Name: Manya Madhu

Batch: RMCA B

Roll no: 17

Date: 15-09-2022

Program No: 4

Aim:

Z score normalization and min-max normalization

Procedure:

a)

```
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])
```

b)

```
from sklearn.datasets import load_iris
from sklearn.preprocessing import MinMaxScaler
import numpy as np
# use the iris dataset
X, y = load_iris(return_X_y=True)
print(X.shape)
# (150, 4) # 150 samples (rows) with 4 features/variables (columns)
# build the scaler model
scaler = MinMaxScaler()
# fit using the train set
scaler.fit(X)
# transform the test test
X_scaled = scaler.transform(X)
# Verify minimum value of all features
X_scaled.min(axis=0)
# array([0., 0., 0., 0.])
# Verify maximum value of all features
X_scaled.max(axis=0)
# array([1., 1., 1., 1.])
# Manually normalise without using scikit-learn
X_manual_scaled = X - X.min(axis=0) / (X.max(axis=0) - X.min(axis=0))
# Verify manually VS scikit-learn estimation
print(np.allclose(X_scaled, X_manual_scaled))
```

```
#True
```

Output

```
(150, 4)
False
```

c)

```
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]])
```

d)

```
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


data.apply(stats.zscore)
```

Output

```
           B          C
 A

 0  -0.790569  -1.195229   1.578410

 1   1.185854  -1.195229  -0.742781

 2  -0.395285   0.597614  -0.278543

 3  -1.185854   1.195229  -1.207020

 4   1.185854   0.597614   0.649934
```

e)

```
from sklearn.preprocessing import MinMaxScaler
>>> data = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]
>>> scaler = MinMaxScaler()
>>> print(scaler.fit(data))
MinMaxScaler()
>>> print(scaler.data_max_)

>>> print(scaler.transform(data))

>>> print(scaler.transform([[2, 2]]))
```

Output

```
MinMaxScaler()
[ 1. 18.]
[[0.   0.  ]
 [0.25 0.25]
 [0.5  0.5 ]
 [1.   1.  ]]
[[1.5 0. ]]
```

f)

```
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.        ]]
```

g)

```
# visualize a minmax scaler transform of the sonar dataset
from pandas import read_csv
from pandas import DataFrame
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot
# load dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/sonar.c
sv"
dataset = read_csv(url, header=None)
# retrieve just the numeric input values
data = dataset.values[:, :-1]
# perform a robust scaler transform of the dataset
trans = MinMaxScaler()
data = trans.fit_transform(data)
# convert the array back to a dataframe
dataset = DataFrame(data)
# summarize
print(dataset.describe())
# histograms of the variables
dataset.hist()
pyplot.show()
```

Output

```
count  208.000000  208.000000  208.000000  208.000000  208.000000  208.000000
mean     0.204011    0.162180    0.139068    0.114342    0.173732    0.253615
std      0.169550    0.141277    0.126242    0.110623    0.140888    0.158843
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.087389    0.067938    0.057326    0.044163    0.079508    0.152714
50%      0.157080    0.129447    0.107753    0.090942    0.141517    0.220236
75%      0.251106    0.202958    0.185447    0.139563    0.237319    0.333042
max      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000

                6           7           8           9  ...          50  \
count  208.000000  208.000000  208.000000  208.000000  ...  208.000000
mean     0.320472    0.285114    0.252485    0.281652  ...    0.160047
std      0.167175    0.187767    0.175311    0.192215  ...    0.119607
min      0.000000    0.000000    0.000000    0.000000  ...    0.000000
25%      0.209957    0.165215    0.132571    0.142964  ...    0.083914
50%      0.280438    0.235061    0.214349    0.244673  ...    0.138446
75%      0.407738    0.361852    0.334555    0.368082  ...    0.207420
max      1.000000    1.000000    1.000000    1.000000  ...    1.000000

               51          52          53          54          55          56
\
count  208.000000  208.000000  208.000000  208.000000  208.000000  208.000000
mean     0.180031    0.265172    0.290669    0.197061    0.200555    0.213642
```

```
std      0.137432    0.183385    0.213474    0.160717    0.147080    0.164361
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.092368    0.118831    0.127924    0.080499    0.102564    0.096591
50%      0.151213    0.235065    0.242690    0.156463    0.165385    0.160511
75%      0.227175    0.374026    0.394737    0.260771    0.260897    0.287642
max      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000

               57          58          59
count  208.000000  208.000000  208.000000
mean     0.175035    0.216015    0.136425
std      0.148051    0.170286    0.116190
min      0.000000    0.000000    0.000000
25%      0.075515    0.098485    0.057737
50%      0.125858    0.173554    0.108545
75%      0.229977    0.281680    0.183025
max      1.000000    1.000000    1.000000

[8 rows x 60 columns]
```