Chapter 6

By Ramesh Tamang

**Introduction to Machine Learning**

• Machine learning (ML) is a branch of artificial intelligence (AI) that focuses on building systems that can learn and improve from experience without being explicitly programmed.

• Machine learning involves training a system using data, allowing it to recognize patterns and make decisions or predictions based on that data.

• By using **algorithms** and **statistical models**, ML enables computers to analyze data, recognize patterns,and make decisions or predictions.

**Goals of Machine Learning**

**1. Prediction**: Use data to predict future outcomes (e.g., stock prices, weather).

**2. Pattern Recognition**: Identify patterns in data (e.g., facial recognition) and classify outcomes based on patterns.

**3. Automation**: Automate decision-making processes (e.g., fraud detection).

**4. Adaptability**: Improve performance over time as new data is encountered.

**5. Insights**: Provide actionable insights by uncovering hidden relationships in data.

Rule-Based Programming vs Machine Learning

• In traditional rule-based systems, you define every rule explicitly, like "If condition A happens, then do B."

• However, in ML, you feed data into the system, and it learns the rules automatically by identifying patterns.

• **For example:** In spam detection, instead of writing rules like "If email contains 'free money', mark it as spam," ML models analyze thousands of spam and non-spam emails to learn patterns that differentiate the two.

Rule-Based Programming vs Machine Learning

**Aspect Rule-Based Programming Machine Learning**

**Approach** Uses manually coded rules and logic.

**Flexibility** Requires updates when conditions change.

**Data Dependency** Works well with deterministic processes.

Learns patterns from data to make predictions.

Adapts automatically as new data is provided.

Suitable for complex, data-rich problems.

**Examples** Spam filter using keywords. Spam filter using trained models on labeled emails.

**Scalability** Hard to scale for large  datasets or complex tasks.

Scales well with large datasets and complex tasks.

**Applications of Machine Learning**

• **Healthcare**: Disease diagnosis, drug discovery, personalized medicine.

• **Finance**: Fraud detection, stock market predictions, credit scoring.

• **Transportation**: Autonomous vehicles, route optimization, traffic prediction.

• **Retail**: Recommendation systems, customer segmentation, demand forecasting.

• **Natural Language Processing (NLP)**: Chatbots, sentiment analysis, translation services.

• **Computer Vision**: Facial recognition, object detection, medical imaging analysis.

**Key Concepts in Machine Learning**

• **Data**:

Data is the foundation of ML. It consists of the information (structured or unstructured) that machines learn from. Examples include numerical data, text, images, and audio. • **Algorithms**:

Algorithms are the methods or procedures used to identify patterns in data and make predictions. Common ML algorithms include linear regression, decision trees, and neural networks.

• **Training and Testing**:

• **Training** involves teaching an ML model by feeding it data and allowing it to learn the relationships within that data.

• **Testing** evaluates the model’s performance on new, unseen data.

• **Features**:

Features are individual measurable properties or characteristics of the data. Selecting the right features is crucial for effective modeling.

• **Models**:

A model is the output of the ML training process. It represents the learned patterns and relationships in the data.

**The ML Workflow**

**1.Data Collection**: Gathering relevant and sufficient data.

**2.Data Preprocessing**: Cleaning and transforming data to make it suitable for modeling.

**3.Model Selection**: Choosing the appropriate ML algorithm. **4.Training/Testing**: Feeding the algorithm with training data to learn patterns and testing model using test data set.

**5.Evaluation**: Measuring model performance using metrics like accuracy, precision, recall, etc.

**6.Deployment**: Integrating the trained model into a real-world system.

**7.Monitoring and Maintenance**: Continuously evaluating and updating the model as needed.

**Introduction to Supervised, Unsupervised, and**

**Reinforcement Learning**

**1. Supervised Learning**

**Definition**: Supervised learning involves training a model on a labeled dataset, where each input (feature) has a corresponding output (label).

**Labeled data**: Data includes input-output pairs (e.g., features and labels).

**1. Supervised Learning (contd.)**

• **Common Algorithms**:

• Linear regression.

• Logistic regression.

• Decision trees and random forests.

• Support vector machines (SVM).

• Neural networks.

• **Examples**:

• **Regression Problem**: Predicting temperature based on historical weather data. •

**Classification Problem**: Identifying whether an image contains a cat or not.

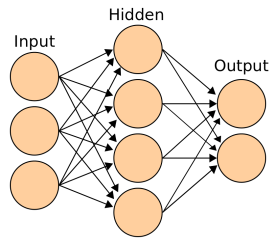
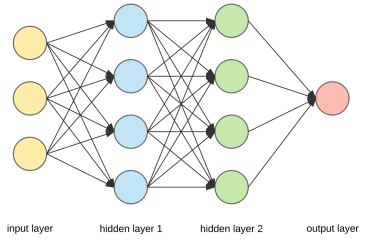
• Predicting house prices (regression).

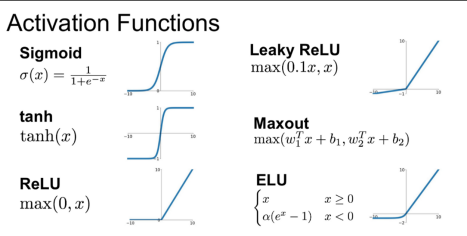
• Classifying emails as spam or not spam (classification).

• Diagnosing diseases from medical test results (classification).

Artificial Neural Networks (ANN)

• ANN is inspired by the structure and functioning of the human brain. It consists of layers of interconnected nodes (neurons) designed to recognize patterns and relationships in data.





Structure of ANN

**1. Input Layer**:

• Takes input features from the dataset.

• Each neuron corresponds to one feature.

**2. Hidden Layers**:

• Perform intermediate computations.

• Extract patterns and relationships in the data.

• Can have one or more layers depending on the problem's complexity.

**3. Output Layer**:

• Produces the final output.

• For classification, it provides class probabilities.

• For regression, it gives a continuous value.

**4. Connections (Weights)**:

• Each connection between neurons has an associated weight that determines the strength of the connection.

**5. Bias**:

• A constant added to the input of a neuron to adjust the output along with the weights.

Components of ANN

**1. Neuron (Node)**:

• Receives input, processes it using an activation function, and sends the output to the next layer.

**2. Activation Function**:

• Introduces non-linearity into the network.

• Common activation functions:

• **Sigmoid**: σ(x) = 1 / (1 + e⁻ᵇ)

• **ReLU (Rectified Linear Unit)**

• **Tanh**

• **Softmax**: Used in the output layer for multi-class classification.

**3. Loss Function**:

• Measures the error between the predicted and actual outputs.

• Common loss functions:

• Mean Squared Error (MSE): For regression.

• Cross-Entropy Loss: For classification.

Components of ANN (contd.)

**4. Optimizer**:

• Updates the weights to minimize the lossfunction

. • Popular optimizers:

• Stochastic Gradient Descent (SGD)

• Adam Optimizer

**5. Learning Rate**:

• Determines the step size for updating weights.

Working of ANN

**1. Forward Propagation**:

• Input data passes through the layers of the network.

• Neurons compute weighted sums of inputs, add biases, and apply activation functions.

• Produces the output.

**2. Loss Calculation**:

• Compares the predicted output with the actual label using the loss function.

**3. Backward Propagation**:

• Computes gradients of the loss function with respect to the weights and biases. • Uses algorithms like **gradient descent** to adjust weights and minimize error.

**4. Iteration**:

• The above steps are repeated for multiple epochs until the model converges or achieves the desired accuracy.

Naive Bayes

• Naive Bayes is a probabilistic classification algorithm based on Bayes' Theorem.

• It assumes that all features are independent of each other given the class label.

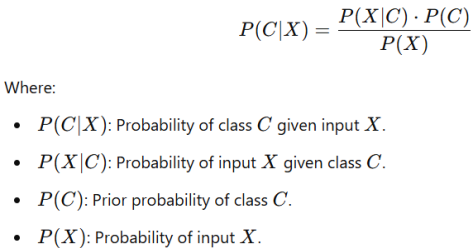
• Bayes’ Theorem



• **Assumptions**: (which might not always hold in real-world scenarios) • Conditional Independence: Each feature contributes independently to the class probability.

• Equal Importance of Features: All features are equally important.

Bayes’ Theorem



Example: Classifying Emails as Spam or Not Spam

• We classify a new email containing the words **"free", "offer", "win"** into **Spam** or **Not Spam**.

• Training Data

| **Email** | **Class** | **Words in Email** |
| --- | --- | --- |
| 1 | Spam | free (3), offer (2), win (1) |
| 2 | Spam | free (2), offer (1), win (3) |
| 3 | Not Spam | free (1), offer (3), win (2) |
| 4 | Not Spam | free (1), offer (2), win (1) |

**Step 1: Count Words in Each Class** • **Spam Class**

• Total Spam Emails = 2

• Words in Spam Emails:

• Email 1: **free (3), offer (2), win (1)**

• Email 2: **free (2), offer (1), win (3)**

• Total Count of Words in Spam:

• **free** = 3+2=5

• **offer** = 2+1= 3

• **win** = 1+3=4

• Total Words in Spam = 5+3+4=12

• **Not Spam Class**

• Total Not Spam Emails = 2

• Words in Not Spam Emails:

• Email 3: **free (1), offer (3), win (2)**

• Email 4: **free (1), offer (2), win (1)**

• Total Count of Words in Not Spam:

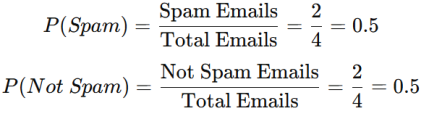
• **free** = 1+1=2

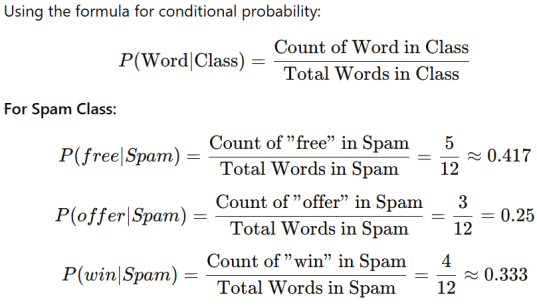
• **offer** = 3+2=5

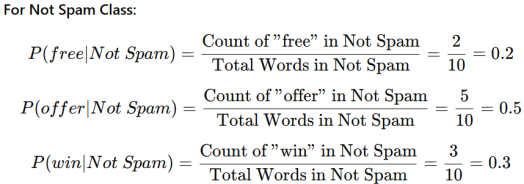
• **win** = 2+1=3

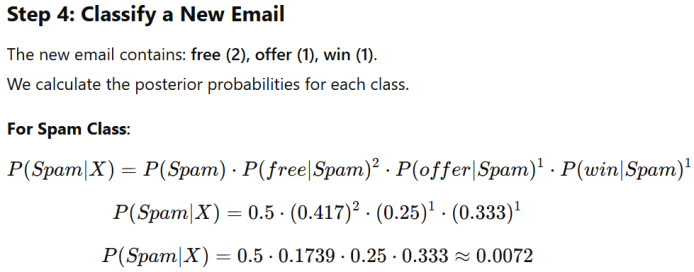
• Total Words in Not Spam = 2+5+3=10

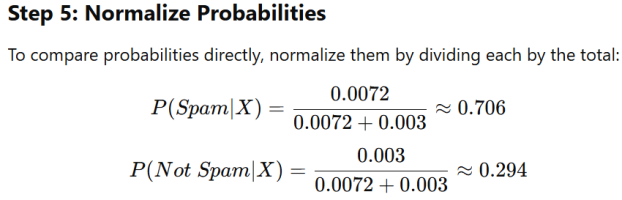
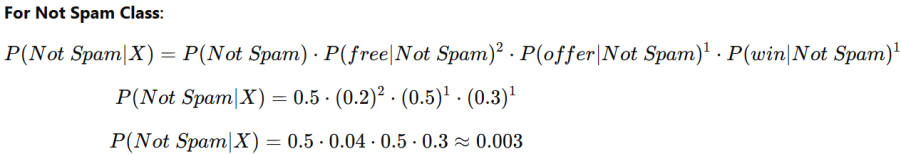
**Step 2: Calculate Prior Probabilities**

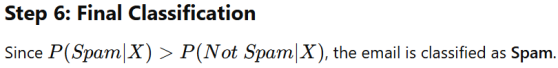
****

**Step 3: Calculate Conditional Probabilities**

****

****

****

****

**2. Unsupervised Learning**

**Definition**:

Unsupervised learning involves training a model on data that lacks labeled outputs. The model identifies hidden patterns, structures, or groupings in the data.

**Unlabeled data**: No explicit labels or outputs are provided.

**Goal**: Discover meaningful patterns or groupings in the data.

**Applications**:

• Market segmentation: Grouping customers with similar buying habits. • Dimensionality reduction: Simplifying large datasets (e.g., PCA). • Anomaly detection: Identifying fraudulent transactions or system failures.

**2. Unsupervised Learning (contd.)**

**Common Algorithms**:

• K-means clustering.

• Hierarchical clustering.

• Principal Component Analysis (PCA).

• Autoencoders.

**Examples**:

• Clustering social media users based on shared interests. • Grouping items in a grocery store by purchase patterns (e.g., milk and bread often bought together).

• Market segmentation: Grouping customers with similar buying habits. • Dimensionality reduction: Simplifying large datasets (e.g., PCA). • Anomaly detection: Identifying fraudulent transactions or system failures.

K-means clustering

• K-Means is an unsupervised machine learning algorithm used to group data into **K clusters** based on their similarities.

• The algorithm minimizes the distance between data points and their corresponding cluster centroids.

• It divides data into K groups

**Steps of the K-Means Algorithm**

**1. Choose the number of clusters (K)**.

**2. Initialize centroids** (randomly select K data points or use specific methods).

**3. Assign each data point to the nearest centroid** based on a distance metric (e.g., Euclidean distance).

**4. Recompute centroids** as the mean of all points in each cluster.

5. Repeat steps 3–4 until the centroids no longer change significantly or the maximum number of iterations is reached.

Example: Grouping Data Points into 2 Clusters (K=2)

**Dataset:** We have the following points in a 2D space:

• A(2,3)

• B(3,3)

• C(6,6)

• D(8,8)

• We want to cluster these points into **2 clusters**.

**Step 1: Initialize Centroids**

Let’s randomly select the initial centroids as:

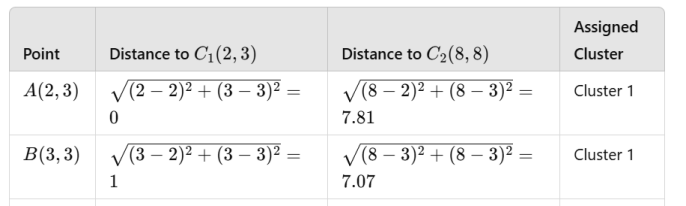
• C1(2,3) (Cluster 1)

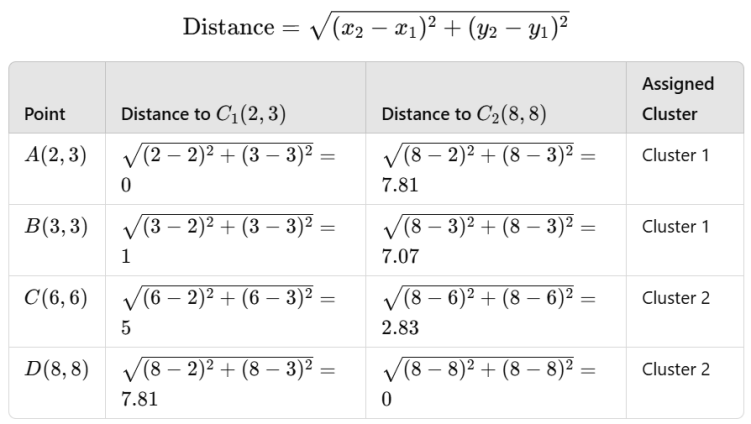
• C2(8,8) (Cluster 2)

**Step 2: Assign Points to Nearest Centroid**

We calculate the Euclidean distance between each point and the centroids using the formula:















**Density-Based Clustering**

• Unlike K-means, it groups data based on regions of high density and identifies outliers as noise.

• Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a popular density-based clustering algorithm.

• OPTICS (Ordering Points To Identify the Clustering Structure) • HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise):

DBSCAN

• DBSCAN defines clusters as areas of high density separated by areas of low density. It relies on two key parameters: **1. Epsilon (ε)**: The radius within which points are considered neighbors. **2. MinPts**: The minimum number of points required to form a dense region (a cluster).

Key Concepts

• Core Point: A point with at least MinPts neighbors within ��. • Border Point: A point within �� of a core point but with fewer than MinPts neighbors.

• Noise: A point that is neither a core point nor a border point.

Steps in DBSCAN

1. Select an unvisited point.

2. If the point is a core point:

• Form a cluster by including all points within ��.

• Expand the cluster by recursively including directly density-reachable points.

3. Mark border points that are directly connected to core points but do not meet MinPts.

4. Points not part of any cluster are labeled as noise.

DBSCAN

• Advantage of DBSCAN

• Identifies Noise: Marks outliers that don't belong to any cluster. • No Need to Specify Clusters (K): Unlike K-means, DBSCAN automatically determines the number of clusters.

• Limitations of DBSCAN

• Parameter Sensitivity:

• �� and MinPts heavily influence clustering results.

• High Dimensionality:

• Distance calculations become less meaningful in high-dimensional spaces.

**Reinforcement Learning (RL)**

• **Reinforcement Learning (RL)** is a type of machine learning where an agent learns to make decisions by interacting with an environment. The agent receives feedback in the form of rewards or penalties and adjusts its actions to maximize cumulative rewards over time.

• **Agent, environment, and reward**: The agent interacts with an environment and learns based on feedback.

• **Goal**: Maximize long-term rewards by learning an optimal policy (strategy).

Basic Concept in Reinforcement Learning

• **Agent**: The decision-maker in the RL system. It interacts with the environment by taking actions and learning from feedback.

• **Environment**: The system or world in which the agent operates. The environment responds to the agent's actions and provides feedback in the form of rewards and new states.

• **Reward**: A scalar value provided to the agent as feedback for its actions. The reward indicates the immediate benefit of an action and guides the agent toward its goal.

• **Policy**: The strategy or mapping from states to actions that the agent follows. The agent's goal is to learn an optimal policy.

• **State**: A representation of the environment's current situation as perceived by the agent. It provides the context for decision-making.

• **Action**: A choice made by the agent at each step, which alters the state of the environment.

Reinforcement Learning: Applications

• **Robotics**: Training robots for tasks like walking, grasping objects, and assembling parts. • **Gaming**: RL is used to create agents that can play games, often achieving superhuman performance.

• **Autonomous Vehicles**: Training vehicles to navigate, avoid obstacles, and follow traffic rules.

• **Finance**: Optimizing stock trading strategies.

• **Healthcare**: Personalized treatment plans and drug discovery using simulations. • **Energy Optimization**: Smart grid systems and efficient energy distribution in power networks.

• **Recommendation Systems**:

• Improving user experiences by learning preferences dynamically.

• Example: Dynamic advertisement placement and personalized content delivery. • **Natural Language Processing (NLP)**:

• Fine-tuning models for conversation generation, machine translation, and summarization.

**Summary**

• **Supervised Learning** is ideal for tasks where labeled data is available, like spam detection or price prediction.

• **Unsupervised Learning** excels in discovering hidden structures, like customer segmentation.

• **Reinforcement Learning** is used for sequential decision-making problems, like training autonomous agents.

**The Machine Learning Modeling Process**

• The machine learning modeling process involves creating a model that can make predictions or decisions based on data. It typically consists of several key steps:

• **1. Defining the Problem**

• Clearly state the problem you want to solve.

• Identify the type of learning required (supervised, unsupervised, or reinforcement).

• Specify the type of output (e.g., classification, regression, clustering).

• **2. Data Collection and Preprocessing**

• **Data Collection**: Gather relevant data from various sources.

• **Data Cleaning**: Handle missing values, outliers, and inconsistencies. • **Feature Engineering**: Select, create, or transform features to improve model performance.

• **Normalization/Scaling**: Adjust the range of features for algorithms sensitive to scale.

• **3. Model Selection**

• Choose an appropriate algorithm based on the problem type, data characteristics, and desired output. Examples:

• Linear Regression for regression problems.

• Decision Trees or SVMs for classification problems.

• K-means for clustering tasks.

• **4. Training the Model**

• Split the dataset into **training** and **testing** subsets (commonly 70%-80% for training, 20%-30% for testing).

• Train the model on the training data by minimizing a loss function (e.g., Mean Squared Error for regression, Cross-Entropy Loss for classification).

• **5. Validating the Model**

• Validation ensures that the model generalizes well to unseen data. This is where **cross-validation** techniques come in.

•

**6. Model Evaluation**

• Evaluate the model’s performance using appropriate metrics: • **Classification**: Accuracy, Precision, Recall,F1-Score,ROC-AUC.

• **Regression**: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R² score.

**Clustering**: Silhouette score, Davies-Bouldin Index.

• **7. Predicting New Observations**

• Once the model is trained and validated, it can predict outcomes for new, unseen observations.

• Ensure that the same preprocessing steps (e.g., scaling, encoding) are applied to the new data as were applied during training.

• **8. Interpreting Predictions**

• Interpreting predictions helps stakeholders understand what the model outputs mean and how reliable they are.

• **1. Regression Predictions**

• Example: Predicting house prices.

• **Output**: A continuous value (e.g., $300,000).

• **Interpretation**: "Based on the input features, the estimated house price is $300,000."

• **Uncertainty**:

• Provide prediction intervals (e.g., "The house price is predicted to be between $290,000 and $310,000 with 95% confidence").

• Analyze residuals (difference between predicted and actual values) to check for systematic biases.

• **8. Interpretation of Predictions**

• Interpreting model predictions involves understanding the results and how the model arrived at them:

**1.Feature Importance**:

1. Determine which features contribute the most to predictions.

2. Algorithms like decision trees and random forests often provide feature importance scores.

**2.Residual Analysis**:

3.Analyze the differences between predicted and actual values (residuals) to detect systematic errors.

Steps in the Modeling Process

The modeling process involves several critical steps to build, validate, and refine a machine learning model:

**1. Define Objectives**: Clearly outline the problem and the desired outcome.

**2. Data Collection**: Gather relevant and high-quality data for the task.

**3. Data Preprocessing**: Clean, transform, and prepare data for modeling.

**4. Feature Selection/Engineering**: Select or create features that capture important information. **5. Model Selection**: Choose a suitable algorithm based on the problem (e.g., regression, classification, clustering).

**6. Model Training:** Fit the model to the training data to learn patterns.

**7. Model Validation and Testing:** Evaluate performance using validation data. **8. Optimization**: Refine the model by tuning hyperparameters.

**9. Deployment**: Integrate the model into a system for real-world use.

**10. Monitoring and Maintenance**: Track performance over time and update as necessary.

Training and Validating Models

Training and validating models ensure they generalize well to unseen data.

• **Training**

• Training a machine learning model involves feeding it with data to learn patterns and relationships. The goal is to optimize the model parameters to minimize the error or loss function.

• **Process**:

• Pass training data through the model.

• Compute the loss using a loss function (e.g., Mean Squared Error forregression,Cross Entropy for classification).• Use an optimization algorithm (e.g., Gradient Descent) to adjust model parameters. • Repeat the process for a set number of iterations or until convergence.

• **Key Outputs**:

• **Trained Parameters**: Coefficients or weights learned by the model.

• **Training Loss**: Measure of error on the training data.

**Validation**

• Validation evaluates the model's performance on a separate dataset (validation set) that the model hasn't seen during training. This step ensures the model generalizes well to unseen data and helps in selecting the best model configuration.

• **Purpose**:

• To prevent overfitting by monitoring the model's performance on unseen data.

• To provide feedback on hyperparameter settings.

• **Process**:

• Train the model on the training set.

• Test the model on the validation set after each training epoch or at regular intervals.

• Use the validation loss or accuracy as a metric to evaluate performance.

Training and Validation Loss

• Training Loss: Error measured on training data.

• Validation Loss: Error measured on validation data. • A small gap between these losses indicates a good fit. • **Validation Loss vs. Training Loss**:

• If validation loss is significantly higher than training loss, the model may be **overfitting**.

• If both training and validation losses are high, the model may be **underfitting**.

Hyperparameter Tuning

• Hyperparameters are settings that are not learned by the model but are set prior to training.

• Examples include the learning rate, number of layers, or regularization strength.

• Hyperparameter tuning is the process of finding the optimal values for these settings to improve model performance.

Common Hyperparameters

• **Learning Rate**: Determines the step size for updating parameters during training.

• **Batch Size**: Number of samples processed before updating the model.

• **Number of Epochs**: Number of complete passes through the training dataset.

• **Regularization Parameters**: Control overfitting (e.g., L1 or L2 regularization strength).

• **Model Architecture**: Number of layers, number of neurons, or activation functions.

Methods for Hyperparameter Tuning

• **Grid Search**:

• Systematically tests all combinations of hyperparameters within a predefined grid.

• Computationally expensive for high-dimensional grids. • **Random Search**

• **Bayesian Optimization**

• **Automated Hyperparameter Tuning Tools**:

• Tools like Optuna, Hyperopt, and AutoML simplify the tuning process.

**Bias-Variance Tradeoff**

• **Bias (underfitting) and Variance (Overfitting)**

• **Bias:**

• **Definition**: Error due to overly oversimplified assumptions in the model.

• **Effect**:

• Leads to underfitting.

• Fails to capture the complexity of the data.

• **Example**: A linear model trying to fit nonlinear data.

• **Solution**: Use a more flexible or complex model.

• **Variance:**

• **Definition**: Error due to the model being overly sensitive to small fluctuations in the training data.

• **Effect**:

• Leads to overfitting.

• Model captures noise and performs poorly on new data.

• **Example**: A high-degree polynomial model fitting a small dataset.

• **Solution**: Simplify the model or use regularization.

• **Tradeoff:**

• Minimize both bias and variance for optimal performance.

• The goal is to find a balance between bias and variance for optimal performance.

Cross-Validation Methods

• Cross-validation is a technique used in machine learning to assess how well a model generalizes to new, unseen data.

• The basic idea is to divide dataset into multiple subsets (called "folds") and then train and test the model on different combinations of these folds.

• Types

▪ k-Fold Cross-Validation

▪ Leave-One-Out Cross-Validation (LOOCV)

▪ Stratified K-Fold Cross-Validation

K-Fold Cross-Validation

**How it works:**

1. The entire dataset (excluding the optional test set) is divided into k equal parts (folds).

2. During training, the model trains on k-1 folds and validates on the remaining 1 fold.

3. This process is repeated k times, with each fold serving as the validation set once.

Example: k-Fold Cross Validation

Let’s say you have a dataset with 1,000 samples, and you choose **k=5 (5-fold cross-validation):**

1. Split the dataset into 5 equal parts (folds):

• Fold 1: 200 samples

• Fold 2: 200 samples

• Fold 3: 200 samples

• Fold 4: 200 samples

• Fold 5: 200 samples

2. Perform training and validation **k times**:

• **Iteration 1:** Train on Folds 2, 3, 4, 5; Validate on Fold 1.

• **Iteration 2:** Train on Folds 1, 3, 4, 5; Validate on Fold 2.

• **Iteration 3:** Train on Folds 1, 2, 4, 5; Validate on Fold 3.

• **Iteration 4:** Train on Folds 1, 2, 3, 5; Validate on Fold 4.

• **Iteration 5:** Train on Folds 1, 2, 3, 4; Validate on Fold 5.

Example: k-Fold Cross Validation (contd.)

3. Combine Results:

• After the 5 iterations, you combine the performance metrics (e.g., accuracy, F1 score) from all the validation sets to calculate the overall performance.

Leave-One-Out Cross-Validation (LOOCV)

• LOOCV is an extreme case of k-fold cross-validation where **k = the number of data points in the dataset.**

• Each data point in the dataset is treated as a separate validation set, and the remaining points are used for training.

• How It Works:

1. For a dataset with N samples (data points):

• Train the model on N−1 samples.

• Test (validate) the model on the **1 remaining sample**.

• Repeat this process N times, with each sample serving as the validation set once. 2. Compute the final performance metric by averaging the results from all N iterations.

Stratified Cross-Validation

• A variation of k-fold cross-validation where the splits are made to ensure that each fold has the same proportion of classes as the original dataset.

• This is especially useful for imbalanced datasets where one class is much more frequent than others.

**How It Works:**

1. Split the dataset into k folds, ensuring that **class distribution is preserved in each fold**.

2. Perform k-fold cross-validation as usual:

1. Train on k−1 folds, Validate on the remaining 1 fold.

2. Repeat k times, each time using a different fold for validation.

3. Compute the final performance metric by averaging results across all folds.

Advantages and limitations of cross validation.

**Advantages**

1. Provides reliable and robust model evaluation.

2. Makes efficient use of the entire dataset.

3. Reduces bias from a single train-test split.

4. Detects overfitting and helps in generalization.

5. Useful for hyperparameter tuning. 6. Works across various machine learning algorithms.

**1. Limitations**

2. Computationally expensive for large datasets.

3. Risk of overfitting in small datasets due to limited training data per fold. 4. Standard cross validation may create imbalanced splits for classification tasks.

5. Not suitable for time-series data 6. Relies on proper data splitting; poor implementation can mislead results. 7. Needs additional resources (memory, disk storage, and computing resources).

Predicting New Observations

• Prediction and inference involve using the trained model to make predictions on new, unseen data.

• **Model deployment** is the process of integrating a trained machine learning model into a production environment

• **Model Inference** involves processing new data, using the trained model to generate predictions, and interpreting those predictions.

Classification Performance Measures

• **Classification Accuracy:** The ratio of correctly predicted instances to the total number of instances.

• **Formula:**

Accuracy = Number of Correct Predictions

Total Number of Predictions

• **Limitation:**

• Accuracy can be misleading in imbalanced datasets (e.g., 95% accuracy in a dataset where 95% of instances belong to the same class doesn't indicate good performance).

Confusion Matrix

• A confusion matrix is a table that summarizes the performance of a classification algorithm. It contains the following:

| **Actual/Predicted** | **Positive** | **Negative** |
| --- | --- | --- |
| **Positive** | True Positive (TP) | False Negative (FN) |
| **Negative** | False Positive (FP) | True Negative (TN) |

• True Positive (TP): Correctly predicted positive instances.

• True Negative (TN): Correctly predicted negative instances.

• False Positive (FP): Negative instances incorrectly classified as positive. • False Negative (FN): Positive instances incorrectly classified as negative.

Derived Metrics

• **Sensitivity (Recall):** The proportion of actual positive instances correctly identified.

• Recall (Sensitivity) = ����

����+����

• Note: Useful in scenarios where missing a positive instance is costly (e.g., detecting diseases).

• **Specificity:** The proportion of actual negative instances correctly identified.

Specificity = ����

����+����

Derived Metrics (contd.)

• **Precision:** The proportion of predicted positive instances that are actually positive.

Specificity = ����

����+����

• **F-Score (or F1-Score):** The harmonic mean of precision and recall.

Specificity = 2 \* ������������������ ∗������������

������������������+������������

Example: Covid





ROC Curve and AUC

• **ROC Curve (Receiver Operating Characteristic):** • A graphical representation of a model’s performance at different threshold levels.

• Each point on the curve corresponds to a different threshold. • A perfect classifier has a point at (0, 1).

• **Axes**:



AUC (Area Under the Curve)

• Definition: The area under the ROC curve.

• Measures the ability of the model to distinguish between classes. • Higher AUC indicates better model performance. • **Range**:

• 0.5: Random classifier (no predictive power).

• 1.0: Perfect classifier.



Clustering Performance Measures

Clustering is an **unsupervised learning** task, and its performance can be evaluated using two main types of measures:

• **Internal measures**

• Evaluate clustering quality based on the **intrinsic properties** of the dataset (e.g., cohesion, separation) without using external labels. • Internal measures assess how well the clusters are formed based on their structure, such as compactness and separation.

• **External measures**

• Compare the clustering results to external ground truth labels (if available).

Internal Measure

**Silhouette Score:** Measures how similar a data point is to its own cluster (cohesion) compared to other clusters (separation).

• **Range**: [−1,1]



• s(i) ≈ 1: Point is well-clustered.

• s(i) ≈ 0: Point is near the decision boundary between clusters. • s(i) ≈ −1: Point is poorly clustered (likely in the wrong cluster).

External Measure

• **Rand Index:** Measures the similarity between the clustering result and ground truth.



Other Measures for Regression and

Classification



Other Measures for Regression and Classification (contd.)

