UNIVERSITÀ DEGLI STUDI DI TRIESTE

Class 3: Python ML Project Principal Components Analysis

Master Course:

Data-driven Systems Engineering (ML Operations)

440MI and 305SM

Agenda & Learning Goals

- PCA & feature engineering
- From prototype ⟶ pickle ⟶ service
- Real-world ML ecosystem

Feature Engineering

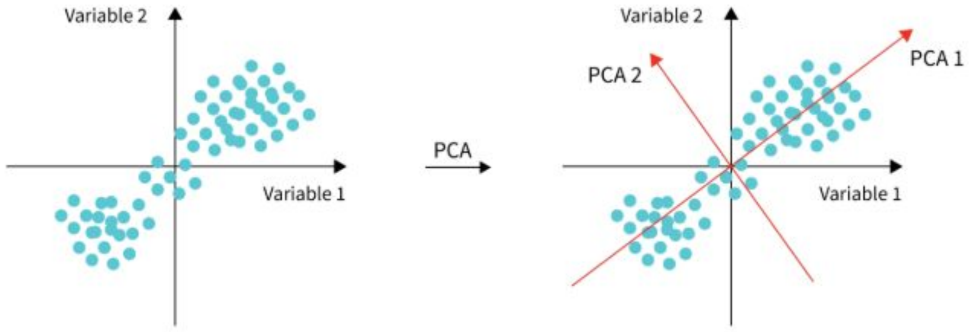
- Creating new variables from raw data
- Handling categorical/numerical features
- Scaling, encoding, transformations

Why Dimensionality Reduction?

- Too many features $\longrightarrow$ overfitting
- Computational cost
- Visualization of high-dimensional data

PCA (Principal Component Analysis) – Intuition

- Projects data onto fewer dimensions
- Maximizes variance retention
- Creates orthogonal principal components



[Reference: Jolliffe, I.T. Principal Component Analysis, 2002]

PCA Advantages & Limitations

- Advantages: reduces noise, speeds up training, useful for visualization
- Limitations: loses interpretability, assumes linearity

PCA Step-by-Step

Consider a dataset with three observations and two features:

$$X = \begin{bmatrix} 2.5 & 2.4 \\ 0.5 & 0.7 \\ 2.2 & 2.9 \end{bmatrix}$$

Step 1: Standardizing the Data:

$$\mu_1 = \frac{(2.5 + 0.5 + 2.2)}{3} = 1.73, \quad \mu_2 = \frac{(2.4 + 0.7 + 2.9)}{3} = 2.0$$

$$X_{centered} = \begin{bmatrix} 0.77 & 0.4 \\ -1.23 & -1.3 \\ 0.47 & 0.9 \end{bmatrix}$$

PCA Step-by-Step

Step 2: Compute the Covariance Matrix

$$C = \frac{1}{n-1} X_{centered}^T X_{centered}$$

$$C = \begin{bmatrix} 0.62 & 0.59 \\ 0.59 & 0.71 \end{bmatrix}$$

Step 3: Compute the Eigenvalues and Eigenvectors

Solving det(C − λI) = 0 gives eigenvalues, considering I is the identity matrix of the same size as C:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C - \lambda I = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = \begin{bmatrix} c_{11} - \lambda & c_{12} \\ c_{21} & c_{22} - \lambda \end{bmatrix}$$

Eigenvalues:

$$\lambda_1 = 1.29, \quad \lambda_2 = 0.04$$

Eigenvectors:

$$v_1 = \begin{bmatrix} 0.67 \\ 0.74 \end{bmatrix}, \quad v_2 = \begin{bmatrix} -0.74 \\ 0.67 \end{bmatrix}$$

PCA Step-by-Step

Transform the Data:

$$X = \begin{bmatrix} 2.5 & 2.4 \\ 0.5 & 0.7 \\ 2.2 & 2.9 \end{bmatrix} \qquad X_{centered} = \begin{bmatrix} 0.77 & 0.4 \\ -1.23 & -1.3 \\ 0.47 & 0.9 \end{bmatrix} \qquad v_1 = \begin{bmatrix} 0.67 \\ 0.74 \end{bmatrix}$$

$$X_{PCA} = X_{centered} \cdot v_1$$

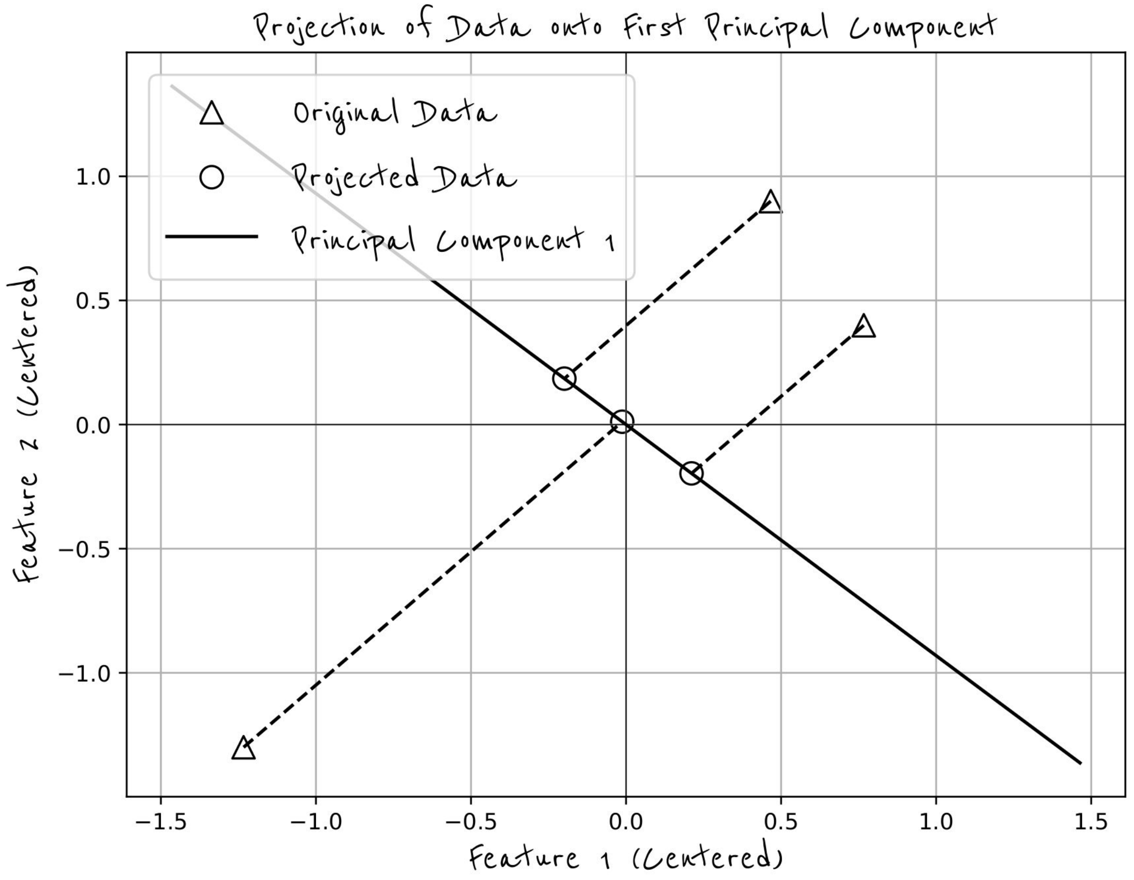
$$X_{PCA} = \begin{bmatrix} 0.89 \\ -1.83 \\ 0.94 \end{bmatrix}$$

Loading Matrix

$$\begin{bmatrix} 0.67 & -0.74 \\ 0.74 & 0.67 \end{bmatrix}$$

PCA Step-by-Step

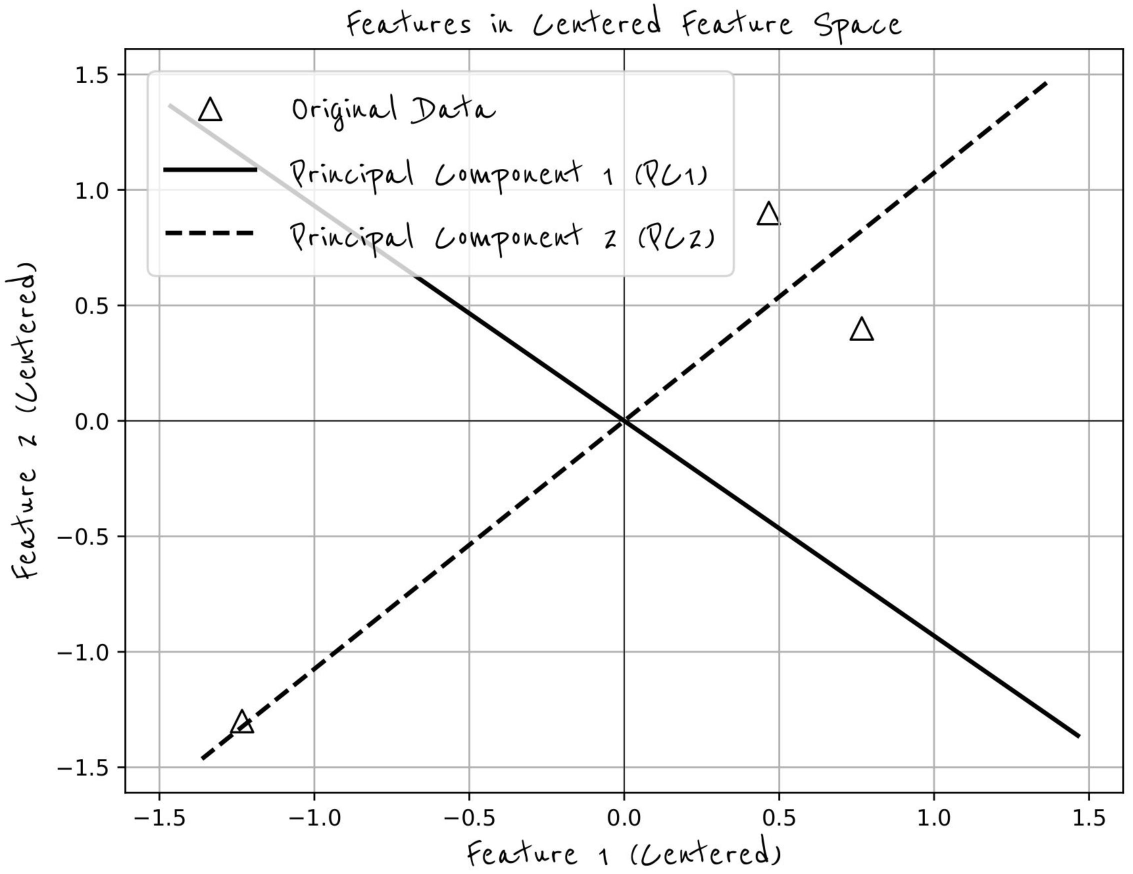
Visual analysis



$$X_{centered} = \begin{bmatrix} 0.77 & 0.4 \\ -1.23 & -1.3 \\ 0.47 & 0.9 \end{bmatrix}$$

PCA Step-by-Step

Visual analysis

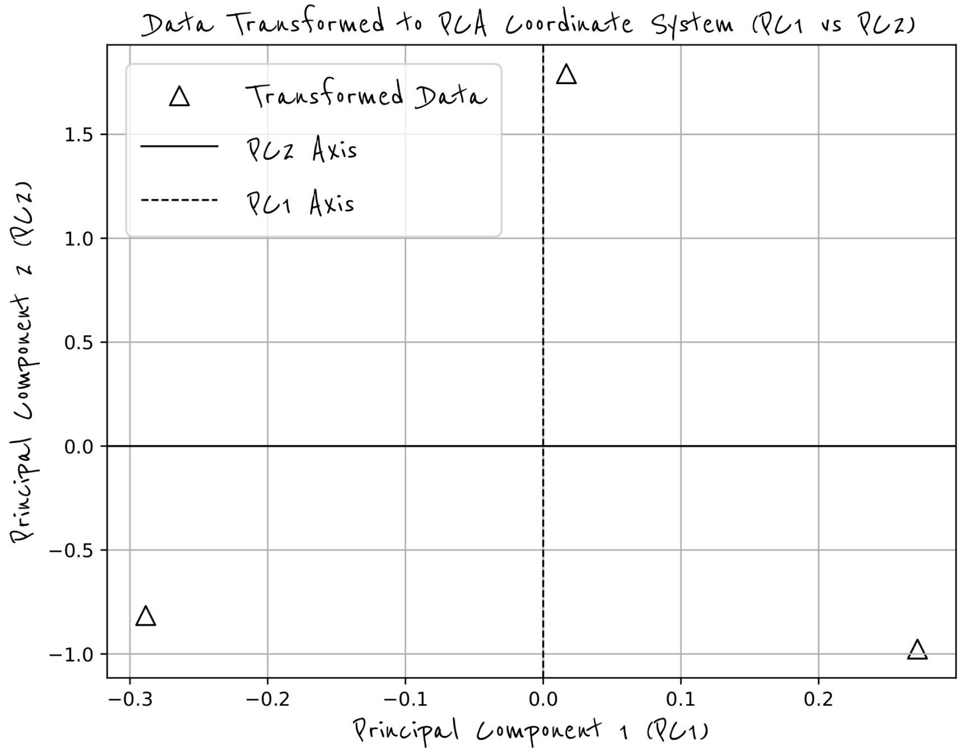


$$X_{centered} = \begin{bmatrix} 0.77 & 0.4 \\ -1.23 & -1.3 \\ 0.47 & 0.9 \end{bmatrix}$$

$$v_1 = \begin{bmatrix} 0.67 \\ 0.74 \end{bmatrix}$$

PCA Step-by-Step

Visual analysis

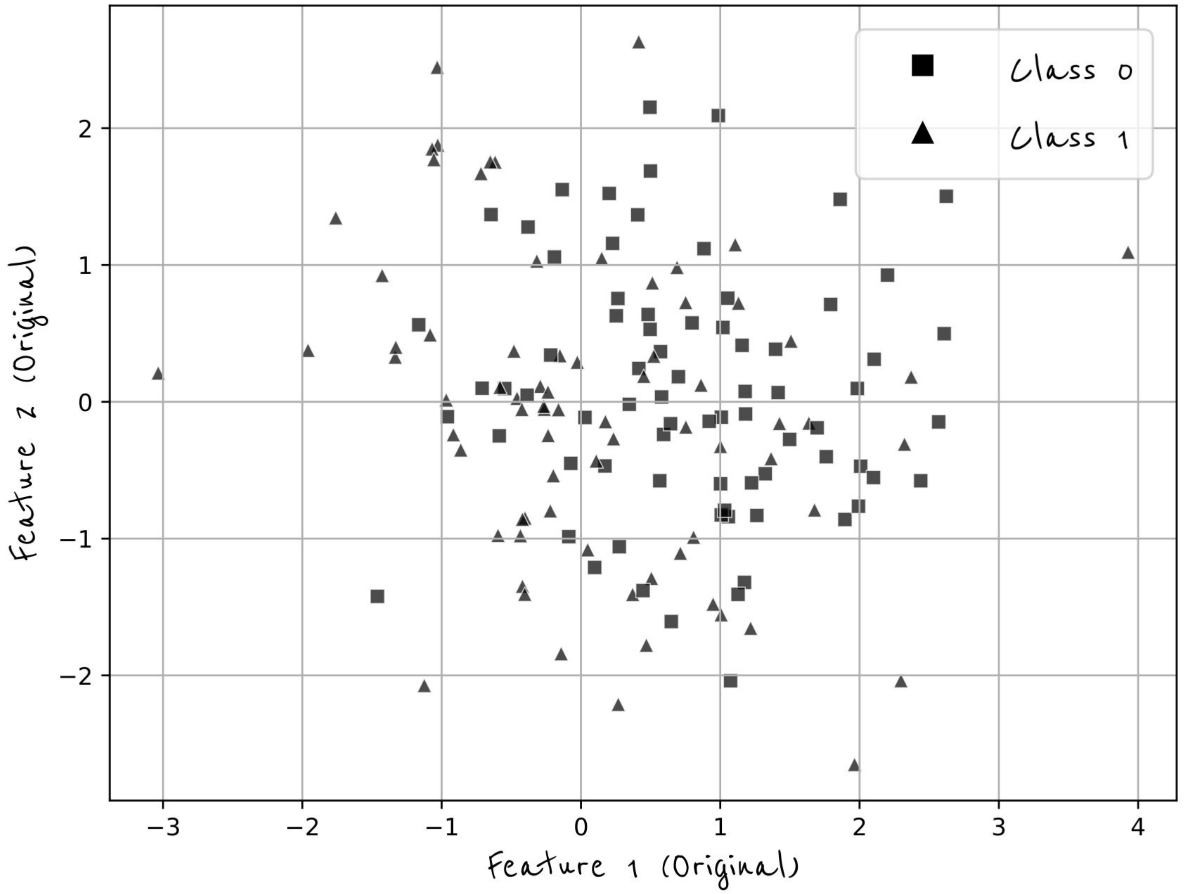


X_PCA = X_centered . v1
= [0.89, -1.83, 0.94]

X_PCA = X_centered . v2
= [-0.30, 0.04, 0.25]

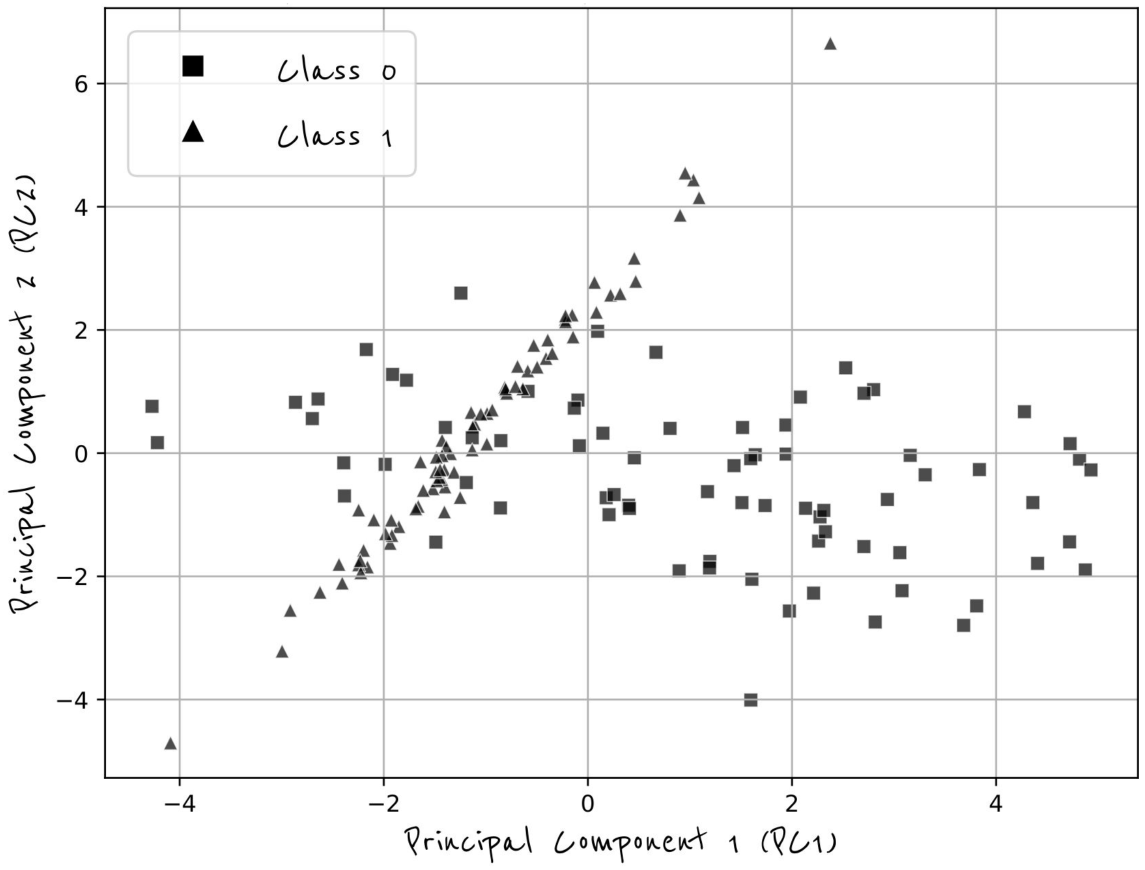
PCA Step-by-Step

Exemple 1:



PCA Step-by-Step

Exemple 1:



PCA Example (Conceptual)

- Dataset: 10 features → reduced to 2 PCs
- 95% variance retained
- Visualization in 2D scatter plot

From Notebook to Prototype

- Jupyter Notebook for experimentation
- Transition to scripts/modules for reproducibility

Model Persistence (Pickle/Joblib)

- Save trained models to disk
- Reload without retraining
- Standard practice in scikit-learn

Risks of Pickle

- Security risks (arbitrary code execution)
- Not cross-platform friendly

Alternatives to Pickle

- ONNX (Open Neural Network Exchange)
- TensorFlow SavedModel
- PyTorch TorchScript

Example Workflow

- Train model
  - → Save with Pickle
  - → Load in Flask API
  - → Serve predictions

Summary & Key Takeaways

- Python dominates ML due to ecosystem & simplicity
- Pipelines & KDD structure workflows
- Data quality & EDA are critical steps
- PCA helps reduce complexity
- Prototypes → pickle → services = production cycle
- MLOps ensures scalability & reliability