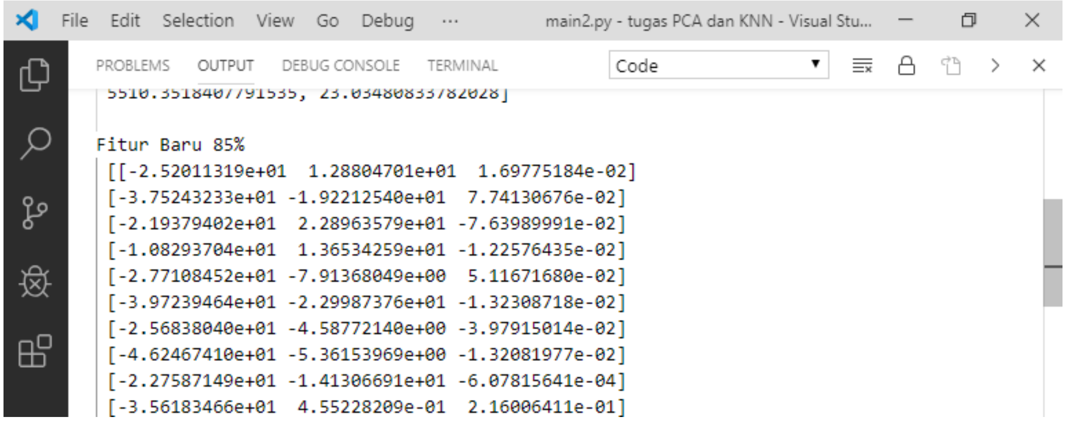
```python

#=========== 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

```

```python
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

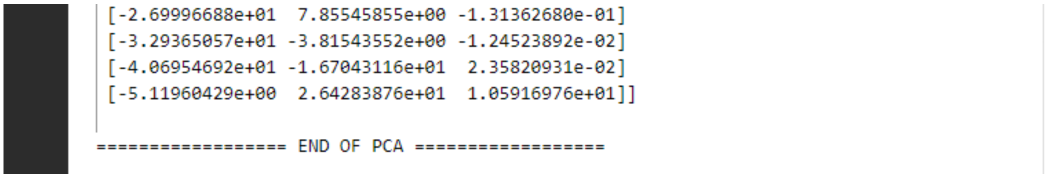


```
Matrix D
 [[ 63.0278175   22.55258597  39.60911701  40.47523153  98.67291675
   -0.25439999]
 [ 39.05695098  10.06099147  25.01537822  28.99595951 114.4054254
    4.56425864]
 [ 68.83202098  22.21848205  50.09219357  46.61353893 105.9851355
   -3.53031731]
 ...
 [ 45.25279209   8.69315736  41.5831264   36.55963472 118.5458418
    0.21475017]
 [ 33.84164075   5.07399141  36.64123294  28.76764934 123.9452436
   -0.19924909]
 [ 70.326584    30.2545481   70.26878     58.90584    135.84812155
    5.29445   ]]

Rata-Rata setiap fitur
 [ 60.52826042  17.58369569  51.98989375  43.00512359 117.97829961
  26.22916311]

ZeroMean
 [[  2.49955708   4.96889028 -12.38077674  -2.52989206 -19.30538286
  -26.48356309]
 [-21.47130944  -7.52270422 -26.97451553 -14.00916408  -3.57287421
  -21.66490446]
 [  8.30376056   4.63478636  -1.89770018   3.60841534 -11.99316411
  -29.75948042]
 ...
 [-15.27546833  -8.89053832 -10.40676735  -6.44548887   0.56754219
  -26.01441294]
 [-26.68661967 -12.50970428 -15.34866081 -14.23747425   5.96694399
  -26.42841219]
 [  9.79832358  12.67085241  18.27888625  15.90071641  17.86982194
  -20.93471311]]

covarian
 [[18379.89707615  6732.82596216 14212.37894874 11683.97905558
  -3475.41218249 25513.98952215]
 [ 6732.82596216  6222.51416879  5012.85348501   558.03986403
    314.51586522  9189.44141274]
 [14212.37894874  5012.85348501 21341.8923895   9268.37765627
  -1161.32838435 22906.51121439]
 [11683.97905558   558.03986403  9268.37765627 11185.83338714
  -3722.61670262 16245.69205284]
 [-3475.41218249   314.51586522 -1161.32838435 -3722.61670262
  11024.45856475  -880.77235795]
 [25513.98952215  9189.44141274 22906.51121439 16245.69205284
   -880.77235795 87267.9885937 ]]

Eigen Value
 [1.10070685e+05 2.14791955e+04 2.30348083e+01 5.51035184e+03
 6.55234657e+03 1.17869705e+04]

Eigen Vector
 [[ 0.32350837  0.47689342 -0.5806346  -0.43810793  0.37259736  0.00333317]
 [ 0.11305681  0.10293335  0.57664244  0.07097828  0.75451971 -0.26413723]
 [ 0.30344804  0.53736411 -0.00111484  0.52056379 -0.33656202 -0.48466194]
 [ 0.21027635  0.37859476  0.57472745 -0.51787727 -0.3827738   0.25966435]
 [-0.03013092 -0.30915222 -0.00537212 -0.50733299 -0.17603831 -0.78429071]
 [ 0.86333989 -0.48405383  0.00228373  0.08033207 -0.03304468  0.11307395]]

Eigen Value Sorted
 [110070.68498751096, 21479.195453796976, 11786.970523773574, 6552.3465658290925,
 5510.3518407791535, 23.03480833782028]

Fitur Baru 85%
```

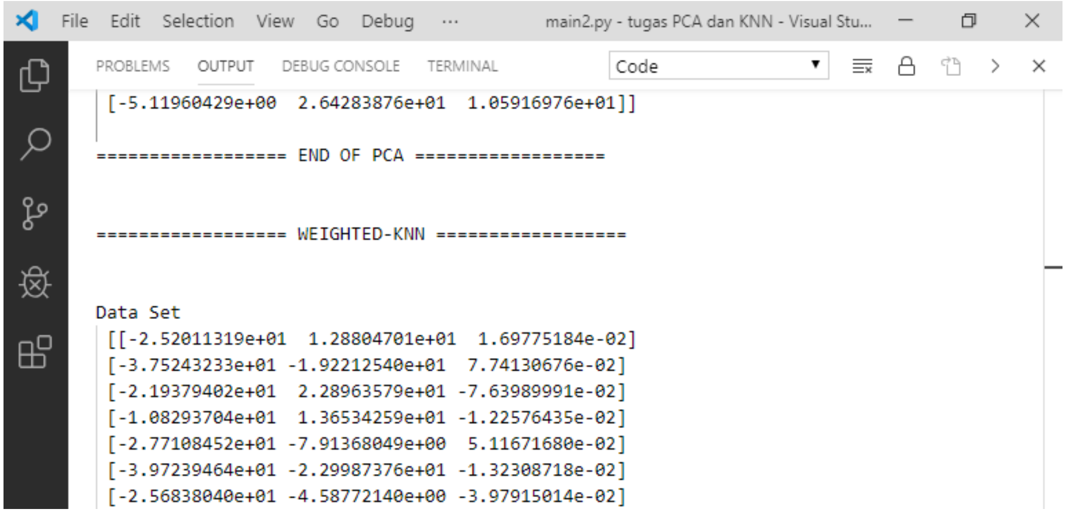


```
 5510.3518407791535, 23.03480833782028]

Fitur Baru 85%
 [[-2.52011319e+01  1.28804701e+01  1.69775184e-02]
 [-3.75243233e+01 -1.92212540e+01  7.74130676e-02]
 [-2.19379402e+01  2.28963579e+01 -7.63989991e-02]
 [-1.08293704e+01  1.36534259e+01 -1.22576435e-02]
 [-2.77108452e+01 -7.91368049e+00  5.11671680e-02]
 [-3.97239464e+01 -2.29987376e+01 -1.32308718e-02]
 [-2.56838040e+01 -4.58772140e+00 -3.97915014e-02]
 [-4.62467410e+01 -5.36153969e+00 -1.32081977e-02]
 [-2.27587149e+01 -1.41306691e+01 -6.07815641e-04]
 [-3.56183466e+01  4.55228209e-01  2.16006411e-01]
```

Sampai

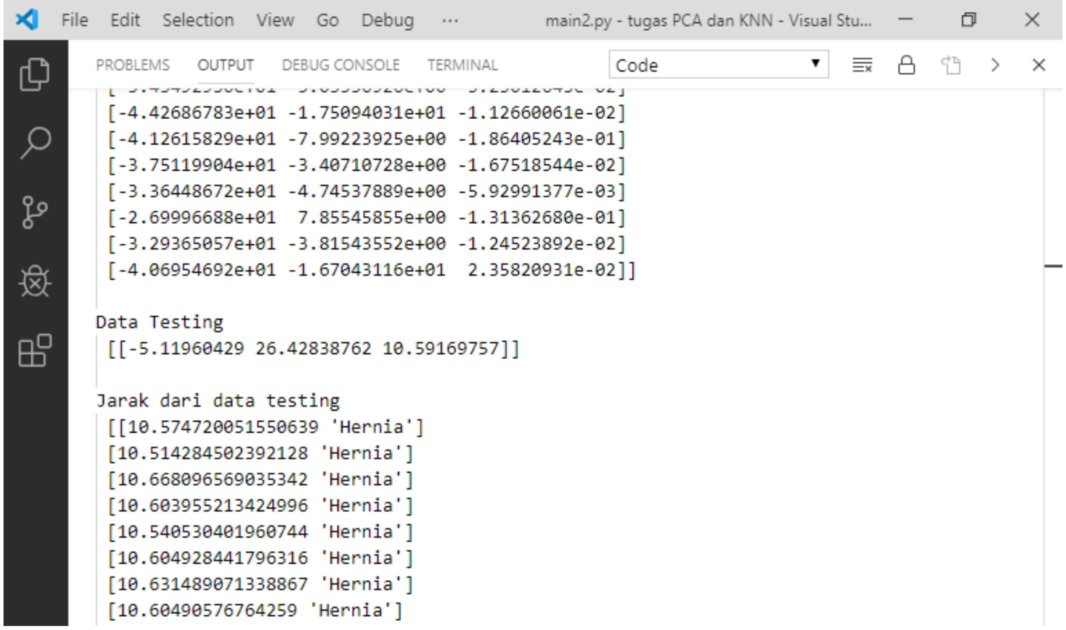


```
 [-2.69996688e+01  7.85545855e+00 -1.31362680e-01]
 [-3.29365057e+01 -3.81543552e+00 -1.24523892e-02]
 [-4.06954692e+01 -1.67043116e+01  2.35820931e-02]
 [-5.11960429e+00  2.64283876e+01  1.05916976e+01]]

================= END OF PCA =================
```



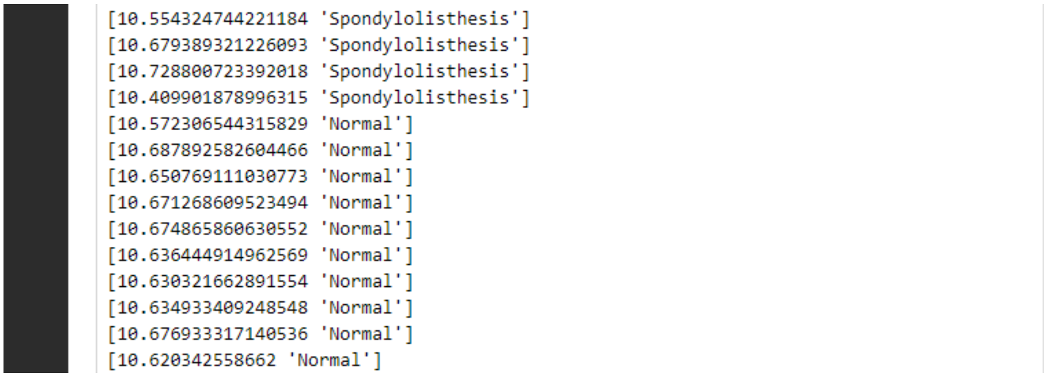
```
 [-5.11960429e+00  2.64283876e+01  1.05916976e+01]]

================= END OF PCA =================


================= WEIGHTED-KNN =================


Data Set
 [[-2.52011319e+01  1.28804701e+01  1.69775184e-02]
 [-3.75243233e+01 -1.92212540e+01  7.74130676e-02]
 [-2.19379402e+01  2.28963579e+01 -7.63989991e-02]
 [-1.08293704e+01  1.36534259e+01 -1.22576435e-02]
 [-2.77108452e+01 -7.91368049e+00  5.11671680e-02]
 [-3.97239464e+01 -2.29987376e+01 -1.32308718e-02]
 [-2.56838040e+01 -4.58772140e+00 -3.97915014e-02]
```

sampai

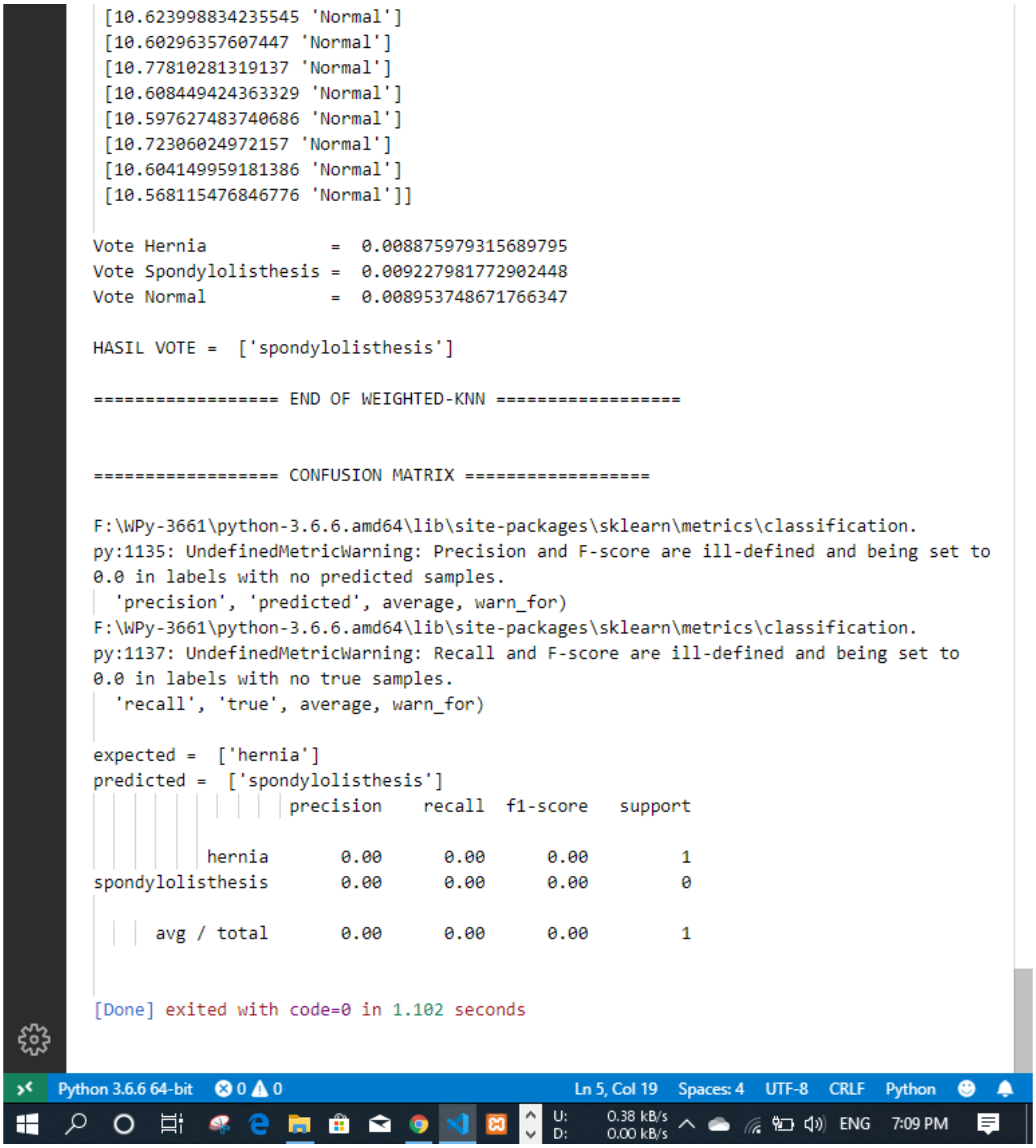


```
 [-4.42686783e+01 -1.75094031e+01 -1.12660061e-02]
 [-4.12615829e+01 -7.99223925e+00 -1.86405243e-01]
 [-3.75119904e+01 -3.40710728e+00 -1.67518544e-02]
 [-3.36448672e+01 -4.74537889e+00 -5.92991377e-03]
 [-2.69996688e+01  7.85545855e+00 -1.31362680e-01]
 [-3.29365057e+01 -3.81543552e+00 -1.24523892e-02]
 [-4.06954692e+01 -1.67043116e+01  2.35820931e-02]]

Data Testing
 [[-5.11960429 26.42838762 10.59169757]]

Jarak dari data testing
 [[10.574720051550639 'Hernia']
 [10.514284502392128 'Hernia']
 [10.668096569035342 'Hernia']
 [10.603955213424996 'Hernia']
 [10.540530401960744 'Hernia']
 [10.604928441796316 'Hernia']
 [10.631489071338867 'Hernia']
 [10.60490576764259 'Hernia']
```

Sampai



```
 [10.554324744221184 'Spondylolisthesis']
 [10.679389321226093 'Spondylolisthesis']
 [10.728800723392018 'Spondylolisthesis']
 [10.409901878996315 'Spondylolisthesis']
 [10.572306544315829 'Normal']
 [10.687892582604466 'Normal']
 [10.650769111030773 'Normal']
 [10.671268609523494 'Normal']
 [10.674865860630552 'Normal']
 [10.636444914962569 'Normal']
 [10.630321662891554 'Normal']
 [10.634933409248548 'Normal']
 [10.676933317140536 'Normal']
 [10.620342558662 'Normal']
```



```
 [10.623998834235545 'Normal']
 [10.60296357607447 'Normal']
 [10.77810281319137 'Normal']
 [10.608449424363329 'Normal']
 [10.597627483740686 'Normal']
 [10.72306024972157 'Normal']
 [10.604149959181386 'Normal']
 [10.568115476846776 'Normal']]

Vote Hernia            =  0.008875979315689795
Vote Spondylolisthesis =  0.009227981772902448
Vote Normal            =  0.008953748671766347

HASIL VOTE =  ['spondylolisthesis']

================== END OF WEIGHTED-KNN ==================


================== CONFUSION MATRIX ==================

F:\WPy-3661\python-3.6.6.amd64\lib\site-packages\sklearn\metrics\classification.
py:1135: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
F:\WPy-3661\python-3.6.6.amd64\lib\site-packages\sklearn\metrics\classification.
py:1137: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to
0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)

expected =  ['hernia']
predicted =  ['spondylolisthesis']
                   precision    recall  f1-score   support

           hernia       0.00      0.00      0.00         1
spondylolisthesis       0.00      0.00      0.00         0

      avg / total       0.00      0.00      0.00         1


[Done] exited with code=0 in 1.102 seconds
```