Name - Makwana Ayush

Roll No - 18BCE107

Practical - 1

Implement normalization method on the given data:

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.

a) Use Min-Max Normalization

b) Use z-score normalization

c) Use decimal scaling

Min-Max

In [5]:

```
import numpy as np

def MinMax(data):
    new_Max = 1
    new_Min = 0
    max_Value = np.max(data)
    min_Value = np.min(data)
    normalizedValues = list()

    print("Max : ",max_Value,"\nMin : ",min_Value,"\n")
    for i in data:
        temp = ((i - min_Value)/(max_Value - min_Value))*(new_Max - new_Min) + new_Min
        normalizedValues.append(temp)

    return normalizedValues
```

Z-Score

In [6]:

```
def zScore(data):
    mean = np.mean(data)
    sd = np.std(data)

    print("\nMean of the data:",mean,"\nStandard deviation of the data",sd)
    normalizedValues = list()

    for i in data:
        temp = (i - mean)/sd
        normalizedValues.append(temp)

    return normalizedValues
```

Decimal Scaling

In [7]:

```
def decimalScaling(data):
    maxnum = np.max(data)
    j = len(str(abs(maxnum)))

    print(j)
    normalizedValues = list()
    for i in data:
        temp = i/(10**j)
        normalizedValues.append(temp)

    return normalizedValues
```

For above dataset

In [9]:

```
data = [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,

print("Min Max normalization:")
resultMinMax = MinMax(data)
print("\nAfter Min Max normalization:\n",np.array(resultMinMax).T)
print("for 25:",resultMinMax[10])
print("for 52:",resultMinMax[-2])

print("\n-----------------------------------\n")
print("\n\nZ-score normalization:")
resultZScore = zScore(data)
print("\nAfter z-score normalization:\n",np.array(resultZScore).T)
print("for 35:",resultZScore[-7])

print("\n-----------------------------------\n")
print("\n\nDecimal Scaling normalization:")
resultDS = decimalScaling(data)
print("\nAfter decimal scaling normalization:\n",np.array(resultDS).T)
print("for 35:",resultDS[-7])
```

```
Min Max normalization:
Max :  70
Min :  13


After Min Max normalization:
 [0.         0.03508772 0.05263158 0.05263158 0.10526316 0.12280702
 0.12280702 0.14035088 0.15789474 0.15789474 0.21052632 0.21052632
 0.21052632 0.21052632 0.29824561 0.35087719 0.35087719 0.38596491
 0.38596491 0.38596491 0.38596491 0.40350877 0.47368421 0.56140351
 0.57894737 0.68421053 1.        ]
for 25: 0.21052631578947367
for 52: 0.6842105263157895

-----------------------------------




Z-score normalization:

Mean of the data: 29.962962962962962
```

```
Standard deviation of the data 12.700193878606099

After z-score normalization:
 [-1.33564599e+00 -1.17816807e+00 -1.09942912e+00 -1.09942912e+00
 -8.63212252e-01 -7.84473297e-01 -7.84473297e-01 -7.05734341e-01
 -6.26995386e-01 -6.26995386e-01 -3.90778520e-01 -3.90778520e-01
 -3.90778520e-01 -3.90778520e-01  2.91625761e-03  2.39133124e-01
  2.39133124e-01  3.96611035e-01  3.96611035e-01  3.96611035e-01
  3.96611035e-01  4.75349990e-01  7.90305812e-01  1.18400059e+00
  1.26273954e+00  1.73517328e+00  3.15247448e+00]
for 35: 0.3966110348537352

------------------------------------




Decimal Scaling normalization:
2

After decimal scaling normalization:
 [0.13 0.15 0.16 0.16 0.19 0.2  0.2  0.21 0.22 0.22 0.25 0.25 0.25 0.25
 0.3  0.33 0.33 0.35 0.35 0.35 0.35 0.36 0.4  0.45 0.46 0.52 0.7 ]
for 35: 0.35
```

Stock Dataset

In [18]:

```python
import pandas as pd
import matplotlib.pyplot as plt
```

In [32]:

```
stk = pd.read_csv('test.csv')
stk.head(437)
```

Out[32]:

	Open	High	Low	Volume
0	67299.00000	67299.00000	66500.39844	4193
1	66500.00000	67100.00000	66150.70313	7531
2	66750.00000	67400.00000	66458.70313	4603
3	66999.70313	67349.89844	66201.00000	4278
4	67100.00000	67345.89844	66666.10156	6823
...	...	...	...	...
432	60800.00000	61100.00000	60076.39844	9028
433	60490.00000	61089.69922	59956.80078	10706
434	60728.89844	61100.00000	59600.00000	9954
435	60677.89844	60677.89844	58542.60156	12645
436	59298.00000	59298.00000	58174.10156	9057

437 rows × 4 columns

Min-Max

In [33]:

```
MinMaxStk = stk.copy()

MinMaxStk['Volume'] = MinMax(MinMaxStk['Volume'])
```

```
Max :  49738
Min :  1613
```

In [13]:

```
MinMaxStk
```

Out[13]:

	Open	High	Low	Volume
0	67299.00000	67299.00000	66500.39844	0.053610
1	66500.00000	67100.00000	66150.70313	0.122971
2	66750.00000	67400.00000	66458.70313	0.062130
3	66999.70313	67349.89844	66201.00000	0.055377
4	67100.00000	67345.89844	66666.10156	0.108260
...	...	...	...	...
433	60490.00000	61089.69922	59956.80078	0.188945
434	60728.89844	61100.00000	59600.00000	0.173319
435	60677.89844	60677.89844	58542.60156	0.229236
436	59298.00000	59298.00000	58174.10156	0.154681
437	58200.10156	59439.19922	58200.10156	0.293590

438 rows × 4 columns

Z-Score

In [34]:

```
ZscoreStk = stk.copy()

ZscoreStk['Volume'] = zScore(ZscoreStk['Volume'])
```

```
Mean of the data: 9054.897260273972
Standard deviation of the data 6396.147769084537
```

In [35]:
```
ZscoreStk
```

Out[35]:

	Open	High	Low	Volume
0	67299.00000	67299.00000	66500.39844	-0.760129
1	66500.00000	67100.00000	66150.70313	-0.238252
2	66750.00000	67400.00000	66458.70313	-0.696028
3	66999.70313	67349.89844	66201.00000	-0.746840
4	67100.00000	67345.89844	66666.10156	-0.348944
...	...	...	...	...
433	60490.00000	61089.69922	59956.80078	0.258140
434	60728.89844	61100.00000	59600.00000	0.140569
435	60677.89844	60677.89844	58542.60156	0.561291
436	59298.00000	59298.00000	58174.10156	0.000329
437	58200.10156	59439.19922	58200.10156	1.045489

438 rows × 4 columns

Decimal Scaling

In [36]:
```
DecimalStk = stk.copy()

DecimalStk['Volume'] = decimalScaling(DecimalStk['Volume'])
```

```
5
```

```
In [17]: DecimalStk
```

Out[17]:

	Open	High	Low	Volume
0	67299.00000	67299.00000	66500.39844	0.04193
1	66500.00000	67100.00000	66150.70313	0.07531
2	66750.00000	67400.00000	66458.70313	0.04603
3	66999.70313	67349.89844	66201.00000	0.04278
4	67100.00000	67345.89844	66666.10156	0.06823
...	...	...	...	...
433	60490.00000	61089.69922	59956.80078	0.10706
434	60728.89844	61100.00000	59600.00000	0.09954
435	60677.89844	60677.89844	58542.60156	0.12645
436	59298.00000	59298.00000	58174.10156	0.09057
437	58200.10156	59439.19922	58200.10156	0.15742

438 rows × 4 columns

CONCLUSION

MinMax - When the feature is more-or-less uniformly distributed across a fixed range.

Zscore - When the feature distribution does not contain extreme outliers.

Decimal - When values are very large but uniform

```
In [ ]:
```