GeeksforGeeks

GATE फर्स्ट

DATA SCIENCE & ARTIFICIAL INTELLIGENCE (DA)

MACHINE LEARNING

SHORT NOTES

ENROLL NOW

TO EXCEL IN GATE
AND ACHIEVE YOUR DREAM IIT OR PSU!

ENROLL NOW

STAR MENTOR CS/DA

KHALEEL SIR
ALGORITHM & OS
29 YEARS OF TEACHING EXPERIENCE

CHANDAN SIR
DIGITAL LOGIC
GATE AIR 23 & 26 / EX-ISRO

SATISH SIR
DISCRETE MATHEMATICS
BE in IT from MUMBAI UNIVERSITY

MALLESHAM SIR
M.TECH FROM IIT BOMBAY
AIR – 114, 119, 210 in GATE
(CRACKED GATE 8 TIMES)
14+ YEARS EXPERIENCE

VIJAY SIR
DBMS & COA
M. TECH FROM NIT
14+ YEARS EXPERIENCE

PARTH SIR
DA
IIIT BANGALORE ALUMNUS
FORMER ASSISTANT PROFESSOR

SAKSHI MA'AM
ENGINEERING MATHEMATICS
IIT ROORKEE ALUMNUS

SHAILENDER SIR
C PROGRAMMING & DATA STRUCTURE
M.TECH in Computer Science
15+ YEARS EXPERIENCE

AVINASH SIR
APTITUDE
10+ YEARS OF TEACHING EXPERIENCE

AJAY SIR
PH.D. IN COMPUTER SCIENCE
12+ YEARS EXPERIENCE

Performance Metrics in Machine Learning – GATE DA Notes

1. Classification Metrics:

- Accuracy: Accuracy = (TP + TN) / (TP + TN + FP + FN) – Overall correctness
- Precision: Precision = TP / (TP + FP) – How many predicted positives are correct
- Recall (Sensitivity): Recall = TP / (TP + FN) – How many actual positives are captured
- Specificity: Specificity = TN / (TN + FP) – How many actual negatives are correctly identified
- F1 Score: F1 Score = 2 * (Precision * Recall) / (Precision + Recall)
- ROC Curve: Graph between True Positive Rate (TPR) and False Positive Rate (FPR)
- AUC (Area Under Curve): Value between 0 and 1 indicating classifier's ability to distinguish between classes
- Log Loss: Negative log-likelihood of true labels given predicted probabilities

Formula (Binary case):

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1-y_i)\log(1-p_i)]$$

- **N:** Total number of samples
- **y_i:** True label (0 or 1) for sample i
- **p_i:** Predicted probability for class 1 for sample i

For multi-class classification, sum the negative log of the predicted probability of the true class for each observation.

- Cohen's Kappa: Measures agreement between predicted and true values, adjusted for chance

Formula:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

- **p_o:** Observed agreement (proportion of times the predicted and actual labels agree)
- **p_e:** Expected agreement by chance (calculated from label distribution)

- Matthews Correlation Coefficient (MCC): Balanced metric even for imbalanced classes

Formula (Binary case):

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

- **TP:** True Positives
- **TN:** True Negatives
- **FP:** False Positives
- **FN:** False Negatives

MCC returns a value between -1 (completely wrong) and +1 (perfect prediction); 0 indicates random prediction.

2. Regression Metrics:

- Mean Absolute Error (MAE): MAE = (1/n) Σ $|y_i - \hat{y}_i|$ – Average absolute error
- Mean Squared Error (MSE): MSE = (1/n) Σ $(y_i - \hat{y}_i)^2$ – Penalizes large errors more
- Root Mean Squared Error (RMSE): RMSE = √MSE – Same unit as target variable
- R^2 Score: R^2 = 1 - (SS_res / SS_tot) – Proportion of variance explained
- Adjusted R^2: Adjusts R^2 for the number of predictors
- Mean Absolute Percentage Error (MAPE): MAPE = (100% / n) Σ $|(y_i - \hat{y}_i) / y_i|$

3. Clustering Metrics:

- Silhouette Score: Measures how close each point in a cluster is to points in neighboring clusters
- Davies-Bouldin Index: Lower value indicates better clustering (separation + compactness)
- Calinski-Harabasz Index: Higher value → better-defined clusters
- Inertia: Sum of squared distances to the nearest cluster center (used in K-Means)
- Adjusted Rand Index (ARI): Measures the similarity between the predicted and true clustering
- Normalized Mutual Information (NMI): Mutual information normalized between predicted and true clusters

Introduction to Machine Learning – GATE DA Short Notes

1. Definition and Basics

Machine Learning (ML) is the science of programming computers to learn from data/experience and improve performance on a task without being explicitly programmed.
A computer program is said to learn from experience (E) with respect to some class of tasks (T) and performance measure (P) if its performance at tasks T, as measured by P, improves with experience E.
Example:
- Handwriting Recognition:
  T = Recognize handwriting
  P = Accuracy (%)
  E = Dataset of labeled handwritten words

Type	Description	Example
Supervised	Learn from labelled data (input-output pairs)	Classification, Regression
Unsupervised	Find patterns in unlabelled data	Clustering, Dim. Reduction
Reinforcement	Learn via feedback (rewards or penalties)	Game playing, Robot control

2. Types of Learning Paradigms

*Supervised learning includes both classification (categorical output) and regression (continuous output).

3. Core Components of ML

- Data Storage: Input source (memory, DBs)
- Abstraction: Extract meaningful representations
- Generalization: Extend learning to unseen data
- Evaluation: Assess prediction accuracy or error

4. Common Learning Models

Logical Models: Use Boolean expressions or IF-THEN rules (e.g., Decision Trees, Random Forest)
Geometric Models: Represent features in n-dimensional space (e.g., Linear Models, KNN)
Probabilistic Models: Use probability distributions (e.g., Naive Bayes, Logistic Regression)
Grouping: Local decisions (e.g., decision trees)
Grading: Global scoring (e.g., neural networks)

5. Designing an ML System

Steps:
1. Define T (task), P (performance metric), E (experience)
2. Choose:
   - Target Function: f: X → Y
   - Representation: Rules, Trees, Linear Models
   - Approximation Algorithm: Learn function (e.g., Gradient Descent)
3. Evaluate and refine using loss/error metrics
Checkers Example:
- T = Win the game
- P = % of games won
- E = Games played against self
- f(b) = Score for a board state
Learned as: $V(b) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_6x_6$

6. Candidate-Elimination and Version Space

- Version Space: All hypotheses consistent with training data
- Candidate-Elimination Algorithm:
- Maintains S (specific boundary) & G (general boundary)
- Prunes hypotheses inconsistent with examples

7. PAC Learning (Probably Approximately Correct)

- Proposed by Leslie Valiant
- Goal: With high probability (1-δ), produce a hypothesis with low error (ε)
- Helps analyze algorithm efficiency and sample complexity

8. VC Dimension (Vapnik-Chervonenkis)

- Measures model capacity: max points that can be shattered
- Shattering: If all 2^n labelings of N points are possible by some hypothesis
- VC(H) = 4 for axis-aligned rectangles in 2D

9. Applications of ML

Domain	Use Case
Retail	Consumer behaviour prediction
Finance	Fraud detection, credit scoring
Manufacturing	Fault detection, process optimization
Healthcare	Medical diagnosis
AI/Robotics	Game playing, control, vision
Web	Recommendation engines, search ranking

10. Algorithms (SUPERVISED LEARNING)

A. Linear Regression

Linear Regression is a supervised learning algorithm used for predicting a continuous target variable based on one or more input features by fitting a linear equation.

Assumptions

- Linearity between x and y
- Homoscedasticity (constant variance of error)
- Independence of errors
- Normally distributed residuals

Form

$y=\beta_0+\beta_1x+\varepsilon$

Where:

- y: Dependent (target) variable
- x: Independent (input) variable
- β_0: Intercept
- β_1: Slope
- ε: Error term

Minimize **Sum of Squared Errors (SSE)**:

- $SSE=\sum(y_i-\hat{Y}_i)^2$

Estimation (Least Squares Method)

$\beta_1=\sum_{i=1}^{N}(xi-\bar{x})(yi-\bar{y})/\sum(xi-\bar{x})^2$

$\beta_0=\bar{y}-\beta_1\bar{x}$

Where

$\bar{x}$= average of x values

$\bar{y}$= average of y values

Multiple Linear Regression

Form

$y=\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_nx_n+\varepsilon$

Predicts the output using **multiple features** $x_1,x_2,\ldots,x_n$

Normal Equation (Closed-form Solution)

$\beta=(x^Tx)^{-1}x^Ty$

Performance Matrix:

- Mean Absolute Error (MAE): MAE = (1/n) Σ $|y_i - \hat{y}_i|$ – Average absolute error
- Mean Squared Error (MSE): MSE = (1/n) Σ $(y_i - \hat{y}_i)^2$ – Penalizes large errors more
- Root Mean Squared Error (RMSE): RMSE = √MSE – Same unit as target variable
- R^2 Score: R^2 = 1 - (SS_res / SS_tot) – Proportion of variance explained
- Adjusted R^2: Adjusts R^2 for the number of predictors
- Mean Absolute Percentage Error (MAPE): MAPE = (100% / n) Σ $|(y_i - \hat{y}_i) / y_i|$

B. Logistic Regression

Definition

- A **discriminative**, **supervised** machine learning algorithm used for classification.
- Computes **P(y|x)** directly, unlike generative models like Naive Bayes that model **P(x|y)P(y)**.

Binary Classification

- Predicts probability:
- $P(y=1|x) = \sigma(w\cdot x+b) = 1/1+e^{-(w,x+b)}$
- Uses **sigmoid** function to map any real-valued input to (0,1).

Multiclass Classification

- Uses Softmax function**:**

$$\mathbf{P(y=k\mid x)}=\frac{e^{w_kx+b_k}}{\sum_{j=1}^{K}e^{w_kx+b_k}}$$

- Also called Multinomial Logistic Regression **or** MaxEnt Classifier**.**

GeeksforGeeks

GATE CSE BATCH

KEY HIGHLIGHTS:

- 300+ HOURS OF RECORDED CONTENT
- 900+ HOURS OF LIVE CONTENT
- SKILL ASSESSMENT CONTESTS
- 6 MONTHS OF 24/7 ONE-ON-ONE AI DOUBT ASSISTANCE
- SUPPORTING NOTES/DOCUMENTATION AND DPPS FOR EVERY LECTURE

COURSE COVERAGE:

- ENGINEERING MATHEMATICS
- GENERAL APTITUDE
- DISCRETE MATHEMATICS
- DIGITAL LOGIC
- COMPUTER ORGANIZATION AND ARCHITECTURE
- C PROGRAMMING
- DATA STRUCTURES
- ALGORITHMS
- THEORY OF COMPUTATION
- COMPILER DESIGN
- OPERATING SYSTEM
- DATABASE MANAGEMENT SYSTEM
- COMPUTER NETWORKS

LEARNING BENEFIT:

- GUIDANCE FROM EXPERT MENTORS
- COMPREHENSIVE GATE SYLLABUS COVERAGE
- EXCLUSIVE ACCESS TO E-STUDY MATERIALS
- ONLINE DOUBT-SOLVING WITH AI
- QUIZZES, DPPS AND PREVIOUS YEAR QUESTIONS SOLUTIONS

ENROLL NOW

TO EXCEL IN GATE AND ACHIEVE YOUR DREAM IIT OR PSU!

ENROLL NOW

Features	Real-valued vector representation of input x=[x1,x2,...,xn]
Weights (w)	Coefficients learned for each feature
Bias (b)	Intercept term
Score (z)	w·x+b also called **logit**
Activation	Sigmoid (binary), Softmax (multiclass)

Training: Learning the Parameters

	Predicted: Class 1	Class 2	.....	Class N
Actual: Class 1	True Positive (TP_1)	False Pred.	.....	False Pred.
Class 2	False Pred.	TP_2	.....	.....
.....	.....	.....	.....	.....
Class N	False Pred.	.....	.....	TP_n

Classification Metrics

Accuracy

Accuracy = (TP + TN) / (TP + TN + FP + FN)
- Measures overall correctness.
- Can be misleading with imbalanced datasets.

Precision

Precision = TP / (TP + FP)
- Of all predicted positives,
how many are actually correct?
- High precision indicates low false positives.

Recall (Sensitivity / True Positive Rate)

Recall = TP / (TP + FN)
- Of all actual positives,
how many were correctly
predicted?
- High recall indicates
low false negatives.

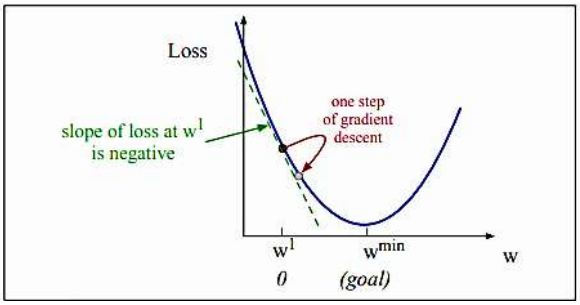
Specificity (True Negative Rate)

Specificity = TN / (TN + FP)
- Of all actual negatives,
how many were correctly predicted?

F1 Score

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

- **Loss Function:** Cross-Entropy
$L(y, \hat{y}) = -[y\log(\hat{y})+(1-y)\log(1-\hat{y})]$
- **Optimization:** Gradient Descent **Gradient Descent Variants**:



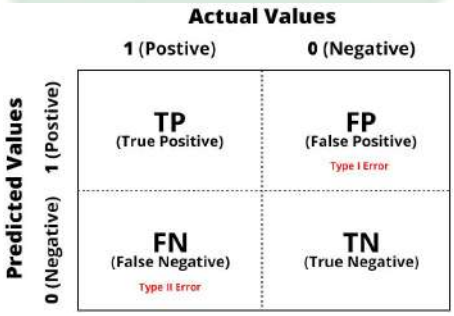
  - Stochastic (per example)
  - Mini-bat

- Harmonic mean of Precision and Recall.
- Useful in imbalanced class settings.

ROC Curve and AUC Score

ROC Curve plots True Positive Rate (Recall) vs. False Positive Rate (FPR = FP / (FP + TN))
AUC (Area Under Curve):
- Ranges between 0 and 1
- 1.0 = perfect classifier, 0.5 = random guess
- Higher AUC indicates better model performance.

C. Support Vector Machine



- Support Vector Machine is a supervised learning algorithm used for classification and regression tasks.
- SVM finds the optimal hyperplane that best separates data points of different classes in a feature space.

Key Concepts

- **Hyperplane**

A hyperplane is a decision boundary that separates data into classes. In 2D, it's a line; in 3D, it's a plane; in higher dimensions, a hyperplane.

- **Margin and Support Vectors**

Margin: The distance between the hyperplane and the closest points from either class. Support Vectors: Data points that lie closest to the hyperplane; they "support" or define the margin.

- **Optimal Hyperplane**

The best hyperplane is the one which has the maximum margin (distance to the support vectors). A larger margin implies better generalization and less overfitting.

- **Types of SVM**

Linear SVM: Works when data is linearly separable.
Non-Linear SVM: Uses Kernel Trick (see below) for data that is not linearly separable.
Mathematical Formulation

- Let data points be (xi,yi)(*xi*,*yi*) where yi=+1*yi*=+1 or −1−1.
- The equation of hyperplane: w⋅x+b=0*w*⋅*x*+*b*=0
- SVM solves:

$$min_{w,b}\frac{1}{2}\,||\ w\ ||^2$$

Subject to:

$$y_i(w\cdot x_i+b)\geq 1\ \forall\ i$$

- It's a **convex optimization** problem.
- Kernel Trick

If data is not linearly separable, SVM uses kernels to map inputs into higher-dimensional space:

- **Linear Kernel:** $K(x,y)=x^Ty$
- **Polynomial Kernel:** $K(x,y)=(x^Ty+c)^d$
- **Radial Basis Function (RBF)/Gaussian:** $K(x,y)=\exp(-\gamma||x-y||^2)$
- **Sigmoid Kernel:** $K(x,y)=\tanh(\alpha x^Ty+c)$

Soft Margin Vs Hard Margin

- Hard Margin SVM: No misclassification allowed.
- Soft Margin SVM: Allows some misclassification using a penalty parameter C*C* (trade-off between margin width and misclassification).

SVM for Multiclass Classification

- SVM is inherently a binary classifier.
- Multiclass handled using strategies like one-vs-one or one-vs-rest.

Advantages

- Effective in high-dimensional spaces.
- Works well with clear margin of separation.
- Memory efficient (only support vectors are used).

Limitations

- Not suitable for very large datasets (training can be slow).
- Performs poorly if classes overlap too much.
- Choice of kernel and parameters (like C and γ) crucial.

Applications

- Text categorization (spam/non-spam)
- Image classification
- Bioinformatics (protein classification)
- Handwriting recognition

C. Decision Tree

- **Definition**: A supervised learning algorithm used for classification and regression.
- **Structure**:
  - Internal nodes → Tests on attributes
  - Branches → Outcomes of the test
  - Leaf nodes → Class labels or outputs

ID3 Algorithm (Iterative Dichotomiser 3)

- **Uses Information Gain to choose attributes**:
  1. Calculate entropy H(S)
  2. For each attribute, calculate $IG(S,A)=H(S)-\sum P(x)H(Sx)$

Criterion	Gini Impurity	Entropy
Formula	$1-\sum pi^2$	$-\sum p_i \log_2(p_i)$
Range	[0, 0.5] (binary case)	[0, 1] (binary case)
Preference	Simpler, faster	More informative (but slower)
Bias	Slightly biased to larger classes	No such bias
Used In	CART	ID3, C4.5

3. Select attribute with maximum IG
4. Repeat recursively until leaf node formed

- **Entropy**: Measures uncertainty in data $H(S)=-\sum p_i \log_2(p_i)$
- The **Gini Impurity** is an alternative to **Entropy** used in decision tree algorithms like **CART (Classification and Regression Tree)**. It measures how often a randomly chosen element from the set would be **incorrectly classified** if it was randomly labeled according to the distribution of class labels in the set.
- Formula for Gini Impurity, For a dataset S with classes $C_1, C_2, \ldots, C_k$, the Gini impurity is:

$$Gini(S) = 1 - \sum_{i=1}^{k} {p_i}^2$$

Where:

- p_i= Proportion of samples belonging to class i

Interpretation

- **Gini = 0** → All elements belong to a **single class** (pure node)
- **Higher Gini** → Higher **class mixture/impurity**

Using Gini in Decision Tree Construction

CART Algorithm Steps:

1. For each feature:
   - Split the dataset on that feature
   - Calculate **weighted Gini impurity** of the resulting subsets
2. Choose the feature that results in the **lowest Gini impurity**
3. Repeat recursively for each branch
4.

Types of Decision Trees

- **Classification Tree**: Output is categorical (Yes/No)
- **Regression Tree**: Output is continuous

CART (Classification and Regression Tree)

- Creates binary trees
- Used in ensemble models like Random Forest

Advantages

- Easy interpretation (if-then rules)
- No need for data distribution assumptions
- Performs feature selection implicitly
- Requires **pruning or ensemble** (e.g., Bagging, Boosting)

Limitations

- Prone to overfitting
- High variance
- Needs pruning or ensemble techniques

C4.5 Algorithm – Successor of ID3

C4.5 improves on ID3. Gain Ratio Formula:

Gain_Ratio(A)=IG(S,A) / Split_Information(A)

C4.5 handles **noise and overfitting** better than ID3.

Feature	ID3	C4.5
Attribute Type	Categorical only	Handles continuous & categorical
Splitting Criterion	Information Gain	Uses Gain Ratio
Pruning	Not supported	Post-pruning using pessimistic error
Handling Missing	Not supported	Supported
Tree Type	Multi-branch	Multi-branch

D. Random Forest

- Random Forest is an **ensemble learning method** for classification and regression.

- It builds **multiple decision trees** and combines their results for more accurate and stable predictions.
- **Main idea:** Combine the predictions of multiple weak learners (trees) to improve generalization.

Key Concepts

1. Decision Tree

- A tree-based model that splits data on features for classification or regression.
- Prone to overfitting when used alone.

2. Ensemble Learning

- Uses multiple models to improve performance over a single model.

3. Bootstrap Aggregating (Bagging)

- Each tree is trained on a bootstrapped (random, with replacement) sample from the data.
- Helps to reduce model variance and prevent overfitting.

4. Random Feature Selection

- At each split in a tree, a subset of features is randomly selected and only those are considered for splitting.
- Ensures de-correlation between trees.

How Random Forest Works

1. For each of the *n* trees:
    - Take a bootstrap sample of the data (sample with replacement).
    - At each split, select the best split from a random subset of features.
    - Grow the tree completely (no pruning), or until a stopping condition is met.
2. For predictions:
    - **Classification:** Aggregate results by majority vote across all trees.
    - **Regression:** Aggregate results by averaging outputs of all trees.

Mathematical Representation:

- Classification Output:
  $\hat{y} = \text{majority\_vote}\{\text{tree}_i(x)\}$
- Regression Output:
  $\hat{y} = \frac{1}{n}\sum_{i=1}^{n}\text{tree}_i(x)$
  where n is the number of trees.

Key Parameters

Parameter	Description
n_estimators	Number of trees in the forest
max_features	Number of features to consider at each split
max_depth	Maximum depth of each decision tree
min_samples_split	Minimum samples required to split a node
bootstrap	Use of bootstrapped samples
random_state	Controls randomness for reproducibility

Advantages

- Robust to overfitting and noise.
- Handles high-dimensional data well.
- Works for both classification and regression.
- Provides feature importance estimates.

Limitations

- Computationally expensive for large datasets.
- Less interpretable than single decision trees.
- Hyperparameter tuning may be necessary for optimal performance.

Feature Importance

- Measures how much each feature improves the split quality in the trees.
- Averaged across the ensemble to rank features.

Applications

- Medical diagnosis
- Credit scoring and financial risk modeling
- Text classification
- Image recognition

Differences from Other Algorithms

Random Forest	Decision Tree	Bagging Ensemble
Ensemble of many trees	Single tree	Ensemble (often trees)
Bagging + feature selection at splits	-	Only bagging
Lower variance, usually higher accuracy	High variance	Somewhat correlated if features not sub-sampled

E. Regularization Techniques

- **Regularization** is a set of techniques used to prevent overfitting in machine learning models by penalizing complex models.
- It discourages learning a model that fits the training data too closely, improving model generalization on unseen data.

Main Regularization Techniques

1. L1 Regularization (Lasso)

- **Adds penalty** proportional to the absolute value of coefficients.
- **Cost Function:** $J = \text{Loss} + \lambda \sum_{j=1}^{n} |w_j|$
- **Effect:** Can shrink some coefficients exactly to zero (feature selection).
- **Applications:** When sparsity is desired or for feature selection.

2. L2 Regularization (Ridge)

- **Adds penalty** proportional to the square of the coefficients.
- **Cost Function:** $J = \text{Loss} + \lambda \sum_{j=1}^{n} w_j^2$
- **Effect:** Shrinks coefficients towards zero but never exactly zero.
- **Applications:** When multicollinearity exists, preferred for continuous predictors.

Closed-Form Solution

The weight vector w*w* that minimizes the regularized loss is given by:

$w^* = (x^T x + \lambda_i)^{-1} x^T y$

- x^T: Transpose of matrix X*X*
- I: Identity matrix (of size n×n*n*×*n*)
- λ: Regularization parameter (greater than zero)

Key Interpretations

- The addition of λI to $x^T x$ makes the matrix invertible even if $x^T x$ is singular or ill-conditioned.
- As λ increases, more penalty is applied, driving weights closer to zero—but not exactly zero.
- When λ=0, this reduces to ordinary least squares (OLS).

Practical Steps

1. Choose λ (often via cross-validation).
2. Solve for w^* using the formula above.
3. Use w^* for predictions: $\hat{Y} = Xw^*$

Summary

- **L2 regularization** penalizes large weights by adding their squared values to the loss.
- The **Ridge solution** is given in closed form for linear regression models.
- This encourages small, spread-out coefficients, improving model generalization and performance on unseen data.

3. Elastic Net Regularization

- **Combination** of L1 and L2 regularization.
- **Cost Function:** $J = \text{Loss} + \lambda_1 \sum_{j=1}^{n} |w_j| + \lambda_2 \sum_{j=1}^{n} w_j^2$
- **Effect:** Inherits benefits of both Lasso and Ridge; works well when there are multiple correlated features.

4. Dropout (for Neural Networks)

- **Randomly drops** units (and their connections) during training.
- Reduces dependency between nodes and prevents co-adaptation.
- Used mostly in deep learning models.

5. Early Stopping

- **Monitors validation loss** during training.
- Stops training once the model's performance on validation data worsens, thus avoiding overfitting.

6. Data Augmentation

- Increases training data by making slight modifications (e.g., rotating images, adding noise).
- Helps model generalize better by exposing more data variability.

Why Use Regularization?

- Reduces model complexity.
- Improves generalization and robustness on unseen data.
- Prevents overfitting, especially in high-dimensional data scenarios.

Key Points

- **L1 (Lasso)**: Feature selection, induces sparsity.
- **L2 (Ridge)**: Handles multicollinearity, shrinks coefficients smoothly.
- **Elastic Net**: Suitable for highly correlated features.
- **Dropout/Early Stopping**: Regularization in deep learning/neural networks.
- **Regularization parameter** (λ): Controls the trade-off between fit and simplicity. Tuned using validation set.

F. NAÏVE BAYES ALGORITHM

Naive Bayes is a family of simple yet effective probabilistic classifiers based on Bayes' Theorem, with a strong (naive) assumption of independence between input features. Widely used for text classification, spam detection, sentiment analysis, and more due to its speed and simplicity.

Key Concepts

1. Bayes' Theorem

Fundamental probability rule for updating beliefs based on new evidence:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Where:

• P(A|B) : Posterior probability of class AA given evidence BB

• P(B|A) : Likelihood of evidence given class

• P(A) : Prior probability of class

• P(B) : Probability of evidence (normalizing constant)

Naive Independence Assumption

- Assumes all features are *independent* given the class label:
- For input vector $x_1, x_2, \ldots, x_n$

$P(x_1,\ldots,x_n|C) = \prod_{i=1}^{n} 1nP(xi \mid C)$

3. Model Steps

- **Training:**
  - Estimate prior probabilities: $P(C_k)$
  - Estimate likelihoods: $P(x_i|C_k)$ for all features
- **Prediction:**
  - Compute posterior for each class using Bayes' Theorem
  - Choose the class with the highest posterior probability

$Class^* = arg\ max P(C) \prod_{i=1}^{N} P(xi \mid Ck)$

Types of Naive Bayes

sType	Dataset Type	Feature Distribution	Example Use
Gaussian	Continuous	Normal/Gaussian	Iris dataset, medical values
Multinomial	Discrete/count	Multinomial (counts, frequencies)	Text classification, NLP
Bernoulli	Binary	Bernoulli (0/1 presence)	Binary attributes, spam filtering

GeeksforGeeks

GATE CSE BATCH

KEY HIGHLIGHTS:

- 300+ HOURS OF RECORDED CONTENT
- 900+ HOURS OF LIVE CONTENT
- SKILL ASSESSMENT CONTESTS
- 6 MONTHS OF 24/7 ONE-ON-ONE AI DOUBT ASSISTANCE
- SUPPORTING NOTES/DOCUMENTATION AND DPPS FOR EVERY LECTURE

COURSE COVERAGE:

- ENGINEERING MATHEMATICS
- GENERAL APTITUDE
- DISCRETE MATHEMATICS
- DIGITAL LOGIC
- COMPUTER ORGANIZATION AND ARCHITECTURE
- C PROGRAMMING
- DATA STRUCTURES
- ALGORITHMS
- THEORY OF COMPUTATION
- COMPILER DESIGN
- OPERATING SYSTEM
- DATABASE MANAGEMENT SYSTEM
- COMPUTER NETWORKS

LEARNING BENEFIT:

- GUIDANCE FROM EXPERT MENTORS
- COMPREHENSIVE GATE SYLLABUS COVERAGE
- EXCLUSIVE ACCESS TO E-STUDY MATERIALS
- ONLINE DOUBT-SOLVING WITH AI
- QUIZZES, DPPS AND PREVIOUS YEAR QUESTIONS SOLUTIONS

ENROLL NOW

TO EXCEL IN GATE AND ACHIEVE YOUR DREAM IIT OR PSU!

ENROLL NOW

1. Gaussian Naive Bayes

- Models each continuous feature as a Gaussian (normal) distribution per class.

$$P(x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x_i - \mu_k)^2}{2\sigma_k^2}\right)$$

2. Multinomial Naive Bayes

- Commonly used for document classification with word frequency.
- Likelihood is based on the relative frequency of feature value in a class.

3. Bernoulli Naive Bayes

- Used for binary/boolean features (e.g., word presence in a document).

Advantages

- **Simple, fast and efficient** even with large datasets.
- **Performs surprisingly well** for many real-world problems, especially with strong independence among features.
- **Requires less training data** compared to more complex models.
- Robust to irrelevant features.

Limitations

- Performance drops if independence assumption is violated.
- Poor estimation for zero-frequency features (seen in Multinomial/Bernoulli NB) — resolved by **Laplace smoothing**.
- Not ideal for learning complex relationships among features.

Laplace Smoothing

- Used to handle zero probabilities for features not seen in the training data.
- Formula for Laplace-smoothed likelihood: $P(x_i|C_k) = \frac{n_{ik}+1}{N_k+d}$
  - n_{ik} : count of feature x_i in class C_k
  - N_k : total number of features in class C_k
  - d: number of unique features

Applications

- Email / SMS spam detection
- Sentiment and topic classification
- Medical diagnosis
- Document categorization

G. KNN ALGORITHM

K-Nearest Neighbors (KNN) is a simple, non-parametric, lazy learning algorithm used for classification and regression. It predicts the class (or value) of a new data point based on the majority class (or mean value) among its 'k' nearest neighbors in the feature space.

Key Concepts

1. Non-parametric Nature

No explicit model training: KNN stores the entire dataset and makes predictions based on proximity. Learning is instance-based:
Classification/regression happens at prediction time, not during training.

2. Intuition

Given a query point, KNN finds the 'k' closest data points (neighbors) using a distance metric. Classification: The majority class among neighbors is assigned. Regression: The average (or weighted average) of neighbors' target values is assigned.

3. Distance Metrics

- **Euclidean Distance:**
  For points x = (x_1, ..., x_n)
- $d(x, y)=\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
- **Manhattan Distance:**
  $d(x, y) = |x_i - y_i|$
- **Minkowski, Hamming (for categorical):**
  Other specialized metrics as needed.

4. Steps in KNN Algorithm

1. **Choose 'k'** (the number of neighbors, e.g., 3, 5).
2. **Compute distance** between the query point and all training examples.
3. **Sort distances** and select the 'k' nearest neighbors.
4. **Predict the label:**

- **Classification:** Most frequent category among the neighbors.
- **Regression:** Mean/median of neighbor values.

5. Choosing 'k'

- Small 'k': Model is sensitive to noise (risk of overfitting).
- Large 'k': Model is more generalized, less affected by noise (risk of underfitting).
- Odd value of 'k' is preferred for binary classification to avoid ties.

Advantages

- Simple to implement and understand.
- No training phase—fast adaptation to new data.
- Effective for multi-class problems.

Limitations

- Computationally expensive when predicting on large datasets (must compute distance to all points).
- Sensitive to irrelevant features and feature scales (feature scaling is necessary).
- Curse of dimensionality: Performance drops as feature space dimensions increase.
- Imbalanced data: Can bias toward majority class.

Practical Considerations

- Features should be normalized/scaled before applying KNN.
- Weighted KNN: Closer neighbors can be given higher weight in prediction.
- Distance functions and 'k' should be selected/tuned via validation.

Applications

- Handwriting/face/image recognition
- Recommendation systems
- Credit scoring
- Medical diagnosis

11. NEURAL NETWORKS

- **Neural Networks** are computational models inspired by the structure of the human brain, made up of interconnected layers of nodes (neurons).
- Used for tasks such as classification, regression, pattern recognition, image processing, and more.

Basic Structure

- **Input Layer:** Receives raw features.
- **Hidden Layer(s):** Intermediate processing units; can be one or many (deep networks have >1).
- **Output Layer:** Produces predictions—e.g., class labels (classification) or numeric value (regression).
- **Weights & Biases:** Parameters that adapt during training to minimize error.

Forward Propagation

- Data is passed from the input layer through each hidden layer to the output, with each neuron applying an activation function (commonly sigmoid, tanh, ReLU).
- Each neuron's output:
  $y = \sum_j w_j x_j + b$

Loss Function

- **Loss function** quantifies the difference between predicted output and true target.
- Common loss functions:
  - **Mean Squared Error (MSE):** Used for regression
    $L = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$
  - **Cross-Entropy Loss:** Used for classification
- The goal is to minimize this loss during training.

Backpropagation

- **Backpropagation** is the process by which neural networks update weights to minimize loss.
- Steps:
  a. Compute output using forward pass.
  b. Calculate loss.
  c. Compute gradients of the loss with respect to each weight using the chain rule.
  d. Update weights in the direction that reduces loss.
- It efficiently distributes gradient computations, enabling learning in deep architectures.

Gradient Descent Optimization

- **Gradient Descent** is the optimization algorithm used to minimize loss by updating parameters in the opposite direction of the gradient.
- **Update rule for weights:**
  $w_{new} = w_{old} - ń \frac{\partial L}{\partial W}$
  - ń : Learning rate (step size)\
- **Variants:**
  - **Batch Gradient Descent:** Uses entire dataset for each step.
  - **Stochastic Gradient Descent (SGD):** Updates weights for each training example.
  - **Mini-batch Gradient Descent:** A compromise, updating weights for small random data subsets.
  - **Momentum, RMSProp, Adam:** Advanced optimizers that speed up and stabilize convergence.

Convolutional Neural Networks (CNN)

Grayscale Image

- **Definition:** An image composed of various shades of gray, ranging from black to white, with no color information.
- **Pixel Data:** Each pixel contains a single intensity value, typically between 0 (black) and 255 (white) in 8-bit images.
- **Channels:** Contains only **one channel** (intensity/luminance).
- **Memory Requirement:** Requires less storage than color images; ideal for tasks like document scanning, edge detection, or medical imaging.
- **Use Cases:** Handwriting recognition, X-ray analysis, thresholding operations, and preprocessing for many image processing tasks.
- **Visualization:** Displayed in shades of gray, making it easy to analyze brightness and contrast without color interference.

RGB Image

- **Definition:** A color image where each pixel is represented by three components corresponding to the Red, Green, and Blue color channels.
- **Pixel Data:** Each pixel contains three intensity values (R, G, B), each typically ranging from 0 to 255 for 8-bit images.
- **Channels:** Contains **three channels** (Red, Green, Blue).
- **Memory Requirement:** Requires three times more memory than a grayscale image of the same size.
- **Use Cases:** Natural scene photography, object detection, facial recognition, and general computer vision applications where color information is important.
- **Visualization:** Can represent millions of colors by mixing R, G, and B intensities.

Comparison Table

Aspect	Grayscale Image	RGB Image
Number of channels	1	3
Memory usage	Lower	Higher
Color information	No (just intensity)	Yes (full color spectrum)
Example pixel value	127 (gray)	(127, 190, 60) (R,G,B components)
Typical file formats	.pgm, .bmp (gray only)	.jpg, .png, .bmp (color)
Common applications	Medical, OCR, edge detection	Photography, vision, graphics

- **CNNs** are specialized neural networks designed for processing data with grid-like topology (such as images).

- **Key Layers:**
  - **Convolutional Layer:** Applies filters/kernels to extract local features.
  - **Pooling Layer:** Reduces spatial dimensionality (e.g., max pooling).
  - **Fully Connected Layer:** Flattens and connects to activity neurons for final output.

- **Applications:** Image classification, object detection, and natural language processing.
- **Advantages:** Local connectivity, parameter sharing, translation invariance.

12. Algorithms (UNSUPERVISED LEARNING)

Clustering

- **Clustering** is an unsupervised learning technique that automatically groups unlabeled data based on similarity.
- Goal: Group similar data points into *clusters* such that intra-cluster similarity is high and inter-cluster similarity is low

Key Principles

- No predefined labels or categories.
- Based on the similarity/distance between data points (Euclidean, Manhattan, etc.).
- Widely used in pattern recognition, market segmentation, anomaly detection, and data exploration.

Main Types of Clustering Algorithms

Algorithm Type	Main Examples	Description / Use
Partition-based/ Centroid	K-means, K-medoids	Assign data to *k* clusters; each with a centroid.
Hierarchical	Agglomerative, Divisive	Builds nested clusters (tree/dendrogram); doesn't need pre-set k.
Density-based	DBSCAN, OPTICS	Finds clusters based on dense regions; handles shape, noise/outliers
Model/Distribution-based	Gaussian Mixture Models	Assumes data from a mix of probability distributions
Fuzzy/Overlapping	Fuzzy K-means	Data points can belong to multiple clusters with membership scores.

1. K-Means Clustering (Partition-based)

- Given number *k*, algorithm assigns every data point to one of *k* clusters by minimizing within-cluster variance.
- **Steps**:
  - Choose initial centroids (randomly or with heuristics).
  - Assign points to nearest centroid.
  - Update centroids as means of assigned points.

    - Repeat until convergence (centroids stable).
- Simple, efficient, but requires pre-selecting *k* and can get stuck in local minima.
- **Applications**: Market segmentation, document classification, image compression.

2. Hierarchical Clustering

- Forms a tree of clusters (dendrogram).
- **Agglomerative**: Bottom-up, each point starts as its cluster, clusters are merged iteratively based on similarity.
- **Divisive**: Top-down, start with one cluster, recursively split.
- Doesn't require pre-specifying number of clusters; can get clusters at different granularity by "cutting" the dendrogram.
- **Linkages**:
    - Single (closest points), Complete (furthest points), Average, Ward's (minimize variance).
- **Advantages**: No need for *k*; dendrogram aids interpretation.
- **Drawbacks**: Computationally intensive for large datasets.

3. Density-Based Clustering: DBSCAN

- Finds clusters as regions of high point density, separates areas by low-density "noise".
- Handles arbitrarily shaped clusters and outliers well.
- Requires specification of two parameters: minimum points per cluster (*minPts*) and neighborhood radius (*epsilon*).
- No need to choose *k*; sometimes tricky to set parameters.
- **Applications**: Spatial data, image analysis, anomaly detection.

4. Distribution-Based: Gaussian Mixture Model (GMM)

- Assumes data is generated from a mixture of several Gaussian (normal) distributions[2][6].
- Uses Expectation-Maximization (EM) algorithm to estimate parameters.
- Points assigned to clusters via probability (soft assignment).
- Suitable when clusters have elliptical or complex shapes.

5. Fuzzy Clustering

- Allows overlap: each data point has a membership strength for each cluster[2][6].
- Fuzzy K-means is the classic algorithm.
- Helpful where data groupings are ambiguous or overlapping.

Steps in a General Clustering Process

1. Preprocessing (scaling, normalization, handling missing data).
2. Choose the clustering algorithm (based on data properties, desired output).
3. Fit the model to data and cluster.
4. Validation/Evaluation (e.g., silhouette score, Davies-Bouldin index, visual checks).
5. Interpretation and use (labeling, further analysis, real-world application).

Common Applications

- Customer Segmentation
- Image Segmentation
- Market Basket Analysis
- Document & Text Clustering
- Anomaly & Outlier Detection

Performance Matrix for Clustering

- Performance of clustering algorithms is assessed using internal and external validation metrics, since "ground truth" labels often do not exist in unsupervised data.

Common Performance (Validation) Metrics

Metric Name	Type	What it Measures	Range / Best Value
Silhouette Score	Internal	Cohesion vs. separation of clusters	–1 to 1; higher is better
Davies–Bouldin Index	Internal	Avg. similarity between each cluster and its most similar one	≥0; lower is better
Dunn Index	Internal	Ratio of minimum inter-cluster to maximum intra-cluster distance	≥0; higher is better
Calinski–Harabasz	Internal	Ratio of between- to within-cluster dispersion	≥0; higher is better
Adjusted Rand Index	External	Similarity to ground truth (if known)	–1 to 1; higher is better
Homogeneity, Completeness, V-Measure	External	Agreement with true labels (if available)	0 to 1; higher is better

Example Performance Matrix Table

Clustering Metric	Description	Formula / Criterion	Preferred Value
Silhouette Score	How similar a point is to its own cluster vs. others	S = {b - a}/{max(a, b)}	Closer to +1
Davies–Bouldin Index	Avg. ratio of intra-cluster to inter-cluster distances	Mean similarity to most similar cluster	Lower
Dunn Index	Cluster separation relative to size	The Dunn Index (DI) is defined as the ratio between: minimum inter-cluster distance and maximum intra-cluster distance	Higher
Adjusted Rand Index	Correctness wrt. true cluster labels (if labels known)	Pairwise agreements minus expected by chance	Higher
Inertia (K-means only)	Within-cluster sum-of-squares	Sum of squared distances between each data point and the centroid of its assigned cluster.	Lower

13. What is Cross Validation?

- **Cross validation** is a method to estimate how well a machine learning model will generalize to unseen data.
- It helps detect overfitting and ensures models adapt to real-world data, not just the training set.
- The data is partitioned into subsets (folds) multiple times so every sample is used for both training and validation

Why Use Cross Validation?

- Provides a more **robust estimate** of model performance compared to a single train-test split.
- Maximizes the use of available data for both training and testing.
- Helps with **model selection** and **hyperparameter tuning**.
- Prevents overfitting by evaluating the model on unseen portions of data.

Main Cross Validation Techniques

1. Holdout Validation

- Simple split: Typically 70–80% data for training, rest for testing.
- Pros: Fast and easy.
- Cons: High variance as performance depends strongly on how data is split; not all data used for training.

2. K-Fold Cross Validation

- Dataset divided into **k** equal folds (common k = 5 or 10).
- Loop: train on k–1 folds, test on 1 fold; repeat k times so every fold is used for testing exactly once.
- Final score is the average across all folds.
- Pros: Each observation gets to be in a test set once, reduces variance.
- Cons: Slightly higher computation time

Step	Task
1	Partition data into k folds
2	Train on k–1, test on 1
3	Repeat for all folds
4	Average results

3. Stratified K-Fold Cross Validation

- Variant of k-fold for classification, preserves class proportions in each fold (especially useful for imbalanced datasets).
- Ensures every fold is representative of the full class distribution.

4. Leave-One-Out Cross Validation (LOOCV)

- Special case: k = n (n = number of data points).
- Each sample is left out once as the test set.
- Maximizes data for training, but is **computationally intensive**.

5. Repeated K-Fold Cross Validation

- K-fold cross validation is repeated multiple times with different splits.
- Results are averaged for an even more robust performance measure.
- Reduces sensitivity to how data is split

6. Leave-P-Out Cross Validation

- Like LOOCV, but leaves p samples out for testing each time.
- Rarely used for large datasets as combinations grow rapidly.

7. Time Series (Rolling/Sliding Window) Cross Validation

- Used for time-dependent data.
- Train on a window of 'past' data, validate on the 'future' window, sliding the window forward

Practical Applications.

- **Model selection:** Identify model that performs best on average across all validation folds.
- **Hyperparameter tuning:** Use cross validation in grid/random search to select best parameters.
- **Bias-variance tradeoff:** Number of folds (k) affects this tradeoff; fewer folds can increase bias, more folds can increase variance.

Advantages and Disadvantages

Advantage	Disadvantage
Robust performance estimation	Computationally expensive for high k/LOOCV
Reduces overfitting risk	Can be time-consuming
Data efficient (all data is used)	Care needed with time series or grouped data

Summary Table

Technique	Key Features	Typical Use Cases
Holdout	Single split	Quick checks, large data
K-Fold	k splits, rotate test/train folds	Most general tasks
Stratified K-Fold	k folds, preserves class balance	Imbalanced classification
LOOCV	n folds, each point tested once	Small data/ bias minimization
Repeated K-Fold	Multiple runs of k-fold with resampling	Robust estimate (small data)
Time Series	Rolling window approach	Time-dependent data

GeeksforGeeks

GATE CSE BATCH

KEY HIGHLIGHTS:

- 300+ HOURS OF RECORDED CONTENT
- 900+ HOURS OF LIVE CONTENT
- SKILL ASSESSMENT CONTESTS
- 6 MONTHS OF 24/7 ONE-ON-ONE AI DOUBT ASSISTANCE
- SUPPORTING NOTES/DOCUMENTATION AND DPPS FOR EVERY LECTURE

COURSE COVERAGE:

- ENGINEERING MATHEMATICS
- GENERAL APTITUDE
- DISCRETE MATHEMATICS
- DIGITAL LOGIC
- COMPUTER ORGANIZATION AND ARCHITECTURE
- C PROGRAMMING
- DATA STRUCTURES
- ALGORITHMS
- THEORY OF COMPUTATION
- COMPILER DESIGN
- OPERATING SYSTEM
- DATABASE MANAGEMENT SYSTEM
- COMPUTER NETWORKS

LEARNING BENEFIT:

- GUIDANCE FROM EXPERT MENTORS
- COMPREHENSIVE GATE SYLLABUS COVERAGE
- EXCLUSIVE ACCESS TO E-STUDY MATERIALS
- ONLINE DOUBT-SOLVING WITH AI
- QUIZZES, DPPS AND PREVIOUS YEAR QUESTIONS SOLUTIONS

ENROLL NOW

TO EXCEL IN GATE AND ACHIEVE YOUR DREAM IIT OR PSU!

ENROLL NOW