Artificial Intellligence: An Introduction

Wei Wang(王伟)

Engineering Research Centre of Applied Technology on Machine Translation and Artificial Intelligence , Macao Polytechnic University

weiwang@mpu.edu.mo; 匯智樓 (WUI CHI)-5/F, N56

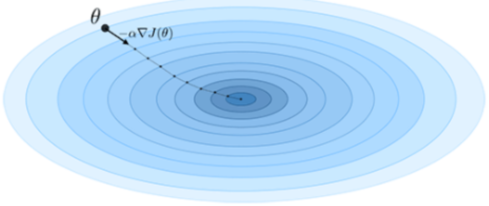
Mar. 02, 2026

AI Models Types

AI models refer to different approaches or frameworks that are used to represent and solve problems in the field of AI.

These models provide a structured way to understand and analyze complex systems and make intelligent decisions.

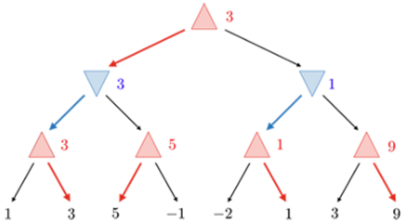
Reflex



- Feature vector, score, margin, residual
- Linear predictors, non-linear predictors
- Loss minimization, stochastic gradient descent, backpropagation, regularization

PDF Web

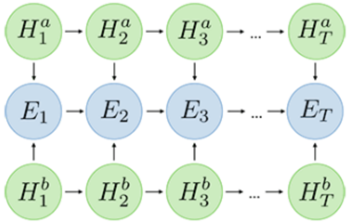
States



- Tree search, graph search, A star search
- Learning costs, relaxation
- Markov decision processes, game playing

PDF Web

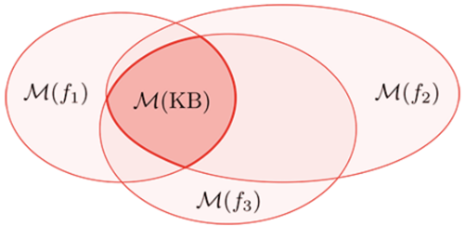
Variables



- Constraint satisfaction problem, backtracking search
- Beam search, Gibbs sampling
- Bayesian networks, probabilistic program, inference

PDF Web

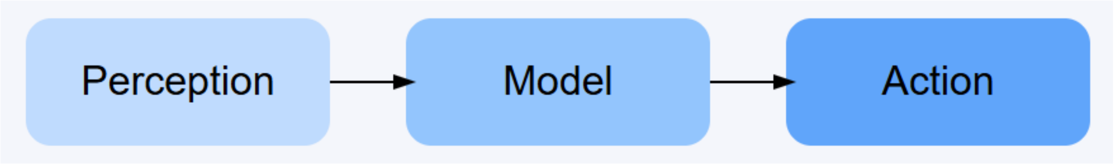
Logic



- Notations, knowledge base, forward inference, rule properties
- Propositional logic, modus ponens, resolution rule, conjunctive normal form
- First-order logic, substitution, unificaion

PDF Web

Source: https://stanford.edu/~shervine/teaching/cs-221/



AI models define how an agent perceives, reasons, and acts.

Different models suit different environments and tasks.

AI Models Types

1. Logic-based models

- Symbolic representation of classes of objects.
- Deductive Reasoning.
- **Apps**: Question Answering Systems, Natural Language Understanding, Expert system
- **Options**: Propositional Logic , First-Order Logic, Knowledge Base.

3. Variables-based models (Uncertainty)

- Solution in an assignment of values for a set of variables.
- **Apps**: Soduko, Speech Recognition, and Face Recognition.
- **Options**: Convolutional Neural Networks, Constraint Satisfaction, Bayesian Networks, Factor Graphs, and Dynamic Ordering.

2. States-based models

- Solutions are defined as a sequence of steps.
- Model a task as a graph of states and a solution as a path in the graph.
- A state captures all of the relevant information about the past in order to act in the future.
- **Apps**: Navigation and Games.
- **Options**: Tree Search (Breadth-first search, Depth-first search, and Iterative deepening), Graph search (Dynamic programming), Markov decision processes, Game playing

4. Reflex-based models

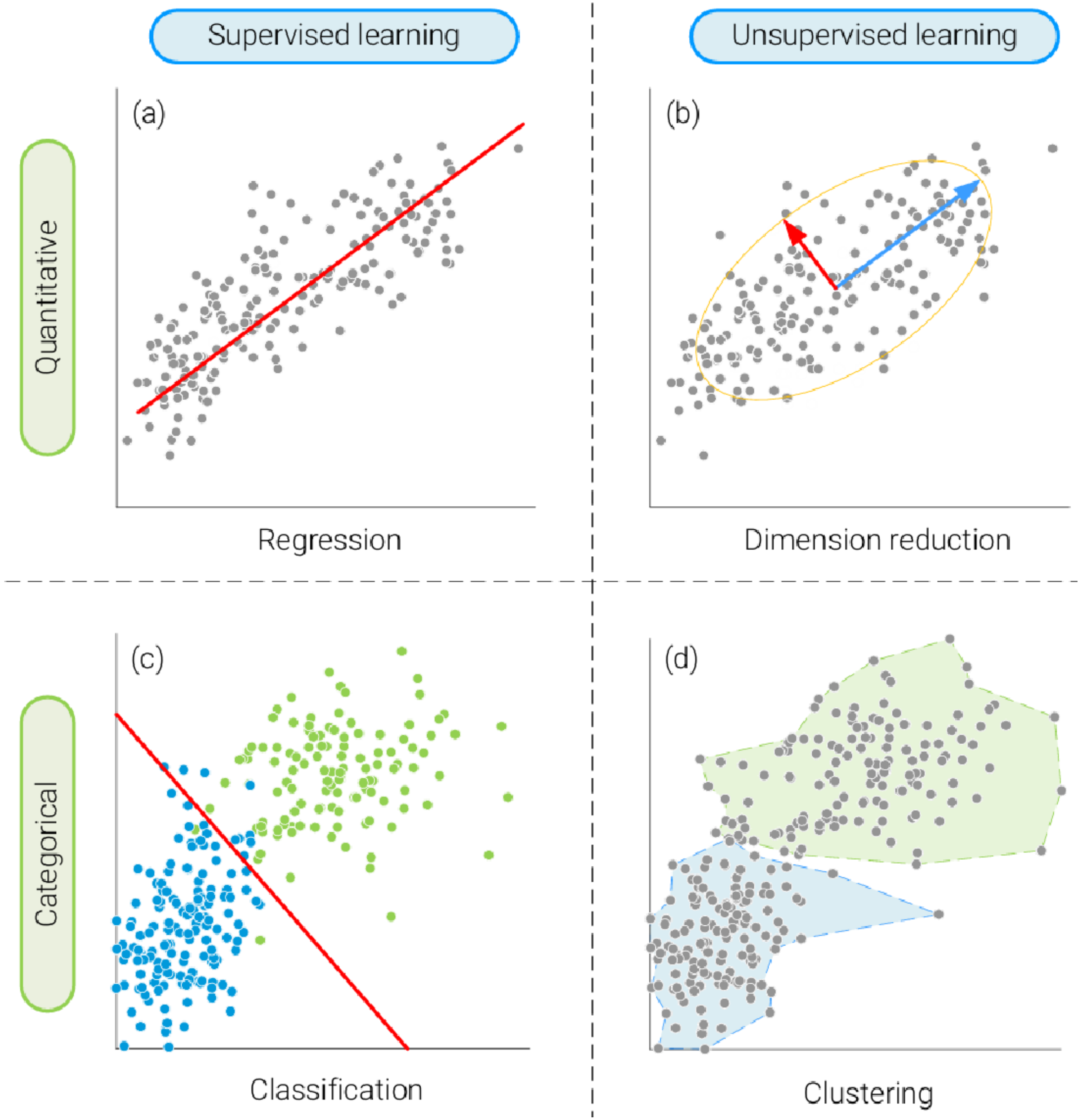
- Given a set of <Input, Output> pairs of training data, learn a set of parameters that will map input to output for future data.
- **Apps**: Classification and Regression.
- **Options**: Artificial Neural Networks (ANN), Decision Trees, Support Vector Machines, Regression, Principal Component Analysis, K-Means Clustering, and K-Nearest Neighbor

Outline

- Reflex-based Models
  Dimension Reduction

AI Models Types

Four Main Machine Learning Methods



AI Models Types

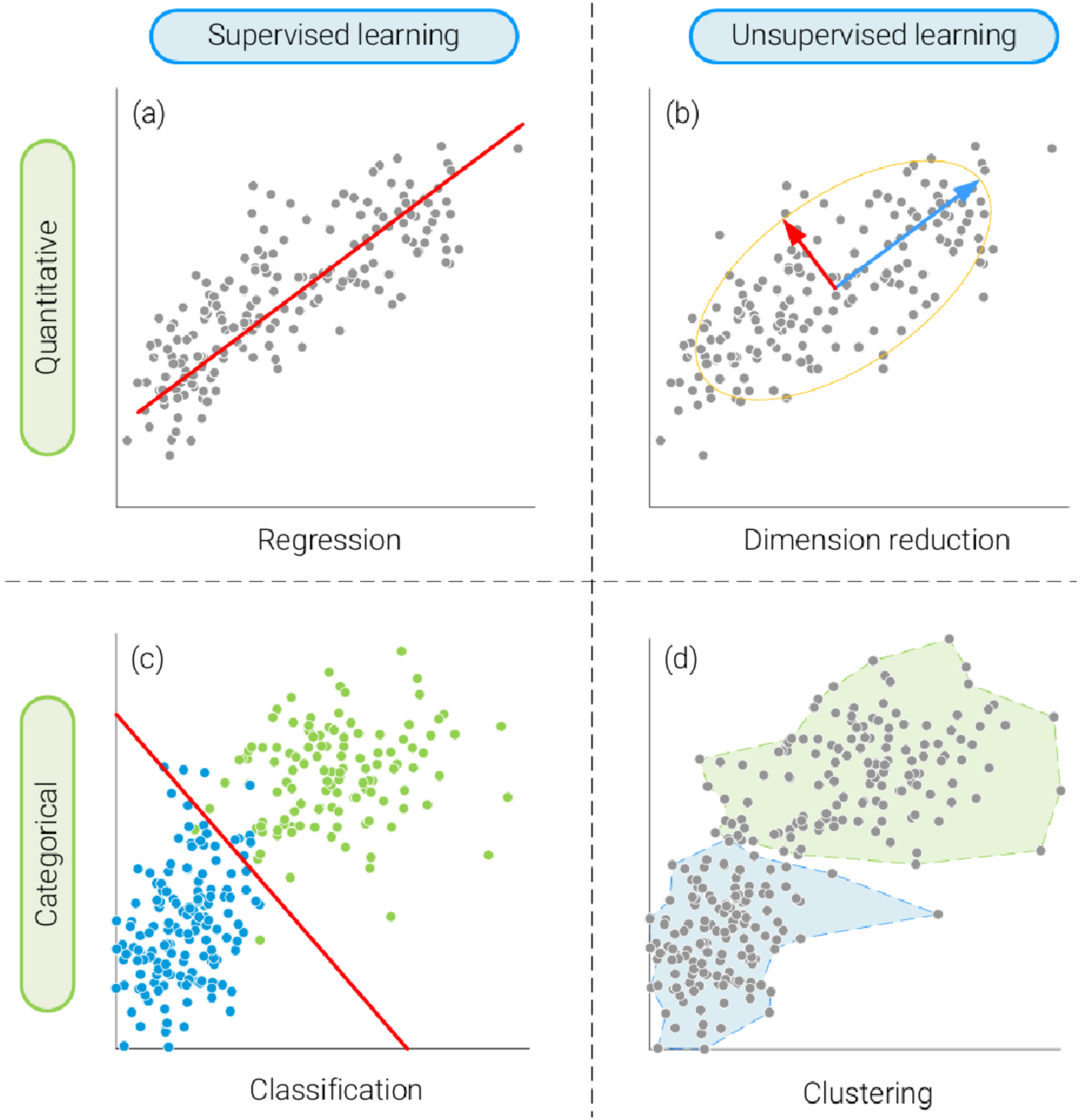
The Big Picture - Learning Paradigms

Supervised Learning: Data comes with Labels (Input X + Output Y)

- → Regression
- → Classification

Unsupervised Learning: Data has No Labels (Input X only)

- → Clustering
- → Dimension Reduction



Why dimension reduction

Modern AI systems deal with:

Images (100,000+ features)

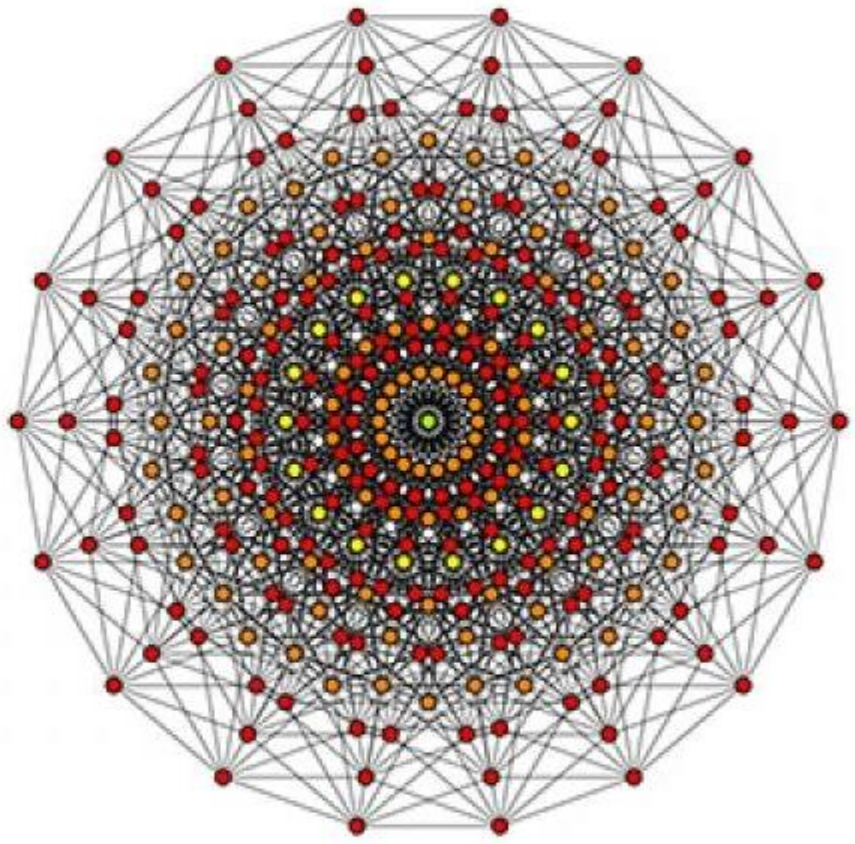
Text (10,000–1,000,000 features)

Genomic data (20,000+ features)

Sensor data (IoT streams)

Embeddings (512–4096 dimensions)

High dimensional data is everywhere.



Example 1: Images

A color image of size:

$$224 \times 224 \times 3 = 150,528 \text{ features}$$

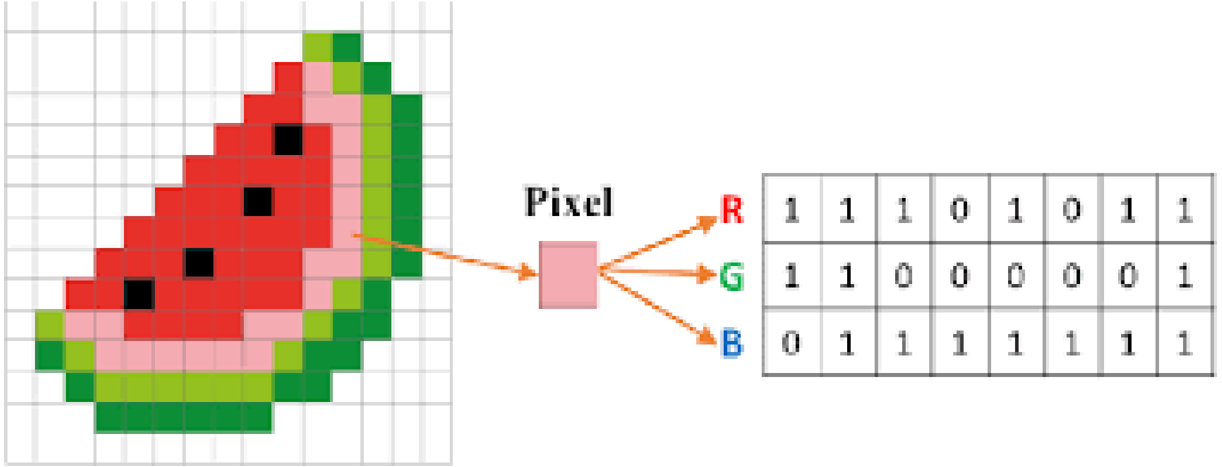
Each pixel is a feature.

If we have:

10,000 images

150,528 dimensions

High dimensional data is everywhere.



The Curse of Dimensionality

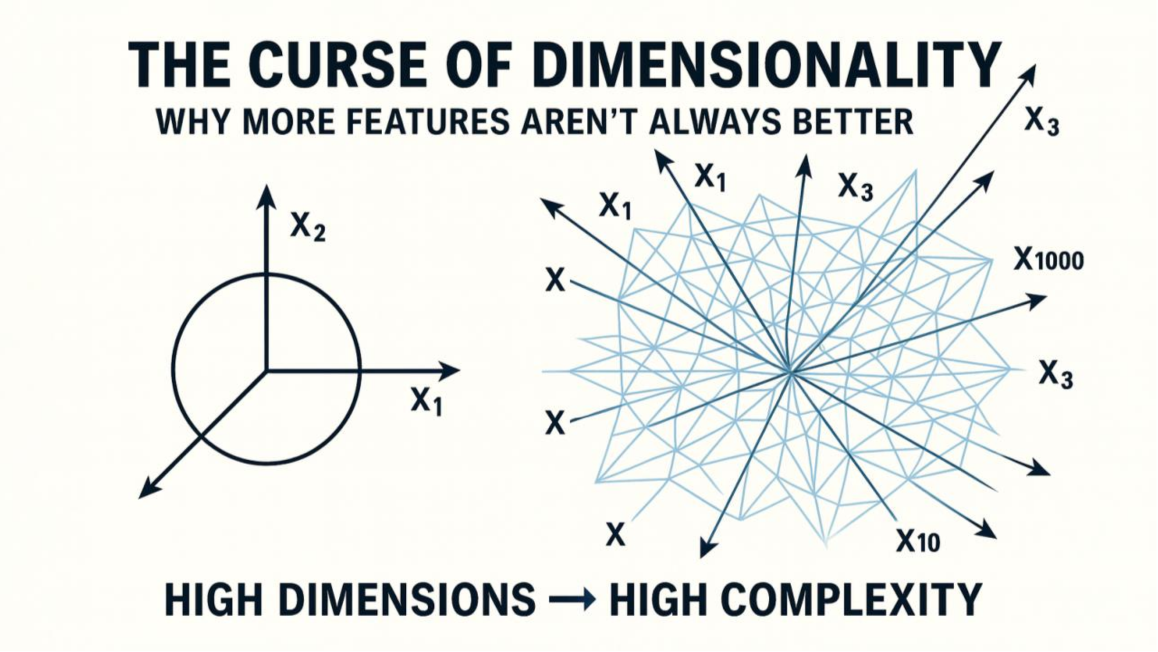
As dimension d increases:

Data becomes sparse

Distance metrics lose meaning

Volume increases exponentially

We need exponentially more data to cover space.



Distance Becomes Meaningless

In high dimensions:

The difference between nearest and farthest neighbor shrinks.

All points become almost equally distant.

$$\frac{D_{max} - D_{min}}{D_{min}} \to 0 \quad \text{as} \quad d \to \infty$$

Implication:

KNN becomes unreliable

Clustering becomes unstable

Similarity loses meaning

Overfitting Explosion

High dimensions → more parameters → higher model complexity

If:

Features = 10,000

Samples = 1,000

Model can memorize data easily.

Generalization error increases.

Bias-variance tradeoff:

$$\text{Total Error} = \text{Bias}^2 + \text{Variance} + \text{Noise}$$

High dimensions → High variance.

Dimension Reduction

Computational Cost

Training complexity often depends on features.

Example:

Linear regression complexity: $O(nd^2)$

Memory cost: $O(nd)$

Dimensionality reduction reduces:

- **CPU time**
- **GPU memory**
- **Storage**
- **Energy consumption**

Noise and Redundancy

In real datasets:

- Many features are correlated
- Many features are irrelevant
- Many features are pure noise

Dimensionality reduction reduces:

- **Removes redundancy**
- **Removes noise**
- **Keeps informative components**

Dimension Reduction

Visualization Problem

Humans can only visualize:

1D, 2D, 3D

But real data may be:

- 300D (word embeddings)
- 2048D (CNN features)
- 4096D (LLM embeddings)

We need projection:

- Understand clusters $\mathbb{R}^{1000} \to \mathbb{R}^2$
- Detect outliers
- Analyze structure

Generalization Improvement

Reducing dimension:

- Simplifies hypothesis space
- Reduces overfitting
- Improves robustness

Intuition:

High-dimensional model = complex boundary

Low-dimensional model = smoother boundary

Simpler models generalize better.

Occam's Razor principle:

Simpler explanations are preferred.

Dimensionality Reduction

Dimensionality reduction is a technique used in machine learning and data analysis to reduce the number of input variables or features in a dataset while preserving the essential information.

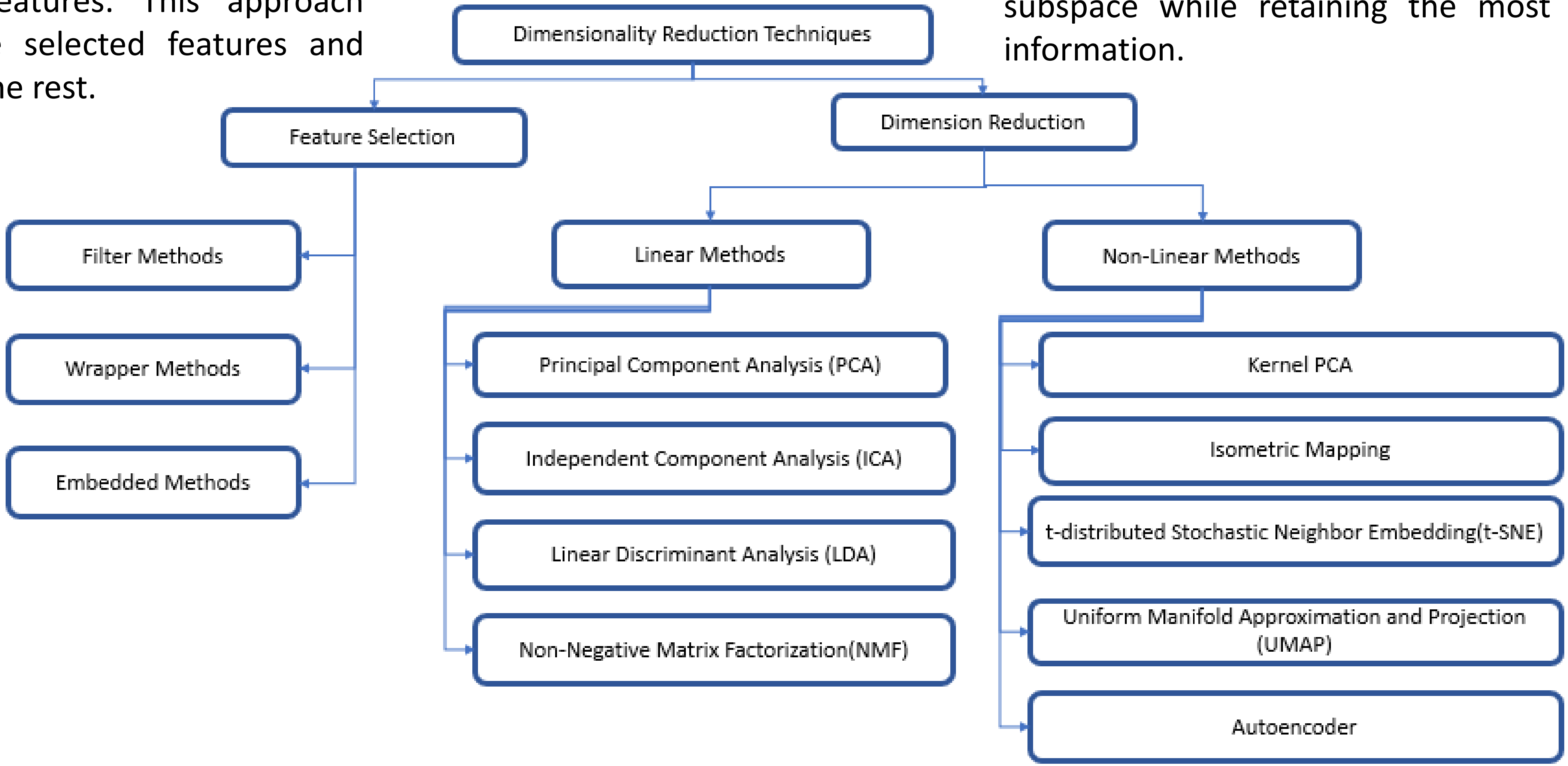
Here are some common reasons why dimensionality reduction is used:

- **Curse of Dimensionality**: As the number of features or dimensions in a dataset increases, the amount of data required to generalize accurately also increases. This can lead to overfitting and increased computational complexity.
- **Improved Visualization**: High-dimensional data is difficult to visualize effectively. By reducing the dimensions, data can be more easily plotted and understood.
- **Feature Engineering**: Dimensionality reduction can help in feature selection by identifying the most relevant features and eliminating redundant or noisy ones.

Dimensionality Reduction Techniques

Involves selecting a subset of the original features. This approach keeps the selected features and discards the rest.

Involves transforming the original features into a lower-dimensional space. This transformation is typically done by projecting the data onto a new subspace while retaining the most important information.



Feature Selection

What is Feature Selection?

Select a subset of original features:

$$X = (x_1, x_2, ..., x_d)$$

Choose:

$$(x_{i_1}, x_{i_2}, ..., x_{i_k}), \quad k < d$$

No transformation — just selection.

Why?

Remove irrelevant features

Remove redundant features

Improve interpretability

Types of Feature Selection

1. Filter Methods

Independent of model.

Examples:

- Correlation
- Mutual Information
- Chi-square test
- Variance threshold

Fast and scalable.

2. Wrapper Methods

Use a model to evaluate subsets.

Examples:

- Recursive Feature Elimination (RFE)

More accurate

More computationally expensive

3. Embedded Methods

Feature selection occurs during training.

Examples:

- LASSO regression
- Decision trees
- Random forests

Types of Feature Selection

Dimensionality Selection Techniques: Subset Selection

- There are 2^d subsets of d features
- *Forward search methods*: Add the best feature at each step
  - Set of features F initially Ø.
  - At each iteration, find the best new feature

    $j = \text{argmin}_i E (F \cup x_i)$
  - Add x_j to F if $E (F \cup x_j) < E (F)$
  - Greedy hill climbing approach

- *Backward search methods*: Start with all features and remove one at a time, if possible.
- *Floating search methods:* (Add k, remove l)

Feature Extraction

What is Feature Extraction?

Transform Data:

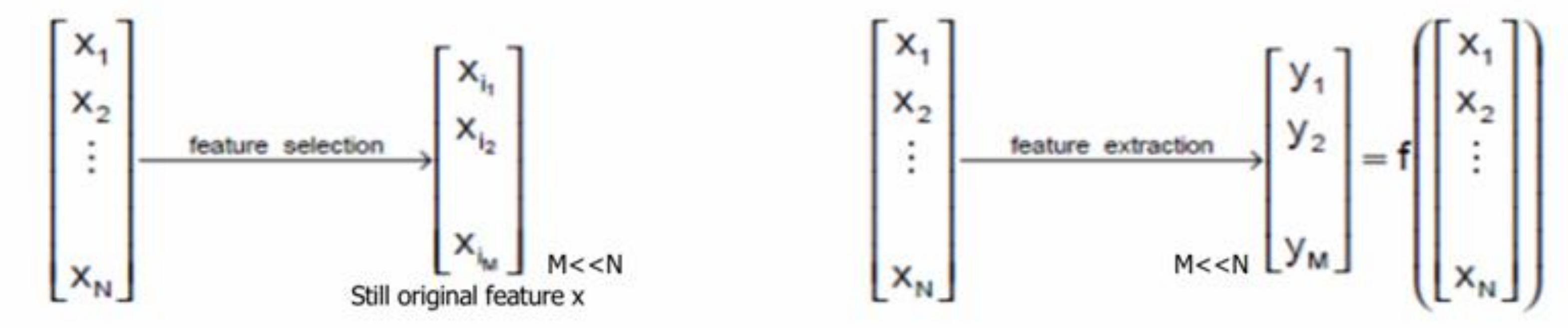
$$X \in \mathbb{R}^d$$

Into:

$$Z \in \mathbb{R}^k, \quad k < d$$

Where:

$$Z = f(X)$$



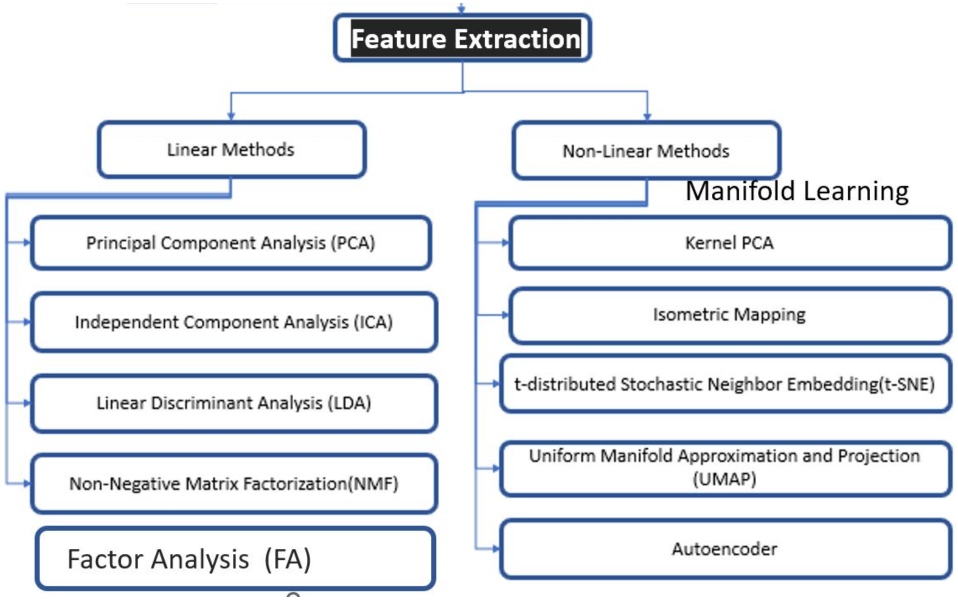
New features are combinations of old ones.

Feature Extraction

Dimensionality extraction methods can broadly be categorized into linear and non-linear techniques based on how they map the high-dimensional data to a lower-dimensional space.

Comparison:

- Linear methods are often faster and easier to interpret but may not capture complex non-linear relationships in the data.
- Non-linear methods can capture complex structures in the data but may be computationally expensive and harder to interpret.



The choice between linear and non-linear methods depends on the nature of the data and the specific goals of the analysis. Linear methods are often a good starting point for dimensionality reduction, while non-linear methods can be employed when the data exhibits complex non-linear relationships.

Feature Extraction

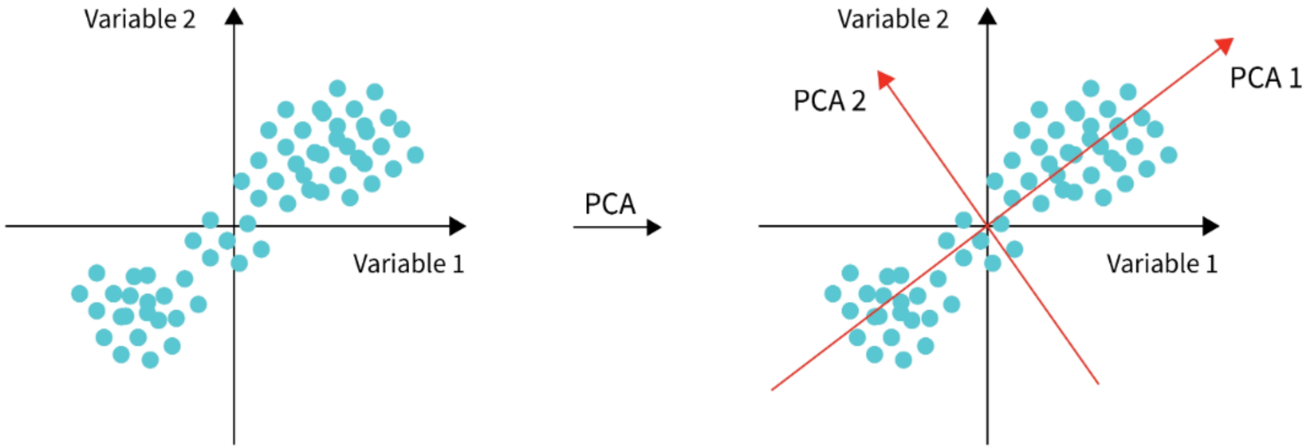
Common techniques for dimensionality extraction include:

- **Principal Component Analysis (PCA)**: A linear technique that finds the directions of maximum variance in the data and projects it onto a new subspace.
- **t-Distributed Stochastic Neighbor Embedding (t-SNE)**: A non-linear technique that aims to preserve the local structure of the data in a lower-dimensional space.
- **Linear Discriminant Analysis (LDA)**: A supervised technique that finds the feature subspace that maximizes class separability.
- **Autoencoders**: Neural network-based models that learn to encode high-dimensional data into a lower-dimensional representation.

Feature Extraction: PCA

Principal Component Analysis (PCA) is a popular dimensionality reduction technique used in machine learning and data analysis to reduce the dimensionality of the data while preserving **most of its variance**. For the problem of dimensionality reduction in high-dimensional data, the basic idea of Principal Component Analysis (PCA) is as follows:

- Firstly, the variables of the data that need to be reduced are standardized (normalized) to create a dataset with a mean of 0 and a variance of 1.
- Subsequently, the standardized data undergoes an orthogonal transformation, converting the original data into new data represented by several linearly independent vectors. These new vectors representing the data not only need to be linearly independent from each other but also should contain the maximum amount of information possible.



Orthogonal
included angle: 90

Source: https://www.scaler.com/topics/nlp/what-is-pca/

Feature Extraction: PCA

$$\boldsymbol{x_1} = \alpha_{1,1} v_1 + \alpha_{1,2} v_2$$

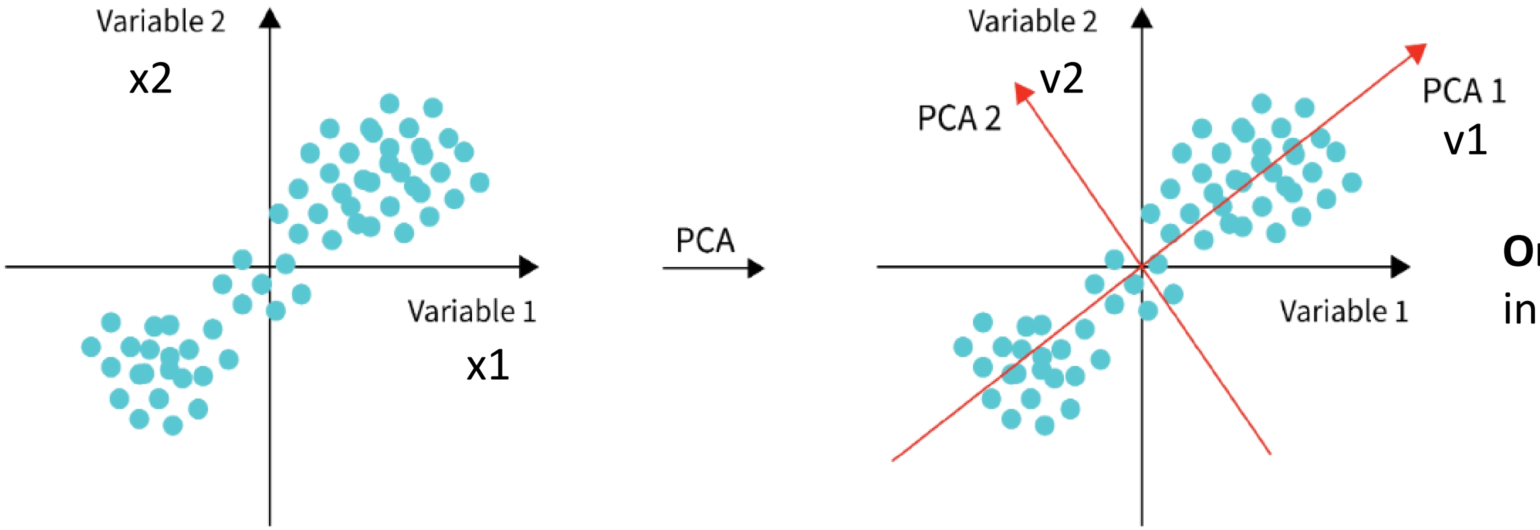
$$\boldsymbol{x_2} = \alpha_{2,1} v_1 + \alpha_{2,2} v_2$$

$$\Longleftrightarrow$$

$$\boldsymbol{v_1} = \rho_{1,1} x_1 + \rho_{1,2} x_2$$

$$\boldsymbol{v_2} = \rho_{2,1} x_1 + \rho_{2,2} x_2$$

- GOAL: account for variance of data in as few dimensions as possible (using linear projection)
- First PC is the projection direction that maximizes the variance of the projected data
- Second PC is the projection direction that is orthogonal to the first PC and maximizes variance of the projected data



Orthogonal
included angle: 90

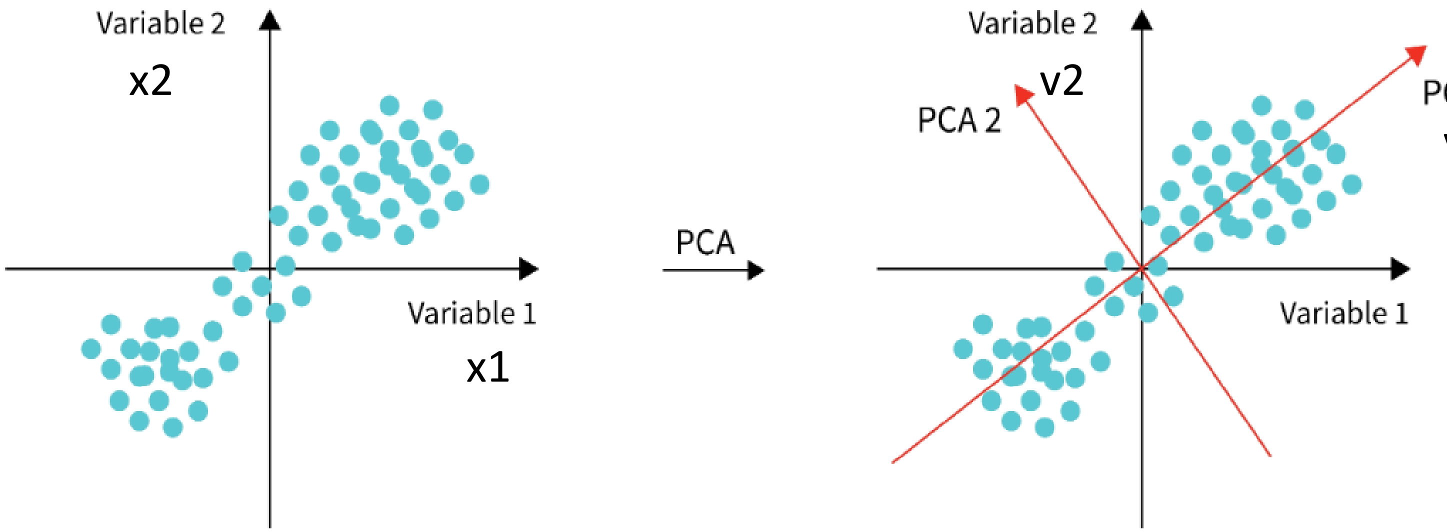
Source: https://www.scaler.com/topics/nlp/what-is-pca/

$$\boldsymbol{x_1} = \alpha_{1,1}v_1 + \alpha_{1,2}v_2 \qquad \Longleftrightarrow \qquad \boldsymbol{v_1} = \rho_{1,1}x_1 + \rho_{1,2}x_2$$

$$\boldsymbol{x_2} = \alpha_{2,1}v_1 + \alpha_{2,2}v_2 \qquad\qquad\qquad \boldsymbol{v_2} = \rho_{2,1}x_1 + \rho_{2,2}x_2$$

- **Eigenvectors** in PCA are the directions along which the data has the maximum variance: v1 and v2.
- **Eigenvalues** in PCA indicate the amount of variance present in the data along the corresponding eigenvector (principal component): λ1=Var($\alpha_{1,1}$, $\alpha_{2,1}$), λ2= Var($\alpha_{1,2}$, $\alpha_{2,2}$)
- **Contribution Rate** indicates the proportion of variance in the original data that is explained by a particular principal component.

$$\text{Contribution Rate}_i = \frac{\lambda_i}{\sum_{j=1}^{k} \lambda_j}$$



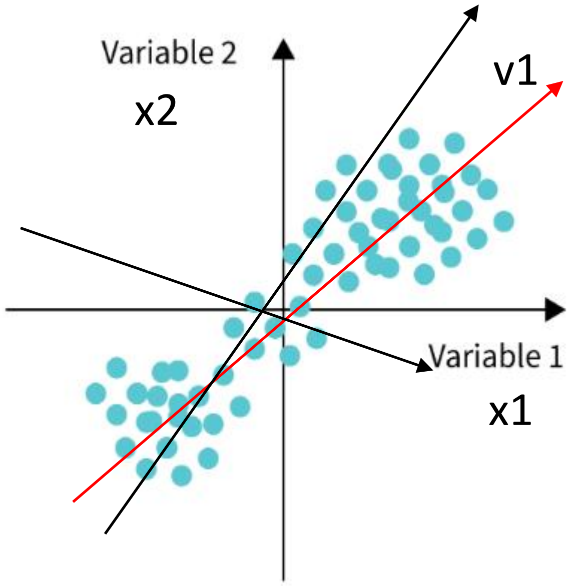
Orthogonal
included angle: 90

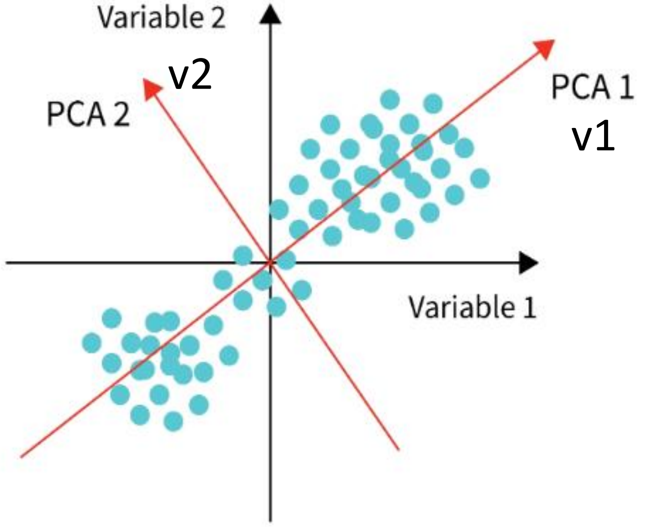
Source: https://www.scaler.com/topics/nlp/what-is-pca/

Feature Extraction: PCA

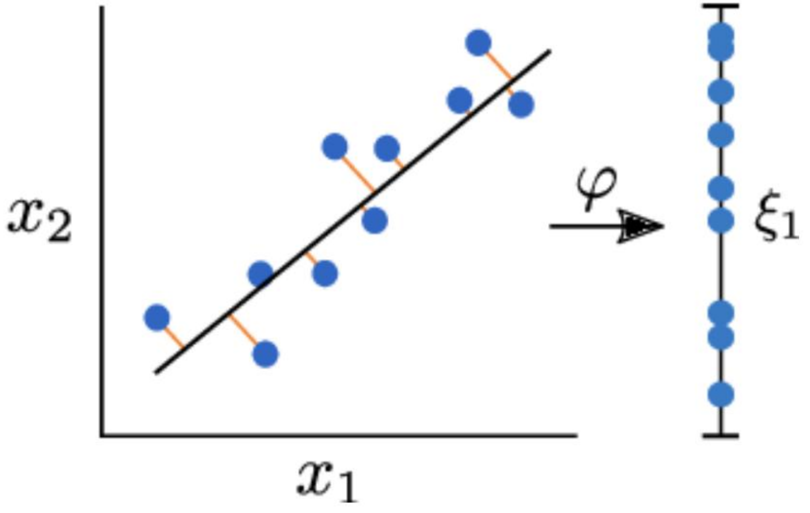
Conceptual algorithm

1. Find a line, such that when the data is projected onto that line, and it has the maximum variance.
2. Find a second line, orthogonal to the first, that has maximum projected variance.
3. Repeat until having k orthogonal lines
4. The projected position of a point on these lines gives the coordinates in the k-dimensional reduced space.





Orthogonal
included angle: 90



Formula derivation

PCA uses variance to measure the information content of new variables, which are sorted by variance in descending order as the first principal component, the second principal component, and so on.

Assuming the original data is an m*m*-dimensional random variable $\boldsymbol{x} = (x_1, x_2, \cdots, x_m)^{\mathrm{T}}$ with mean vector $\boldsymbol{\mu} = \mathrm{E}(\boldsymbol{x}) = (\mu_1, \mu_2, \cdots, \mu_m)^{\mathrm{T}}$, and covariance matrix as described below:

$$\boldsymbol{\Sigma} = \mathrm{cov}(\boldsymbol{x}, \boldsymbol{x}) = \mathrm{E}[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^{\mathrm{T}}]$$

From an *m*-dimensional random variable x to an *m*-dimensional random variable $\boldsymbol{v} = (v_1, v_2, \cdots, v_m)^{\mathrm{T}}$ through a linear transformation:

$$v_i = \boldsymbol{\alpha}_i^{\mathrm{T}} \boldsymbol{x} = \alpha_{1i} x_1 + \alpha_{2i} x_2 + \cdots + \alpha_{mi} x_m$$

After the linear transformation, the mean, variance, and covariance statistics of the random variable v_i can be expressed as:

$$\mathrm{E}(v_i) = \boldsymbol{\alpha}_i^{\mathrm{T}} \mu_i, \qquad i = 1, 2, \cdots, m$$

$$\mathbf{var}(\boldsymbol{v_i}) = \boldsymbol{\alpha_i^T \Sigma \alpha_i}, \qquad i = 1, 2, \cdots, m$$

$$\mathrm{cov}(v_i, v_j) = \boldsymbol{\alpha}_i^{\mathrm{T}} \boldsymbol{\Sigma} \alpha_j, \qquad i, j = 1, 2, \cdots, m$$

Feature Extraction: PCA

From an *m*-dimensional random variable x to an m*m*-dimensional random variable $\boldsymbol{v} = (v_1, v_2, \cdots, v_m)^{\mathrm{T}}$ through a linear transformation:

$$v_i = \boldsymbol{\alpha}_i^{\mathrm{T}} \boldsymbol{x} = \alpha_{1i} x_1 + \alpha_{2i} x_2 + \cdots + \alpha_{mi} x_m$$

After the linear transformation, the mean, variance, and covariance statistics of the random variable v_i can be expressed as:

$$\mathbf{var}(\boldsymbol{v_i}) = \boldsymbol{\alpha}_i^T \boldsymbol{\Sigma} \boldsymbol{\alpha_i}, \qquad i = 1, 2, \cdots, m$$

When the linear transformation from random variable x to random variable **v** satisfies the following conditions the transformed variables $v_1, v_2, \cdots, v_m$ respectively represent the first principal component, the second principal component, ..., up to the *m*-th principal component.

- The coefficient vector $\boldsymbol{\alpha}_i^{\mathrm{T}}$ of the linear transformation is a unit vector, satisfying $\boldsymbol{\alpha}_i^{\mathrm{T}} \alpha_i = 1,\ i = 1, 2, \cdots, m$.
- The transformed variables v*i* and v*j* are linearly independent, meaning $\mathrm{cov}(v_i, v_j) \neq 0 (i \neq j)$.
- Variable v1 has the largest variance among all possible linear transformations of random variable **x**, and v2 has the largest variance among all linear transformations independent of v1.

Taking the first principal component as an example, the mathematical expression of the optimization problem for the first principal component can be formulated as:

$$\max\ \boldsymbol{\alpha}_1^{\mathrm{T}} \boldsymbol{\Sigma} \boldsymbol{\alpha}_1$$
$$s.t. \boldsymbol{\alpha}_1^{\mathrm{T}} \boldsymbol{\alpha}_1 = 1$$

Feature Extraction: PCA

Install User Guide API Examples Community More

DictionaryLearning
FactorAnalysis
FastICA
IncrementalPCA
KernelPCA
LatentDirichletAllocation
MiniBatchDictionaryLearning
MiniBatchNMF
MiniBatchSparsePCA
NMF
PCA
SparseCoder
SparsePCA
TruncatedSVD
dict_learning
dict_learning_online
fastica
non_negative_factorization

API Reference > sklearn.decomposition > PCA

PCA

class **sklearn.decomposition.PCA**(*n_components=None*, *, *copy=True*, *whiten=False*, *svd_solver='auto'*, *tol=0.0*, *iterated_power='auto'*, *n_oversamples=10*, *power_iteration_normalizer='auto'*, *random_state=None*) [source]

Principal component analysis (PCA).

Linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space. The input data is centered but not scaled for each feature before applying the SVD.

It uses the LAPACK implementation of the full SVD or a randomized truncated SVD by the method of Halko et al. 2009, depending on the shape of the input data and the number of components to extract.

With sparse inputs, the ARPACK implementation of the truncated SVD can be used (i.e. through `scipy.sparse.linalg.svds`). Alternatively, one may consider `TruncatedSVD` where the data are not centered.

Notice that this class only supports sparse inputs for some solvers such as “arpack” and “covariance_eigh”. See `TruncatedSVD` for an alternative with sparse data.

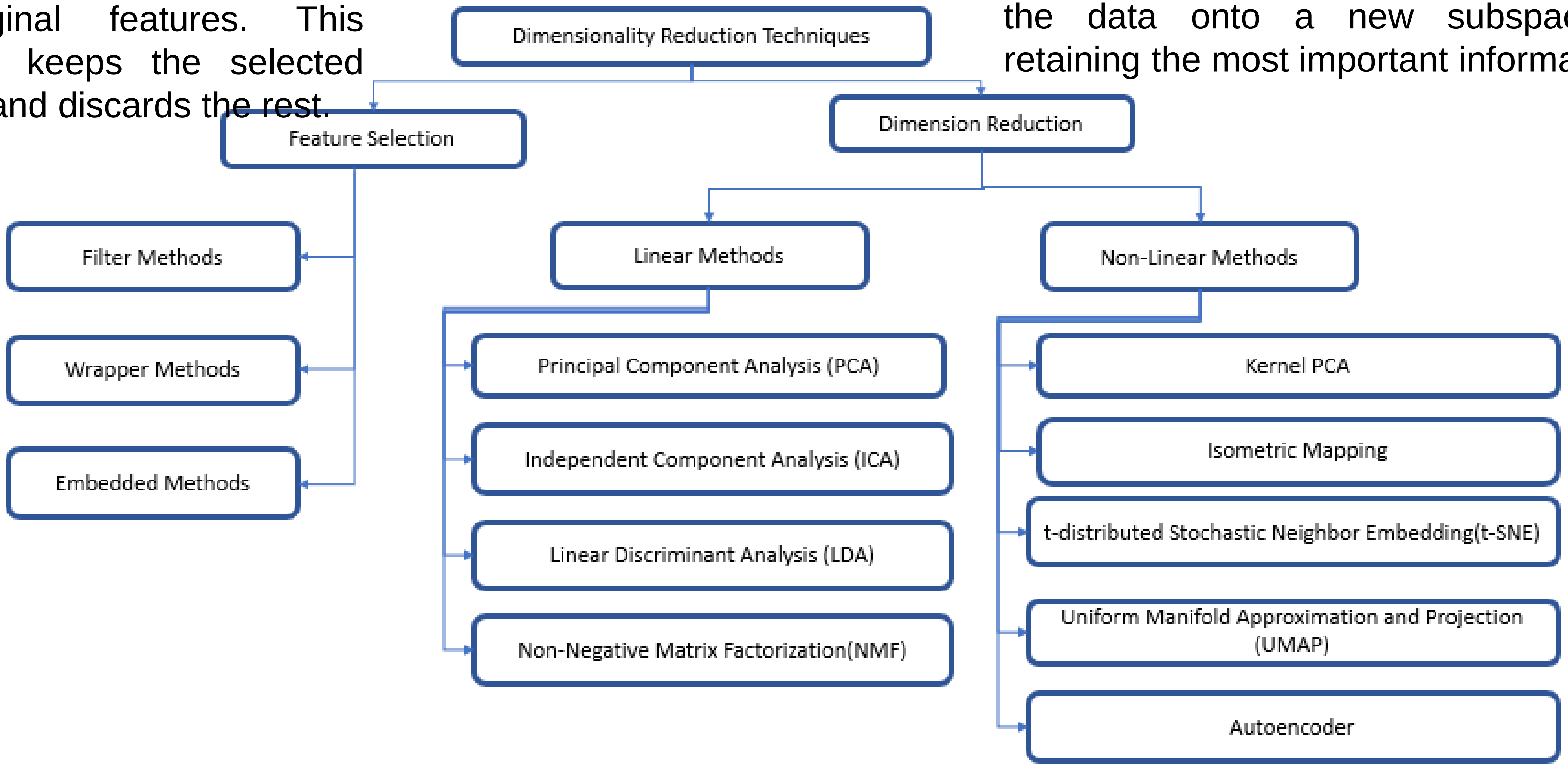
PCA Evaluation Metrics

1. **Explained Variance Ratio**: This metric indicates the proportion of variance captured by each principal component. It helps in understanding how much information each component retains from the original data.
2. **Cumulative Explained Variance**: This metric shows the cumulative variance explained by all principal components. It helps in deciding how many components are needed to retain a certain amount of variance in the data.
3. **Scree Plot**: A scree plot is a graphical representation of the eigenvalues of the principal components. It helps in visualizing the amount of variance explained by each component and identifying the point where adding more components does not significantly improve variance retention.
4. **Reconstruction Error**: This metric evaluates how well the original data can be reconstructed from the reduced dimensions. A lower reconstruction error indicates that the principal components are effectively capturing the variability in the data.

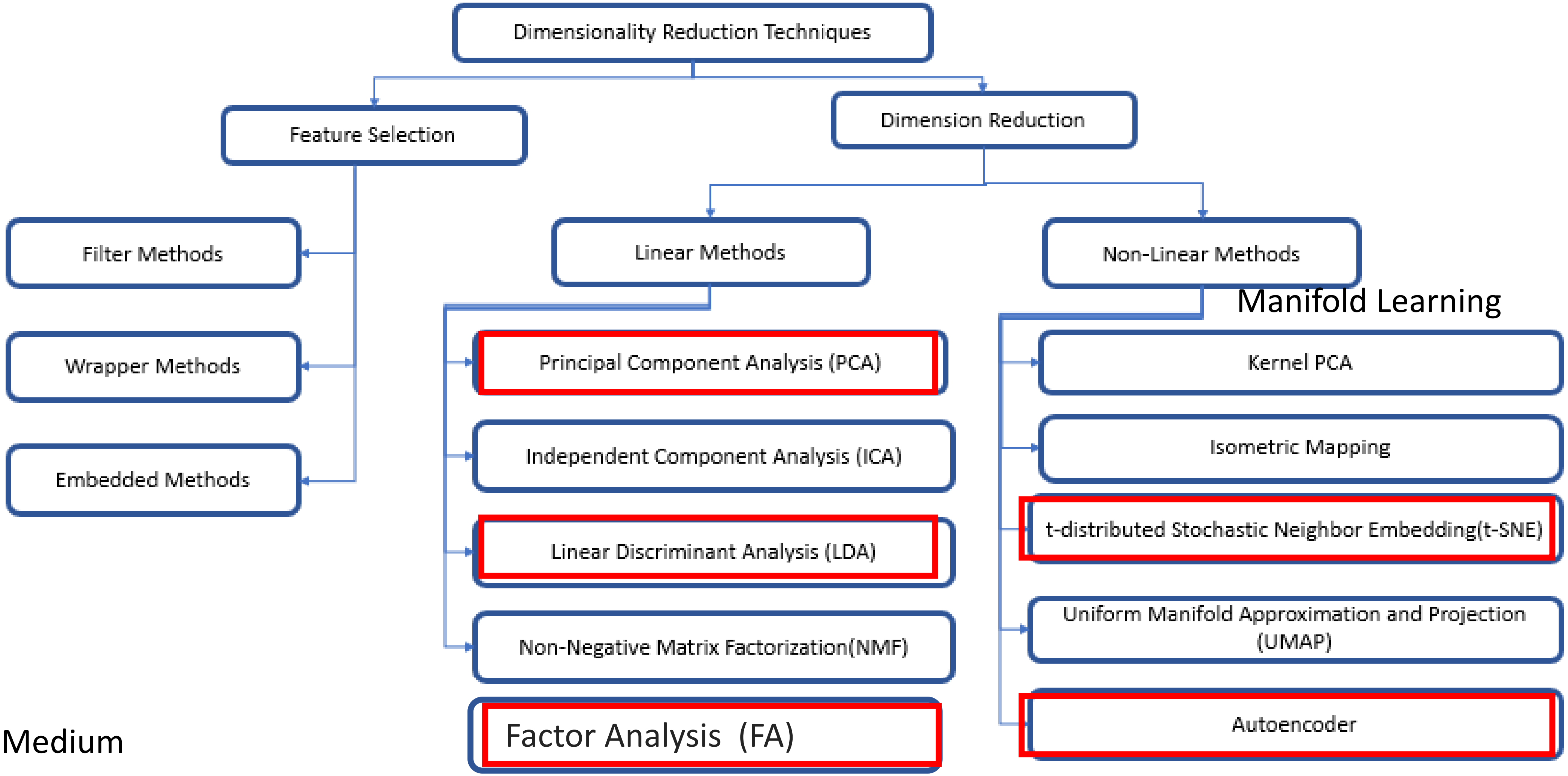
Feature Extraction

Involves selecting a subset of the original features. This approach keeps the selected features and discards the rest.

Involves transforming the original features into a lower-dimensional space. This transformation is typically done by projecting the data onto a new subspace while retaining the most important information.



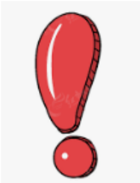
Feature Extraction



Source: Medium

https://www.youtube.com/watch?v=FD4DeN81ODY

Classwork



You can sign out at any time after **completing this assignment**.

OR

If you do not complete it, you can sign out **at 5 PM**.

Please use PCA to reduce the IRIS data to 3 dimensions, and then use KNN for classification, setting K to 3. Output a visualization scatter plot for comparison.

This assignment has nothing to do with the grade!

Run the Logistic Regression of Iris

https://mirror.tuna.tsinghua.edu.cn/help/anaconda/

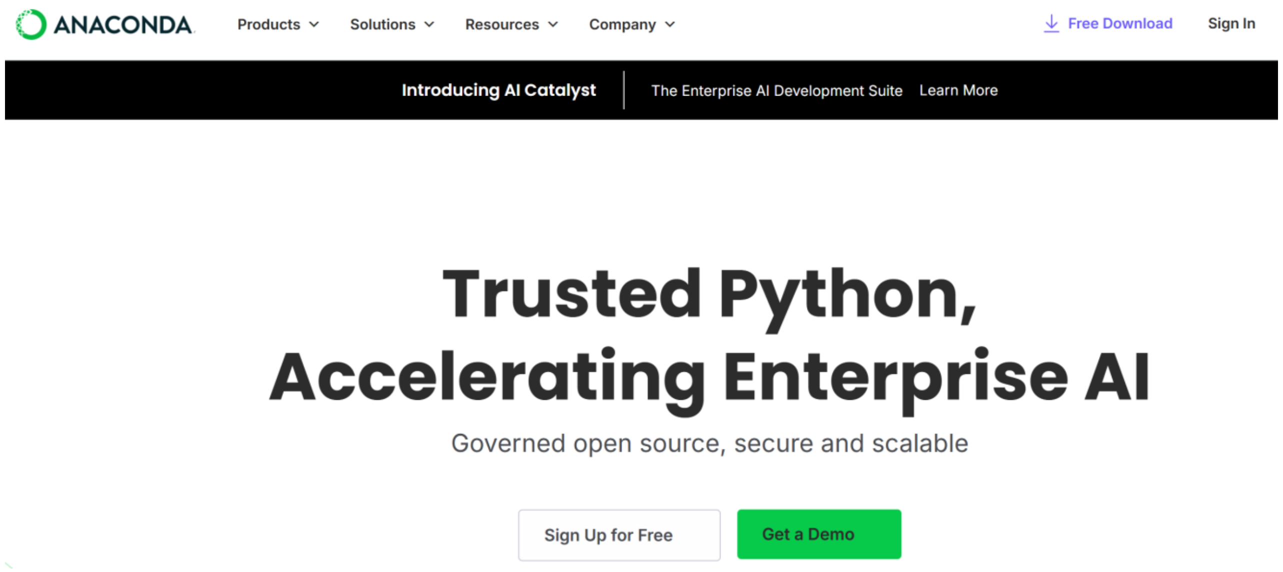
1. What is Anaconda?

Anaconda is a Python data science distribution, essentially an ‘integrated toolkit’. Its core components include:

- **Python interpreter**: The foundational program for directly executing Python code.
- **Hundreds of pre-installed libraries**: Covering common tools for data processing, numerical computation, visualisation, machine learning, and more.

Conda package manager: for installing, updating, and uninstalling libraries, offering greater capabilities than Python's built-in pip (manages non-Python dependencies); Environment management tool: enables creation of multiple independent virtual environments (e.g., one environment using Python 3.8, another using 3.10), preventing dependency conflicts between different projects. 2. How to install Anaconda

1. Download from the official website
2. Download via mirror sites



Anaconda Install

Run the Logistic Regression of Iris

https://www.jetbrains.com/pycharm/

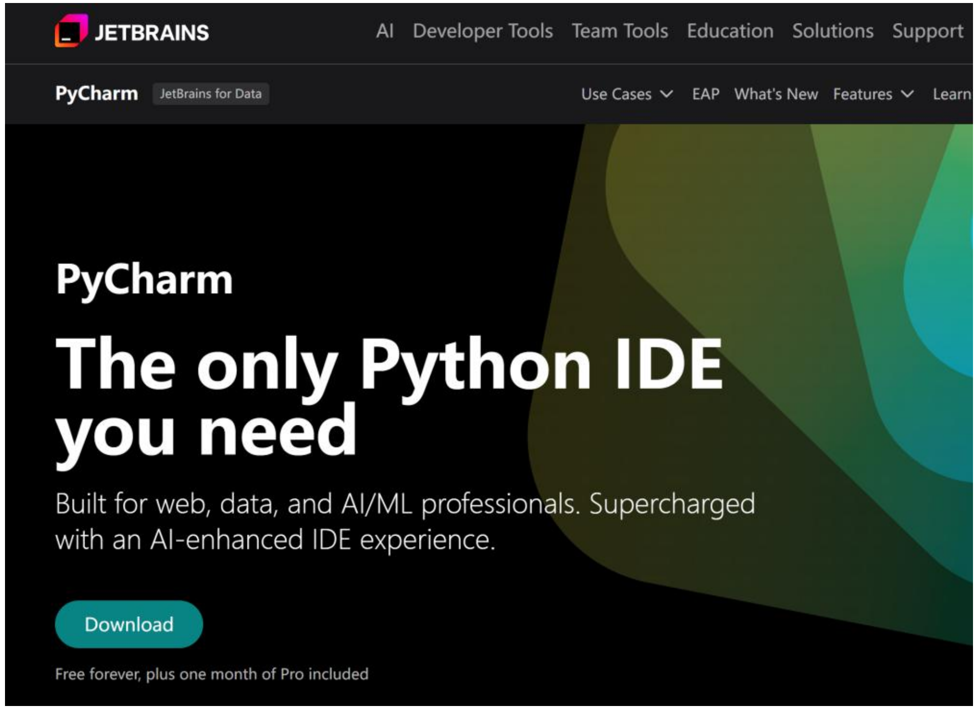
What is PyCharm?

PyCharm is a professional Python integrated development environment (IDE) developed by JetBrains, specifically designed for the Python programming language.
It offers extensive features and tools aimed at enhancing developers' programming efficiency and code quality.

It is widely used across various Python development domains, including web development, data analysis, artificial intelligence, and scientific computing.

It is available in two editions:
Community Edition and **Professional Edition**.

Opt for a **more stable cracked version** during PyCharm installation.



Pycharm Install

Thank you!

Innovating into the Future