

**Experiment No.: 5**

**Aim**

To implement Z-score normalization, Min-max normalization using iris dataset

1. **Program and Output**

import pandas as pd

import numpy as np

import scipy.stats as stats

data = np.array([6, 7, 7, 12, 13, 13, 15, 16, 19, 22])

stats.zscore(data)

**OUTPUT**

array([-1.39443338, -1.19522861, -1.19522861, -0.19920477, 0. , 0. , 0.39840954, 0.5976143 , 1.19522861, 1.79284291])

1. **Program and Output**

import pandas as pd

import numpy as np

import scipy.stats as stats

data = np.array([[5, 6, 7, 7, 8],

                 [8, 8, 8, 9, 9],

                 [2, 2, 4, 4, 5]])

stats.zscore(data, axis=1)

**OUTPUT**

array([[-1.56892908, -0.58834841, 0.39223227, 0.39223227, 1.37281295], [-0.81649658, -0.81649658, -0.81649658, 1.22474487, 1.22474487], [-1.16666667, -1.16666667, 0.5 , 0.5 , 1.33333333]])

1. **Program and Output**

import pandas as pd

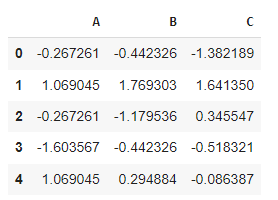
import numpy as np

import scipy.stats as stats

data = pd.DataFrame(np.random.randint(0, 10, size=(5, 3)), columns=['A', 'B', 'C'])

data.apply(stats.zscore)

**OUTPUT**



1. **Program and Output**

from numpy import asarray

from sklearn.preprocessing import MinMaxScaler

# define data

data = asarray([[100, 0.001],

        [8, 0.05],

        [50, 0.005],

        [88, 0.07],

        [4, 0.1]])

print(data)

# define min max scaler

scaler = MinMaxScaler()

# transform data

scaled = scaler.fit\_transform(data)

print(scaled)

**OUTPUT**

[[1.0e+02 1.0e-03]

[8.0e+00 5.0e-02]

[5.0e+01 5.0e-03]

[8.8e+01 7.0e-02]

[4.0e+00 1.0e-01]]

[[1. 0. ]

[0.04166667 0.49494949]

[0.47916667 0.04040404]

[0.875 0.6969697 ]

[0. 1. ]]

1. **Program and Output**

from numpy import asarray

from sklearn.preprocessing import StandardScaler

# define data

data = asarray([[100, 0.001],

        [8, 0.05],

        [50, 0.005],

        [88, 0.07],

        [4, 0.1]])

print(data)

# define standard scaler

scaler = StandardScaler()

# transform data

scaled = scaler.fit\_transform(data)

print(scaled)

**OUTPUT**

[[1.0e+02 1.0e-03]

[8.0e+00 5.0e-02]

[5.0e+01 5.0e-03]

[8.8e+01 7.0e-02]

[4.0e+00 1.0e-01]]

[[ 1.26398112 -1.16389967]

[-1.06174414 0.12639634]

[ 0. -1.05856939]

[ 0.96062565 0.65304778]

[-1.16286263 1.44302493]]

1. **Program and Output**

# load and summarize the sonar dataset

from pandas import read\_csv

from pandas.plotting import scatter\_matrix

from matplotlib import pyplot

# Load dataset

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/sonar.csv"

dataset = read\_csv(url, header=None)

# summarize the shape of the dataset

print(dataset.shape)

# summarize each variable

print(dataset.describe())

# histograms of the variables

dataset.hist()

pyplot.show()

**OUTPUT**

(208, 61)

0 1 2 3 4 5 \

count 208.000000 208.000000 208.000000 208.000000 208.000000 208.000000

mean 0.029164 0.038437 0.043832 0.053892 0.075202 0.104570

std 0.022991 0.032960 0.038428 0.046528 0.055552 0.059105

min 0.001500 0.000600 0.001500 0.005800 0.006700 0.010200

25% 0.013350 0.016450 0.018950 0.024375 0.038050 0.067025

50% 0.022800 0.030800 0.034300 0.044050 0.062500 0.092150

75% 0.035550 0.047950 0.057950 0.064500 0.100275 0.134125

max 0.137100 0.233900 0.305900 0.426400 0.401000 0.382300

6 7 8 9 ... 50 \

count 208.000000 208.000000 208.000000 208.000000 ... 208.000000

mean 0.121747 0.134799 0.178003 0.208259 ... 0.016069

std 0.061788 0.085152 0.118387 0.134416 ... 0.012008

min 0.003300 0.005500 0.007500 0.011300 ... 0.000000

25% 0.080900 0.080425 0.097025 0.111275 ... 0.008425

50% 0.106950 0.112100 0.152250 0.182400 ... 0.013900

75% 0.154000 0.169600 0.233425 0.268700 ... 0.020825

max 0.372900 0.459000 0.682800 0.710600 ... 0.100400

51 52 53 54 55 56 \

count 208.000000 208.000000 208.000000 208.000000 208.000000 208.000000

mean 0.013420 0.010709 0.010941 0.009290 0.008222 0.007820

std 0.009634 0.007060 0.007301 0.007088 0.005736 0.005785

min 0.000800 0.000500 0.001000 0.000600 0.000400 0.000300

25% 0.007275 0.005075 0.005375 0.004150 0.004400 0.003700

50% 0.011400 0.009550 0.009300 0.007500 0.006850 0.005950

75% 0.016725 0.014900 0.014500 0.012100 0.010575 0.010425

max 0.070900 0.039000 0.035200 0.044700 0.039400 0.035500

57 58 59

count 208.000000 208.000000 208.000000

mean 0.007949 0.007941 0.006507

std 0.006470 0.006181 0.005031

min 0.000300 0.000100 0.000600

25% 0.003600 0.003675 0.003100

50% 0.005800 0.006400 0.005300

75% 0.010350 0.010325 0.008525

max 0.044000 0.036400 0.043900

[8 rows x 60 columns]