Here is a **clear and exam-ready definition of Naive Bayes Network**:

Naive Bayes Network – Definition

A **Naive Bayes Network** (or **Naive Bayes Classifier**) is a *probabilistic machine learning model* based on **Bayes' Theorem** that assumes all features (predictor variables) are **conditionally independent** of each other given the class label.
Because of this simplified independence assumption, it becomes "naive," yet it works surprisingly well in many real-world applications.

Formal Definition

A **Naive Bayes Network** is a **Bayesian network classifier** where the class node is the parent of all feature nodes, and **each feature is independent of every other feature** given the class.
Mathematically:

```
P(C|X_1, X_2, ..., X_n) = \frac{P(C)\prod_{i=1}^{n} P(X_i |
C)}{P(X_1,X_2,...,X_n)}
```

Key Characteristics

- Based on **Bayes' theorem**
- Assumes **independence between features**
- Uses **probability distributions** to classify data
- Simple, fast, and works well for large datasets
- Used in **text classification, spam filtering, sentiment analysis**, etc.

Here are **clear, exam-oriented notes on Feature Selection Techniques**:

Feature Selection Techniques – Definition

Feature Selection is the process of selecting a subset of relevant, important, and non-redundant features from a dataset to improve model accuracy, reduce complexity, and avoid overfitting.

It helps in choosing only the most meaningful attributes before training a model.

Types of Feature Selection Techniques

Feature selection techniques are mainly divided into **three categories**:

---

1. Filter Methods

These methods use **statistical measures** to score features and select the best ones.

Common Filter Techniques

- **Correlation Coefficient**
- **Chi-Square Test**
- **Mutual Information**
- **ANOVA F-test**
- **Information Gain**
- **Variance Threshold**

Advantages

- Fast and simple
- Independent of machine learning algorithm
- Works well on very large datasets

---

2. Wrapper Methods

These methods **evaluate subsets of features** using a predictive model.
They select features based on **model performance**.

Common Wrapper Techniques

- **Forward Selection**
- **Backward Elimination**
- **Recursive Feature Elimination (RFE)**
- **Exhaustive Search / Best Subset Selection**

- **Genetic Algorithms**

Advantages

- More accurate because features are evaluated using actual model performance

Disadvantages

- Computationally expensive
- Slow for large datasets

---

3. Embedded Methods

Feature selection that happens **during model training**.
Algorithms automatically select the most important features.

Common Embedded Techniques

- **LASSO (L1 regularization)**
- **Ridge Regression (L2 regularization)**
- **Elastic Net**
- **Tree-based models (e.g., Random Forest, Decision Trees feature importance)**
- **Gradient Boosting feature importance**

Advantages

- Fast and efficient
- Built into the model
- Good balance between filter and wrapper methods

Why Feature Selection is Important?

- Reduces overfitting
- Improves accuracy
- Reduces training time
- Simplifies model
- Improves interpretability

- Removes noise and irrelevant data

Examples of Use Cases

- High-dimensional text data
- Gene expression data (thousands of features)
- Image processing
- Signal processing

Here are **clear, exam-ready notes on SVD (Singular Value Decomposition)**:

Singular Value Decomposition (SVD)

Definition

Singular Value Decomposition (SVD) is a matrix factorization technique that decomposes any real or complex matrix **A** into three matrices:

```
A = U \Sigma V^T
```

It is widely used in dimensionality reduction, noise reduction, latent semantic analysis, and recommendation systems.

Components of SVD

For any matrix of size :

1. **U (Left Singular Vectors)**
   - Size:
   - Orthogonal matrix
   - Represents directions of maximum variance in the original space
2. **Σ (Sigma – Singular Values)**

    - Size: diagonal matrix
    - Contains singular values (non-negative) in descending order
    - Indicates the importance (strength) of each corresponding dimension
3. **V^T (Right Singular Vectors)**
    - Size:
    - Orthogonal matrix
    - Represents directions of maximum variance in feature space

---

Formula

```
A = U \Sigma V^T
```

Where:

- Columns of **U** = eigenvectors of
- Columns of **V** = eigenvectors of
- Diagonal of **Σ** = square roots of eigenvalues of or

Uses of SVD

1. Dimensionality Reduction (PCA uses SVD)

Reduce the size of data while keeping important information.

2. Noise Reduction

Small singular values represent noise → can be removed.

3. Latent Semantic Analysis (LSA) in NLP

Find hidden relationships between documents and terms.

4. Image Compression

Lower-rank approximation of images using top singular values.

5. Recommendation Systems

Used in collaborative filtering (e.g., Netflix, Amazon).

Properties of SVD

- Works for **any matrix**, not only square matrices
- Singular values are always **non-negative**
- Rank of the matrix = number of non-zero singular values
- Provides optimal **low-rank approximation**

Low-Rank Approximation using SVD

If we keep only top **k** singular values:

$$A_k = U_k \Sigma_k V_k^T$$

This gives best approximation of A with rank k.

Small Example

For a matrix

SVD will decompose it into:

$$U = [...],\ \Sigma = [...],\ V^T = [...]$$

(Full matrices depend on computation.)

Short Definition (for exams)

SVD is a matrix decomposition method that factorizes a matrix into U (orthogonal), Σ (singular values), and V^T (orthogonal), and is used for dimensionality reduction, noise reduction, and extracting important patterns from data.

Here are **clear, concise, and exam-ready notes on SVM (Support Vector Machine):**

Support Vector Machine (SVM)

Definition

A **Support Vector Machine (SVM)** is a **supervised machine learning algorithm** used for **classification and regression** that finds the **best separating hyperplane** which maximizes the **margin** between different classes.

Key Concepts of SVM

1. Hyperplane

A hyperplane is a decision boundary that separates different classes.

- In 2D: a line
- In 3D: a plane
- In higher dimensions: a hyperplane

SVM finds the **optimal hyperplane**.

---

2. Margin

The margin is the distance between the hyperplane and the nearest data points from each class.

```
\text{SVM tries to maximize this margin}
```

---

3. Support Vectors

The data points that are closest to the hyperplane and influence its position are called **support vectors**.
They determine the optimal boundary.

---

4. Types of SVM

a) Linear SVM

Used when data is linearly separable.

b) Non-linear SVM (Using Kernel Trick)

Used when data is **not linearly separable**.

Kernel Trick

SVM uses kernel functions to transform data into a higher-dimensional space where it becomes linearly separable.

Common Kernels

- **Linear Kernel**
- **Polynomial Kernel**
- **Radial Basis Function (RBF)** / Gaussian Kernel
- **Sigmoid Kernel**

SVM Objective

To find:

```
\text{Hyperplane with maximum margin}
```

This leads to better generalization and accuracy.

Advantages of SVM

- Works well on high-dimensional data
- Effective for classification problems
- Uses support vectors → efficient
- Good for small and medium datasets
- Works well even when classes are not linearly separable (using kernels)

Applications

- Face detection
- Image classification
- Text categorization
- Bioinformatics
- Handwriting recognition

Short Exam Definition

SVM is a supervised classification algorithm that finds an optimal separating hyperplane with maximum margin, and uses kernel functions to classify both linear and non-linear data.

Here are **clear, exam-ready notes on HMM (Hidden Markov Model):**

Hidden Markov Model (HMM)

Definition

A **Hidden Markov Model (HMM)** is a **statistical model** in which the system is assumed to follow a **Markov process** with **hidden (unobservable) states**, but produces **observable outputs (symbols)** that depend probabilistically on those hidden states.

It is widely used in sequence modeling such as speech recognition, handwriting recognition, bioinformatics, and NLP.

Components of HMM

An HMM consists of **five elements**:

1. **N – Number of hidden states**
   Example: Rainy, Sunny
2. **M – Number of observation symbols**
   Example: Walking, Shopping, Cleaning
3. **State Transition Probability Matrix (A)**

```
A = \{a_{ij}\} = P(q_{t+1} = j \mid q_t = i)
```

4. **Observation Probability Matrix (B)**

```
B = \{b_j(k)\} = P(O_t = k \mid q_t = j)
```

5. **Initial State Probability (π)**

```
\pi_i = P(q_1 = i)
```

Two Layers in HMM

1. Hidden Layer

- Contains states that are **not directly observable**
- Example: emotional state, weather condition, biological gene state

2. Observation Layer

- What we actually observe
- Example: speech signal, word, activity

Key Problems Solved by HMM

HMM deals with three classical problems:

1. Evaluation Problem

Find the probability of an observation sequence, given the model.
Solved using **Forward or Backward Algorithm**.

2. Decoding Problem

Find the most probable sequence of hidden states.
Solved using **Viterbi Algorithm**.

3. Learning Problem

Estimate model parameters (A, B, π) from training data.
Solved using **Baum-Welch Algorithm (EM algorithm)**.

Properties of HMM

- Based on **Markov assumption**: next state depends only on current state
- Observation depends only on current state
- Both states and observations follow probability distributions

- Good for modeling **time-series** and **sequential data**

Applications of HMM

- Speech recognition (e.g., Siri, Google Assistant)
- POS tagging in NLP
- Handwriting recognition
- DNA sequence analysis
- Gesture recognition
- Weather prediction

Mathematical Representation

An HMM is represented by:

```
\lambda = (A, B, \pi)
```

Where:

- **A** = transition probabilities
- **B** = emission probabilities
- **π** = initial probabilities

Short Exam Definition

A Hidden Markov Model (HMM) is a statistical model with hidden states that follow a Markov process and observable outputs that depend on those hidden states. It is used to model sequential and time-series data.

Here are **clear, concise, and exam-focused notes on Maximum Likelihood Estimation (MLE):**

Maximum Likelihood Estimation (MLE)

Definition

Maximum Likelihood Estimation (MLE) is a statistical method used to estimate the parameters of a probability distribution by **maximizing the likelihood function**, i.e., by finding the parameter values that make the observed data most probable.

Concept

Suppose data points:

```
x_1, x_2, ..., x_n
```

Assume they come from a probability distribution with parameter .
The **likelihood function** is:

```
L(\theta) = P(x_1, x_2, ..., x_n \mid \theta)
```

MLE chooses such that is maximum.

Steps of MLE

1. **Write the probability distribution** based on the problem
   (e.g., Normal, Bernoulli, Poisson)
2. **Form the likelihood function**

   ```
   L(\theta) = \prod_{i=1}^n f(x_i \mid \theta)
   ```

3. **Convert to log-likelihood** (simplifies multiplication → addition)

```
\ln L(\theta)
```

4. **Differentiate** log-likelihood with respect to
5. **Set derivative = 0** and solve for θ
   → gives MLE estimate

Example (Simple)

Data from a Bernoulli distribution (coin toss):

```
x \in \{0,1\}
```

Likelihood:

```
L(p) = p^k (1-p)^{n-k}
```

Log-likelihood:

```
\ln L(p) = k\ln p + (n-k) \ln (1-p)
```

Derivative = 0 gives:

```
\hat{p} = \frac{k}{n}
```

(Probability of heads = number of heads / total tosses)

Why Use MLE?

- Gives best parameter estimates
- Works for many probability distributions
- Has good statistical properties (consistent, efficient)

Properties of MLE

- **Consistency**: as n → ∞, estimate → true value
- **Efficiency**: minimum variance among all estimators
- **Asymptotic Normality**
- **Invariant**: If is MLE, then is also MLE of

Applications of MLE

- Machine learning models
- Naive Bayes parameter estimation
- Logistic regression
- Hidden Markov Models
- Statistical inference
- Signal processing

Short Exam Definition

Maximum Likelihood Estimation is a method of estimating the parameters of a statistical model by choosing the values that maximize the probability of observing the given sample data.

Here are **clear, detailed, and exam-ready notes on the Deep Learning Approach**:

Deep Learning Approach

Definition

Deep Learning is a subfield of machine learning that uses **artificial neural networks with multiple layers** (deep neural networks) to automatically learn features and patterns from large amounts of data.

It mimics the working of the human brain and is capable of learning complex representations.

Key Characteristics of Deep Learning

1. Multi-Layer Architecture

Deep learning models consist of many layers:

- Input layer
- Hidden layers (multiple)
- Output layer

Each layer extracts higher-level features.

---

2. Automatic Feature Learning

Unlike traditional machine learning, **no manual feature extraction** is needed.
The network learns features automatically.

---

3. Non-Linear Transformations

Every layer applies non-linear functions (ReLU, Sigmoid, Tanh) to learn complex relationships.

---

4. Training with Large Data

Deep learning performs best with:

- Large datasets
- High computational power (GPUs/TPUs)

Main Deep Learning Architectures

1. Artificial Neural Networks (ANN)

Basic neural network with multiple layers used for classification/regression.

2. Convolutional Neural Networks (CNN)

Best for:

- Image processing
- Object detection
- Medical image analysis

Uses *convolution* and *pooling* layers.

3. Recurrent Neural Networks (RNN)

Best for sequential or time-series data:

- Text
- Speech
- Financial data

Variants:

- LSTM (Long Short-Term Memory)
- GRU (Gated Recurrent Unit)

4. Autoencoders

Used for:

- Dimensionality reduction
- Denoising
- Feature learning

5. Generative Adversarial Networks (GANs)

Two networks (Generator + Discriminator) compete to generate realistic data.
Used in:

- Image generation
- Deepfake creation
- Data augmentation

6. Transformer Models

State-of-the-art for NLP:

- BERT
- GPT
- Vision Transformer (ViT)

Deep Learning Workflow

1. **Collect Data**
2. **Preprocess Data** (normalization, encoding, etc.)
3. **Build Model**
4. **Choose Loss Function & Optimizer** (e.g., cross-entropy + Adam)
5. **Train the Network** (forward pass + backpropagation)
6. **Evaluate Model**
7. **Deploy Model**

Advantages

- Learns complex patterns
- Automatic feature extraction
- High accuracy with large data
- Excellent for images, text, audio
- Supports transfer learning

Challenges

- Requires a lot of data
- High computational cost
- Hard to interpret (“black box”)
- Risk of overfitting
- Long training time

Applications

- Computer vision (face recognition, medical imaging)
- Natural language processing (chatbots, translation)
- Speech recognition
- Autonomous vehicles
- Robotics
- Recommendation systems

Short Exam Definition

Deep Learning is a machine learning approach that uses multi-layered neural networks to automatically learn hierarchical features from large datasets, enabling high accuracy in complex tasks such as image recognition, speech processing, and NLP.

Here are **clear, concise, and exam-ready notes on Autoencoders**:

Autoencoder

Definition

An **Autoencoder** is a type of **unsupervised deep learning neural network** used to learn efficient data representations (encoding) by reconstructing the input data at the output.
It compresses the input into a **lower-dimensional latent space** and then reconstructs it back.

It consists of two parts:

1. **Encoder** – compresses data
2. **Decoder** – reconstructs data

Architecture of Autoencoder

1. Encoder

Transforms the input into a smaller representation .

```
Z = f_{\theta}(X)
```

2. Latent Space (Bottleneck)

- Contains compressed/encoded information
- Lowest dimension
- Forces the model to learn the most important features

3. Decoder

Reconstructs input from latent representation.

```
X' = g_{\theta}(Z)
```

Goal:

```
X' \approx X
```

The model learns to minimize the **reconstruction error**.

Loss Function

Most common:

```
\text{Loss} = \| X - X' \|^2
```

(MSE – Mean Squared Error)

Types of Autoencoders

1. Undercomplete Autoencoder

Latent space is smaller → forces feature learning.

2. Sparse Autoencoder

Uses sparsity constraints → activates only few neurons.

3. Denoising Autoencoder

Input is intentionally corrupted; output must be clean.

Used for:

- Noise removal
- Image enhancement

4. Variational Autoencoder (VAE)

Generative model → produces new samples similar to input data.

5. Convolutional Autoencoder

Uses convolution layers for image data.

Applications of Autoencoders

- **Dimensionality reduction** (alternative to PCA)
- **Image denoising**
- **Image compression**
- **Anomaly detection**
- **Feature extraction**
- **Data generation** (with VAEs)
- **Recommender systems**

Advantages

- Learns non-linear feature representations
- Very useful for image and high-dimensional data
- Can remove noise effectively
- Works without labeled data

Disadvantages

- Needs large data for training
- Risk of learning the identity function (if too large)
- Hard to interpret latent space
- Reconstruction quality may vary

Short Exam Definition

An autoencoder is an unsupervised neural network that learns compressed representations of data using an encoder–decoder architecture and reconstructs the input from its latent space representation.

Here are **clear, concise, and exam-ready notes** on **Applications of Clustering in Pattern Recognition**:

Applications of Clustering in Pattern Recognition

Clustering is an **unsupervised learning technique** used to group similar patterns or data points into clusters.
In **pattern recognition**, clustering plays an important role in discovering structures and hidden patterns without prior class labels.

Major Applications

1. Image Segmentation

- Clustering groups pixels with similar color, intensity, or texture.
- Used in:
    - Medical image analysis
    - Satellite image processing
    - Object detection

2. Document and Text Clustering

- Groups similar documents together.
- Used for:
    - Topic modeling
    - Search optimization
    - News categorization

3. Pattern Classification (Preprocessing)

- Clustering helps identify natural groupings before classification.
- Useful when class labels are unknown or incomplete.

4. Customer Segmentation

- In marketing, clustering groups customers by:

    - Buying behavior
    - Interests
    - Demographics

Used for targeted advertising.

5. Anomaly / Outlier Detection

- Data points that do not fit in any cluster are identified as outliers.
- Used in:
    - Fraud detection
    - Network intrusion detection
    - Fault detection in machines

6. Handwriting and Character Recognition

- Groups similar handwritten characters for feature extraction.
- Useful in OCR (Optical Character Recognition).

7. Bioinformatics / Gene Expression Analysis

- Clustering groups genes with similar expression patterns.
- Helps in disease classification and drug discovery.

8. Face Recognition and Image Retrieval

- Clusters images with similar facial features.
- Used in photo-tagging, surveillance, biometrics.

9. Recommender Systems

- Groups users or items based on similarity.
- Helps recommend similar products or content.

10. Speech and Audio Pattern Recognition

- Clustering segments phonemes or similar sound patterns.
- Used in:
    - Speech recognition
    - Speaker identification

Short Exam Answer

Clustering in pattern recognition is used for image segmentation, text/document grouping, pattern classification, anomaly detection, customer segmentation, handwriting recognition, gene expression analysis, face recognition, speech processing, and building recommender systems.

Here are **clear, detailed, and exam-ready notes** on **Pattern Recognition in various domains**:

Pattern Recognition in Image Processing, Speech Recognition, Biometrics, NLP, Object Detection & Recognition

Pattern recognition techniques are widely used across different fields to identify patterns, classify data, and make intelligent decisions. Below are domain-wise applications.

---

1. Pattern Recognition in Image Processing

Pattern recognition helps in analyzing and understanding images by extracting meaningful information.

Applications

- **Image Classification** – Categorizing images (cat, dog, car, etc.)
- **Image Segmentation** – Separating objects from background
- **Face Recognition**
- **Medical Imaging** – Detecting tumors, diseases (CT, MRI, X-ray)

- **Optical Character Recognition (OCR)** – Reading printed/handwritten text
- **Texture Analysis** – Identifying patterns in satellite or industrial images
- **Image Compression using PCA/SVD**

---

2. Pattern Recognition in Speech Recognition

Pattern recognition converts human speech into text or commands.

Applications

- **Automatic Speech Recognition (ASR)** – Google Assistant, Siri
- **Speaker Identification** – Detecting who is speaking
- **Speech-to-Text Transcription**
- **Emotion Detection from Voice**
- **Voice Biometrics** for authentication
- **Language Translation Systems** (speech-to-speech translators)

Techniques

- HMM (Hidden Markov Models)
- Dynamic Time Warping (DTW)
- Deep Learning (LSTM, Transformer models)

---

3. Pattern Recognition in Biometrics

Biometrics identifies individuals based on unique biological and behavioral traits.

Applications

- **Fingerprint Recognition**
- **Face Recognition**
- **Iris Recognition**
- **Voice Biometrics**
- **Palmprint & Vein Recognition**
- **Gait Recognition**
- **Signature Verification**

Usage Areas

- Security systems
- Smartphones
- Banking / Authentication
- Access control

---

4. Pattern Recognition in Natural Language Processing (NLP)

Pattern recognition techniques extract meaning, structure, and patterns from text.

Applications

- **Text Classification**
    - Spam detection
    - Sentiment analysis
    - Topic detection
- **Named Entity Recognition (NER)**
- **Machine Translation**
- **Speech-to-text & Text-to-speech**
- **Chatbots and Virtual Assistants**
- **Keyword Extraction**
- **Document Clustering**

Models used

- Naive Bayes
- SVM
- Transformers (BERT, GPT)

---

5. Pattern Recognition in Object Detection and Recognition

Identifying and locating objects in images or videos.

Applications

- **Self-driving cars** – detect pedestrians, vehicles, traffic signs
- **Surveillance systems** – detect suspicious activities
- **Industrial Automation** – quality inspection, defect detection
- **Medical Field** – detecting organs, lesions, tumors
- **Retail** – automatic checkout systems
- **Robotics** – environment perception
- **Augmented Reality (AR)** – object tracking in real time

Common Techniques

- CNNs (Convolutional Neural Networks)
- YOLO (You Only Look Once)
- R-CNN, Fast/Faster RCNN
- SSD (Single Shot Detector)

Short Exam Answer (Compact Form)

Pattern recognition is used in image processing (object detection, segmentation, OCR), speech recognition (ASR, speaker identification), biometrics (fingerprint, face, iris recognition), NLP (text classification, sentiment analysis, translation), and object detection (self-driving cars, surveillance, robotics).

Here are **clear, concise, and exam-ready notes on KNN (K-Nearest Neighbors)**:

K-Nearest Neighbors (KNN)

Definition

K-Nearest Neighbors (KNN) is a **supervised machine learning algorithm** used for **classification and regression**.

It predicts the class of a new data point based on the **majority class of its K nearest neighbors** in the feature space.

Key Idea

“**Similar data points exist close to each other**.”

To classify a new sample:

1. Find the **K nearest data points** (neighbors)
2. Use **majority voting** (classification) or **average value** (regression)

How KNN Works (Steps)

1. Choose number of neighbors **K**
2. Calculate distance between new point and all training points
   - Euclidean distance (most common)
   - Manhattan distance
   - Minkowski distance
3. Select **K nearest neighbors**
4. Classification → choose the **majority class**
   Regression → take **average**
5. Output the prediction

Distance Formula

Euclidean Distance

```
d = \sqrt{\sum (x_i - y_i)^2}
```

Choosing K

- Small K → sensitive to noise (overfitting)
- Large K → smooth decision boundary (may underfit)

Common choice: **K** = √**N** (heuristic)

Advantages of KNN

- Simple and easy to understand
- No training required (lazy learner)
- Works well with small datasets
- Good for non-linear decision boundaries

Disadvantages of KNN

- Slow for large datasets (needs distance calculation)
- Sensitive to irrelevant features
- Must choose K carefully
- Memory intensive
- Bad for high-dimensional data (curse of dimensionality)

Applications of KNN

- Recommendation systems
- Pattern recognition
- Medical diagnosis
- Text classification
- Image recognition
- Fraud detection

Short Exam Definition

KNN is a supervised learning algorithm that classifies a new data point based on the majority class among its K closest neighbors using a distance measure.

Here is a clear and concise explanation of **PGM (Probabilistic Graphical Models)** for your Pattern Recognition notes:

PGM (Probabilistic Graphical Model)

A **Probabilistic Graphical Model (PGM)** is a framework that uses graphs to represent complex probability distributions. It combines **graph theory** and **probability theory** to model the relationships among random variables efficiently.

Definition

A **Probabilistic Graphical Model** is a graphical representation where:

- **Nodes** represent **random variables**
- **Edges** represent **probabilistic dependencies** between variables

PGMs provide a compact way to encode joint probability distributions and allow efficient inference and learning.

Types of PGMs

1. Bayesian Networks (Directed Graphical Models)

- Directed acyclic graphs (DAGs)
- Edges show **causal** or **conditional dependencies**
- Example: Disease → Symptoms

2. Markov Random Fields / Markov Networks (Undirected Graphical Models)

- Undirected graphs
- Represent **mutual** or **spatial dependencies**
- Example: Image pixels dependency in image segmentation

3. Factor Graphs

- Bipartite graphs connecting variables and factors
- Useful for complex inference algorithms

Key Concepts

1. Conditional Independence

PGMs simplify large probability distributions by exploiting conditional independence properties.

2. Inference

Finding unknown variables given observed data.
Techniques include:

- Variable elimination
- Belief propagation
- Sampling-based methods (MCMC)

3. Learning

Estimating:

- **Parameters** (probabilities)
- **Structure** (graph connections)

Applications of PGMs

- **Image Processing** → Image denoising, segmentation
- **Speech Recognition** → Modeling temporal dependencies
- **Natural Language Processing** → POS tagging, information extraction
- **Medical Diagnosis** → Predicting diseases

- **Robotics** → Localization, SLAM
- **Computer Vision** → Object detection & recognition

Advantages

- Represents complex data dependencies compactly
- Supports efficient computation
- Handles uncertainty well
- Integrates prior knowledge

Conclusion

PGMs are powerful tools in pattern recognition for modeling uncertain, structured, and high-dimensional data. They form the foundation for many modern AI and ML systems.
