Name : Naman Thaker

Roll no : 20BCE529

Subject : Data Mining

Division D

Practical 4

Implement the following normalization techniques for the given dataset (age): 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70. Use Min-Max Normalization to transform the value 25 and 52 for age onto the range [0.0, 1.0]. Use z-score normalization to transform the value 35 for age, where the standard deviation of age is 12.94 years. Use normalization by decimal scaling to transform the value 35 for age.

```
# Import Statements
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
%matplotlib inline
```

```
bank_data = pd.read_csv("Automobile_insurance_fraud.csv")
bank_data
```

	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_
0	328	48	521585	17-10-2014	OH	250/

```
bank_data.describe()
```

**Min- Max Normalization **

1000 rows × 40 columns

```
min_inc = bank_data['policy_annual_premium'].min()
max_inc = bank_data['policy_annual_premium'].max()
bank_data['income_norm'] = [(x-min_inc)/(max_inc-min_inc) for x in bank_data['policy_annua

min_age = bank_data['age'].min()
max_age = bank_data['age'].max()
bank_data['age_norm'] = [(x-min_age)/(max_age-min_age) for x in bank_data['age']]

min_kid = bank_data['vehicle_claim'].min()
max_kid = bank_data['vehicle_claim'].max()
bank_data['vehicle_norm'] = [(x-min_kid)/(max_kid-min_kid) for x in bank_data['vehicle_cla
last_column = bank_data.iloc[: , -3],bank_data.iloc[: , -2],bank_data.iloc[: , -1]
last_column
```

```
(0      0.603112
 1      0.473214
 2      0.606972
 3      0.608582
 4      0.712760
          ...
 995    0.543574
 996    0.621622
 997    0.588604
 998    0.572145
 999    0.206200
```

```
Name: income_norm, Length: 1000, dtype: float64, 0      0.644444
1      0.511111
2      0.222222
3      0.488889
4      0.555556
         ...
995    0.422222
996    0.488889
997    0.333333
998    0.955556
999    0.911111
Name: age_norm, Length: 1000, dtype: float64, 0      0.654296
1      0.043276
2      0.289722
3      0.637187
4      0.056359
         ...
995    0.767015
996    0.908919
997    0.659580
998    0.458800
999    0.045415
Name: vehicle_norm, Length: 1000, dtype: float64)
```

Using Z score Normalization

```
# stats.zscore() method
import numpy as np
from scipy import stats

arr1 = [bank_data['vehicle_claim'],
        bank_data['property_claim']]

# arr2 = [[50, 12, 12, 34, 4],
#       [12, 11, 10, 34, 21]]

print ("\narr1 : ", arr1)
#print ("\narr2 : ", arr2)

print ("\nZ-score for arr1 : \n", stats.zscore(arr1))
print ("\nZ-score for arr1 : \n", stats.zscore(arr1, axis = 1))
```

```
arr1 :  [0      52080
1       3510
2      23100
3      50720
4       4550
       ...
995    61040
996    72320
997    52500
998    36540
999     3680
Name: vehicle_claim, Length: 1000, dtype: int64, 0      13020
```

```
1        780
2       3850
3       6340
4        650
       ...
995     8720
996    18080
997     7500
998     5220
999      920
Name: property_claim, Length: 1000, dtype: int64]

Z-score for arr1 :
 [[ 1.  1.  1. ...  1.  1.  1.]
 [-1. -1. -1. ... -1. -1. -1.]]

Z-score for arr1 :
 [[ 0.74965272 -1.82334593 -0.78556451 ...  0.77190224 -0.07357971
  -1.81434017]
 [ 1.16550497 -1.37269599 -0.73607206 ...  0.0208261  -0.45197603
  -1.34366428]]


# Problem Statement
# Following data (in increasing order) is provided for the attribute ‘age’: 13, 15,
# 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45,
# 52, 70.
# (a) Use smoothing by bin means to smooth these data, using a bin depth of 3.
# (b) Use min-max normalization to transform the value 35 for age onto the range
# [0.0, 1.0].
# (c) Use z-score normalization to transform the value 35 for age, where the standard
# deviation of age is 12.94 years.
# (d) Use normalization by decimal scaling to transform the value 35 for age.
# '''


import statistics

def bin(list,depth):
    maxSize=len(list)
    i=0
    newList = []
    while (i<maxSize-depth+1):
        sum = 0
        #Sum of a BIN
        for j in range (i,i+depth):
            sum+=list[j]
        #Smoothing a BIN
        ans = sum / depth
        ans=round(ans,2)

        #Smoothed Data in a list
        for j in range (i,i+depth):
            newList.append(ans)
        i+=depth

    #if number of elements in list is not a multiplier of depth
```

```
        if(maxSize%depth!=0):
            sum=0
            for j in range (i,maxSize):
                sum+=list[j]
            ans=sum/(maxSize-i)
            ans = round(ans, 2)

            # Smoothed Data in a list
            for j in range(i, maxSize):
                newList.append(ans)
        return newList


    def minMaxNor(num,list):
        ans=round((num-list[0])/(list[len(list)-1]-list[0]),3)
        return ans

    def zNor (num,mean,stdDv):
        return round((num-mean)/stdDv,3)

    def decNor(num,maxNum):
        digit=len(str(maxNum))
        div=pow(10,digit)
        return num/div


    list = [13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35
    depth=3

    minAns=minMaxNor(35,list)

    print("\nAfter doing min-max normalization : \t",minAns)
    zNormalization=zNor(35,statistics.mean(list),12.94)
    print("\nAfter doing z-score normalization : \t", zNormalization)
    decimalNormalitation=decNor(35,max(list))
    print("\nAfter doing normalization by decimal scaling :\t",decimalNormalitation)
```

```
After doing min-max normalization :      0.386

After doing z-score normalization :      0.389

After doing normalization by decimal scaling :   0.35
```