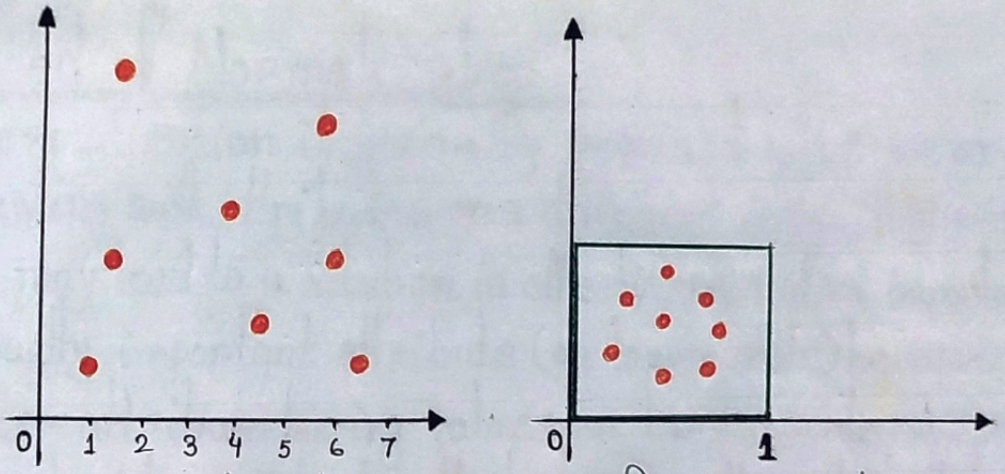
Data Normalization

Data normalization is a technique used in data mining to transform the values of a dataset into a common scale. This is important because many machine learning algorithms are sensitive to the scale of the input features and can produce better results when the data is normalized.



The different normalization techniques

1. Min-Max normalization.
2. Z-Score normalization.
3. Decimal Scaling.
4. Logarithmic transformation.
5. Root transformation.

In conclusion, normalization is an important step in data mining, as it can help to improve the performance of machine learning algorithms by

scaling the input features to a common scale. This can help to reduce the impact of outliers and improve the accuracy of the model.

Normalization is used to scale the data of an attribute so that it falls in a smaller range, such as -1.0 to 1.0 or 0.0 to 1.0. It is generally useful for classification algorithms.

Need of Normalization

Normalization is generally required when we are dealing with attributes on a different scale, otherwise, it may lead to a dilution in effectiveness of an important equally important attribute (on lower scale) because of other attribute having values on larger scale. In simple words, when multiple attributes are there but attributes have values on different scales, this may lead to poor data models while performing data mining operations. So they are normalized to bring all the attributes on the same scale.

Person_name	Salary	Year of experience	Expected Position Level
Suman	100000	10	2
Bumba	78000	7	4
Sourav	32000	5	8
Sudip	55000	6	7
Raj	92000	8	3

The attributes salary and year_of_experience are on different scale and hence attribute salary can take high priority over attribute year_of_experience in the model.

Min-Max Normalization

Min-Max normalization scales the data to a fixed range, typically between 0 and 1.

The formula for Min-Max normalization is

$X_{norm} = (X - X_{min}) / (X_{max} - X_{min})$

Where

- X is the original data point,
- Xmin is the minimum value in the dataset,
- Xmax is the maximum value in the dataset.

This technique is useful when you want to preserve the shape of the distribution and the exact values of the minimum and maximum. However, it is sensitive to outliers and may not work well if the dataset has extreme values.

Min-Max Normalization Implementation in Python

```
from sklearn.preprocessing import MinMaxScaler
# Normalize the data using Min-Max Normalization
scaler = MinMaxScaler()
X_normalized = scaler.fit_transform(dataset)
```

Z-score Normalization

Z-score normalization, also called Standardization, scales the data so that it has a mean of 0 and a standard deviation of 1.

The formula for Z-score normalization is,

Xnorm = (x - mu) / sigma

Where

- X is the original data point,
- mu is the mean of the dataset,
- sigma is the standard deviation of the dataset.

This technique is useful when you want to compare data points across different datasets or when you want to identify outliers.

It's also robust to outliers since it's based on the mean and standard deviation of the dataset. However, it may not preserve the shape of the distribution and can make it difficult to interpret the original values.

Z-Score Normalization Implementation In Python

```
from sklearn.preprocessing import StandardScaler
# Normalize the data using Z-score Normalization
scaler = StandardScaler()
X_normalized = scaler.fit_transform(datasets)
```

Decimal Scaling

Decimal scaling scales the data by multiplying it by a power of 10 to shift the decimal point to a fixed position.

The formula for decimal scaling is $X_{norm} = X / 10^j$.
Where

- X is the original data point,
- j is the smallest integer such that $\max(|x_{norm}|) < 1$.

This technique is useful when you want to preserve the relative size of the values while simplifying the data. However, it may not work well with small datasets or datasets with extreme values.

Decimal Scaling Normalization Implementation In Python

```
import pandas as pd
# Normalize the data using Decimal Scaling Normalization
X_normalized = dataset/10**(int(pd.np.ceil(pd.np.log10(pd.np.abs
(datasets)))))
```

Log Transformation

Log transformation scales the data by taking the logarithm of the values. The formula for log transformation is $X_{norm} = \log(x)$

Where X is the original data point.

This technique is useful when the data is skewed or has a heavy tail and when you want to compress a wide range of values into a smaller range. However, it may not work well with negative or zero values.

Log Transformation Implementation In Python

```
import numpy as np
# Normalize the data using Log Transformation
X_normalized = np.log(dataset)
```

Root Transformation

This technique applies a square root transformation to the values of a feature. This can be useful for data with a wide range of values, as it can help to reduce the impact of outliers.

Implementation In Python

```
import numpy as np
transformed_data = np.sqrt(data)
```

Uses

1. Min-Max normalization is useful when you want to preserve the shape of the distribution and the exact values of the minimum and maximum.

2. Z-score normalization is useful when you want to compare data points across different datasets or when you want to identify outliers.

3. Decimal scaling is useful when you want to preserve the relative size of the values while simplifying the data.

4. Log transformation is useful when the data is skewed or has a heavy tail, and when you want to compress a wide range of values into a smaller range.

5. Root transformation performs so well on the left-skewed data and efficiently transfromed the left-skewed data into normally distributed data.

Advantages and Disadvantages :

Advantages

1. Improved performance of machine learning algorithms : Normalization can help to improve the performance of machine learning algorithms by scaling the input features to a common scale. This can help to reduce the impact of outliers and improve the accuracy of the model.

2. Better handling of outliers : Normalization can help to reduce the impact of outliers by scaling the data to a common scale, which can make the outliers less influential.

3. Improved interpretability of results : Normalization can make it easier to interpret the results of a machine learning model, as the inputs will be on a common scale.

4. Better generalization : Normalization can help to improve the generalization of a model, by reducing the impact of outliers and by making the model less sensitive to the scale of the inputs.

Disadvantages

1. Loss of information : Normalization can result in a loss of information if the original scale of the input features is important.

2. Impact on outliers : Normalization can make it harder to detect outliers as they will be scaled along with the rest of the data.

3. Impact on interpretability : Normalization can make it harder to interpret the results of a machine learning model, as the inputs will be on a common scale, which may not align with the original scale of the data.

4. Additional computational costs : Normalization can add additional computational costs to the data mining process, as it requires additional processing time to scale the data.