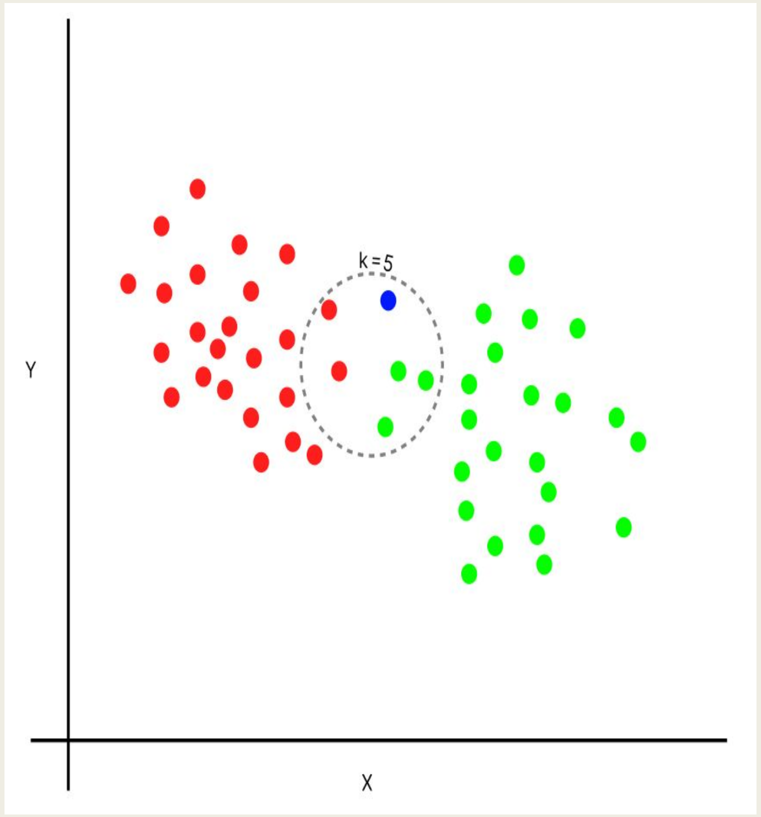
MACHINE LEARNING ALGORITHMS

K-NEAREST NEIGHBOURS,RANDOM FOREST,NAIVE BAYES,GRADIENT BOOSTING

K NEAREST NEIGHBOURS

- **KNN** is a **non-parametric, instance-based** machine learning algorithm that performs classification or regression by finding the k closest neighbors by averaging(in case of regression) or voting (in case of classification).
- **Core Principles:**
  - **Similarity drives prediction** i.e the points closer together are seen as alike.
  - **No training phase**: The algorithm stores the dataset and all computations happened during prediction time.
  - **Flexible distance metrics** :Euclidean, Manhattan, Cosine, or learned metrics can be used.
- **Key Properties:**
  - **Non-parametric:** Doesn’t assume any fixed data pattern.
  - **Lazy learning:** Stores data and computes only at prediction time.
  - **Versatility:** Can be used for both classification and regression.
  - **Local decision-making:** Looks only at nearby points to decide.
- E-commerce platforms like **Amazon** use KNN in their recommendation systems , when you view a product, the system finds similar products (nearest neighbors) based on features like category, price range, and user ratings, then suggests “Customers who bought this also bought…”.



- **KNN Workflow:**
  - **Store Training Data**:Stores the entire dataset in memory , no parameters are learned or no computations are carried out in this phase.
  - **Choose k**: Number of nearest neighbors to consider.
  - **Select Distance Metric:**
    - Euclidean : used for continuous features.
    - Manhattan : used for grid-like data.
    - Cosine similarity : used for text features.
  - **Find k Nearest Neighbors:** Compute distance from the query point to all training points and select the closest k points.
  - **Predict**:
    - Classification: Majority voting .
    - Regression: Mean of neighbors values.

- ## Impact of k:
  - **Small k (e.g., 1–3):**
    - **Pros:** Captures local patterns and more flexible.
    - **Cons:** Sensitive to noise and outliers.
  - **Large k:**
    - **Pros:** More stable predictions and less variance.
    - **Cons:** May over smooth boundaries and miss local structure.

- **Choosing k:**
  - **Rule of thumb:** Start with square root of n (n = training samples).
  - **Cross-validation:** Try multiple k values (1–20 or more) and pick the one with the highest validation score.
  - **Odd k for classification:** To avoid ties in binary classification.
- **Why Scaling is Important:**
  - Distance plays a huge role in this algorithm and if scaling is not applied, features having larger values will dominate.
  - **Solution:** Standardize (z-score) or normalize (min-max) all features before KNN.
- **Distance Metric Selection**

METRIC	WHEN TO USE
Euclidean	Continuous, numeric features, similar scale.
Manhattan	Sparse features, high dimensionality, grid-like data.
Cosine	Text embeddings, high-dim vectors where magnitude isn’t important.
Mahalanobis	Features with correlation, accounts for covariance structure.

- **Tie-Breaking Strategies in KNN**
  - **Closest Neighbour Wins**:Prediction is the label of the nearest point
  - **Distance-Weighted Voting**:Closer points have more weight.
  - **Reduce k**:Reduce the number of neighbors to make the model more sensitive.

Performance Evaluation

- **Regression Metrics**
  - **MSE**: Penalizes large errors more.
  - **RMSE**: MSE in same units as target.
  - **MAE**: Less sensitive to outliers than MSE.
  - **R^2**: Variance explained by the model.
- **Classification Metrics**
  - Accuracy: Tells overall correctness but can be misled if class imbalance is present.

    $$\frac{TP + TN}{TP + TN + FP + FN}$$

  - Precision: Tells us out of all the predicted positives ,which were truly positive.

    $$\frac{TP}{TP + FP}$$

  - Recall: Tells us out of actual positives , how many the model predicted correct.

    $$\frac{TP}{TP + FN}$$

  - F1: Harmonic mean of precision and recall.

PROS AND CONS

- **PROS**
  - Simple to understand and implement.
  - No training phase, just store the data.
  - Works for both classification and regression.
  - Adapts well to complex decision boundaries.
- **CONS**
  - Slow predictions for large datasets.
  - Sensitive to noisy data and outliers.
  - Performance drops in high-dimensional data.
  - Needs careful choice of k and distance metric.

Use Cases & Why KNN Works

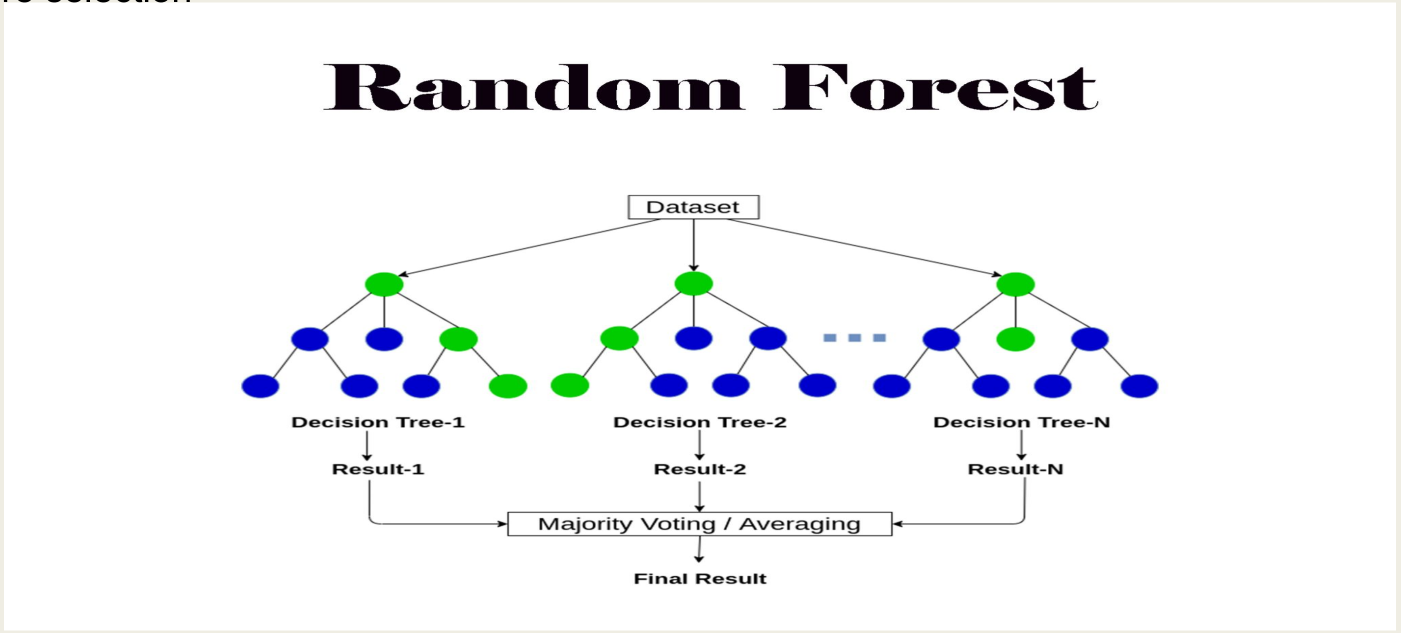
USE CASE	WHY KNN WAS CHOSEN
Medical Diagnosis	1. Finds similar patient cases using multiple health metrics. 2. Easy for doctors to interpret results ("patients like you had this outcome").
Recommendation Systems	1. Quickly identifies similar users/items using feature similarity (ratings, categories, embeddings). 2. Works well when patterns are local and non-linear.
Anomaly Detection	1. Outliers have few or no close neighbors, making them easy to detect. 2. No assumption about the shape of normal data distribution.

RANDOM FOREST

- Random Forest is an ensemble learning technique that combines the outputs of multiple decision trees.Each decision tree is trained on a random subset of the actual data,along with subset of features.
- A single decision tree might lead to overfitting and provide unstable results.
- Random Forest reduces this instability by building many independent trees and averaging or voting their predictions.
- In simple terms, we can say that Diversity makes better decisions.
- Characteristics:
  - Works for both classification and regression tasks
  - Uses randomness in data sampling and feature selection
  - No assumption about data distribution
  - Robust to noisy data and irrelevant features

EXAMPLE:Imagine asking 100 different friends for advice.

Some may be wrong, but if most agree on an answer, it’s probably the right one.



- **STEP-BY-STEP PROCESS**
  - **Bootstrap Sampling:**
    In this step , we create random subsets of the original dataset with replacement, i.e.some rows many appear more than once and some may not appear at all.
  - **Grow a Decision Tree:**
    In this step, independent decision trees are constructed and trained on the individual bootstrapped samples. At every node while splitting , it will select a subset of the original feature set.
  - **Repeat:**
    Train a number of trees.
  - **Aggregate Predictions:**
    Classification: Majority vote from all trees.
    Regression: Average predictions from all trees.
- **Random Forest help in reducing the variance and predicting more accurate and stable outputs. How?**
  - **Diversity :** Training individual trees on bootstrapped random data subsets and subset of features for splitting, helps reduce the correlation between the trees , thus not making the same mistakes.
  - **Averaging/Voting:** This further reduces variance , by taking into consideration all the outputs rather than looking at one given decision tree output.
- If the individual decision trees are **underfitting** (high bias), adding more trees will not improve performance. This is because bias comes from the model’s **wrong assumptions or oversimplification**, and if every tree makes the same wrong assumptions, averaging them just repeats the same mistake instead of fixing it.
- Random Forest measures feature importance by tracking how much each feature **reduces impurity** (like Gini or entropy) across all splits in all trees, and averaging this reduction.

Parameter	Role	Effect
n_estimators	Number of trees	● More trees will give more stable predictions ● Adding more trees might also increase training time.
max_depth	Tree depth	● Trees that are too deep tend to overfit. ● Trees that are too shallow tend to underfit.
max_features	Features per split	● A lower value will provide more generalized predictions. ● A higher value might lead to correlation among the decision trees and overfit.
min_samples_split	Min samples to split a node	● A lower value might lead to deeper more complex trees. ● A higher value leads to less complex trees.
bootstrap	Use bagging or not	● Setting value to true, the algorithm uses bagging and results in more diverse trees, whereas false would lead to pasting.
oob_score	Out-of-bag validation	● Estimates accuracy without separate test set. ● Saves time during evaluation.

PROS AND CONS

- **PROS**
  - Handles complex, non-linear data without requiring much transformations.
  - Can be used for both classification and regression.
  - Less sensitive to missing values and noisy data.
  - Provides feature importance, which further helps us to determine the strong performing features.

- **CONS**
  - Less interpretable than individual decision trees.
  - If the number of estimators is large ,it will require higher memory.
  - Does not fix high bias.
  - Computationally heavy and slower predictions.

Performance Evaluation

- **Regression Metrics**
  - **MSE**: Penalizes large errors more.
  - **RMSE**: MSE in same units as target.
  - **MAE**: Less sensitive to outliers than MSE.
  - **R^2**: Variance explained by the model.
- **Classification Metrics**
  - Accuracy: Tells overall correctness but can be misled if class imbalance is present.

    $$\frac{TP + TN}{TP + TN + FP + FN}$$

  - Precision: Tells us out of all the predicted positives ,which were truly positive

    $$\frac{TP}{TP + FP}$$

  - Recall: Tells us out of actual positives , how many the model predicted correct.

    $$\frac{TP}{TP + FN}$$

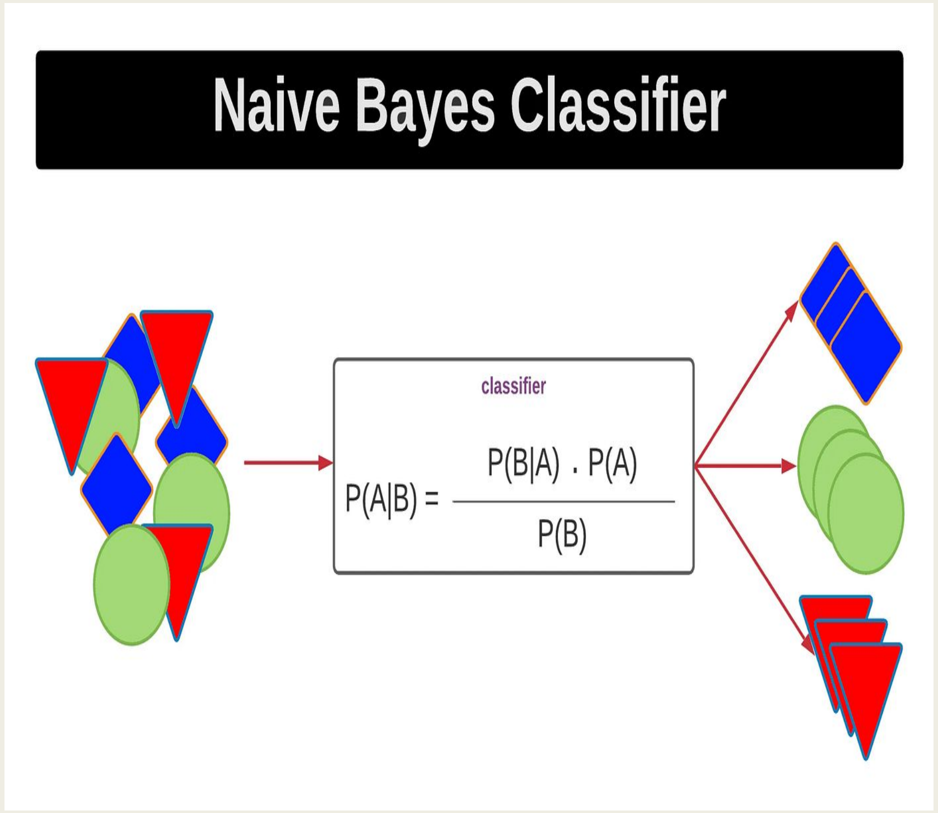
  - F1: Harmonic mean of precision and recall.
  - Confusion Matrix : Table showing TP, FP, TN, FN counts for detailed error analysis.

Use Cases & Why Random Forest Works

USE CASE	WHY RANDOM FOREST WAS CHOSEN
Customer Churn Prediction	1. Handles large datasets with many features without heavy preprocessing. 2. Captures complex, non-linear relationships between customer actions and churn risk.
Credit Risk Scoring	1. Robust to noisy data and missing records. 2.Provides feature importance to explain risk factors to compliance teams.
Medical Diagnosis Support	1.Can handle mixed data types (categorical symptoms and numerical test results). 2. Reduces overfitting compared to single decision trees, improving generalization on unseen patients.

NAIVE BAYES

- Naive Bayes is a machine learning algorithm based on Bayes Theorem , it has a strong assumption that all the features are independent given the class. It is a family of probabilistic algorithms.
- It is used for classification tasks , specially in cases where the data is high dimensional like text data.
- **CORE PRINCIPLES:**
  - **Bayes' Theorem**: Uses prior probabilities and new evidence to update the likelihood of a hypothesis being true.
  - **Naive Independence:** Simplifies calculations by assuming all features are independent.
  - **Class Conditional Probabilities**: Estimates the probability of observing certain features if the data belongs to a given class.
- **KEY PROPERTIES:**
  - **Probabilistic model:**Makes predictions by calculating and comparing probabilities for each class.
  - **Fast training**:Requires simple calculations, making it quick to train even on large datasets.
  - **Works well with small datasets**:Can produce reliable results even when training data is limited.



EXAMPLE:

Email services use Naive Bayes in **spam filtering**. The algorithm calculates the probability of the email being spam email based on the features(words and patterns).

Types of Naïve Bayes

- **Gaussian NB:** Used when features are **continuous** and assumed to follow a **normal (Gaussian) distribution**.
- **Multinomial NB:** Best for **count-based features** like word frequencies in text classification. Assumes features represent discrete counts.
- **Bernoulli NB:**Designed for **binary features** (0 or 1), such as the presence or absence of a word in a document.

WHY DOES NAIVE BAYES WORK WELL?

- **Strong with independence assumption** :Even if features are not perfectly independent, it often works surprisingly well in practice.
- **Low data requirement** :Learns effectively from small datasets.
- **Handles high-dimensional data** : Especially effective in text classification where vocabulary size is large.
- **Fast & efficient** : Requires only counting and probability calculations.

COST FUNCTION:

Naive Bayes minimizes the **Negative Log-Likelihood (NLL)**, which measures how well the predicted probabilities match the true classes.It is derived from Bayes theorem under the conditional independence assumption.

$$\mathrm{NLL} = -\sum_{i=1}^{n} \log P(y_i \mid x_i; \theta)$$

■ STEP-BY-STEP WORKFLOW:

- **Calculate Prior Probabilities:**
  The **prior probability** of each class C_k is

$$P(C_k) = \frac{\text{Number of samples in class } C_k}{\text{Total number of samples}}$$

- **Calculate Likelihoods**
  For each feature x_i, compute the probability of seeing that feature given a class C_k

$$P(x_i \mid C_k) = \frac{\text{Count of feature } x_i \text{ in class } C_k}{\text{Total count of features in class } C_k}$$

- **Apply the Naive Independence Assumption:**
  We assume features are independent given the class

$$P(x_1, x_2, \ldots, x_n \mid C_k) = \prod_{i=1}^{n} P(x_i \mid C_k)$$

- **Apply Bayes Theorem:**
  For each class C_k, compute the **posterior probability**

$$P(C_k \mid x_1, x_2, \ldots, x_n) = \frac{P(C_k) \cdot \prod_{i=1}^{n} P(x_i \mid C_k)}{P(x_1, x_2, \ldots, x_n)}$$

- **Class Prediction:**
  Pick the class with the **maximum posterior**

$$\hat{C} = \arg\max_{C_k} \; P(C_k) \cdot \prod_{i=1}^{n} P(x_i \mid C_k)$$

PROS AND CONS

- **PROS**
  - Extremely fast to train and predict.
  - Works well for high-dimensional data like text.
  - Requires less training data.
  - Provides probabilistic output for interpretability.

- **CONS**
  - Assumption of independence is often unrealistic.
  - Struggles with correlated features.
  - For continuous variables, performance depends on correct distributional assumptions.

Performance Evaluation

- **Classification Metrics**
  - Accuracy: Tells overall correctness but can be misled if class imbalance is present.

$$\frac{TP + TN}{TP + TN + FP + FN}$$

  - Precision: Tells us out of all the predicted positives ,which were truly positive

$$\frac{TP}{TP + FP}$$

  - Recall: Tells us out of actual positives , how many the model predicted correct.

$$\frac{TP}{TP + FN}$$

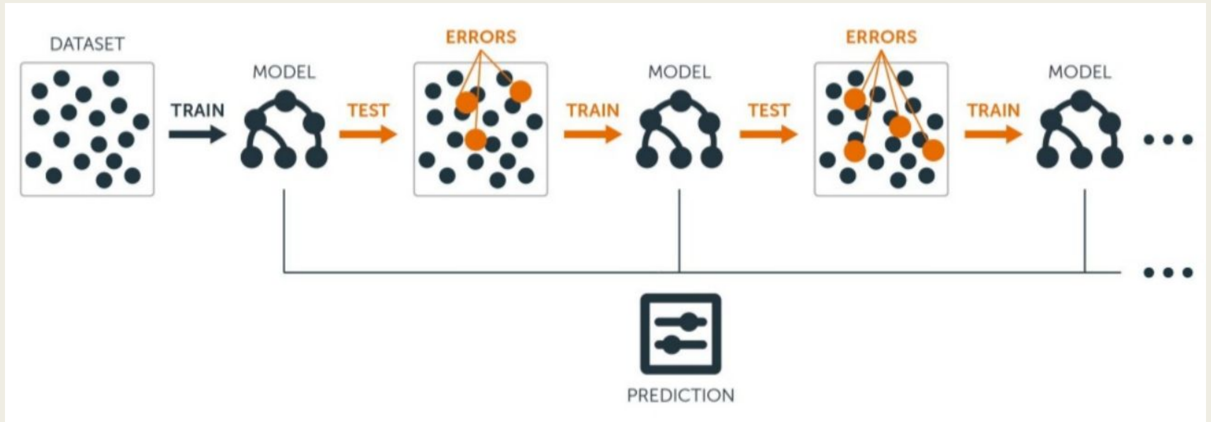
  - F1: Harmonic mean of precision and recall.
  - Confusion Matrix : Table showing TP, FP, TN, FN counts for detailed error analysis.

Use Cases & Why Naive Bayes Works

USE CASE	WHY NAIVE BAYES WAS CHOSEN
Spam Filtering	1. Handles high-dimensional data efficiently. Eg, Text from emails 2. Probabilistic output helps rank emails by spam likelihood.
Sentiment Analysis	1. Works well with sparse text features. 2. Fast enough for real-time classification.
Document Categorization	1. Scales to large numbers of categories. 2. Simple, interpretable probability-based predictions.

GRADIENT BOOSTING

- Gradient Boosting is an ensemble learning technique that is sequential and sequentially adds weak learners to form one strong model. Each succeeding learner tries to correct the error of its previous weak learner.
- The gradient from Gradient Boosting refers to using gradient descent to improve the previous errors.
- Each added learner tries to minimize the loss function by correcting the previously caused errors.
- **CORE PRINCIPLES:**
  - **Boosting**:Models are trained sequentially where each added model tries correcting the errors caused by the previous combined models.
  - **Gradient Descent**:Uses the gradient of the loss function to decide which direction to move to in order to minimize the error.
- **KEY PROPERTIES:**
  - **Versatile**:Can handle both regression and classification tasks.
  - **Flexible with features**:Can handles both categorical and numerical variables.
  - **Non-parametric**: No strict assumptions about the underlying data distribution.



- **STEP-BY-STEP PROCESS**
  - **Start simple:**
    We can start with an easy guess such as for regression we can use the mean whereas for classification we can use basic probability.
  - **Compute Residuals:**
    Calculate the residuals(difference between predictions and actual values) for regression and pseudo-residuals for classification.
  - **Fit Weak Learner:**
    Train a small decision tree that tries to predict just those mistakes.
  - **Update Model:**
    Add the new learner’s predictions to the existing model, scaled by a learning rate.
  - **Keep improving:**
    Repeat the process until you’ve done it enough times or there’s hardly any improvement left.

- **Common Variants of Gradient Boosting**
  - **XGBoost:** A super-fast, high-performance version of Gradient Boosting with extra tricks to avoid overfitting.
  - **LightGBM:** Grows trees in a clever “leaf-first” way, making it quick on big datasets and able to handle categories without extra encoding.
  - **CatBoost:** Great at working with categorical data straight away, and uses a special boosting method to prevent overfitting.

EXAMPLE:

Banks use Gradient Boosting to figure out how risky it is to give someone a loan. It looks at patterns in people’s information and predicts the chances they might not pay back, often doing a better job than simpler models.

PROS AND CONS

- **PROS**
  - High predictive accuracy.
  - Handles complex non-linear relationships.
  - Works well with a mix of features and missing values.
  - Built-in feature importance.
- **CONS**
  - More sensitive to overfitting than Random Forest if not tuned well.
  - Longer training times.
  - Requires careful parameter tuning.

Parameter	Role	Effect
n_estimators	Number of boosting stages	More trees can make the model more accurate but might overfit.
learning_rate	How much each tree affects the final prediction	Smaller values make the model learn slowly and generalize better, but you need more trees.
max_depth	Controls tree complexity	Bigger trees can catch more patterns but may overfit.
subsample	Fraction of data used per tree	Using less than 100% adds randomness and helps the model generalize.
loss	Defines optimization target	For example, squared error for regression or log loss for classification.

Performance Evaluation

- **Regression Metrics**
  - **MSE**: Penalizes large errors more.
  - **RMSE**: MSE in same units as target.
  - **MAE**: Less sensitive to outliers than MSE.
  - **R^2**: Variance explained by the model.
- **Classification Metrics**
  - Accuracy: Tells overall correctness but can be misled if class imbalance is present.

    $$\frac{TP + TN}{TP + TN + FP + FN}$$

  - Precision: Tells us out of all the predicted positives ,which were truly positive

    $$\frac{TP}{TP + FP}$$

  - Recall: Tells us out of actual positives , how many the model predicted correct.

    $$\frac{TP}{TP + FN}$$

  - F1: Harmonic mean of precision and recall.
  - Confusion Matrix : Table showing TP, FP, TN, FN counts for detailed error analysis.

Use Cases & Why Gradient Boosting Works

USE CASE	WHY GRADIENT BOOSTING WAS CHOSEN
Credit Risk Scoring	1. Finds complex patterns between different risk factors. 2. Gives accurate predictions and shows which features matter most.
Customer Churn Prediction	1. Works well with both numbers and categories. 2. Picks up subtle signals in customer behavior.
Search Ranking	1. Learns small but important differences for ranking. 2. Handles many different types of features effectively.