Q1. Min-Max Scaling

Definition:
Min-Max scaling (also called normalization) transforms features to a **fixed range**, usually [0, 1] or [-1, 1]. It preserves the shape of the original distribution.

Formula:

Xscaled=X−XminXmax−Xmin×(new_max−new_min)+new_minX_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \times (new_max - new_min) + new_minXscaled=Xmax−XminX−Xmin×(new_max−new_min)+new_min

Use:

- Ensures all features contribute equally to ML algorithms
- Required for algorithms sensitive to scale: **KNN, SVM, Neural Networks**

Python Example:

```
from sklearn.preprocessing import MinMaxScaler
import numpy as np

data = np.array([[10], [20], [30]])
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(data)
print(scaled_data)
```

Q2. Unit Vector Scaling

Definition:
Unit Vector scaling (or normalization) scales a **feature vector to have length 1**, i.e., the Euclidean norm is 1.

Formula:

Xunit=X∣∣X∣∣2=X∑Xi2X_{\text{unit}} = \frac{X}{||X||_2} = \frac{X}{\sqrt{\sum X_i^2}}Xunit=∣∣X∣∣2X=∑Xi2X

Difference from Min-Max Scaling:

- Min-Max: scales features to a fixed range, retains relative distances
- Unit Vector: scales **entire feature vector**, focuses on direction rather than magnitude

Python Example:

```
from sklearn.preprocessing import Normalizer
import numpy as np

data = np.array([[3,4]])
scaler = Normalizer(norm='l2')
normalized_data = scaler.transform(data)
print(normalized_data)  # Output: [[0.6, 0.8]] because sqrt(3^2+4^2)=5
```

Q3. PCA (Principal Component Analysis)

Definition:
PCA is a **dimensionality reduction technique** that transforms correlated features into a **smaller set of uncorrelated variables** called **principal components**, which retain most of the variance.

Steps:

1. Standardize the data
2. Compute covariance matrix
3. Calculate eigenvectors and eigenvalues
4. Select top `k` components based on explained variance
5. Transform data to lower dimension

Python Example:

```
from sklearn.decomposition import PCA
import numpy as np

data = np.array([[2.5,2.4],[0.5,0.7],[2.2,2.9],[1.9,2.2]])
```

```
pca = PCA(n_components=1)
reduced_data = pca.fit_transform(data)
print(reduced_data)
```

Q4. Relationship between PCA and Feature Extraction

- **Feature Extraction:** Creating **new features** from original features
- **PCA as Feature Extraction:** Principal components are **new features** that summarize the original features while preserving most variance

Example:

- Original features: height, weight, age
- PCA generates 2 principal components:
    - PC1 = combination of height & weight
    - PC2 = combination of age & weight

```
# The same PCA code as above converts 3 features → 2 PCs
```

Q5. Min-Max Scaling in Recommendation System

- Dataset: price, rating, delivery time
- Problem: features have **different scales** (price in $, rating 0-5, delivery time in minutes)
- **Solution:** Apply Min-Max scaling so all features are in the same range (e.g., 0-1), allowing fair comparison for algorithms like KNN or cosine similarity.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled_features =
scaler.fit_transform(df[['price','rating','delivery_time']])
```

Q6. PCA for Stock Price Prediction

- Dataset: company financials, market trends → **many correlated features**
- Problem: high-dimensional data → increases computation, risk of overfitting
- **Solution:**
    - Standardize features
    - Apply PCA → retain top k components explaining ~95% variance
    - Train model on reduced dimensions

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

X_scaled = StandardScaler().fit_transform(X)
pca = PCA(n_components=0.95)  # retain 95% variance
X_reduced = pca.fit_transform(X_scaled)
```

Q7. Min-Max Scaling Example: Transform [1,5,10,15,20] to [-1,1]

Formula:

Xscaled=(X−Xmin)/(Xmax−Xmin)×(new_max−new_min)+new_minX_{\text{scaled}} = (X - X_{\min}) / (X_{\max} - X_{\min}) \times (new_max - new_min) + new_minXscaled=(X−Xmin)/(Xmax−Xmin)×(new_max−new_min)+new_min

- Xmin=1X_{\min}=1Xmin=1, Xmax=20X_{\max}=20Xmax=20, new_min=−1new_min=-1new_min=−1, new_max=1new_max=1new_max=1

Xscaled=X−119×2−1X_{\text{scaled}} = \frac{X-1}{19} \times 2 - 1Xscaled=19X−1×2−1

Calculation:

Original X	Scaled X
1	-1
5	-0.5789
10	0.0526
15	0.6842
20	1

Q8. PCA Feature Extraction for [height, weight, age, gender, blood pressure]

Steps:

1. Standardize features
2. Compute PCA
3. Choose **principal components** that explain most variance (e.g., 90-95%)

Number of components to retain:

- Depends on **explained variance ratio**
- Example: If first 2 PCs explain 92% variance → retain **2 components**

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

X_scaled = StandardScaler().fit_transform(X)
pca = PCA(n_components=0.90)  # retain 90% variance
X_pca = pca.fit_transform(X_scaled)
print(pca.n_components_)  # e.g., 2
```

✅ Summary Table

Concept	Purpose	Example
Min-Max Scaling	Normalize features to range [0,1] or [-1,1]	price, rating, delivery time
Unit Vector Scaling	Scale feature vector to length 1	text TF-IDF vectors
PCA	Reduce dimensionality, extract uncorrelated features	Stock prices → top PCs
Feature Extraction via PCA	Create new features summarizing original ones	height & weight → PC1
