Notebooks

Principal Component Analysis

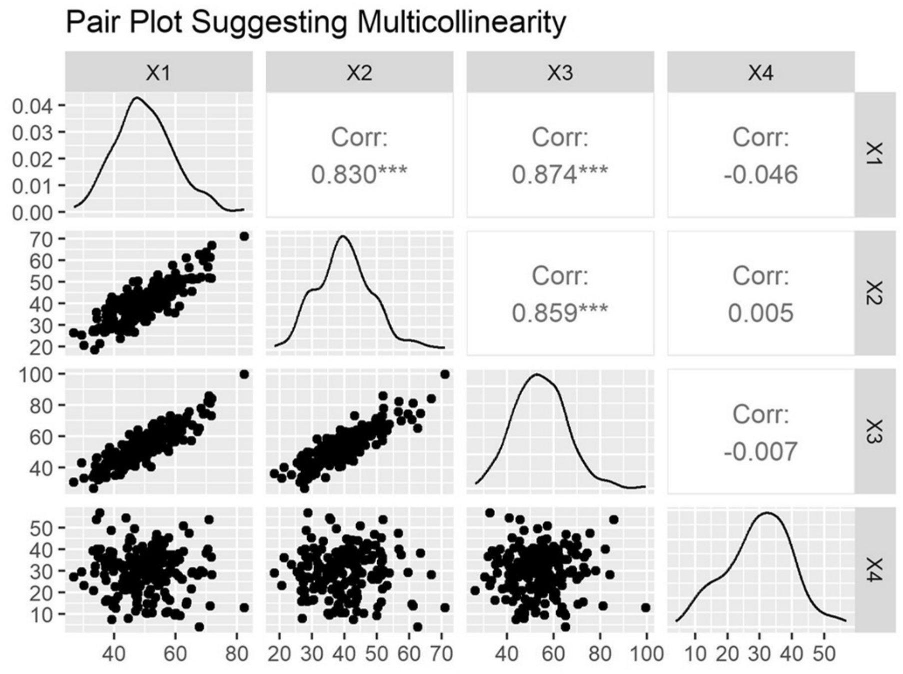
Understanding What, Why, How & When

Multicollinearity

Problem with connected features

What is Multicollinearity?

- Strong correlation among input features
- Redundant information in predictors



Real time examples of multicolinearity

- **Real Estate Pricing: square footage** , **number of rooms** to predict house price.

→Larger houses generally have more rooms, causing high correlation.

- **Economic Indicators: income** and **consumer spending** predict GDP,

→strongly rise or fall together.

- **Health and Fitness: body weight** and **body fat percentage to predict** cardiovascular health

→ they are frequently correlated.

- **Energy Consumption: family income** and **number of appliances to predict** electricity usage

→as higher income often leads to owning more appliances.

- **Retail/Sales: temperature** and **sunny hours** together to predict ice cream sales.

Impact on Regression: Coefficients

- Coefficients become unstable
- Small data changes → large coefficient changes
- Coefficient signs may flip
- Physical interpretation breaks

Impact on Regression: Prediction

- Overall prediction accuracy often unaffected
- Redundant features still encode signal
- Inference and sensitivity analysis become unreliable

Impact on Classification: Linear Models

- Logistic regression coefficients unstable
- Odds ratios unreliable
- Decision boundary often unchanged
- feature attribution becomes misleading

Impact on Tree-Based Models

- Prediction largely unaffected
- Trees select one of correlated features
- Feature importance diluted
- Multicollinearity hidden, not solved

When Multicollinearity is Critical

- Physical interpretation required
- Sensitivity studies
- Causal reasoning

Key Takeaway

- Multicollinearity does not break predictions.
- It breaks the story your model tells about the inputs.

What is PCA?

- **Definition:** A statistical technique that reduces dimensionality while preserving maximum variance
- **Core Concept:** Transforms correlated variables into new uncorrelated variables called *principal components*
- **Mathematical Basis:** Finds eigenvectors and eigenvalues from covariance matrix
- **Output:** Ranked principal components ordered by variance explained

Why Use PCA?

Reduce Complexity

Simplify high-dimensional data without losing key patterns

Improve Performance

Speed up ML models, reduce overfitting, enhance accuracy

Enable Visualization

Project data into 2D/3D for intuitive exploration

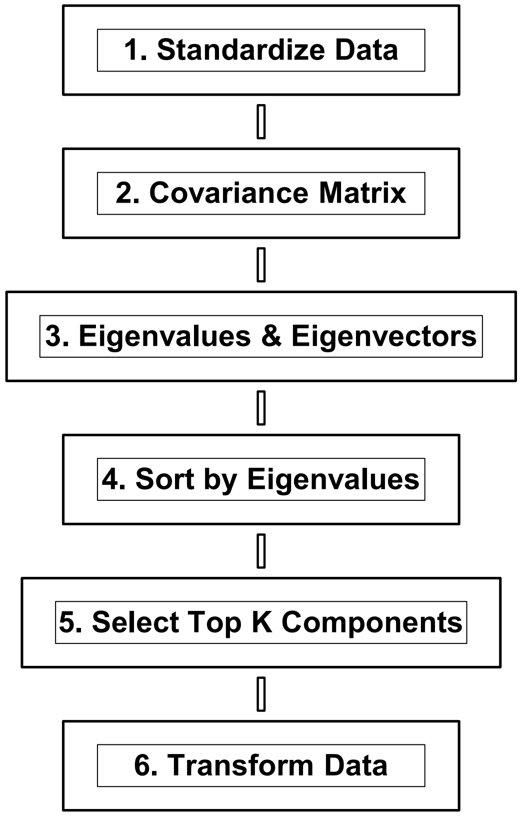
How Does PCA Work?

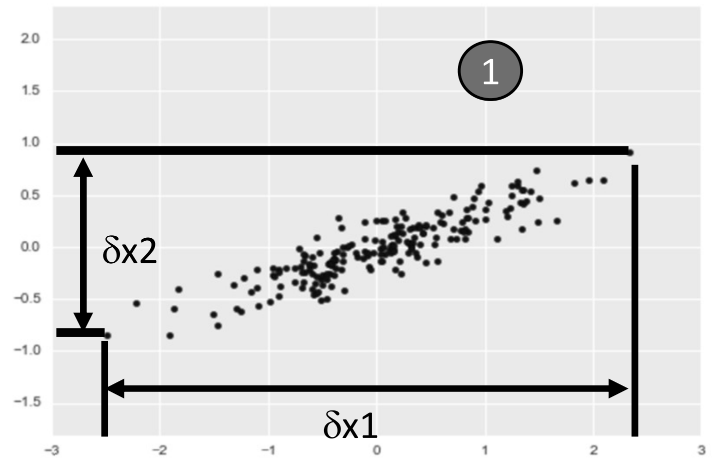
- **Standardize Data:** Normalize variables to mean=0, std=1 (essential for fair comparison)
- **Compute Covariance Matrix:** Calculate relationships between all variable pairs
- **Find Eigenvectors & Eigenvalues:** Eigenvectors are directions; eigenvalues measure variance captured

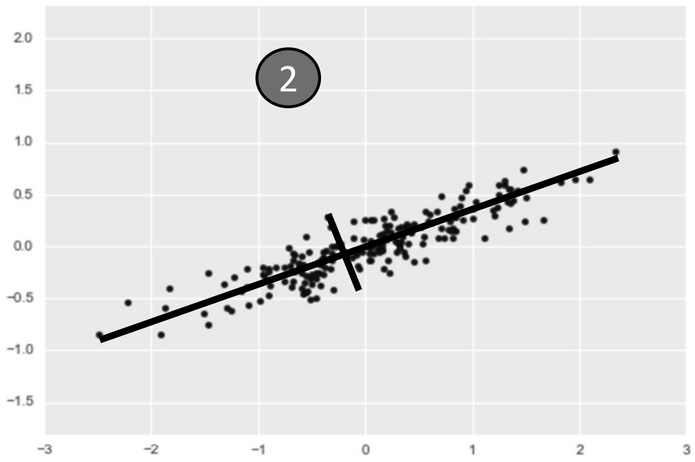
How Does PCA Work?

- **Sort by Variance:** Rank components by eigenvalue (largest variance first)
- **Select Components:** Choose top K components retaining 95%+ variance
- **Project Data:** Transform original data onto new principal component axes

Complete PCA Process





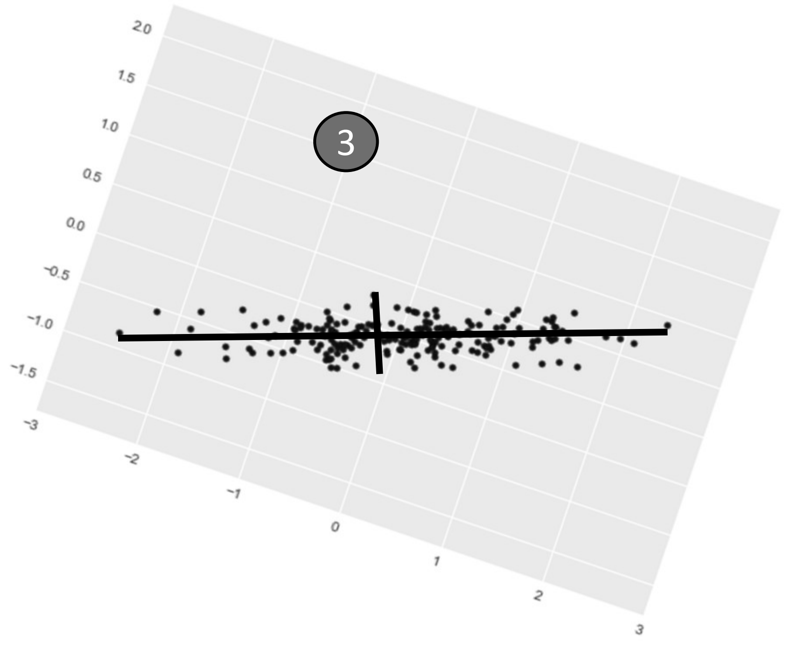


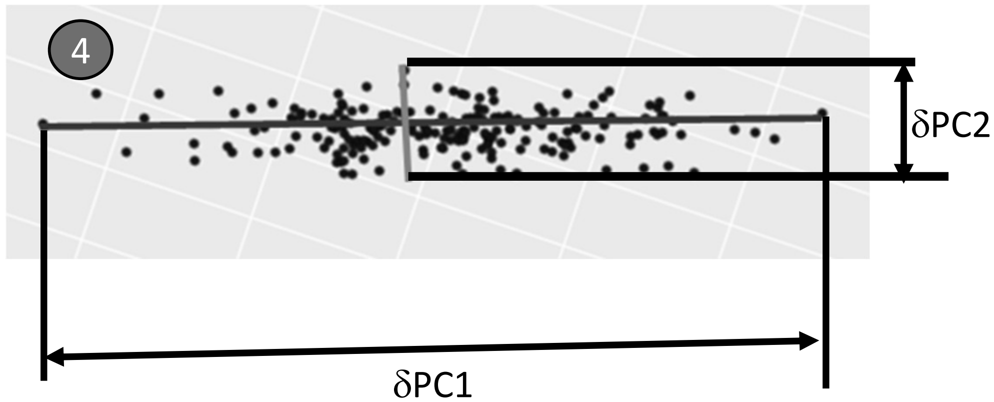
y= beta0+beta1*x1

PC1=w11x1+w12X2

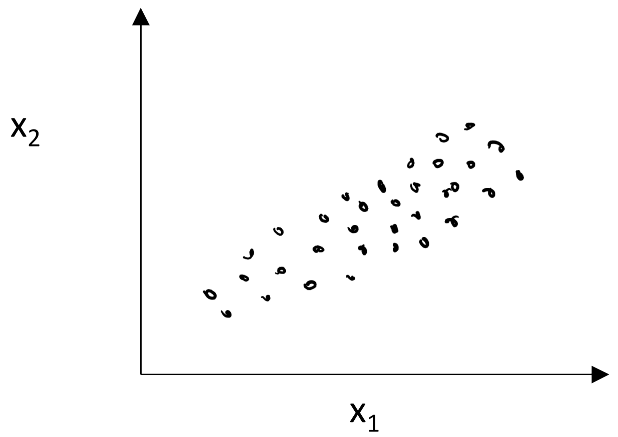
y= beta0'+beta1'*PC1

Correlated variables to uncorrelated Components

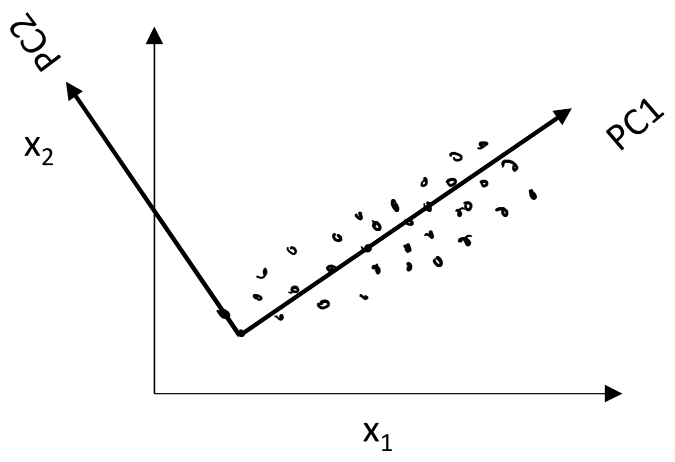




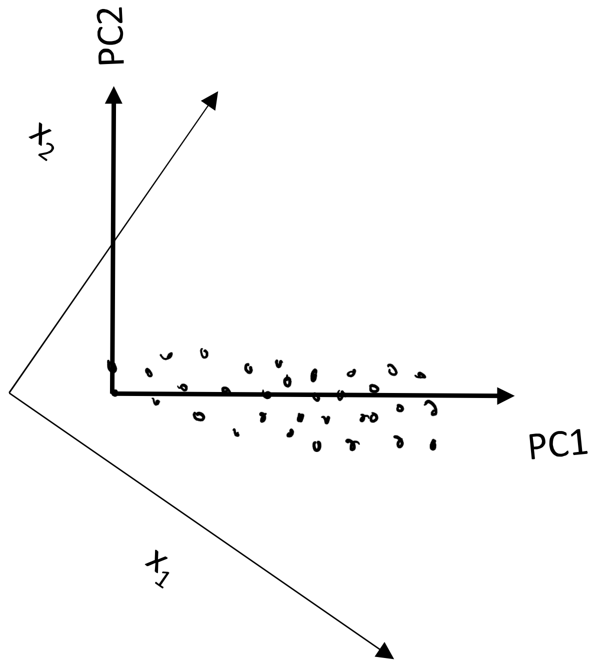
Note: no longer the data points are in terms of x_1,x_2. The plot is in terms of PC_1 and PC_2



x_2
x_1



PC2
x_2
PC1
x_1

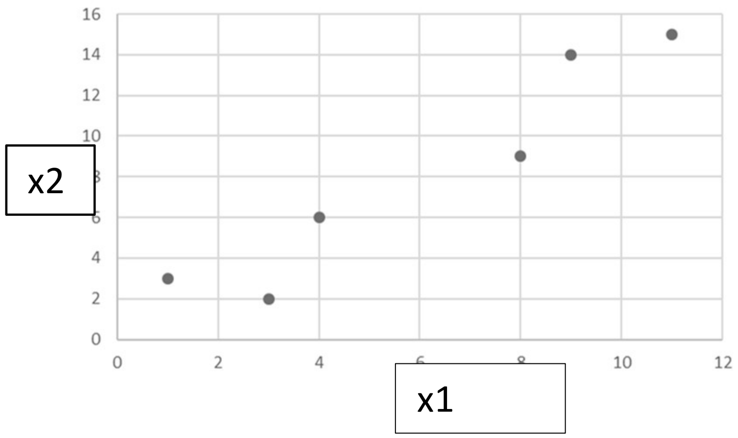


PC2
x_2
PC1
x_1

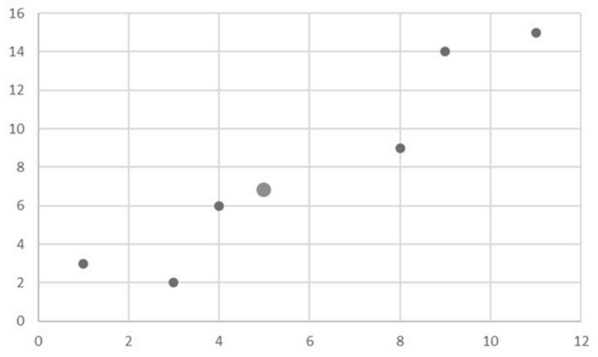
How PCA works

x1	1	4	3	8	9	11
x2	3	6	2	9	14	16

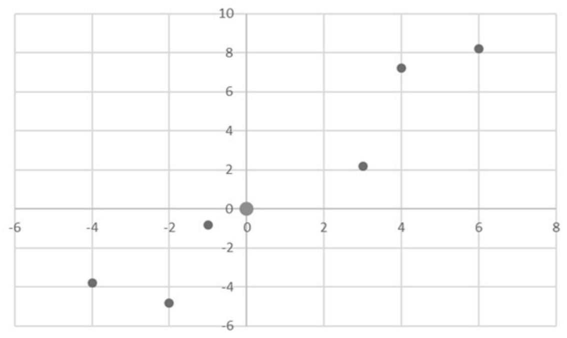
Step-1



Step-2

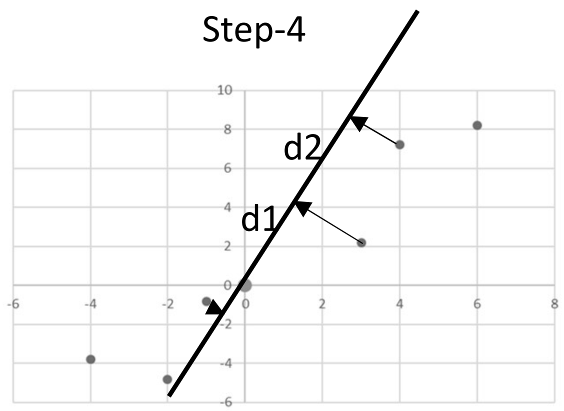


Step-3



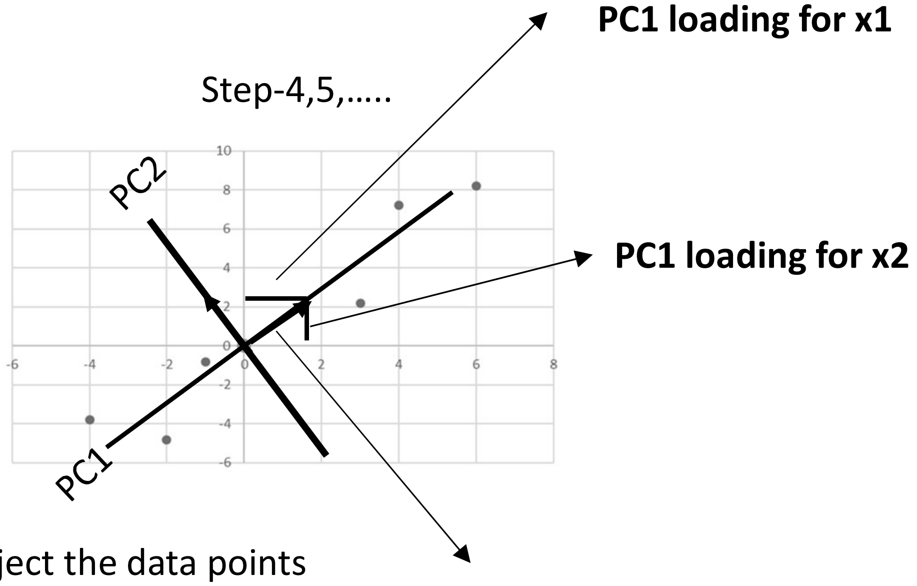
How PCA works

x1	1	4	3	8	9	11
x2	3	6	2	9	14	16



SSD = $d_1^2+d_2^2+....d_6^2$

Maximize (SSD)



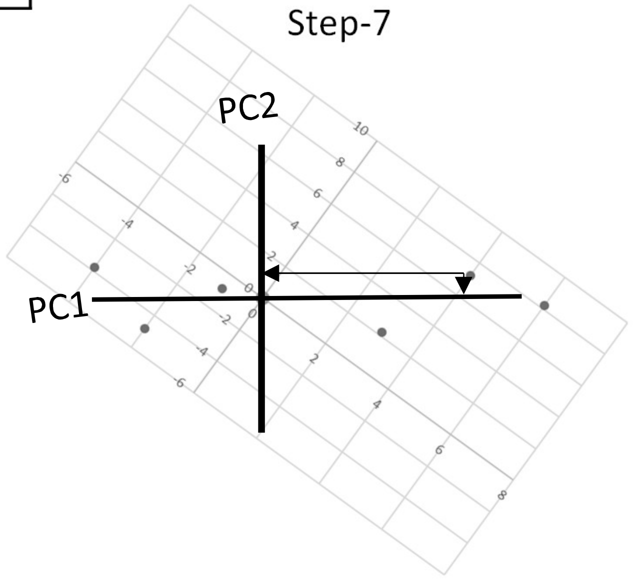
Step-4→ draw a random line passing through the origin and project the data points on the new line. Measure the distances of the projected points along the new line
Step-5→ rotate the fit line about the origin such that SSD is maximized. The maximum SSD line is the PC1 axis
Step-6→ draw an orthogonal to PC1 to get PC2

Eigen vector

Eigen Value → SSD

How PCA works

x1	1	4	3	8	9	11
x2	3	6	2	9	14	16

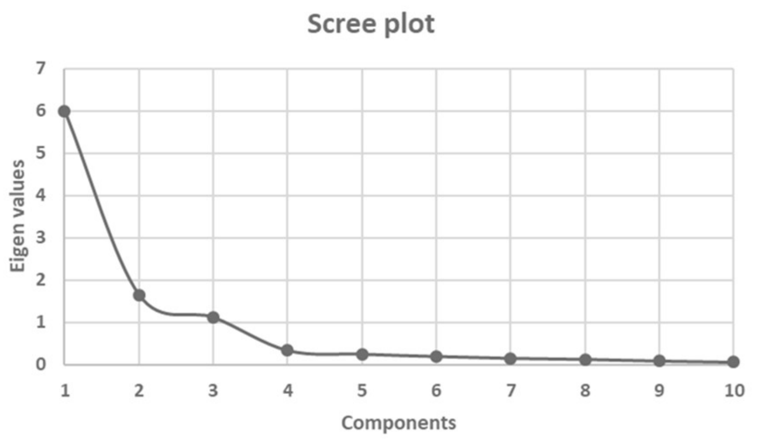


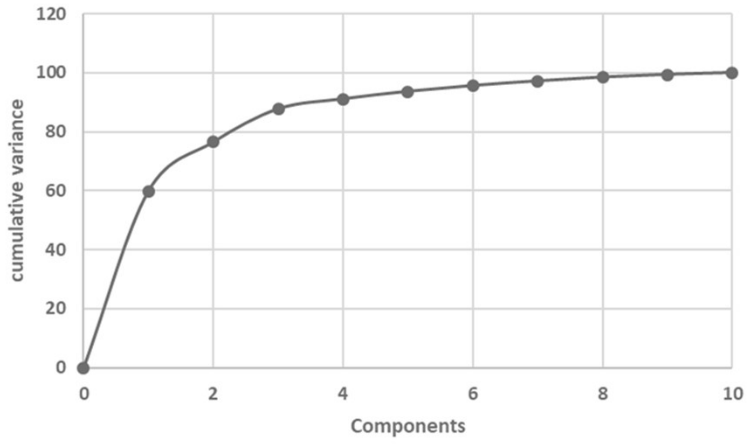
Step-7: rotate such that PC1 is horizontal

Loadings=Eigenvectors×sqrt (Eigenvalues)

Determining the number of PCs/Fs

Component	Initial Eigenvalues		
	Total	% of Variance	Cumulative %
1	5.994	59.938	59.938
2	1.654	16.545	76.482
3	1.123	11.227	87.709
4	.339	3.389	91.098
5	.254	2.541	93.640
6	.199	1.994	95.633
7	.155	1.547	97.181
8	.130	1.299	98.480
9	.091	.905	99.385
10	.061	.615	100.000





Hands-on

- PCA for DR in character recognition

- PCA for Noise reduction

When to Use PCA?

- High-dimensional data (more features than samples)
- Correlated features causing redundancy
- Need to visualize complex data in 2D/3D
- Machine learning preprocessing to reduce overfitting
- Computational efficiency is critical

When NOT to Use PCA?

- Features are already uncorrelated and interpretable
- Domain requires interpretability of individual features
- Low-dimensional data (curse of dimensionality doesn't apply)
- Nonlinear relationships dominate (use t-SNE or UMAP instead)
- Categories are highly imbalanced or overlapping

Example 1: Image Compression

Scenario: A semiconductor wafer inspection system captures 1000×1000 pixel SEM images (1M dimensions)

- PCA extracts top 50 principal components capturing 98% variance
- Reduces storage by **95%** (1M → 50 dimensions)
- Enables faster defect detection algorithms
- Reconstructed images show minimal quality loss

Example 2: multi-LiDAR data fusion

Scenario: Lidar data from self driven vehicles (3-6 sensors, 1000s samples)

- Each LiDAR outputs 100K+ points/sec × 6 sensors = 600K+ dimensions. Raw data overwhelms compute

- PCA extracts top 50-200 principal components capturing 95%+ varianc.

- High reconstruction error flags dirty lenses, misalignment, or occlusions

Key Advantages of PCA

✓ Linear & Fast
Computationally efficient, scales to large datasets

✓ Variance Focus
Captures maximum information in fewer dimensions

✓ Unsupervised
Works without labeled data

Key Limitations of PCA

✗ Linear Only

Misses nonlinear patterns in data

✗ Interpretability

PCs are combinations, not original features

✗ Outlier Sensitivity

Outliers inflate variance, distort components

Quick PCA Implementation (Python)

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
# Standardize data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Apply PCA
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)
# Variance explained
print(pca.explained_variance_ratio_)
```

PCA Best Practices

- **Always standardize** data before PCA (unit variance)
- **Handle outliers** using robust scaling or removal
- **Choose components** retaining 95%+ cumulative variance
- **Validate** on test data; track reconstruction error

Key Takeaways

✓ **WHAT:** Linear transformation reducing dimensions via uncorrelated principal components

✓ **WHY:** Simplify data, speed models, enable visualization, reduce noise

✓ **HOW:** Standardize → Covariance matrix → Eigenvectors → Project data

✓ **WHEN:** High-dimensional, correlated data needing efficiency or visualization