Q1:-

There is another way of data scaling, where the minimum of feature is made equal to zero and the maximum of feature equal to one. MinMax Scaler shrinks the data within the given range, usually of 0 to 1. It transforms data by scaling features to a given range.

In [1]:
```
# import module
from sklearn.preprocessing import StandardScaler

# create data
data = [[11, 2], [3, 7], [0, 10], [11, 8]]

# compute required values
scaler = StandardScaler()
model = scaler.fit(data)
scaled_data = model.transform(data)

# print scaled data
print(scaled_data)
```

```
[[ 0.97596444 -1.61155897]
 [-0.66776515  0.08481889]
 [-1.28416374  1.10264561]
 [ 0.97596444  0.42409446]]
```

Q2:- This usually means dividing each component by the Euclidean length of the vector (L2 Norm). In some applications (e.g., histogram features), it can be more practical to use the L1 norm of the feature vector. Like Min-Max Scaling, the Unit Vector technique produces values of range [0,1]. In both cases, you're transforming the values of numeric variables so that the transformed data points have specific helpful properties. The difference is that: in scaling, you're changing the range of your data, while. in normalization, you're changing the shape of the distribution of your data.

For example, a scale of 1:100 means that one unit on the drawing equals 100 units in reality. Scale units can be expressed in different ways, such as fractions, decimals, or ratios.

Q3 Principal component analysis, or PCA, is a dimensionality reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

In [2]:
```
import pandas as pd
import numpy as np

# Here we are using inbuilt dataset of scikit learn
from sklearn.datasets import load_breast_cancer

# instantiating
cancer = load_breast_cancer(as_frame=True)
# creating dataframe
df = cancer.frame
```

```
# checking shape
print('Original Dataframe shape :',df.shape)

# Input features
X = df[cancer['feature_names']]
print('Inputs Dataframe shape   :', X.shape)
```

```
Original Dataframe shape : (569, 31)
Inputs Dataframe shape   : (569, 30)
```

Q4 PCA can be used to identify the most important features in a dataset, which can be used to build predictive models. Visualization: PCA can be used to visualize high-dimensional data in two or three dimensions, making it easier to understand and interpret. Principal Component Analysis (PCA) is a popular linear feature extractor used for unsupervised feature selection based on eigenvectors analysis to identify critical original features for principal component.

PCA is one of the most used linear dimensionality reduction technique. When using PCA, we take as input our original data and try to find a combination of the input features which can best summarize the original data distribution so that to reduce its original dimensions.

Q5 These steps include: Transforming and normalizing data. Training models. Evaluating model performance. Selecting the optimal model. Before we deep dive into this topic, first we'll think of how we can recommend items to users:

We can recommend items to a user which are most popular among all the users We can divide the users into multiple segments based on their preferences (user features) and recommend items to them based on the segment they belong to Both of the above methods have their drawbacks. In the first case, the most popular items would be the same for each user so everybody will see the same recommendations. While in the second case, as the number of users increases, the number of features will also increase. So classifying the users into various segments will be a very difficult task.

Q6 Machine learning (ML) is playing an increasingly significant role in stock trading. Predicting market fluctuations, studying consumer behavior, and analyzing stock price dynamics are examples of how investment companies can use machine learning for stock trading f you study prices over a long period of time, you will be able to see all three types of trends on the same chart. Watch the slope – The slope of a trend indicates how much the price should move each day. Steep lines, moving either upward or downward, indicate a certain trend.

In [5]:
```
#Q7
# import module
from sklearn.preprocessing import StandardScaler

# create data
data = [[5, 1], [5, 10], [10, 15], [15, 20]]
```

```
# compute required values
scaler = StandardScaler()
model = scaler.fit(data)
scaled_data = model.transform(data)

# print scaled data
print(scaled_data)
```

```
[[-0.90453403 -1.49618805]
 [-0.90453403 -0.21374115]
 [ 0.30151134  0.49872935]
 [ 1.50755672  1.21119985]]
```

In [10]:

```
#Q8
#Standardize the data (X_std)
#Calculate the Covariance-matrix
#Calculate the Eigenvector & Eigenvalues for the Covariance-matrix.
#Arrange all Eigenvalues in decreasing order.
#Normalize the sorted Eigenvalues.
#Horizontally stack the Normalized_ Eigenvalues =W_matrix
#X_PCA=X_std .dot (W_matrix)


import pandas as pd
import numpy as np

# Here we are using inbuilt dataset of scikit learn
from sklearn.datasets import load_breast_cancer

# instantiating
cancer = load_breast_cancer(as_frame=True)
# creating dataframe
df = cancer.frame

# checking shape
print('Original Dataframe shape :',df.shape)

# Input features
X = df[cancer['feature_names']]
print('Inputs Dataframe shape   :', X.shape)


# Mean
X_mean = X.mean()

# Standard deviation
X_std = X.std()

# Standardization
Z = (X - X_mean) / X_std
```

```
Original Dataframe shape : (569, 31)
Inputs Dataframe shape   : (569, 30)
```

In [ ]: