1. Basic Definitions

Machine Learning

A field of study where computers learn patterns from data without being explicitly programmed.

Model

A mathematical function that maps input to output.

Training

Process of learning model parameters from data.

Testing

Evaluating the trained model on new unseen data.

Feature

An input variable used to describe data.

Label

The output or target variable.

---

2. Linear Algebra (For ML)

Linear algebra provides mathematical tools for representing data and models.

Vector

Ordered list of numbers.
Example: x = (x1, x2, …, xn)

Matrix

2D array of numbers.
Used for: datasets, transformations.

Dot Product

```
w^T x = Σ (w_i * x_i)
```

Matrix Multiplication

```
A (m×n) * B (n×p) = C (m×p)
```

Norm (Length of vector)

```
||x|| = sqrt( Σ x_i^2 )
```

Eigenvalues and Eigenvectors

Used in PCA, Spectral Clustering.

Equation:

```
A v = λ v
```

Properties Useful in ML

- Linear regression uses matrix inversion
- PCA uses eigen decomposition
- Neural networks use matrix multiplications

---

3. Statistical Learning Theory

Statistical learning theory explains how learning works mathematically.

Generalization

How well a model performs on unseen data.

Risk

Expected loss on new data:

```
R(f) = E[ L(y, f(x)) ]
```

Empirical Risk

Loss on training data:

```
R_emp(f) = (1/N) Σ L(y_i, f(x_i))
```

Goal

Minimize risk (true error), not just training error.

Bias–Variance Tradeoff

- High bias → underfitting
- High variance → overfitting

VC Dimension

Capacity or complexity of a hypothesis class.
High VC ⟶ flexible but risky.

---

4. Types of Learning

1. Supervised Learning

Input + Output given.
Example: classification, regression.

2. Unsupervised Learning

Only input, no labels.
Example: clustering, PCA.

3. Semi-Supervised Learning

Some labels + many unlabeled samples.

4. Reinforcement Learning

Learns by interacting with environment and receiving rewards.

5. Self-Supervised Learning

Uses data to generate its own labels.

---

5. Hypothesis Space and Inductive Bias

Hypothesis Space (H)

Set of all models an algorithm can choose from.

Example:
For linear models:

```
H = { w^T x + b }
```

Inductive Bias

The assumptions the learner uses to choose one hypothesis over others.

Examples:

- Linear models assume data is linearly separable
- Decision trees assume hierarchical structure
- KNN assumes nearby points have similar labels

Why Needed?

Without inductive bias, infinite hypotheses fit training data.

---

6. Evaluation and Cross Validation

Train-Test Split

Split data into:

1. Training set
2. Test set

Accuracy

```
Accuracy = correct_predictions / total_predictions
```

Confusion Matrix

TP, TN, FP, FN.

Precision & Recall

```
Precision = TP / (TP + FP)
Recall = TP / (TP + FN)
```

F1 Score

```
F1 = 2 * (Precision * Recall) / (Precision + Recall)
```

Cross Validation

K-Fold Cross Validation

- Divide data into k equal parts
- Train on k−1 folds
- Test on the remaining fold
- Repeat for all folds
- Average performance

Advantages

- Reduces variance
- Uses data efficiently

Leave-One-Out CV (LOOCV)

Special case where k = number of samples.

---

7. Optimization (in ML)

Optimization means finding model parameters that minimize the loss function.

Objective Function

Commonly:

```
minimize  L(w)
```

Gradient Descent

Updates parameters in direction of negative gradient.

Update rule:

```
w = w - η * gradient( L(w) )
```

η = learning rate

Variants

- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini-batch Gradient Descent
- Momentum
- Adam optimizer

Convex Optimization

Loss function is convex → global minimum exists.
Example: Linear regression.

Non-Convex Optimization

Deep neural networks → many local minima.

Below is a **clean, exam-oriented explanation** of all topics.
Plain text only — **easy to copy**, no formatting issues.

---

1. Statistical Learning Theory

Statistical Learning Theory studies how a model learns from data and how well it **generalizes** to unseen data.

Key Concepts

(a) Generalization

Ability of a model to perform well on new data (not just training data).

(b) Risk (True Error)

Expected loss over the whole data distribution:

```
R(f) = E[ L(y, f(x)) ]
```

(c) Empirical Risk (Training Error)

Average loss on training samples:

```
R_emp(f) = (1/N) Σ L(y_i , f(x_i))
```

(d) Empirical Risk Minimization (ERM)

Choose the hypothesis that minimizes training error.

(e) Overfitting

Model fits noise in training data → poor generalization.

(f) Underfitting

Model is too simple → high error on both training and test data.

(g) VC Dimension

A measure of model complexity.
Higher VC → more expressive → risk of overfitting.

Goal of Learning

Balance between model complexity and generalization.

---

2. Types of Learning

(a) Supervised Learning

Data has input + output labels.
Tasks: Classification, Regression.

(b) Unsupervised Learning

Only input data, no labels.
Tasks: Clustering, Dimensionality Reduction.

(c) Semi-Supervised Learning

Few labeled samples + many unlabeled samples.

(d) Reinforcement Learning

Agent interacts with environment and receives rewards.
Goal: maximize cumulative reward.

(e) Self-Supervised Learning

Model generates its own labels using data structure (e.g., autoencoders).

(f) Online Learning

Learning happens continuously with data coming in streams.

---

3. Hypothesis Space and Inductive Bias

Hypothesis Space (H)

Set of all possible models that a learning algorithm can choose.

Example:
For linear models:

```
H = { w^T x + b }
```

Inductive Bias

Assumptions the learning algorithm makes to choose a hypothesis.

Examples of inductive bias:

- Linear models assume data is linearly separable
- KNN assumes nearby points have similar labels
- Decision trees assume hierarchical structure in data
- SVM assumes margin maximization improves generalization

Why Needed?

Without assumptions, infinite hypotheses fit the training data → learning becomes impossible.

---

4. Evaluation and Cross Validation

Model Evaluation Methods

(a) Train–Test Split

Dataset divided into:

- Training set
- Test set (unseen)

(b) Accuracy

```
Accuracy = correct_predictions / total_samples
```

(c) Confusion Matrix

Contains: TP, TN, FP, FN.

(d) Precision

```
Precision = TP / (TP + FP)
```

(e) Recall

```
Recall = TP / (TP + FN)
```

(f) F1 Score

```
F1 = 2 * (Precision * Recall) / (Precision + Recall)
```

(g) ROC and AUC

Used to analyze classifier performance.

Cross Validation

K-Fold Cross Validation

- Split data into k equal parts
- Train on k−1 parts
- Test on remaining 1 part
- Repeat k times
- Average the results

Advantages

- Uses all data efficiently
- Reduces variance of evaluation
- More reliable than simple train-test split

Types

1. **k-Fold**
2. **Stratified k-Fold** (keeps class ratio same)
3. **Leave-One-Out (LOOCV)**
4. **Repeated k-Fold**

---

5. Optimization (in Machine Learning)

Optimization is the process of finding model parameters that minimize the loss function.

Goal

```
minimize L(w)
```

Where w = model parameters.

Gradient Descent

Most common optimization method.

Update rule:

```
w = w - η * ∇L(w)
```

η = learning rate
$\nabla L(w)$ = gradient of loss

Types of Gradient Descent

- **Batch Gradient Descent**
  Uses whole dataset each step
- **Stochastic Gradient Descent (SGD)**
  Uses one sample per step
- **Mini-Batch Gradient Descent**
  Uses small batch per step (most common)

Advanced Optimizers

- Momentum
- RMSProp
- Adam
- Adagrad

Convex vs Non-Convex Optimization

- Linear models → convex (easy, one minimum)
- Deep networks → non-convex (many minima)

.

---

1. Statistical Decision Theory

Statistical Decision Theory provides a **mathematical framework for making decisions under uncertainty**.

It answers one question:

"Given a set of observations, how do we choose the best decision that minimizes loss?"

Key Concepts

1. Decision

An action you choose based on data.

Example:
Decide whether an email is *spam* or *not spam*.

---

2. States of Nature

The true but unknown condition.

Example:
Actual class of email (spam or not spam).

---

3. Loss Function (L)

Measures the **cost** of making a wrong decision.

Examples:

- $L = 0$ → correct decision
- $L > 0$ → wrong decision
- Medical test false negative → high loss
- Spam filter misclassification → low loss

Loss function defines how “bad” each decision is.

---

4. Risk Function (R)

Expected loss over all possible states.

$$R(\theta, d) = \mathbb{E}[L(\theta, d)]$$

Goal:

Choose the decision that minimizes risk.

---

5. Bayes Risk

Average risk using a prior probability distribution.

$$R_B(d) = \int L(\theta, d) \cdot p(\theta)\, d\theta$$

Bayes Decision Rule → choose decision with **minimum Bayes risk**.

---

2. Bayesian Learning

Bayesian learning uses **Bayes’ Theorem** to update beliefs (probabilities) based on data.

Bayes’ Theorem:

$$p(\theta \mid D) = [\ p(D \mid \theta) \cdot p(\theta)\ ] \,/\, p(D)$$

Where:

- → Prior
- → Likelihood
- → Posterior
- → Evidence (normalizing constant)

3. Maximum Likelihood (ML) Estimation

ML finds parameters that **maximize the likelihood** of observed data.

```
θ_ML = arg max_θ p(D | θ)
```

Key Idea

Choose parameters that make the observed data “most probable.”

Example:

Given coin flips: H H H T H
Estimate the probability of heads (θ):

```
\hat{\theta}_{ML} = \frac{\text{Number of Heads}}{\text{Total flips}}
```

ML ignores prior knowledge.

4. Maximum A Posteriori (MAP) Estimation

MAP estimation maximizes the **posterior** probability.

```
θ_MAP = arg max_θ p(θ | D)
```

Using Bayes’ rule:

```
θ_MAP = arg max_θ [ p(D | θ) · p(θ) ]
```

Key Idea

ML + Prior
MAP uses both:

- Data (likelihood)
- Prior belief

Example

Coin flips + belief that coin is fair.
MAP will push θ closer to 0.5, unlike ML.

5. Bayes Estimates

Bayes estimate is the **expected value** of parameters using the posterior distribution.

```
θ_Bayes = E[ θ | D ]
```

Key Idea

Bayes estimate uses **full posterior**, not just mode (like MAP).

Depends on loss function:

- For squared loss → Posterior mean
- For absolute loss → Posterior median
- For 0–1 loss → Posterior mode (MAP)

---

6. Conjugate Priors

A prior is a **conjugate prior** if the posterior has the **same distribution family** as the prior.

Why Useful?

- Posterior is easy to compute
- Closed-form solution
- No complicated integrals

Common Conjugate Priors

1. Bernoulli / Binomial Likelihood

Likelihood: Coin flips (success/failure)
Conjugate Prior: **Beta distribution**

Posterior:

```
p ~ Beta(α, β)
```

---

2. Gaussian Likelihood

Likelihood: Normal data
Conjugate Prior:

- Prior mean → **Normal distribution**
- Prior variance → **Inverse-Gamma**

Posterior is also Gaussian (for known variance).

---

3. Poisson Likelihood

Conjugate Prior: **Gamma distribution**

Why Conjugate Priors Matter?

- Simple formulas
- Easy posterior updates
- Used in Naive Bayes, Bayesian Networks, Hidden Markov Models

---

Summary Table

Concept	Meaning
Statistical Decision Theory	Framework for decision-making under uncertainty
ML Estimate	Maximizes likelihood P(D
MAP Estimate	Maximizes posterior P(θ
Bayes Estimate	Posterior expectation of θ
Conjugate Prior	Prior + likelihood → same family posterior

Below is a **simple, clear, exam-oriented explanation** of **Linear Regression, Ridge Regression, and Lasso Regression** with formulas, intuition, differences, and applications.

☆ 1. Linear Regression

Linear Regression is a **supervised learning** method used to predict a continuous value.

✔ Goal

Find a straight line (or hyperplane) that best fits the data.

✔ Model

```
y = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n
```
Where:

- → predicted output
- → input features
- → model weights

☆ Cost Function: Mean Squared Error (MSE)

```
J(w) = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \hat{y_i})^2
```

✔ Goal

Minimize MSE by choosing appropriate weights.

✔ Limitations

- Overfitting when many features
- Weights can become too large
- Sensitive to multicollinearity
- Not good when features are highly correlated

☆ 2. Ridge Regression (L2 Regularization)

Ridge Regression solves the limitations of Linear Regression by adding a penalty term.

✔ Ridge Cost Function

```
J(w) = \frac{1}{2m}\sum (y_i - \hat{y_i})^2 + \lambda \sum_{j=1}^{n} w_j^2
```

Where:

- = Regularization parameter
- = L2 penalty

☆ Intuition

Ridge **shrinks coefficients** but **does not make them zero**.

✔ Prevents:

- Overfitting
- Large coefficients
- Multicollinearity issues

☆ Properties

- Coefficients become small (shrinked).
- Model becomes more stable.
- Works well when many small/medium-sized effects exist.

✔ When to Use Ridge?

- When all features contribute a little
- When features are highly correlated
- When we want to reduce model complexity but not remove features

3. Lasso Regression (L1 Regularization)

Lasso (Least Absolute Shrinkage and Selection Operator) adds **L1 penalty** on weights.

Lasso Cost Function

```
J(w) = (1 / (2m)) * Σ (y_i - y_hat_i)^2  +  λ * Σ_{j=1 to n} |w_j|
```

Intuition

Lasso **shrinks some coefficients to zero**, thus performing **feature selection**.

Properties

- Produces **sparse models**[1]
- Selects important features automatically
- Good for high-dimensional data (many features)

When to Use Lasso?

- When many features are irrelevant
- When automatic feature selection is needed
- When model must be simple and interpretable

4. Comparison Table

Feature	Linear Regression	Ridge	Lasso
Regularization	❌ No	✓ L2	✓ L1
Cost Function	MSE	MSE + λ Σ w^2	MSE + λ Σ
Shrinks Coefficients	❌ No	✓ Yes	✓ Yes
Can set coefficients to zero	❌ No	❌ No	✓ Yes
Handles multicollinearity	Poor	Good	Average
Feature Selection	No	No	Yes
Model Complexity	High	Medium	Low (sparse)

☆ 5. Visual Intuition

- Linear Regression → No penalty → Large weights possible
- Ridge → Circular constraint → Shrinks weights
- Lasso → Diamond constraint → Corners cause weights to become **exact zero**

☆ 6. Key Formulas

✓ Linear Regression solution

```
w = (X^TX)^{-1}X^Ty
```

✓ Ridge Regression solution

```
w = (X^TX + \lambda I)^{-1}X^Ty
```

✓ Lasso Regression solution

No closed-form solution → solved using **coordinate descent**.

☆ 7. Applications

✓ Linear Regression

- Salary prediction
- House price prediction
- Trend analysis

✓ Ridge Regression

- Medical prediction models
- Marketing prediction
- Problems with multicollinearity

✓ Lasso Regression

- High-dimensional datasets
- Feature selection in genetics
- Sparse models in finance

Below are **clear, simple, exam-ready explanations** of **Principal Component Analysis (PCA)** and **Partial Least Squares (PLS)** — **no extra design**, only clean theory and points.

---

1. Principal Component Analysis (PCA)

Definition

PCA is an **unsupervised dimensionality reduction technique** that transforms high-dimensional data into a smaller set of new uncorrelated variables called **principal components**, while retaining maximum variance.

Key Ideas

- Reduces number of features.
- Removes correlation between variables.
- Captures maximum variance in first few components.
- Useful in noise reduction and visualization.

How PCA Works (Steps)

- **Standardize the data**
  Convert features to same scale.
- **Compute the covariance matrix**
  Shows how features vary with each other.
- **Find Eigenvalues and Eigenvectors**
  - Eigenvectors → directions of maximum variance (principal components)
  - Eigenvalues → amount of variance captured
- **Sort eigenvalues in descending order**
  Select top k components.
- **Project original data onto new components**
  Reduced-dimension dataset is formed.

Properties

- Principal components are **orthogonal** (uncorrelated).
- First PC captures **maximum variance**.
- Second PC captures next maximum variance, and so on.
- PCA is **unsupervised** — it ignores class labels.

Applications

- Image compression
- Face recognition (Eigenfaces)
- Data visualization (2D plots)
- Noise reduction
- Removing multicollinearity in regression

---

2. Partial Least Squares (PLS)

Definition

PLS is a **supervised dimensionality reduction method** used when:

- There are many features
- Features are highly correlated
- Goal is prediction (regression or classification)

PLS finds new components that maximize the **covariance between predictors (X) and output (Y)**.

Key Difference from PCA

- PCA → maximizes variance in **X only** (unsupervised)
- PLS → maximizes covariance between **X and Y** (supervised)

This means PLS chooses components that are **most useful for predicting Y**.

How PLS Works (Concept)

- Computes latent components (called **PLS components**).
- Each component is chosen to have:
  - High variance in X
  - High correlation with Y
- Reduces dimensions while preserving predictive power.

PLS vs PCA

Feature	PCA	PLS
Type	Unsupervised	Supervised
Uses Y (target)?	❌ No	✅ Yes
Objective	Maximize variance in X	Maximize covariance of X & Y
Best for	Data compression	Prediction problems
Feature correlation	Removes	Uses correlation for prediction

Applications of PLS

1. Chemometrics (spectral data)

2. Bioinformatics
3. Quantitative structure–activity relationships (QSAR)
4. Predictive modeling with many features
5. Situations where number of predictors >> number of samples

---

Below is a **clean, exam-oriented explanation** of all topics.
Plain text only — **no design**, no symbols that break while copying.

---

1. Linear Classification

Linear classification uses a **linear decision boundary** to separate classes.

Decision Function

```
f(x) = w^T x + b
```

Prediction Rule

```
y = +1  if  f(x) >= 0
y = -1  if  f(x) < 0
```

Examples

- Perceptron
- Linear SVM
- Logistic Regression (though probabilistic)

Advantages

- Fast
- Interpretable
- Works well when classes are linearly separable

---

2. Logistic Regression

Logistic Regression is a **classification model**, not regression.
It predicts **probability of class 1**.

Linear Score

```
z = w^T x + b
```

Sigmoid Function

```
sigmoid(z) = 1 / (1 + exp(-z))
```

Class Probability

```
P(y=1 | x) = sigmoid(w^T x + b)
```

Loss Function (Binary Cross Entropy)

```
L = - [ y * log(p) + (1 - y) * log(1 - p) ]
```

Training

Use **gradient descent** to minimize loss.

Decision Rule

```
Predict 1 if P >= 0.5
Else predict 0
```

Advantages

- Output probability
- Simple and widely used
- Works well for binary classification

---

3. Linear Discriminant Analysis (LDA)

LDA is a **generative probabilistic classifier** based on Bayes' rule.

It assumes:

1. Each class is normally distributed
2. **Same covariance matrix** for all classes
3. Different mean vectors

Class Conditional Density

```
p(x | y=k) = Normal( mean = μ_k , covariance = Σ )
```

Discriminant Function

```
δ_k(x) = x^T Σ^-1 μ_k - 0.5 μ_k^T Σ^-1 μ_k + log(π_k)
```

Where π_k = prior probability of class k.

Decision Rule

Assign x to class with largest δ_k(x).

Decision Boundary

Because covariance is same for all classes →
Boundary is linear.

When LDA Works Best

- Classes are normally distributed
- Equal covariance assumption holds
- Large sample size

---

4. Quadratic Discriminant Analysis (QDA)

QDA is similar to LDA but **each class has its own covariance matrix**.

Assumption:

```
p(x | y=k) = Normal( mean = μ_k , covariance = Σ_k )
```

Discriminant Function

```
δ_k(x) =
- 0.5 * log |Σ_k|
- 0.5 * (x - μ_k)^T Σ_k^-1 (x - μ_k)
+ log(π_k)
```

Decision Boundary

Since Σ_k is different for each class →
Decision boundary becomes quadratic.

Difference Between LDA and QDA

Feature	LDA	QDA
Covariance	Same Σ for all classes	Different Σ_k
Boundary	Linear	Quadratic
Parameters	Fewer	Many
Need large dataset	No	Yes
More flexible	Less	More

Advantages of QDA

- More flexible
- Can fit complex class boundaries

Disadvantages

- Requires more data
- Overfits if dataset is small

---

5. Summary Table

Method	Boundary	Uses Y?	Distribution Assumption	Covariance	Linear/Non-linear
Linear Classification	Linear	Yes	None	None	Linear
Logistic Regression	Linear	Yes	None	None	Linear
LDA	Linear	Yes	Gaussian	Same for all classes	Linear

Method	Boundary	Uses Y?	Distribution Assumption	Covariance	Linear/Non-linear
QDA	Quadratic	Yes	Gaussian	Different per class	Non-linear

---

1. Support Vector Machines (SVM)

Definition

SVM is a **supervised classification algorithm** that finds the **best separating boundary** between classes.

Key Idea

SVM tries to find a **hyperplane** that maximizes the **margin** (distance) between classes.

Hyperplane

A decision boundary:

```
w^T x + b = 0
```

Margin

Distance of the closest points (called **support vectors**) from the hyperplane.

Goal of SVM

Maximize the margin:

```
Maximize 2 / ||w||
```

This reduces overfitting and improves generalization.

Soft Margin SVM

Allows some misclassification using penalty parameter **C**.

☆ Kernels in SVM

When data is not linearly separable, SVM uses **kernel tricks** to map data into higher dimensions.

Common Kernels:

1. **Linear Kernel**

```
K(x, z) = x^T z
```

- **Polynomial Kernel**

```
K(x, z) = (x^T z + c)^d
```

- **RBF / Gaussian Kernel**

```
K(x, z) = exp(-||x - z||^2 / (2σ^2))
```

- **Sigmoid Kernel**

```
K(x, z) = tanh(k * x^T z + c)
```

Why use kernels?

They allow SVM to learn **non-linear boundaries** without increasing dimensionality manually.

---

2. Artificial Neural Networks (ANN)

Definition

ANN is a machine learning model made of **neurons** organized in layers:

- Input layer
- Hidden layers
- Output layer

Each neuron computes:

```
z = w^T x + b
a = activation(z)
```

Common Activation Functions:

1. Sigmoid: `1 / (1 + exp(-z))`
2. ReLU: `max(0, z)`
3. Tanh: `(e^z - e^-z) / (e^z + e^-z)`

☆ Backpropagation

Backpropagation is the **learning algorithm** for neural networks.

Goal

Minimize the error using gradient descent.

Steps

- **Forward pass**
  Network computes predictions.
- **Calculate error**
  Example: Mean Square Error or Cross Entropy.
- **Backward pass**
  Compute gradients of weights using chain rule.

- **Update weights**

```
w = w - learning_rate * gradient
```

Backpropagation updates weights layer-by-layer starting from output back to input.

---

3. Decision Trees

Definition

A tree-based model that splits data into groups based on feature values.

How it works

At each node, the algorithm selects the **best feature** to split data to create pure subsets.

Metrics used

1. **Gini Index**

```
Gini = 1 - sum(p_i^2)
```

2. **Entropy**

```
Entropy = - sum( p_i * log2(p_i) )
```

3. **Information Gain**

```
IG = Entropy(parent) - Weighted entropy(children)
```

Important Terms

4. Root node: first split
5. Internal node: decision
6. Leaf node: final class

Advantages

- Easy to interpret
- Handles non-linear relations
- Works for classification and regression

Disadvantages

- Overfitting
- Unstable to small data changes

---

4. Bayes Optimal Classifier

Definition

The Bayes Optimal Classifier is the **best possible classifier** because it uses the full posterior probability distribution.

Decision Rule

Choose class with **maximum posterior probability**:

```
Choose class k if P(class_k | x) is highest.
```

Key Point

1. It always achieves **minimum possible error**.
2. But it is often **impossible** to compute (posterior distribution is unknown).

This is a theoretical ideal classifier.

---

5. Naive Bayes Classifier

Definition

Naive Bayes uses **Bayes theorem** assuming that all features are **conditionally independent** given the class.

Bayes Rule

```
P(y | x) = [ P(x | y) * P(y) ] / P(x)
```

Naive Assumption

```
P(x1, x2, x3…| y) = P(x1|y) * P(x2|y) * P(x3|y) * ...
```

Prediction rule

```
Choose the class y with highest P(y | x)
```

Types of Naive Bayes

1. **Gaussian Naive Bayes** (continuous data)
2. **Multinomial Naive Bayes** (text classification)
3. **Bernoulli Naive Bayes** (binary features)

Advantages

2. Fast
3. Works well even with limited data
4. Excellent for text (spam, sentiment)

Disadvantages

3. Assumption of independence rarely true
4. May perform poorly if features strongly correlated

Hypothesis Testing

Hypothesis testing is a statistical method used to make decisions about a population based on sample data.

Steps

- **Define hypotheses**
    - Null hypothesis (H0): No effect, no difference.
    - Alternative hypothesis (H1): Effect exists.
- **Choose significance level (α)**
  Usually α = 0.05.
- **Select test statistic** (z-test, t-test, chi-square, F-test etc.)
- **Compute p-value**
- **Decision Rule**
    - If p-value ≤ α → reject H0
    - If p-value > α → fail to reject H0

Types

- **One-tailed test**: checks only one direction.
- **Two-tailed test**: checks both directions.

Errors

- Type-I error: rejecting true H0
- Type-II error: failing to reject false H0

Ensemble Methods

Ensemble methods combine multiple models to improve accuracy and reduce overfitting.

Why Ensemble?

- Individual models have errors.
- Combining many models reduces variance and improves generalization.

Types of Ensemble

1. **Bagging**
2. **Boosting**
3. **Stacking**

Bagging (Bootstrap Aggregating)

Bagging reduces **variance** by training multiple models on different bootstrap samples.

Process

- Randomly sample dataset with replacement → bootstrap samples

- Train same model (e.g., decision tree) on each sample.
- Combine predictions:
  - Classification → majority voting
  - Regression → average

Example

Random Forest = Bagging + Decision Trees

Advantages

- Reduces overfitting
- Works well with unstable models (trees)

AdaBoost (Adaptive Boosting)

Boosting reduces **bias** by focusing on difficult samples.

Process

1. Start with equal weights on all training samples.
2. Train weak learner (e.g., small decision tree).
3. Increase weights of misclassified points.
4. Train next classifier focusing on hard points.
5. Final prediction = weighted sum of all weak learners.

Important Formulas

Error of classifier:

```
err_t = sum(w_i * I(y_i != h_t(x_i)))
```

Classifier weight:

```
alpha_t = 0.5 * ln((1 - err_t) / err_t)
```

Weight update:

```
w_i ← w_i * exp(-alpha_t * y_i * h_t(x_i))
```

Normalize weights.

Characteristics

- Sequential learning
- Reduces bias
- Sensitive to noise

Gradient Boosting

Boosting method based on **gradient descent** in function space.

Idea

Build model step-by-step by minimizing loss function:

```
F_0(x) = initial prediction
F_t(x) = F_(t-1)(x) + learning_rate * h_t(x)
```

Where h_t(x) = model trained to fit **negative gradient of loss**.

Steps

- Initialize model with constant prediction.
- Compute residuals (negative gradient of loss).
- Train weak learner (tree) to predict residuals.
- Update model.

Used in

- XGBoost
- LightGBM
- CatBoost

Advantages

1. Very high accuracy
2. Works with any differentiable loss

Disadvantages

1. Slower than bagging
2. Sensitive to overfitting (use regularization)

Summary Table

Method	Purpose	How it Works	Reduces	Example
Bagging	Improve stability	Train on bootstrap samples	Variance	Random Forest
AdaBoost	Improve accuracy	Focus on misclassified samples	Bias	AdaBoost
Gradient Boosting	Sequential error correction	Fit gradients of loss function	Bias & variance	XGBoost, LightGBM

---

1. Clustering

Clustering is an **unsupervised learning** method that groups similar data points into clusters.

Goal

Maximize **intra-cluster similarity** and minimize **inter-cluster similarity**.

Types of Clustering

- Partition-based
- Hierarchical
- Density-based
- Grid-based
- Model-based
- Spectral methods

---

2. K-means Clustering

K-means is a **partition-based**, **centroid-based** clustering algorithm.

Steps

1. Choose number of clusters k
2. Initialize k centroids (randomly)
3. Assign each point to nearest centroid using Euclidean distance
4. Update centroids by taking the **mean** of points in each cluster
5. Repeat assignment + update until convergence

Objective Function

Minimize **sum of squared distances**:

$J = \Sigma \Sigma || x_i - \mu_j ||^2$
where μ_j = centroid of cluster j.

Pros

- Simple and fast
- Works well for spherical clusters

Cons

1. Requires k in advance
2. Sensitive to outliers
3. Cannot detect arbitrary shapes

---

3. K-medoids (PAM – Partitioning Around Medoids)

K-medoids is similar to K-means but uses **medoid** (actual data point) instead of mean.

Medoid

A representative point in the cluster with **minimum total distance** to all other points.

Steps

- Choose k medoids (initial)
- Assign points to nearest medoid
- Try swapping medoids with non-medoids to reduce cost
- Continue until no improvement

Objective Function

Minimize sum of pairwise dissimilarities:

```
J = Σ Σ distance(x_i , medoid_j)
```

Pros

- Robust to outliers
- Works with any distance metric

Cons

- Slower than K-means
- Not suitable for very large datasets

---

4. Density-Based Clustering (DBSCAN)

DBSCAN = **Density-Based Spatial Clustering of Applications with Noise**

Key Concepts

1. **eps**: radius
2. **minPts**: minimum number of points for dense region
3. **Core point**: ≥ minPts within eps
4. **Border point**: < minPts but inside eps of a core point
5. **Noise point**: not core, not border

Steps

1. Select an unvisited point
2. If it is a core point → form a cluster
3. Expand the cluster by visiting reachable density points
4. Mark others as noise

Pros

- Finds clusters of arbitrary shape
- Handles noise and outliers
- No need to specify k

Cons

- EPS and minPts selection is hard
- Not good for varying density clusters

---

5. Hierarchical Clustering

Builds a hierarchy (tree-like structure) of clusters.

Two Types

1. **Agglomerative (bottom-up)**
   Start with each point as its own cluster → merge repeatedly.
2. **Divisive (top-down)**
   Start with one cluster → repeatedly split.

Distance (Linkage) Methods

- **Single linkage**: min distance between points
- **Complete linkage**: max distance
- **Average linkage**: average distance
- **Ward's method**: minimize variance

Output

A **dendrogram** that shows how clusters are merged/split.

Pros

- No need for k
- Good for small datasets

Cons

- Expensive for large datasets
- Hard to undo a bad merge

---

6. Spectral Clustering

Spectral clustering uses **eigenvalues** of a similarity matrix to find clusters.

Steps

1. Construct similarity graph (S)
   Example: $S(i,j) = \exp(- ||xi - xj||^2 / (2\sigma^2))$
2. Compute **graph Laplacian matrix**:
3. `L = D - S`
   where D = diagonal degree matrix

4. Compute **top k eigenvectors** of L
   → forms a low-dimensional representation

5. Apply **K-means** on these eigenvectors

Why Spectral Clustering?

- Works well for **non-convex** and **complex** cluster shapes
- Uses graph theory to detect cluster structure

Pros

- Can detect complex shapes
- More accurate than K-means for tricky data

Cons

1. Needs eigen-decomposition
2. Expensive for large datasets

---

1. Expectation–Maximization (EM) Algorithm

EM is an iterative optimization algorithm used when data has **missing values**, **latent variables**, or **hidden structure**.

Goal

Maximize the **likelihood function** when direct computation is difficult.

Steps

EM alternates between two steps:

E-Step (Expectation)

Compute the expected value of hidden variables using current parameters.

Mathematically:

```
Q(θ | θ_old) = E[ log L(θ ; X, Z) | X, θ_old ]
```

Where:
X → observed data
Z → hidden variables
θ → model parameters

M-Step (Maximization)

Maximize this expected log-likelihood to update parameters.

```
θ_new = argmax_θ Q(θ | θ_old)
```

Repeat E-step and M-step until convergence.

Where Used

- Gaussian Mixture Models
- Hidden Markov Models
- Missing data estimation

---

2. Gaussian Mixture Models (GMMs)

GMM represents data as a mixture of multiple Gaussian distributions.

Probability Model

```
p(x) = Σ_k π_k * N(x | μ_k, Σ_k)
```

Where:
π_k = mixing weight
μ_k = mean of k-th Gaussian
Σ_k = covariance matrix

Goal

Estimate parameters {π_k, μ_k, Σ_k}.

How GMM is Learned

Using **Expectation-Maximization**:

E-Step

Compute probability each point belongs to each cluster (responsibility):

```
γ(z_k) = π_k * N(x | μ_k, Σ_k) / Σ_j π_j * N(x | μ_j, Σ_j)
```

M-Step

Update parameters:

```
N_k = Σ_i γ_i(k)
μ_k = (1/N_k) Σ_i γ_i(k) x_i
Σ_k = (1/N_k) Σ_i γ_i(k) (x_i − μ_k)(x_i − μ_k)^T
π_k = N_k / N
```

Advantages

- Soft clustering
- Handles complex-shaped clusters
- More flexible than K-means

---

3. Learning Theory (Introduction)

Learning theory studies **how algorithms learn**, **how well they generalize**, and **what guarantees exist**.

Key Concepts

Hypothesis Class (H)

Set of all possible models an algorithm can choose from.

Generalization

Performance on unseen data.

Overfitting

Good on training data, bad on test data.

VC Dimension

Measures capacity/complexity of a model.
Higher VC → high flexibility → risk of overfitting.

Bias–Variance Tradeoff

- High bias → underfitting
- High variance → overfitting

PAC Learning (Probably Approximately Correct)

A framework saying that learning is possible if:

1. Hypothesis class is not too large
2. Enough samples are available

---

4. Introduction to Reinforcement Learning (RL)

Reinforcement Learning is a learning framework where an **agent** learns by interacting with an **environment** to maximize **reward**.

Key Components

- **Agent**: learner
- **Environment**: where agent acts
- **State (s)**: current situation
- **Action (a)**: choice made by agent
- **Reward (r)**: feedback
- **Policy (π)**: mapping from state → action
- **Value Function (V or Q)**: expected future reward

Goal

Maximize **cumulative reward**.

Types of RL Methods

- **Model-free**
  - Q-learning
  - SARSA
  - Deep Q Networks (DQN)
- **Model-based**
  - Learn transition probabilities

Exploration vs Exploitation

- Explore: try new actions
- Exploit: choose best-known action

---

5. Bayesian Networks (Bayes Nets)

Bayesian Networks are **probabilistic graphical models** that represent **joint probability distributions** using a **Directed Acyclic Graph (DAG)**.

Nodes

Random variables.

Edges

Dependencies / causal relationships.

Each node has a Conditional Probability Table (CPT):

```
P(X | Parents(X))
```

Joint Probability Distribution

Computed using:

```
P(X1, X2, …, Xn) = Π P(X_i | Parents(X_i))
```

Advantages

- Represents uncertainty
- Supports inference
- Explains causal structure
- Works with incomplete data

Applications

- Medical diagnosis
- Fraud detection
- Speech recognition
- Robotics