6. Modeling and Validation Processes

Table of content

- Introduction to Machine Learning
- Introduction to Supervised, Unsupervised & Reinforcement Learning.
- Modeling Process, Training /Validating model, Cross Validation methods, Predicting new observations Interpretation
- Measures for Model Performance and Evaluation:

6.1 Introduction to Machine Learning

1 Definition and Goals of Machine Learning

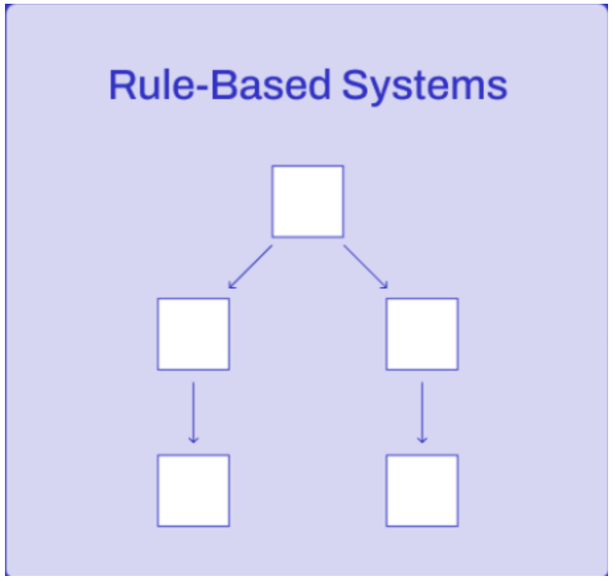
- Machine learning (ML) is a branch of artificial intelligence that enables systems to learn from data without explicit programming.

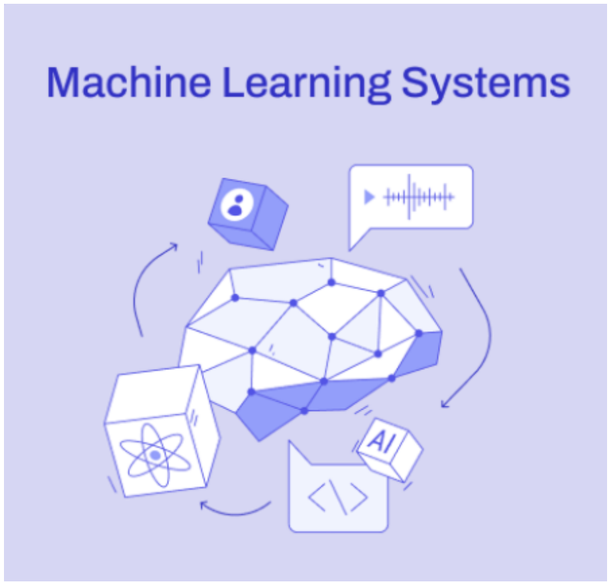
- Goals: Identify patterns, make predictions, automate decision-making, and improve over time.

2 Rule-Based Programming vs. Machine Learning

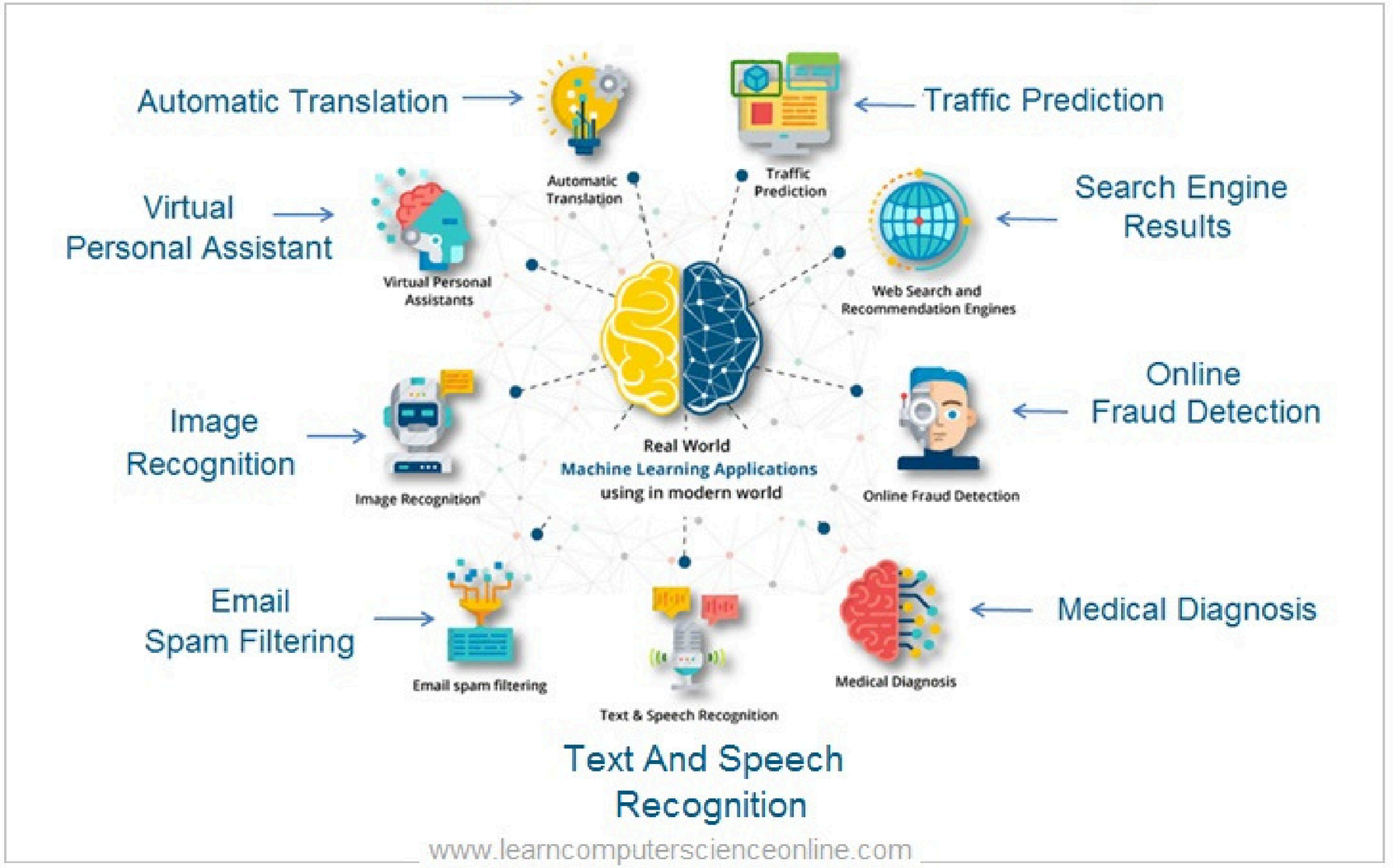
- Rule-Based Programming: Explicitly defined instructions and logic.

- Machine Learning: Learns patterns from data to make decisions.





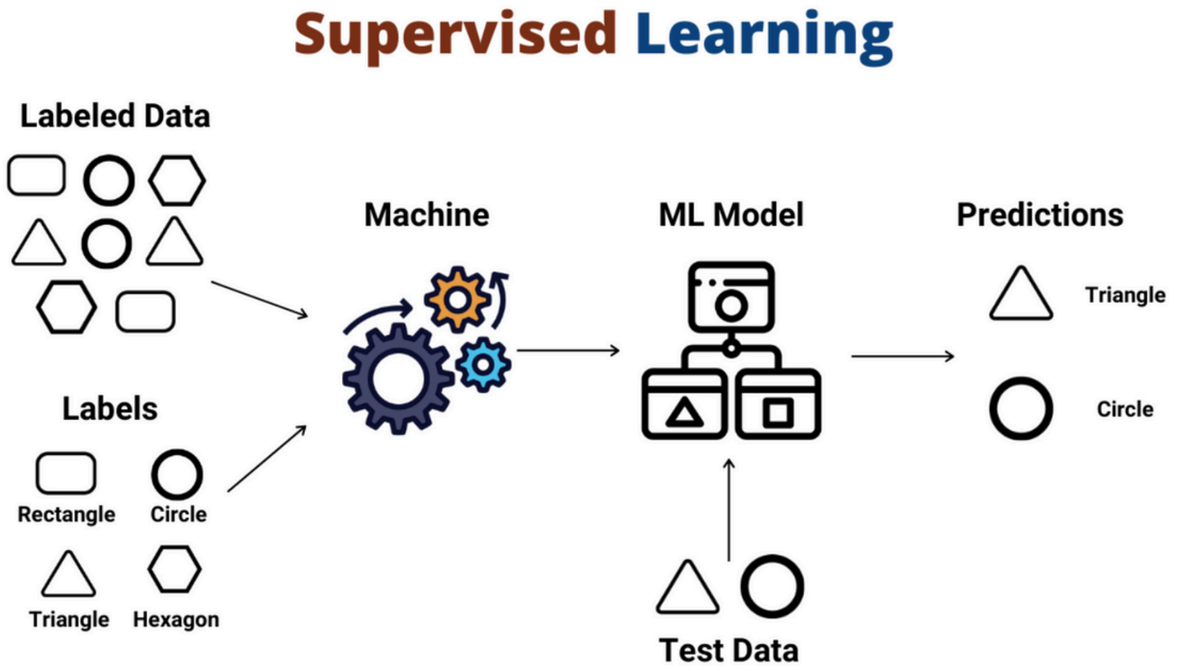
Real World Applications Of Machine Learning



6.2 Introduction to Supervised, Unsupervised & Reinforcement Learning.

1.1 Supervised Learning

Supervised learning is a type of machine learning where a model is trained using labeled data. The algorithm learns from input-output pairs, mapping inputs to desired outputs based on patterns in the training data. The goal is to make accurate predictions or classifications when given new, unseen data.



Applications

- Spam Detection
- Fraud Detection
- Medical Diagnosis
- Speech Recognition
- Image Classification

1.2 Algorithms in Supervised Learning

Algorithm	Regression, Classification	Purpose	Method	Use Cases
Linear Regression	Regression	Predict continuous output values	Linear equation minimizing sum of squares of residuals	Predicting continuous values
Logistic Regression	Classification	Predict binary output variable	Logistic function transforming linear relationship	Binary classification tasks
Decision Trees	Both	Model decisions and outcomes	Tree-like structure with decisions and outcomes	Classification and Regression tasks
Random Forests	Both	Improve classification and regression accuracy	Combining multiple decision trees	Reducing overfitting, improving prediction accuracy
support vector machine(SVM)	Both	Create hyperplane for classification or predict continuous values	Maximizing margin between classes or predicting continuous values	Classification and Regression tasks
k nearest neighbor(KNN)	Both	Predict class or value based on k closest neighbors	Finding k closest neighbors and predicting based on majority or average	Classification and Regression tasks, sensitive to noisy data
Gradient Boosting	Both	Combine weak learners to create strong model	Iteratively correcting errors with new models	Classification and Regression tasks to improve prediction accuracy
Naive Bayes	Classification	Predict class based on feature independence assumption	Bayes' theorem with feature independence assumption	Text classification, spam filtering, sentiment analysis, medical

1.3 Artificial Neural Networks (ANN)

Overview:

Artificial Neural Networks (ANNs) are computing systems inspired by the human brain. They consist of layers of interconnected neurons that process input data to produce outputs.

Architecture:

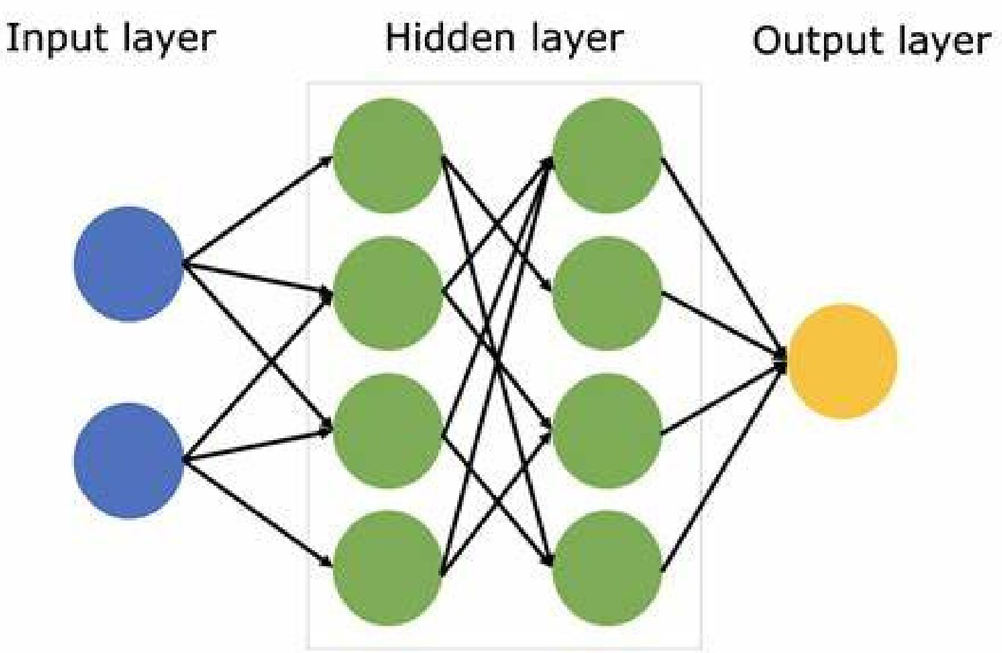
- Input Layer: Receives the input features.
- Hidden Layers: Perform computations using activation functions (ReLU, Sigmoid, Tanh).
- Output Layer: Produces the final prediction (classification or regression).

Training Process:

- Forward propagation to compute predictions.
- Loss function to measure prediction error.
- Backpropagation to adjust weights using optimization algorithms like Gradient Descent.

Applications:

- Image and speech recognition.
- Time-series forecasting.
- Autonomous vehicles.



Artificial neural networks

1.4 Naïve Bayes Classifier

Overview:

Naïve Bayes is a classification algorithm that predicts the category of data based on probabilities. It is based on Bayes' Theorem and assumes that all features (characteristics of data) are independent of each other.

Mathematical Formula (Bayes' Theorem):

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Here,

- A is called the hypothesis.
- B is the evidence.
- P(A) is termed as prior probability. It is the probability of occurrence of the hypothesis.
- P(B) is termed marginal probability. It is the probability of occurrence of the evidence.
- P(B/A) is called likelihood. It is the probability of occurrence of B given that A has already occurred.
- P(A/B) is called posterior probability. It is the probability of occurrence of A given that B has already occurred.

Applications:

- Text classification (e.g., spam detection, sentiment analysis).
- Medical diagnosis.
- Recommender systems.

1.4 Simple example of Naïve Bayes Classifier

We want to find the probability that a card is a King, given that we already know it is a Face Card.

Step 1: Define the Events

- Event A = The card is a Face Card (Jack, Queen, or King).
- Event B = The card is a King.

We need to find P(B | A), which is the probability that a card is a King, given that it is a Face Card.

Step 2: Probabilities

- P(A) = Probability of picking a Face Card = 12/52 (Face Cards: 4 Jacks, 4 Queens, 4 Kings)
- P(B) = Probability of picking a King = 4/52 (Kings in the deck)
- P(A | B) = Probability of a King being a Face Card = 1 (Every King is a Face Card)

Step 3: Use Bayes' Theorem

$$P(B|A) = \frac{P(A|B) \times P(B)}{P(A)}$$

$$= \frac{1 \times (4/52)}{12/52} = \frac{4}{12} = \frac{1}{3}$$

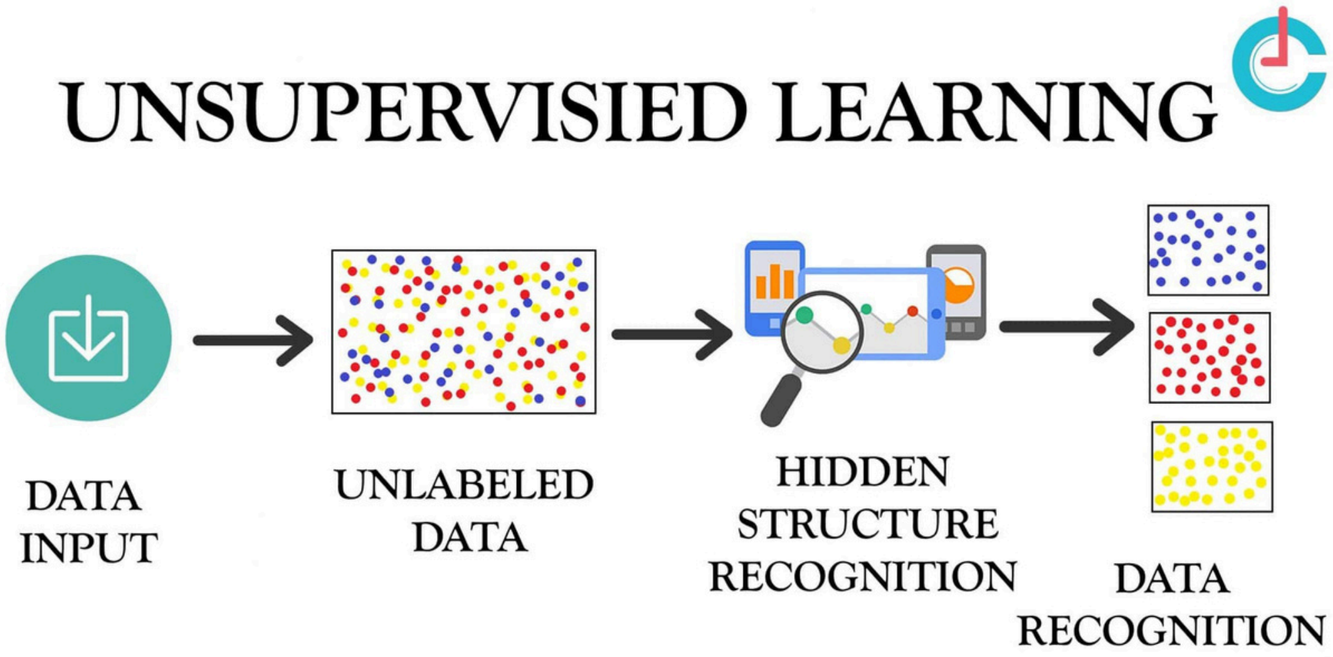
✅ If we already know that a card is a Face Card, the chance of it being a King is 1/3 (or 33.3%).

2.1 Unsupervised Learning

Unsupervised learning involves training a model on unlabeled data, where the algorithm identifies patterns, structures, and relationships within the dataset without explicit supervision.

Applications

- Customer Segmentation
- Anomaly Detection
- Market Basket Analysis
- Recommendation Systems



2.2 Algorithms in Unsupervised Learning

Algorithm	Purpose	Method	Use Cases
K-Means Clustering	Cluster data into groups	Assigns data points to k clusters by minimizing variance	Customer segmentation, anomaly detection
Hierarchical Clustering	Create a hierarchy of clusters	Builds a tree of clusters based on similarity	Gene expression analysis, document classification
DBSCAN	Identify clusters of varying density	Groups points based on density and distance criteria	Noise filtering, geospatial data analysis
Gaussian Mixture Models (GMM)	Model data as a mixture of distributions	Uses probability distributions to model clusters	Speaker identification, fraud detection
Principal Component Analysis (PCA)	Reduce dimensionality while preserving variance	Transforms data into a lower-dimensional space	Feature reduction, image compression, exploratory data analysis
Autoencoders	Learn efficient data representations	Neural network that compresses and reconstructs data	Anomaly detection, feature learning, image denoising
Association Rule Learning	Discover relationships between variables	Identifies frequent patterns, associations, or correlations	Market basket analysis, recommendation systems

2.3 K mean clustering

Overview:

K-Means is a popular unsupervised clustering algorithm used to partition data into K distinct clusters. The algorithm minimizes the variance within clusters and assigns data points to the nearest cluster center (centroid).

Key Steps:

1. **Initialization**: Choose K initial centroids (either randomly or using specific techniques like K-Means++ to improve convergence).
2. **Assignment Step:** Assign each data point to the nearest centroid based on distance (usually Euclidean distance).
3. **Update Step:** Compute the new centroids by calculating the mean of all points assigned to each centroid.
4. **Repeat:** Repeat the assignment and update steps until convergence (centroids no longer change significantly or a set number of iterations is reached).

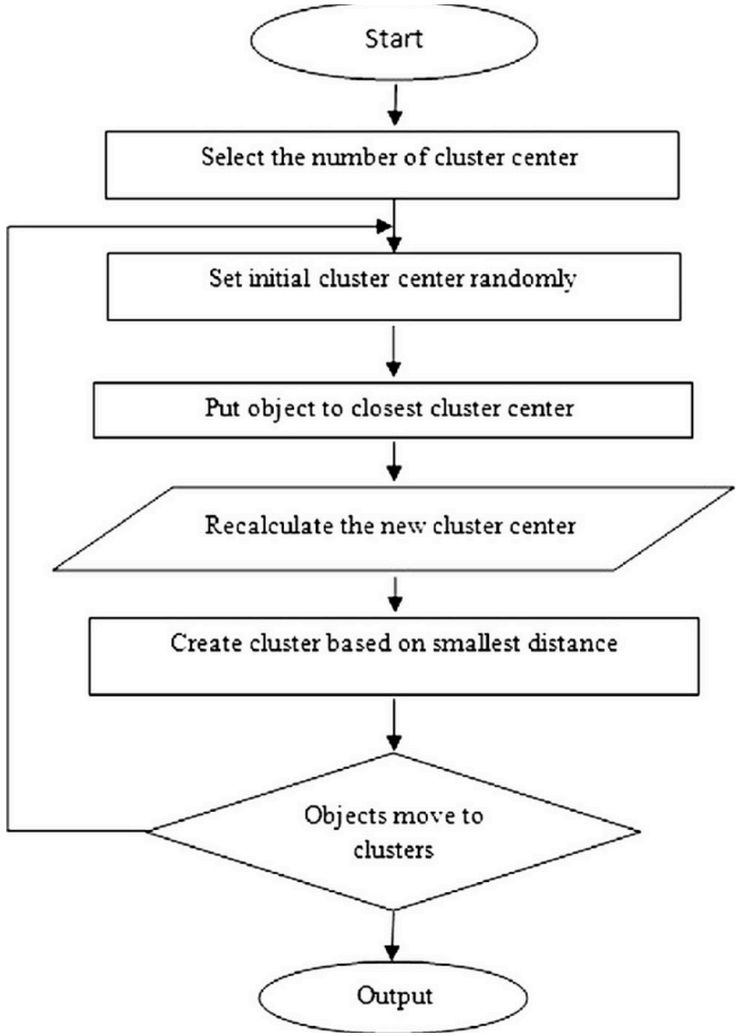
Advantages:

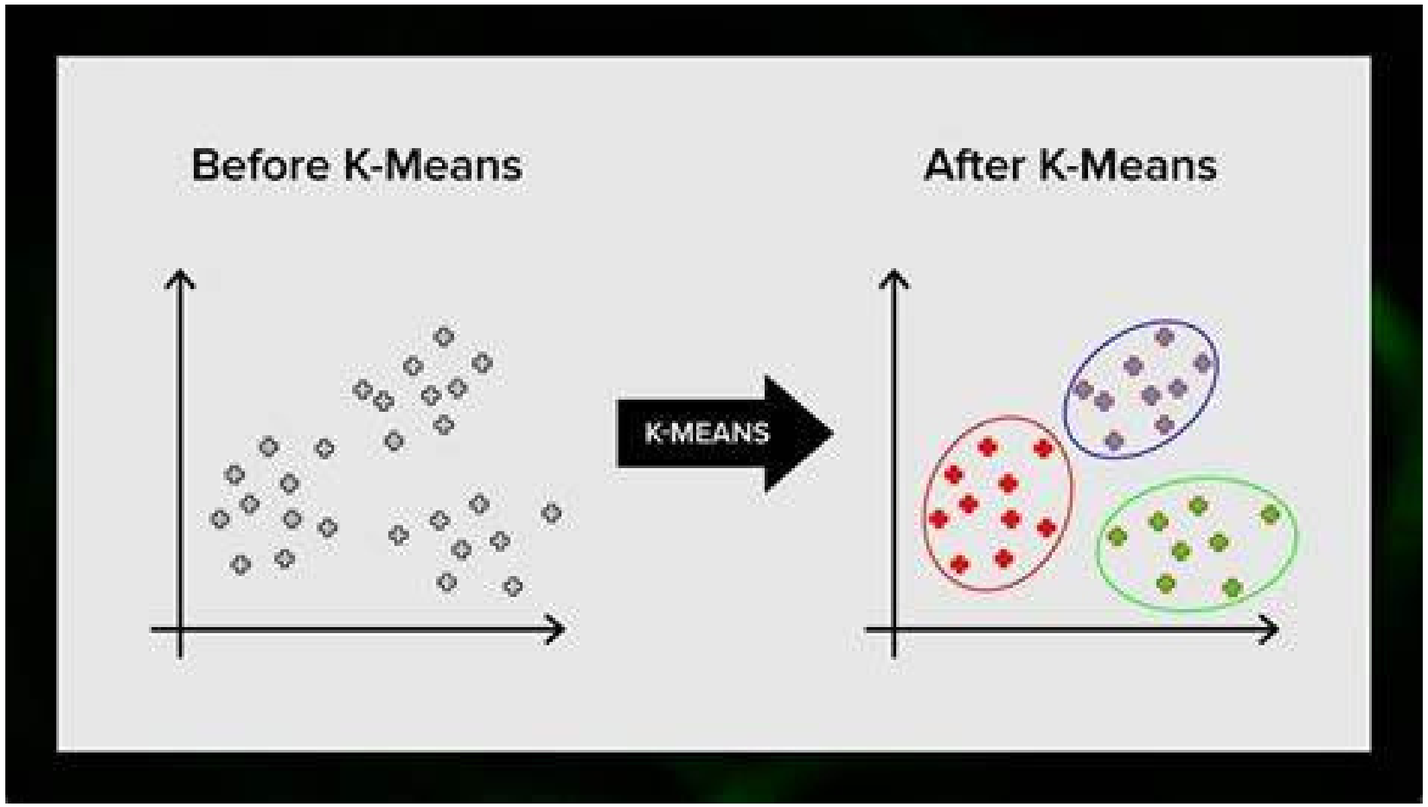
- Simple and fast for small to medium-sized datasets.
- Works well when clusters are roughly spherical and have similar sizes.
- Scalable for large datasets.

Disadvantages:

- Requires the number of clusters (K) to be specified beforehand.
- Sensitive to the initial placement of centroids.
- Struggles with clusters of non-convex shapes or varying sizes.

2.3 K mean clustering





2.4 Density based Spatial clustering of Applications with Noise(DBSCAN)

Overview:

DBSCAN is a density-based clustering algorithm that identifies regions of high point density and separates them from regions of low density. Unlike K-Means, DBSCAN does not require specifying the number of clusters upfront.

Key Steps:

1. **Core Points:** A point is a core point if it has more than a specified number of points (minPts) within a given radius (epsilon, ε).
2. **Border Points:** A point that is not a core point but lies within the epsilon distance of a core point.
3. **Noise Points:** Points that do not satisfy either of the above criteria and are considered noise or outliers.
4. **Clustering:** Clusters are formed from core points that are reachable from each other.
5. **Expand:** Once a core point is found, expand the cluster to include neighboring points (within epsilon distance) and repeat until no more points can be added.

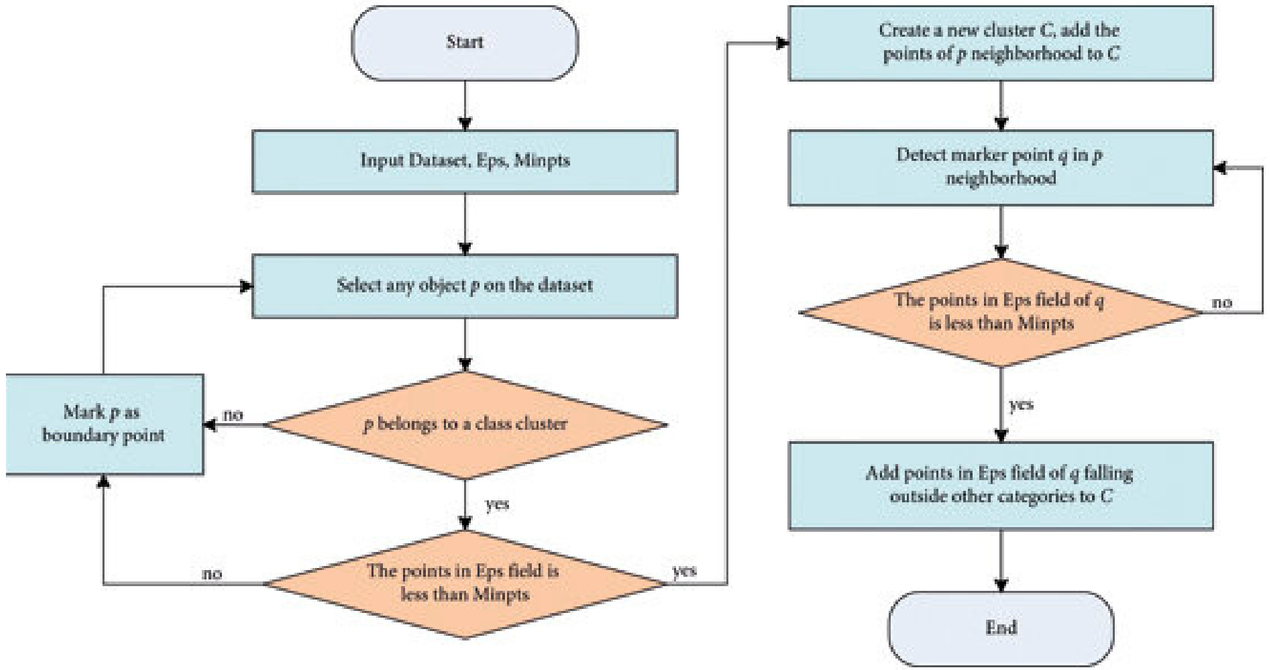
Advantages:

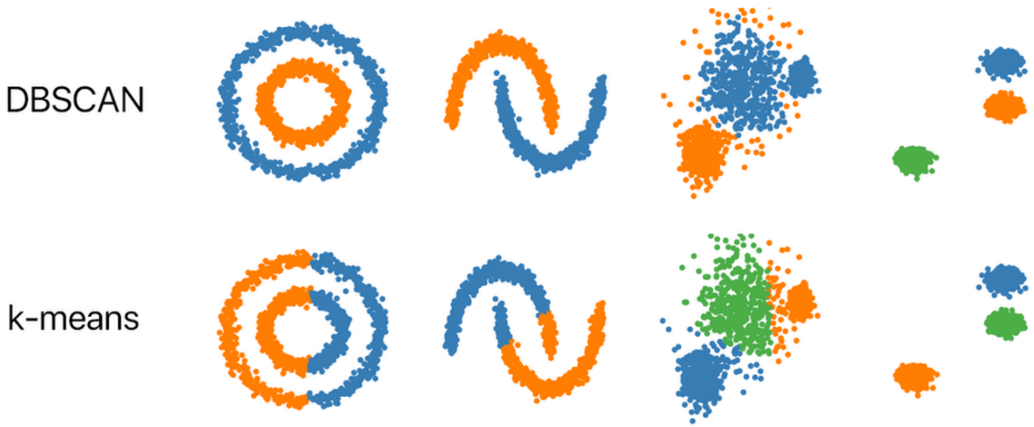
- Does not require specifying the number of clusters.
- Can find arbitrarily shaped clusters.
- Handles noise and outliers well (points not belonging to any cluster are considered noise).
- Good for data with varying densities.

Disadvantages:

- Sensitive to the choice of parameters (epsilon and minPts).
- Struggles with clusters of varying densities.
- May have difficulties when the dataset has very high dimensionality.

2.4 Density based Spatial clustering of Applications with Noise(DBSCAN)





3

Reinforcement Learning

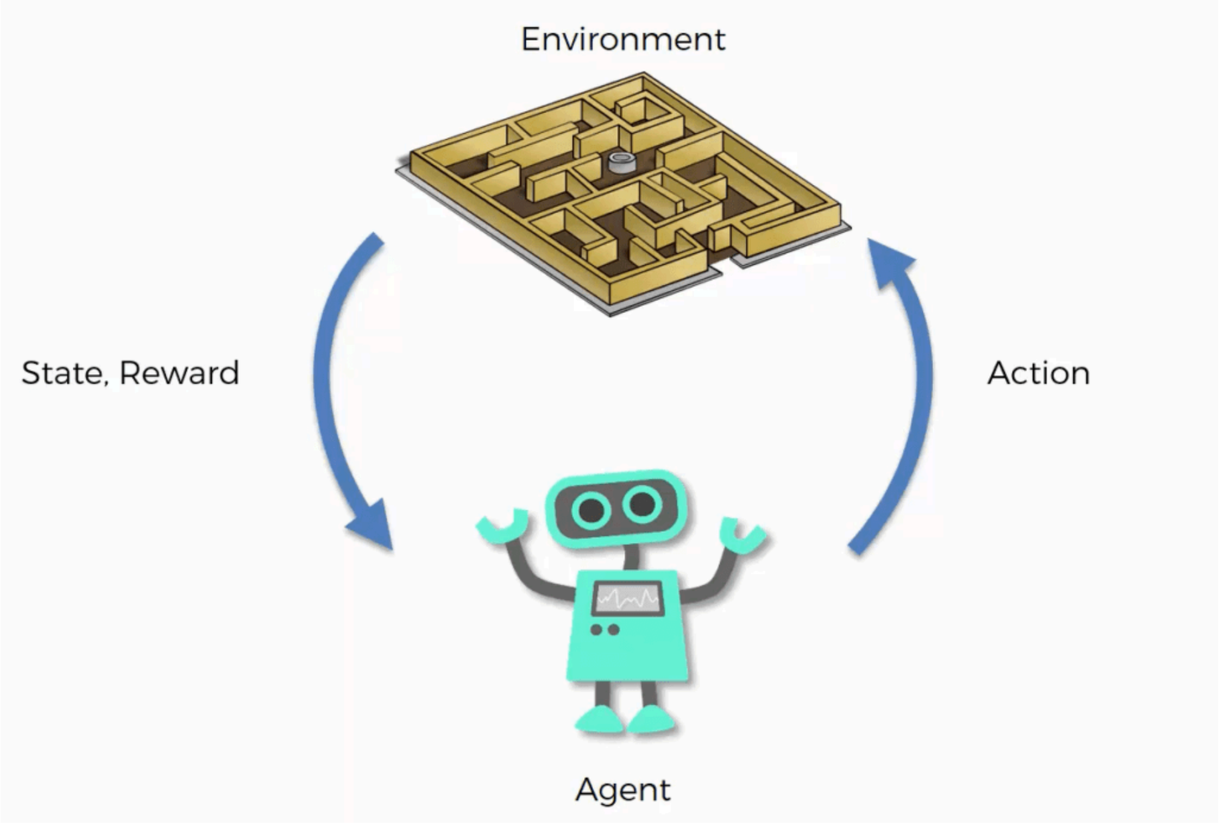
Reinforcement Learning (RL) is a type of machine learning where an agent learns by interacting with an environment and getting rewards or penalties based on its actions.

Basic Concepts:

- **Agent:** Learner or decision-maker.
- **Reward:** Feedback signal.
- **Environment:** The system the agent interacts with.
- **Policy:** Strategy that the agent follows.

Applications

- Robotics
- Self-Driving Cars
- Game Playing



Assignments

- Difference between supervised, unsupervise and reinforcement learning.

6.3 Modeling Process, Training /Validating model, Cross Validation methods, Predicting new observations Interpretation

a What is a model?

A model in machine learning is a mathematical representation of a real-world process that learns patterns from data and makes predictions or decisions.

b Components of a Model

A machine learning model consists of:

- Inputs (Features, X): Variables used for prediction.
- Parameters (w,b): Values adjusted during training to minimize error.
- Prediction Function ($\hat{y}=f(X)$): Maps inputs to outputs.
- Loss Function ($L(y,\hat{y})$): Measures the error in predictions.
- Optimization Algorithm: Adjusts parameters to minimize loss (e.g., Gradient Descent).

3.1 Modeling Process

Definition:

Modeling in ML refers to the overall process of designing, selecting, and building a mathematical or statistical model to learn from data.

Process:

1. **Problem Definition & Hypothesis Development**
2. **Type Identification**
3. **Platform & Infrastructure Selection**
4. **Data Collection & Preprocessing**
5. **Model & Algorithm Selection**
6. **Hyperparameter Tuning**
7. **Model Training**
8. **Model Testing & Validation**
9. **Model Evaluation**
10. **Model Deployment**
11. **Model Monitoring & Maintenance**

3.2 Training and Validating Models

1. Concepts of Training, Validation, and Hyperparameter Tuning

Model Training

- The process where the model learns patterns from labeled training data.
- Uses optimization algorithms like Gradient Descent to adjust model parameters (weights & biases).
- The goal is to minimize loss (error) on training data.

Model Validation

- Used to check how well the model generalizes to unseen data.
- Helps in hyperparameter tuning to avoid overfitting or underfitting.
- Commonly done using validation sets or cross-validation.

3.2 Training and Validating Models

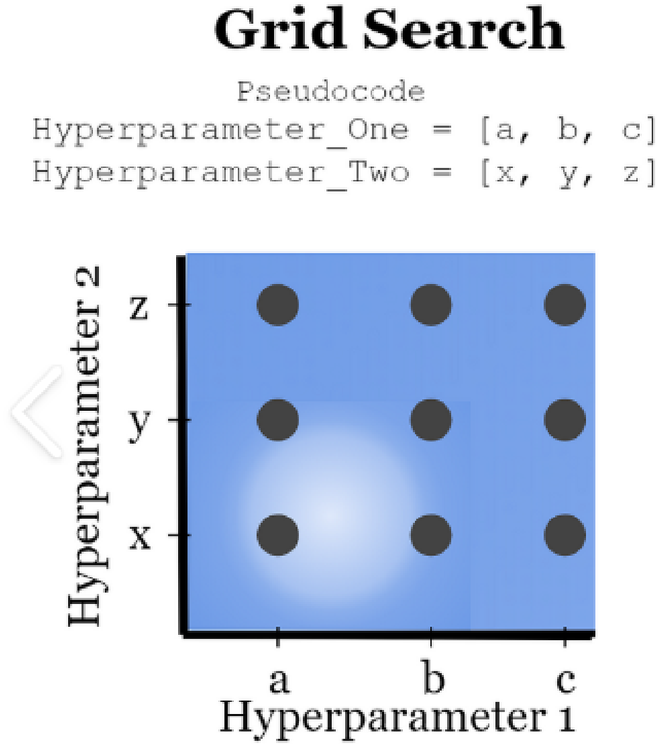
Hyperparameter Tuning

- Hyperparameters are not learned from data (e.g., learning rate, number of layers, dropout rate, max tree depth, Number of trees in the forest, etc).
- **Tuning methods:**
  - **Grid Search**
  - **Random Search**
  - **Bayesian Optimization**

Tuning methods:

a) Grid Search:

- **What it is:** This is an exhaustive search over a specified parameter grid. You define a grid of hyperparameters to try (e.g., different values of max_depth, min_samples_split), and grid search will evaluate all possible combinations.
- **How it works:**
  - You define the values you want to test for each hyperparameter.
  - Grid Search tests every possible combination of these values and selects the combination that performs best based on cross-validation or a validation set.
- **Pros:**
  - Simple and easy to understand.
  - Guarantees finding the best combination (within the defined grid).
- **Cons:**
  - Computationally expensive, especially with large datasets or many hyperparameters.
  - Does not scale well with increasing hyperparameter dimensions.



Tuning methods:

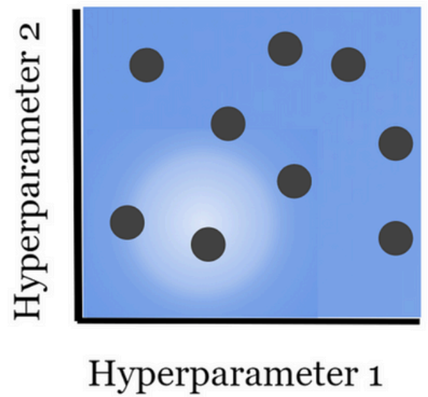
b) Random Search:

- **What it is:** Random search picks random combinations of hyperparameters from a predefined range and evaluates their performance.
- **How it works:**
  - You define a distribution (e.g., range of values) for each hyperparameter.
  - Random Search samples a combination of hyperparameters randomly and tests it.
  - This process continues for a fixed number of iterations or until computational limits are reached.
- **Pros:**
  - More efficient than grid search, especially when you don't know the best range for each hyperparameter.
  - Can potentially find better results faster by exploring a broader search space.
- **Cons:**
  - There's no guarantee of finding the absolute best hyperparameters.
  - You may miss important combinations.

Random Search

Pseudocode

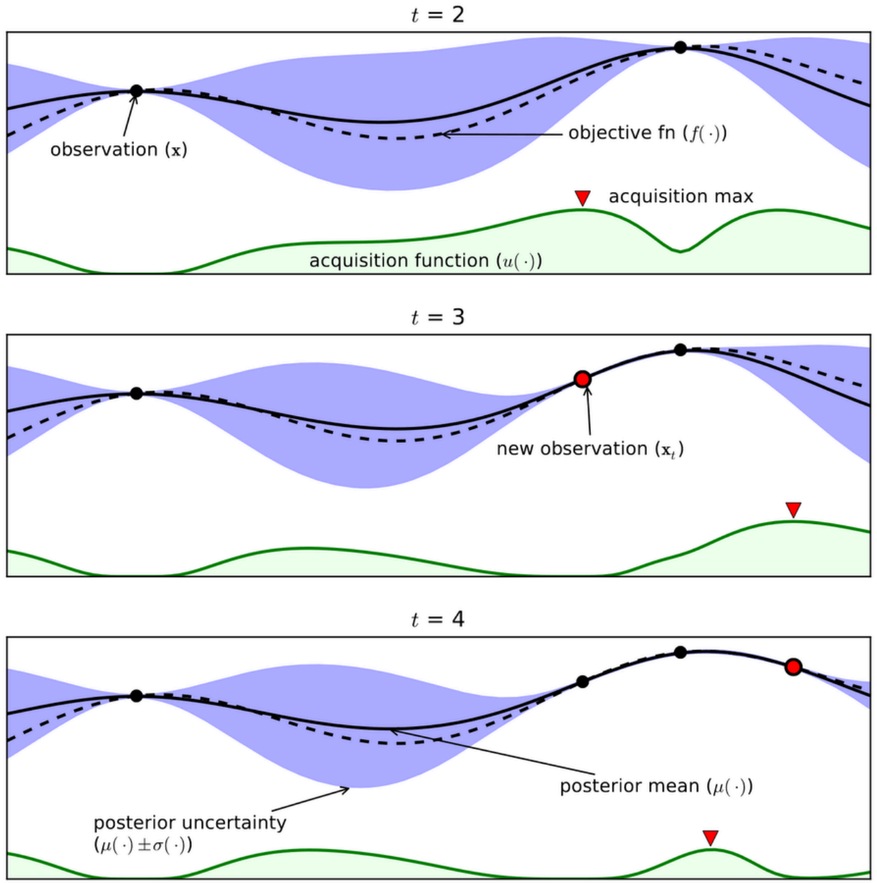
```
Hyperparameter_One = random.num(range)
Hyperparameter_Two = random.num(range)
```



Tuning methods:

c) Bayesian Optimization:

- **What it is:** A more advanced technique where a probabilistic model is used to predict the performance of hyperparameters and guide the search for better ones.
- **How it works:**
  - The algorithm models the objective function (e.g., validation score) as a probabilistic function.
  - Based on the previous results, it uses this model to suggest the next set of hyperparameters to try.
  - It is more efficient than Grid and Random Search because it explores promising regions of the hyperparameter space more intensively.
- **Pros:**
  - More efficient and faster, especially in high-dimensional spaces.
  - Can be computationally cheaper than grid search.
- **Cons:**
  - Requires more advanced understanding and is more complex to implement.
  - Can be computationally expensive in some cases.



Tuning methods:

Model	Key Hyperparameters to Tune	Recommended Tuning Method
Linear Regression	alpha (for Ridge/Lasso regularization)	Grid Search (if using regularization)
Logistic Regression	C (Inverse of regularization strength), penalty (L1/L2/ElasticNet)	Grid Search (small space)
Decision Trees	max_depth, min_samples_split, min_samples_leaf, criterion	Grid Search (small dataset) / Random Search (large dataset)
Random Forests	n_estimators, max_depth, min_samples_split, min_samples_leaf, max_features	Random Search (fast) / Grid Search (small dataset)
Support Vector Machines (SVM)	C (regularization), gamma (for RBF kernel), kernel (linear, RBF, poly)	Grid Search (small space) / Random Search (large space)
K-Nearest Neighbors (KNN)	n_neighbors, weights (uniform/distance), metric (euclidean, manhattan)	Grid Search (small dataset) / Random Search (for large n_neighbors)
Gradient Boosting (XGBoost, LightGBM, CatBoost)	learning_rate, n_estimators, max_depth, min_child_weight, subsample	Random Search (fast) / Bayesian Optimization (for fine-tuning)
K-Means Clustering	n_clusters, init (k-means++, random), max_iter, tol	Grid Search (small n_clusters) / Elbow Method & Silhouette Score
DBSCAN	eps, min_samples	Grid Search (small dataset) / Manual tuning using visualization

3.3 Training and Validation Loss

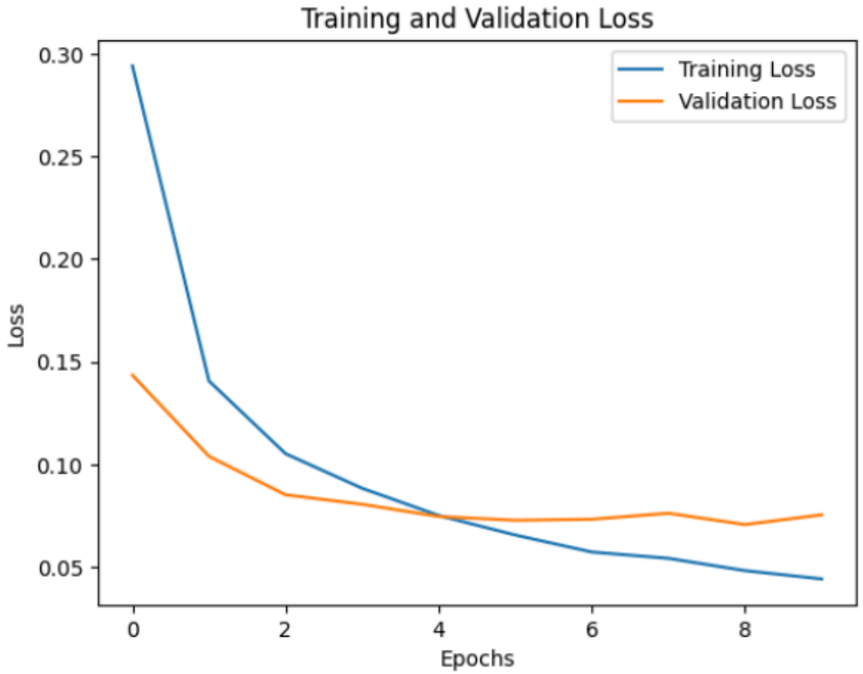
- **Training Loss**

➔ Measures error on the training data (should decrease over time).

- **Validation Loss**

→ Measures error on validation data (helps detect overfitting).

- **Ideally, both should decrease. If validation loss increases while training loss decreases, it suggests overfitting.**
- **A well-trained model has both losses decreasing smoothly.**
- **A bad model shows divergence (validation loss increasing, training loss decreasing).**



3.4 Overfitting and Underfitting

- **Overfitting (High Variance)**

🔴 **Problem:**

- The model memorizes training data instead of learning patterns.
- High accuracy on training data but poor performance on new data.

🔧 **Solutions:**

- Use more training data.
- Apply Regularization (L1/L2, Dropout).
- Reduce model complexity.
- Use Early Stopping.

- **Underfitting (High Bias)**

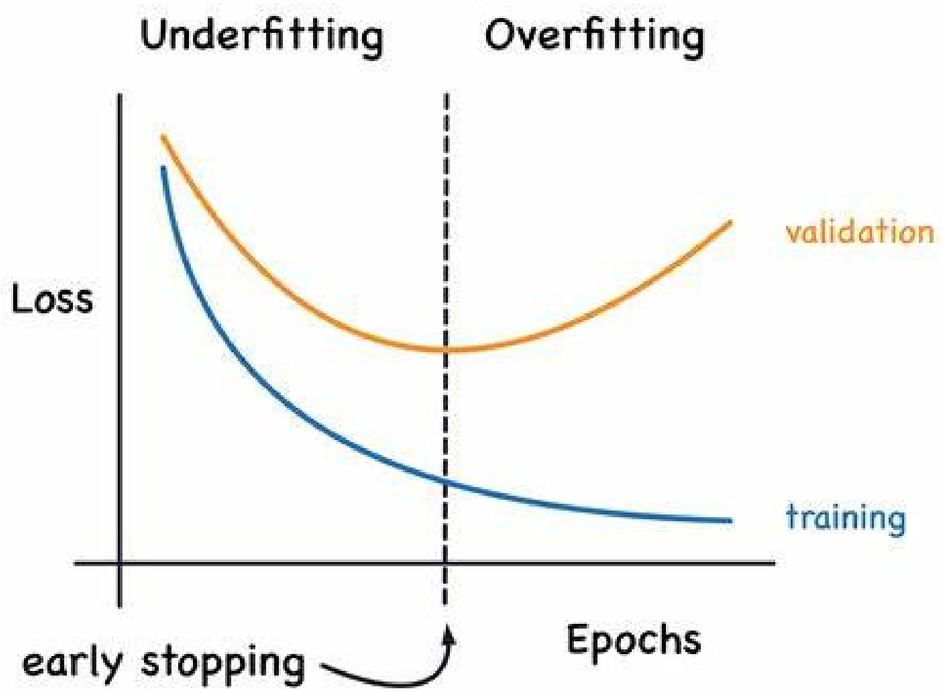
🔴 **Problem:**

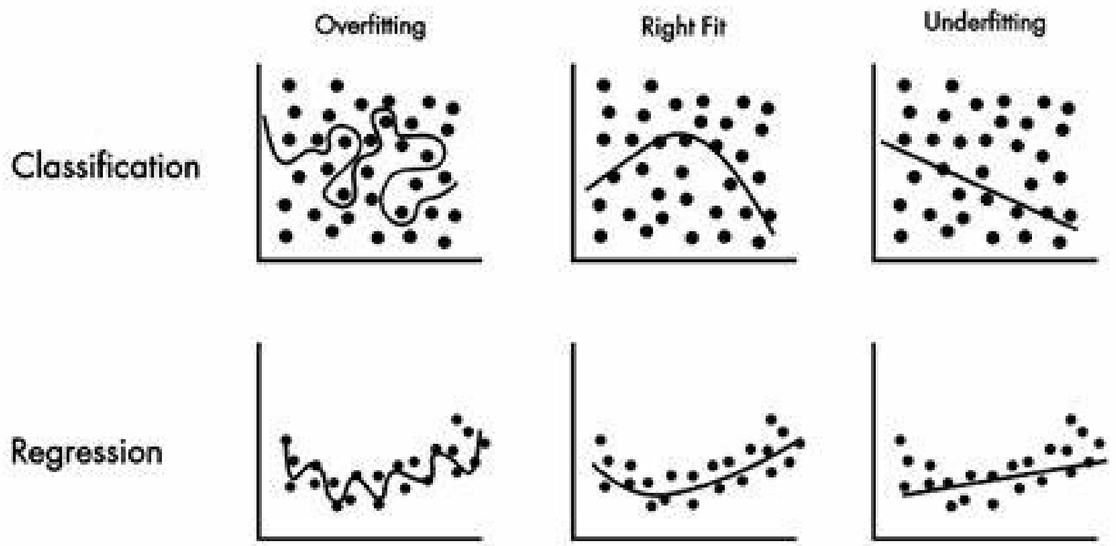
- The model is too simple to capture important patterns.
- Poor performance on both training and validation data.

🔧 **Solutions:**

- Use a more complex model (e.g., from Linear Regression → Neural Network).
- Train for more epochs.
- Use feature engineering to extract better patterns.

3.4 Overfitting and Underfitting





3.5 Bias-variance tradeoff.

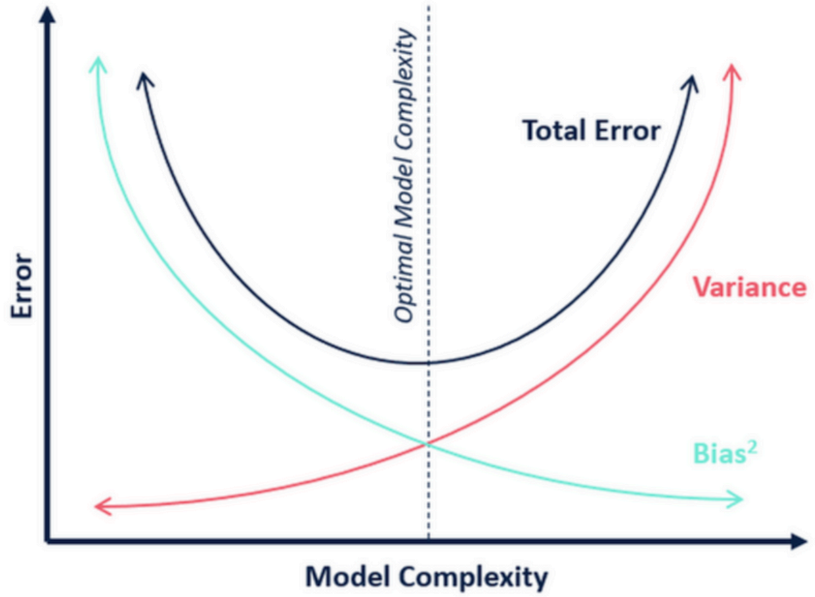
What is Bias?

- The bias is known as the difference between the prediction of the values by the Machine Learning model and the correct value.

What is Variance?

- The variability of model prediction for a given data point which tells us the spread of our data is called the variance of the model.

Concept	High Bias (Underfitting)	High Variance (Overfitting)
Definition	Model is too simple, misses patterns.	Model is too complex, memorizes noise.
Training Error	High	Low
Validation Error	High	High



🛠️ **Goal:**

Find the optimal balance between bias and variance to ensure generalization.

3.5 Bias-variance tradeoff.

How to Manage the Tradeoff?

Reducing Bias (Underfitting)

- Use more complex models (e.g., decision trees → random forests → neural networks).
- Increase training time (e.g., train for more epochs).
- Perform feature engineering to provide more relevant information to the model.

Reducing Variance (Overfitting)

- Use regularization (L1, L2, dropout in neural networks).
- Reduce model complexity (e.g., reduce number of parameters or layers).
- Use cross-validation to better assess model performance and prevent overfitting on the training set.
- Increase training data (more data helps the model generalize better)

3.6 Cross Validation

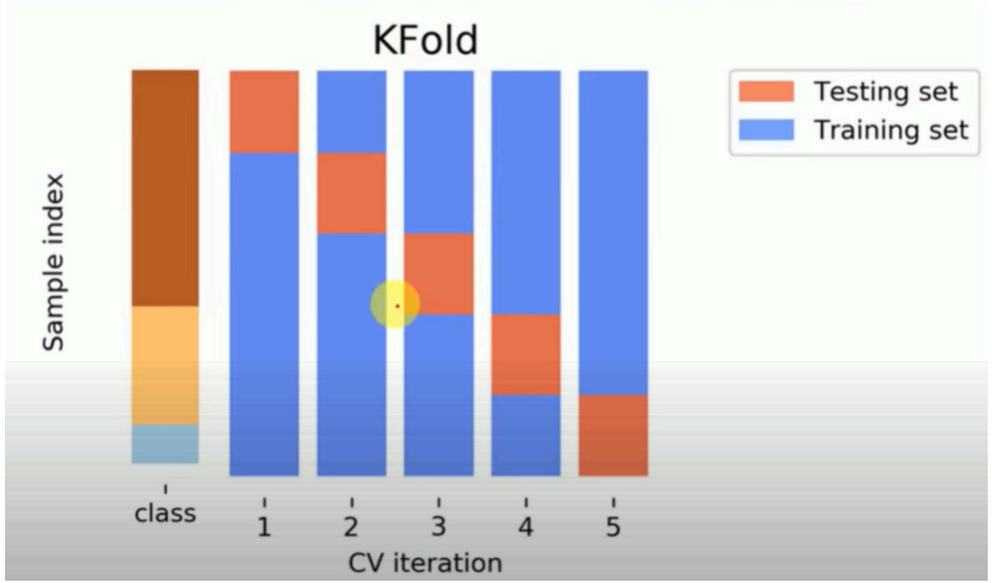
Cross-validation is a resampling technique used to evaluate machine learning models by splitting the data into multiple subsets. It helps estimate how well a model will generalize to unseen data.

3.6.1 Types of Cross-Validation

1. K-Fold Cross-Validation

✅ **Process:**

- The dataset is divided into **K** equal-sized folds.
- The model is trained on **K-1** folds and tested on the remaining fold.
- This process is repeated **K** times, and the final score is the average of all K iterations.



🔹 **Example:**

For 5-Fold CV, the dataset is split into 5 parts:

- Final Score = Average of all 5 test scores

✔️ **Advantages:**

- Reduces variance by using multiple training and test splits.
- Works well even for small datasets.

❌ **Limitations:**

- Computationally expensive for large datasets.

3.6 Cross Validation

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

✅ **Process:**

- A special case of K-Fold CV where K = number of samples.
- Each iteration, the model is trained on (n-1) samples and tested on 1 sample.
- The final score is the average of all test results.

✓ **Advantages:**

- Uses maximum training data, making it a low-bias approach.
- Ideal for very small datasets.

✕ **Limitations:**

- Extremely slow for large datasets (since it runs n times).
- High variance (a single test point can strongly influence the result).

3.6 Cross Validation

3. Stratified K-Fold Cross-Validation

✅ **Process:**

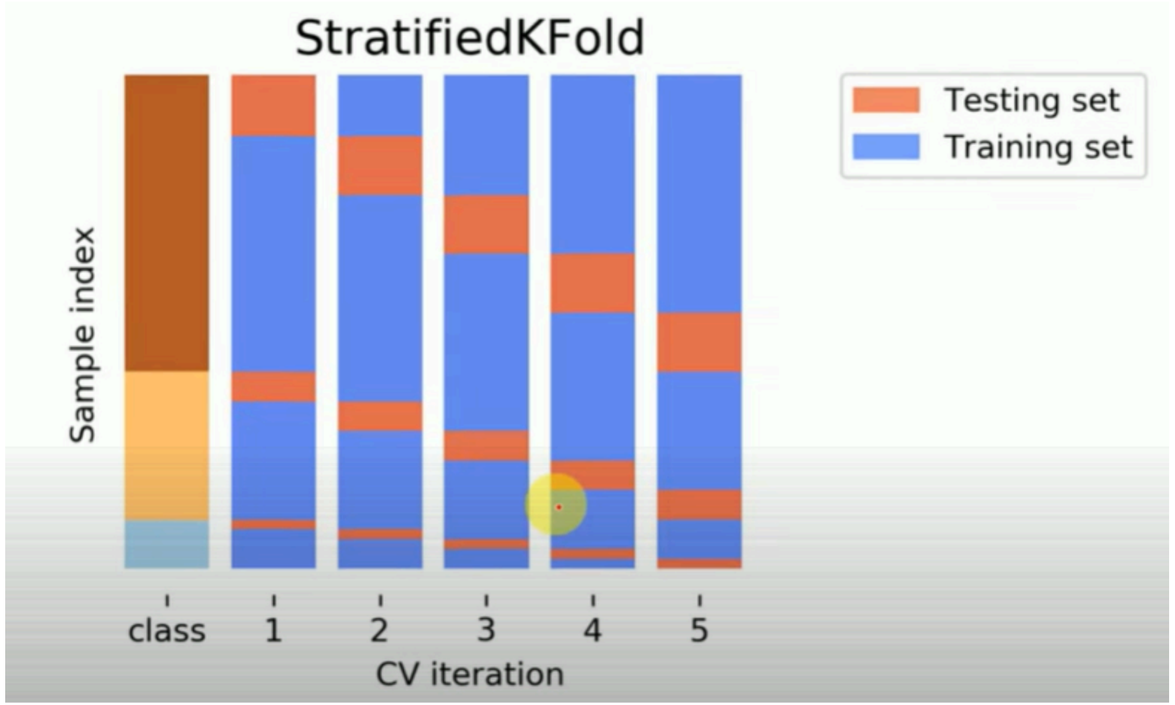
- Similar to K-Fold CV, but ensures that each fold maintains the same proportion of classes as in the original dataset.
- Useful for imbalanced datasets (e.g., 90% class A, 10% class B).

✔ **Advantages:**

- Ensures each fold has a representative class distribution.
- Reduces the risk of misleading performance metrics in imbalanced classification tasks.

❌ **Limitations:**

- Slightly more complex to implement than simple K-Fold.



3.7 Advantages & Limitations of Cross-Validation

✔️ **Advantages:**
✅ **Improves model generalization by training and testing on multiple subsets.**
✅ **Reduces variance in performance estimates.**
✅ **Avoids overfitting on a single train-test split.**
✖️ **Limitations:**
❌ **Computationally expensive (especially LOOCV).**
❌ **Does not fix data leakage if preprocessing isn't done properly.**

Which Cross-Validation Method to Use?

Scenario	Recommended Method
Small dataset	LOOCV
Large dataset	K-Fold (K=5 or 10)
Imbalanced dataset	Stratified K-Fold
High computation cost	K-Fold (lower K, e.g., 5)

3.8 Predicting New Observations: Model Deployment and Inference

1. Model Deployment

Deployment refers to integrating the trained model into a system where it can make real-time or batch predictions.

Steps in Model Deployment:

1. **Save the trained model**
   - Example formats:
     - joblib or pickle for scikit-learn models
     - .h5 for TensorFlow/Keras models
     - .pt for PyTorch models
2. **Set up an inference pipeline**
   - Preprocessing steps must be the same as used during training.
   - Example:
     - Convert categorical features
     - Scale/normalize numerical data
3. **Deploy to a server, cloud, or local system**
   - Local deployment: Run as a script
   - Web API: Use Flask, FastAPI, or Django to serve the model
   - Cloud deployment: Use AWS, GCP, or Azure
4. **Monitor and maintain the model**
   - Track model performance over time
   - Handle data drift (when new data distribution changes)
   - Update the model as needed

3.8 Predicting New Observations: Model Deployment and Inference

2. Inference in Unseen Data

Once deployed, the model makes predictions on new, unseen data.

Challenges in Inference:

- Data format mismatch (must match training format)
- Concept drift (new data distribution changes)
- Latency issues (in real-time applications)

Best Practices for Inference:

✅ Ensure preprocessing consistency between training and inference
✅ Use batch inference for large datasets to improve performance
✅ Monitor real-world predictions to detect errors or drift

6.4 Measures for Model Performance and Evaluation:

4.1 Classification Performance Measures

1. Classification Accuracy

Accuracy measures how many predictions were correct out of all predictions made:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \qquad \text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

✅ Pros: Simple and intuitive

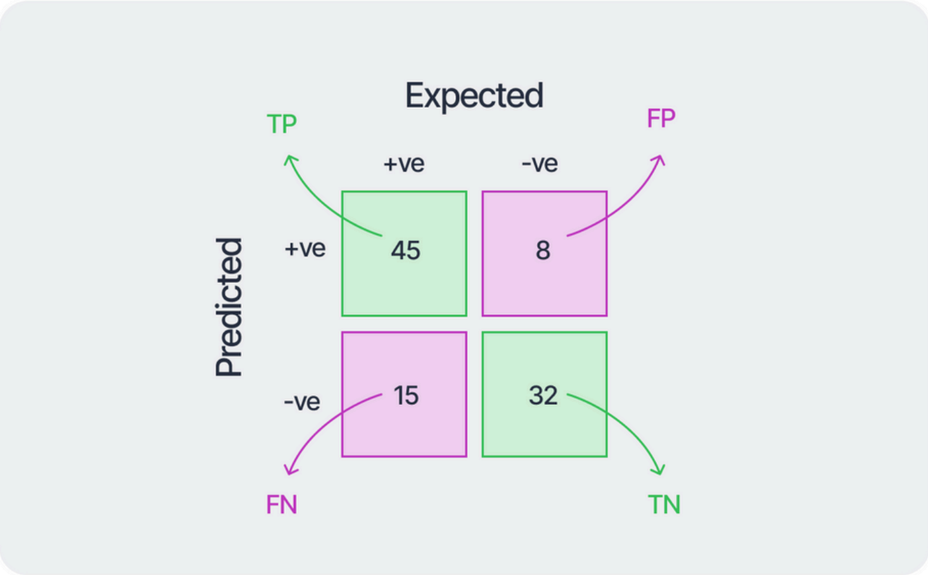
❌ Cons: Not reliable for imbalanced datasets (e.g., if 95% of samples belong to one class, a model predicting only that class will have 95% accuracy but is useless).

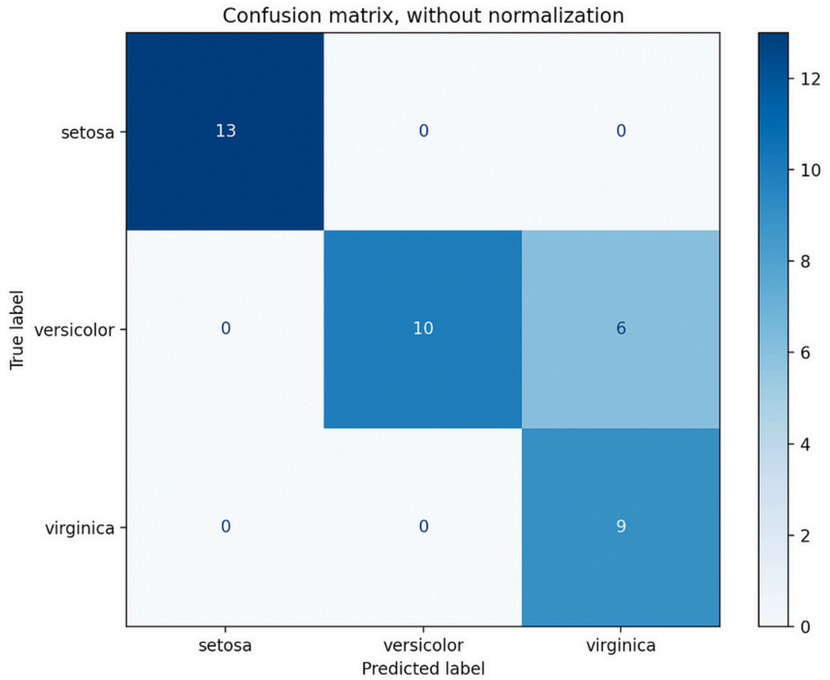
6.4 Measures for Model Performance and Evaluation:

4.1 Classification Performance Measures

2. Confusion Matrix and derived metrics:

A confusion matrix breaks down model predictions into four categories:





2. Confusion Matrix and derived metrics:

a) Recall (Sensitivity, True Positive Rate - TPR)
b) Specificity (True Negative Rate - TNR)
c) Precision (Positive Predictive Value - PPV)
d) F1-Score (Harmonic Mean of Precision & Recall)
e) ROC Curve & AUC (Area Under Curve)

2. Confusion Matrix and derived metrics:

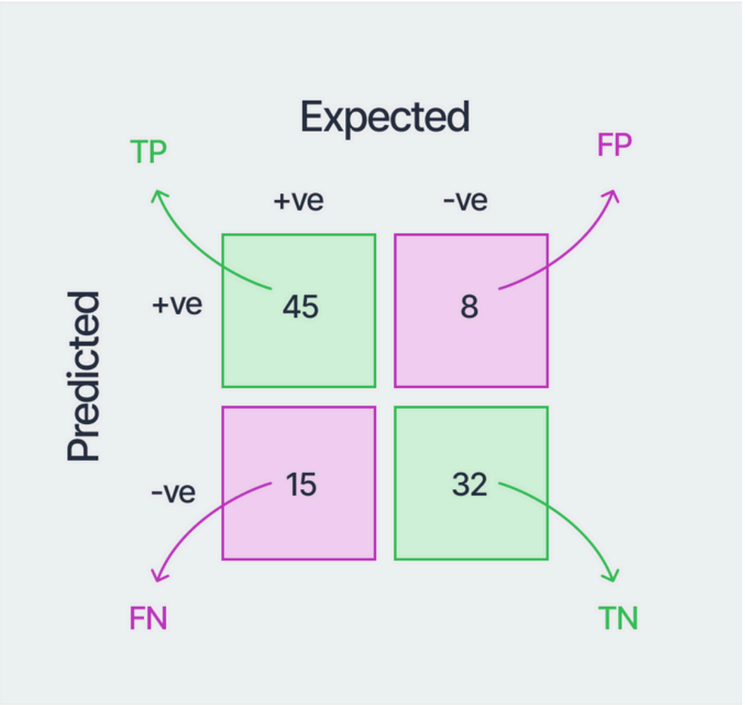
a) Recall (Sensitivity, True Positive Rate - TPR)

$$\text{Recall} = \frac{TP}{TP + FN}$$

Recall = 45 / (45 + 15)
= 0.75

- Measures how many actual positives were correctly identified.
- High recall means fewer false negatives.

✅ **Useful when False Negatives are costly (e.g., in cancer detection, missing a positive case is critical).**



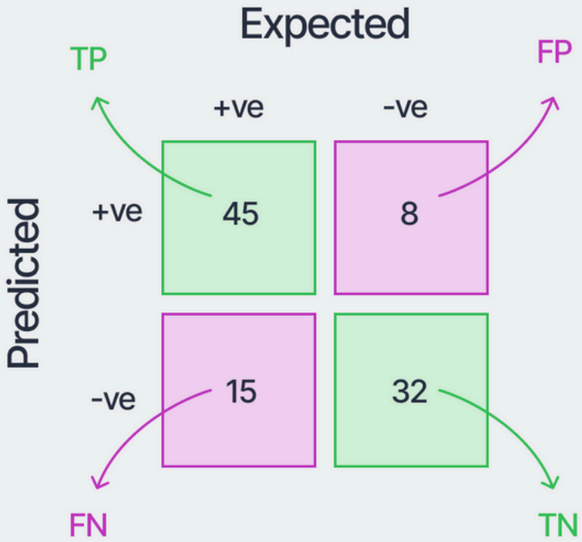
2. Confusion Matrix and derived metrics:

b) Specificity (True Negative Rate - TNR)

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Specificity = 32/(32+8)
= 0.8

- Measures how well the model avoids false alarms



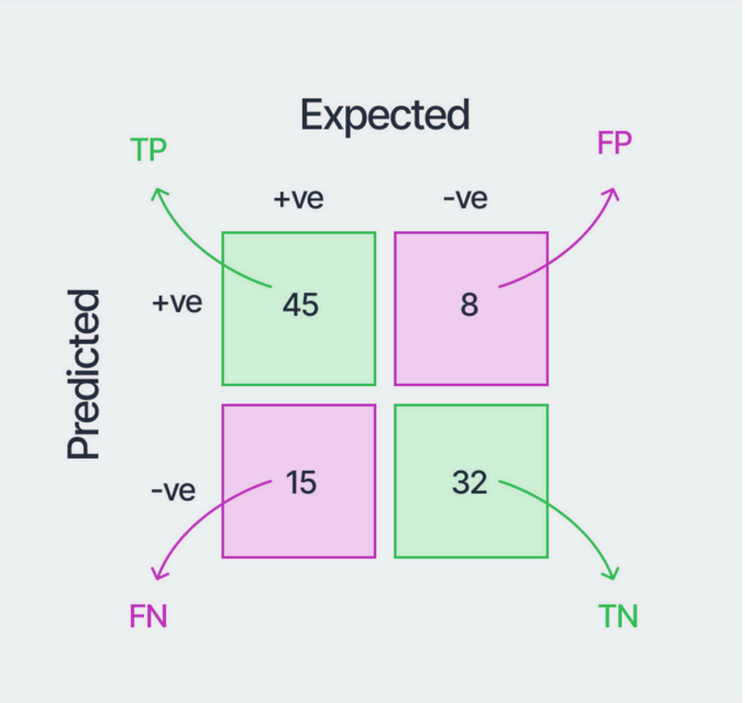
2. Confusion Matrix and derived metrics:

c) Precision (Positive Predictive Value - PPV)

$$\text{Precision} = \frac{TP}{TP + FP}$$

Precision = 45/(45+8)
= 0.849

- Measures how many predicted positives are actually correct.
- High precision means fewer false positives.



✅ **Useful when False Positives are costly (e.g., in spam detection, we want fewer non-spam emails classified as spam).**

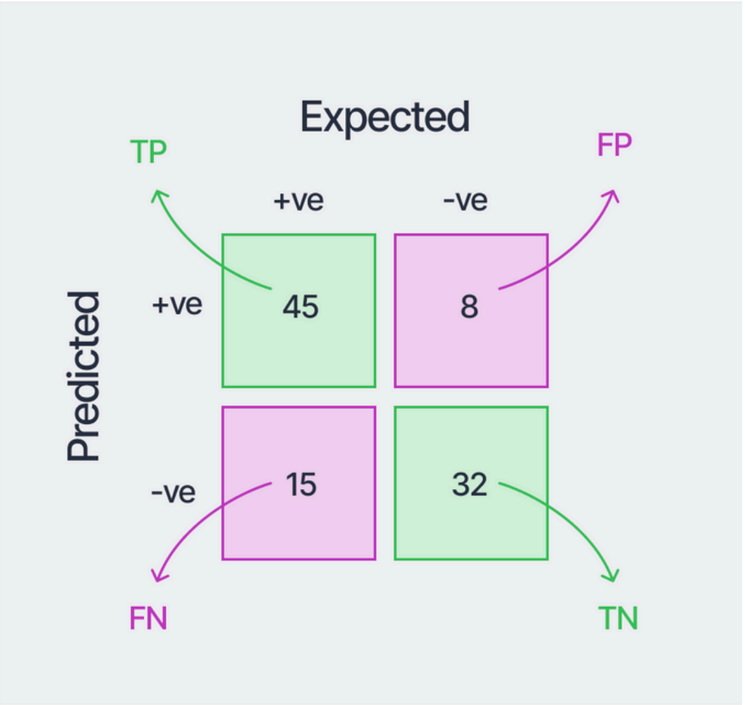
2. Confusion Matrix and derived metrics:

d) F1-Score (Harmonic Mean of Precision & Recall)

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1-score = (2 * (0.849 * 0.75))/(0.849 + 0.75)
= 0.796

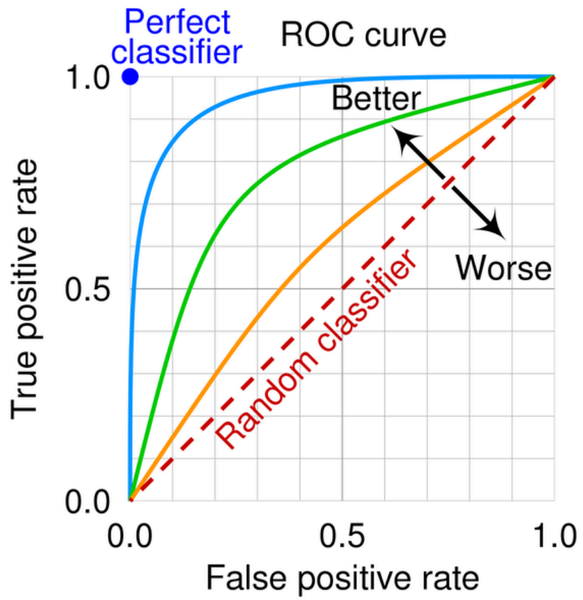
- A balance between Precision and Recall
- Best when dataset is imbalanced

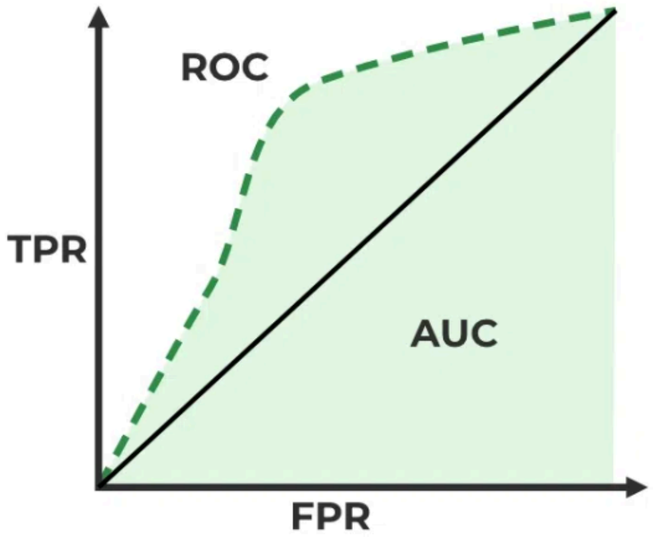


2. Confusion Matrix and derived metrics:

e) ROC Curve & AUC (Area Under Curve)

- Receiver Operating Characteristic (ROC) curve plots TPR vs. FPR
- AUC (Area Under Curve) quantifies how well the model differentiates classes





ROC-AUC Classification Evaluation Metric

6.4 Measures for Model Performance and Evaluation:

1. Internal Measures (Based on cluster structure)

Used when no ground truth labels are available.

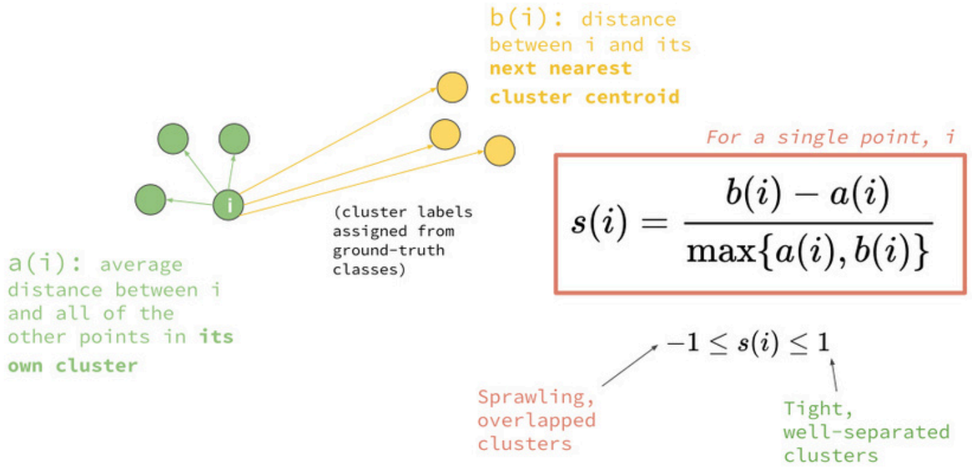
a) Silhouette Score

- Measures how well a data point fits in its assigned cluster vs. other clusters.
- Formula:

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

- where:
  - a(i) = average intra-cluster distance (within its own cluster)
  - b(i) = average nearest-cluster distance (to the next best cluster)
- Ranges from -1 to 1:
  - 1 → Well-clustered
  - 0 → Borderline
  - -1 → Misclustered

✅ Best for comparing different clustering results on the same dataset.



1. Internal Measures (Based on cluster structure)

b) Davies-Bouldin Index (DBI)

The Davies-Bouldin Index (DBI) measures the compactness and separation of clusters. Unlike the Silhouette Score, which is based on distances between individual points and clusters, DBI evaluates clustering quality based on cluster centroids and cluster spreads.

Formula

The DBI is calculated as:

$$DBI = \frac{1}{N} \sum_{i=1}^{N} \max_{j \neq i} R_{i,j}$$

where:

$$R_{i,j} = \frac{S_i + S_j}{D_{i,j}}$$

- S_i = Average distance between **points in cluster** i and **centroid of cluster** i (Measures **compactness**).
- S_j = Average distance between **points in cluster** j and **centroid of cluster** j.
- $D_{i,j}$ = Distance between **centroids of clusters** i **and** j (Measures **separation**).
- $R_{i,j}$ is the **relative similarity** between cluster i and cluster j.

DBI is the **average of the worst-case (maximum) cluster similarity** $R_{i,j}$ **for each cluster.**

Interpreting the DBI

- **Lower DBI values indicate better clustering because:**
- **Higher DBI values suggest poor clustering as clusters are overlapping or not well-defined.**

Range : 0 to ∞

2. External Measures (Uses ground truth labels)

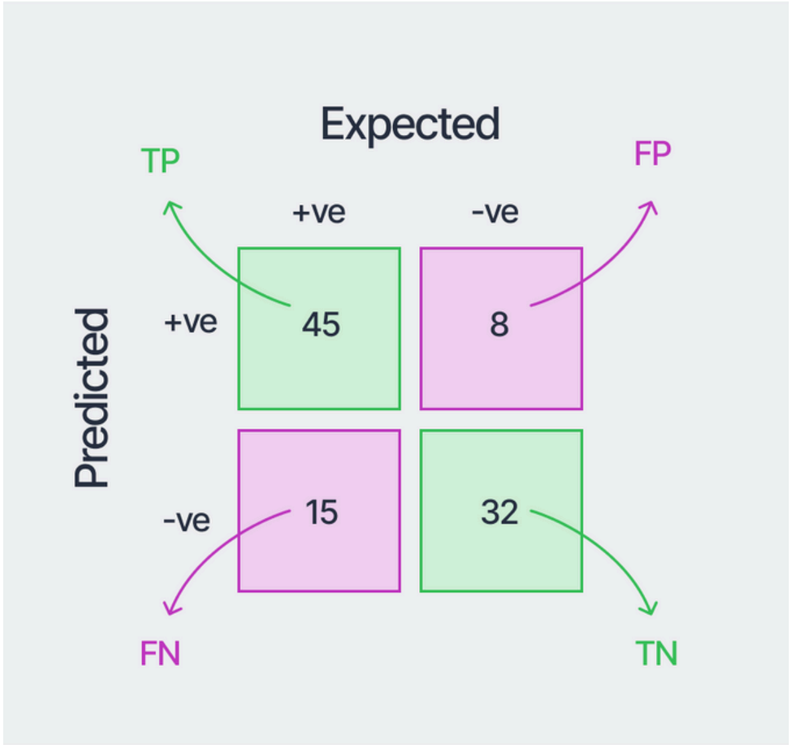
Used when true cluster assignments are known.

a) Rand Index

The Rand Index (RI) measures the similarity between two clusterings by counting the number of agreements between them.

Formula for Rand Index

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



RI = (45 + 32) / (45 + 8 + 15 + 32)
= 0.77

Rand Index values range from [0,1]:

- 1 (perfect clustering) → All pairs are correctly assigned.
- 0.5 (random clustering) → Similar to random assignment.
- 0 (worst case) → Completely incorrect clustering.

- Compares predicted clusters with actual labels.
- Counts agreements (true positives & true negatives).
- Adjusted Rand Index (ARI) corrects for random chance.

✅ Useful when evaluating clustering performance against labeled data.

2. External Measures (Uses ground truth labels)

b) Adjusted Rand Index (ARI)

- Since RI does not account for random chance, the Adjusted Rand Index (ARI) adjusts for it:

$$ARI = \frac{RI - E[RI]}{\max(RI) - E[RI]}$$

- where E[RI] is the expected value of RI under random clustering.
- **ARI values range from [-1,1]:**
  - 1 (perfect agreement)
  - 0 (random clustering)
  - Negative (worse than random clustering

3. Other Measures for regression and classification

For regression, we use:

a) Mean Squared Error (MSE)

b) Root Mean Squared Error (RMSE)

c) Mean Absolute Error (MAE)

d) R^2 Score (Explained Variance)

a) Mean Squared Error (MSE)

Definition:

MSE measures the average squared difference between the actual and predicted values. It penalizes larger errors more than smaller ones due to squaring, making it sensitive to outliers.

Formula:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where:

- y_i = actual value (ground truth)
- $\hat{y}_i$ = predicted value
- N = total number of observations

✅ **Pros:**

- Penalizes large errors (due to squaring).
- Differentiable, making it useful for optimization in machine learning.

❌ **Cons:**

- Sensitive to outliers, as large errors get amplified.

a) Root Mean Squared Error (RMSE)

Definition:

RMSE is the square root of MSE and provides error in the same units as the target variable. It is useful when we want a more interpretable error metric.

Formula:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

✅ **Pros:**

- Easier to interpret because it's in the same units as the target variable.

❌ **Cons:**

- Still sensitive to outliers, like MSE.

c) Mean Absolute Error (MAE)

Definition:

MAE calculates the average absolute difference between actual and predicted values. Unlike MSE, it does not square the errors, so it treats all errors equally.

Formula:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

✅ **Pros:**

- Less sensitive to outliers compared to MSE.
- More interpretable (gives an estimate of the average error).

❌ **Cons:**

- Does not penalize large errors as much as MSE/RMSE.

d) R^2 Score (Coefficient of Determination)

Definition:

R^2 measures how well the model explains the variance in the data. It compares the model's performance to a simple mean baseline model.

Formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

where:

- $\bar{y}$ = mean of actual values

✅ **Pros:**

- Measures goodness-of-fit in a normalized way.
- Ranges from 0 to 1 (higher is better), and it can be negative if the model is worse than predicting the mean.

❌ **Cons:**

- Does not indicate whether predictions are overestimated or underestimated.
- Can be misleading for non-linear relationships.

📌 **Interpreting R^2:**

- $R^2 = 1$ → Perfect prediction.
- $R^2 = 0$ → Model performs as poorly as predicting the mean.
- $R^2 < 0$ → Model performs worse than predicting the mean.

Assignments

- Given the following actual and predicted values:

Actual (yi)	Predicted (yi')
3	2
5	5
7	8
9	7
6	6

Calculate the following metrics for this dataset:

- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- R^2 Score (Coefficient of Determination)