1. Introduction and Basics of Machine Learning

What is Machine Learning?

Machine Learning (ML) is a field of artificial intelligence that allows computers to learn patterns from data and make decisions or predictions without being explicitly programmed for specific tasks.

Example: Teaching a spam filter to classify emails as spam or not based on previous emails.

Types of Machine Learning

- Supervised Learning: Learns from labeled data (input-output pairs).
  - Example: Predicting house prices from features like size, location.
- Unsupervised Learning: Finds patterns or structure in unlabeled data.
  - Example: Grouping customers into segments based on purchase behavior.
- Semi-supervised Learning: Uses a small amount of labeled data with a large amount of unlabeled data.
- Reinforcement Learning: Learning by interacting with an environment and receiving rewards or penalties.
  - Example: Teaching a robot to navigate a maze.

Applications of Machine Learning

- Spam detection
- Image and speech recognition
- Fraud detection
- Recommendation systems (Netflix, Amazon)
- Autonomous vehicles

Machine Learning Workflow

1. Collect Data
2. Preprocess Data (cleaning, feature engineering)
3. Split Data into training and testing sets
4. Choose a Model (e.g., decision tree, neural network)
5. Train Model on training data
6. Evaluate Model on testing data
7. Tune Hyperparameters
8. Deploy and Monitor

Overfitting vs Underfitting

- Overfitting: A model learns the training data too well, including noise, so it performs poorly on new, unseen data.
- Underfitting: A model is too simple and cannot capture the underlying patterns well, performing poorly even on the training data.

Bias-Variance Tradeoff

- Bias: The error due to overly simplistic assumptions in the model (often leads to underfitting).
- Variance: The error due to a model's sensitivity to small fluctuations in the training data (often leads to overfitting).
- Goal: Find a balance between bias and variance to minimize the total error and achieve good generalization.

Train/Test Split and Cross-Validation

- Train/Test Split: Dividing the dataset into a training set (e.g., 80%) to train the model and a testing set (e.g., 20%) to evaluate its performance on unseen data.
- Cross-Validation: A more robust evaluation technique where the data is split into k folds. The model is trained and tested k times, with each fold serving as the test set once. The results are then averaged for a more reliable performance estimate.

2. Supervised Learning

What is Supervised Learning?

Supervised learning trains a model on labeled data, where each example has an input and a known output (target). The goal is to learn a function that maps inputs to inputs.

Regression

- **Linear Regression:** Predicts a continuous target variable as a weighted sum of input features.
  - **Example:** Predict house price based on size, number of bedrooms, etc.
  - **Formula:** y=w0+w1x1+w2x2+⋯+wnxn
- **Polynomial Regression:** Extends linear regression by adding polynomial terms to capture non-linear relationships.
- **Logistic Regression:** Used for binary classification (output is 0 or 1). Models the probability that an input belongs to a class using a sigmoid function.

Classification

- **k-Nearest Neighbors (k-NN):** Classifies based on the majority class of the k closest training examples in the feature space.
- **Support Vector Machines (SVM):** Finds the best boundary (hyperplane) that separates classes with maximum margin.
- **Decision Trees:** Splits data based on feature thresholds to form a tree where leaves represent class labels.
- **Random Forests:** An ensemble of decision trees trained on different subsets of data and features, voting for the final class.
- **Gradient Boosting Machines:** Builds models sequentially to correct errors of previous models (e.g., XGBoost).
- **Neural Networks:** Layers of interconnected nodes that can model complex relationships.

Performance Metrics

- **Accuracy:** Fraction of correct predictions.
- **Precision:** How many predicted positives are actually positive.
- **Recall:** How many actual positives are correctly predicted.
- **F1-score:** Harmonic mean of precision and recall.
- **ROC Curve & AUC:** Plots true positive rate vs false positive rate; AUC summarizes performance.
- **Confusion Matrix:** Table showing true positives, false positives, true negatives, false negatives.

Hyperparameter Tuning

Hyperparameters (like number of trees, max depth, learning rate) are parameters set before training.
Techniques:
- **Grid Search:** Try all combinations exhaustively.
- **Randomized Search:** Randomly sample combinations to save time.

3. Unsupervised Learning

What is Unsupervised Learning?

It finds patterns or structure in data without labeled outputs.

Clustering

- **K-means Clustering:** Assigns data points into k clusters by minimizing the distance between points and cluster centers.

- **Hierarchical Clustering:** Builds a tree of clusters by either merging (agglomerative) or splitting (divisive) clusters.
- **DBSCAN:** Density-based clustering that groups points that are closely packed and marks points in low-density areas as noise.

Dimensionality Reduction

- **Principal Component Analysis (PCA):** Projects data into fewer dimensions while preserving variance.
- **t-SNE:** Visualizes high-dimensional data by reducing dimensions while preserving local structure.
- **Autoencoders:** Neural networks trained to compress and then reconstruct data, learning efficient representations.

Association Rule Learning

- **Apriori Algorithm:** Finds frequent itemsets in data to identify association rules (e.g., market basket analysis).

4. Reinforcement Learning

Basics of RL

An agent interacts with an environment through actions, receives rewards, and learns a policy to maximize cumulative rewards.

Markov Decision Processes (MDP)

Framework defining states, actions, transition probabilities, and rewards.

Q-Learning

A value-based method where the agent learns a function Q(s,a) that estimates the expected return of taking action a in state s.

Deep Reinforcement Learning

Combines deep neural networks with RL (e.g., Deep Q-Networks) to handle high-dimensional inputs like images.

5. Deep Learning

Neural Networks Basics

Composed of layers of neurons (nodes). Each neuron receives inputs, multiplies by weights, adds bias, applies an activation function, and passes output to next layer.

Activation Functions

- **ReLU (Rectified Linear Unit):** Outputs input if positive, else zero.
  - $f(x)=max(0,x)$
  - Popular for hidden layers.
- **Sigmoid:** Outputs between 0 and 1, useful for probabilities.
  - $f(x)=1+e-x1$
- **Tanh:** Outputs between -1 and 1, zero-centered.

Feedforward Neural Networks

Information flows from input to output layer through hidden layers. Used for regression/classification on tabular data.

Backpropagation and Gradient Descent

- **Backpropagation:** Calculates gradients of loss with respect to weights.
- **Gradient Descent:** Updates weights to minimize loss.

Convolutional Neural Networks (CNNs)

Designed for images. Uses convolutional layers that apply filters to detect edges, shapes, textures. Followed by pooling layers to reduce spatial size.

Recurrent Neural Networks (RNNs), LSTM, GRU

Designed for sequential data (time series, text). RNNs have loops to maintain a state. LSTM and GRU are special units that handle long-term dependencies better by controlling information flow.

Transfer Learning

Using a pretrained model on a new but related task. Saves training time and improves performance.

Generative Adversarial Networks (GANs)

Two networks: Generator creates fake data, Discriminator tries to distinguish real vs fake. Trains both networks simultaneously to improve data generation.

6. Feature Engineering & Selection

Feature Scaling

- **Normalization:** Scale features to [0,1].

- **Standardization:** Scale features to have zero mean and unit variance.

Handling Missing Data

Techniques: Removing rows, imputing with mean/median/mode, using models to predict missing values.

Encoding Categorical Variables

- **One-Hot Encoding:** Convert categories to binary vectors.
- **Label Encoding:** Assign integer values.

Feature Selection Methods

- **Filter Methods:** Select features based on statistics (correlation, chi-square).
- **Wrapper Methods:** Use model performance to select features (recursive feature elimination).
- **Embedded Methods:** Feature selection during model training (Lasso regression).

Feature Extraction

Creating new features from raw data (e.g., PCA).

7. Model Evaluation & Validation

Cross-Validation Techniques

- **k-fold:** Split data into k parts; train on k-1 and validate on 1; repeat.
- **Stratified k-fold:** Maintains class distribution in folds.

Bias-Variance Decomposition

Helps understand error from bias and variance components.

Learning Curves

Plots of training and validation performance versus data size or epochs. Diagnose overfitting/underfitting.

Model Interpretability

Understanding how models make decisions.

Explainable AI (SHAP, LIME)

Tools that explain prediction impact of features on individual predictions.

9. Machine Learning Tools & Libraries

Scikit-learn

- **Type:** Python library for classical machine learning.
- **Focus:** Easy-to-use, efficient tools for data mining and data analysis.
- **Features:**
    - Implements popular ML algorithms: linear/logistic regression, SVM, decision trees, random forests, k-NN, clustering, PCA, etc.
    - Tools for preprocessing, model selection, hyperparameter tuning (GridSearchCV), evaluation metrics, and pipelines.
- **Use Cases:**
    - Great for beginners and prototyping classic ML models on tabular data.
    - Used for standard ML workflows that don’t require deep learning.
- **Example:**

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

TensorFlow

- **Type:** Open-source deep learning framework developed by Google.
- **Focus:** Building and training large-scale deep learning models.
- **Features:**
    - Flexible computation graphs for complex model building.
    - Supports CPUs, GPUs, and TPUs.
    - High-level APIs (Keras) built on top for easier model design.
    - TensorBoard for visualization.
- **Use Cases:**
    - Deep learning projects like image recognition, NLP, reinforcement learning.
    - Production-level deployment with TensorFlow Serving and TensorFlow Lite for mobile.
- **Example:**

```
import tensorflow as tf
model = tf.keras.Sequential([
  tf.keras.layers.Dense(128, activation='relu', input_shape=(input_dim,)),
  tf.keras.layers.Dense(10, activation='softmax')
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

```
model.fit(X_train, y_train, epochs=10)
```

PyTorch

- **Type:** Open-source deep learning framework developed by Facebook.
- **Focus:** Dynamic computation graphs, flexibility, and ease of use for research.
- **Features:**
    - Dynamic graph allows modification on-the-fly, great for debugging.
    - Strong Python integration, intuitive API.
    - TorchVision for computer vision tasks.
- **Use Cases:**
    - Research and experimentation in deep learning.
    - Rapid prototyping and complex model architectures.
    - Production-ready with TorchScript and deployment tools.
- **Example:**

```
import torch
import torch.nn as nn

class SimpleNN(nn.Module):
    def __init__(self):
        super(SimpleNN, self).__init__()
        self.fc1 = nn.Linear(input_dim, 128)
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.softmax(self.fc2(x), dim=1)
        return x

model = SimpleNN()
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters())
# Training loop would follow here
```

Keras

- **Type:** High-level neural network API written in Python.
- **Focus:** User-friendly API to build and train deep learning models.
- **Features:**
    - Runs on top of TensorFlow (mostly), Theano, or CNTK backends.
    - Simplifies model building with Sequential and Functional APIs.
    - Easy to use for beginners and prototyping.

- **Use Cases:**
  - Quick prototyping of neural networks.
  - Ideal for beginners in deep learning.
  - Production-ready since it integrates well with TensorFlow ecosystem.
- **Example:**

```
from tensorflow import keras
model = keras.Sequential([
    keras.layers.Dense(128, activation='relu', input_shape=(input_dim,)),
    keras.layers.Dense(10, activation='softmax')
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10)
```

Summary Table

Library	Best For	Strengths	Typical Use Cases
Scikit-learn	Classical ML algorithms	Easy to use, extensive algorithms	Tabular data, prototyping
TensorFlow	Deep Learning	Scalable, production-ready	Large scale DL, production
PyTorch	Deep Learning research	Flexible, dynamic graphs	Research, experimentation
Keras	Deep Learning beginners/prototyping	Simple, high-level API	Quick prototyping, education