SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V12I4P102

Volume 12 Issue 4, 12-46, April 2025
© 2025 Seventh Sense Research Group®

*Review Article*

# Machine Learning: Key Algorithms, Practical Applications, and Current Research Directions

Mohammad Nazmul Alam[1], Vijay Laxmi[2], Abhishek Sharma[3], Sarishma Dangi[4]

[1]*Department of CSE, Guru Kashi University, Talwandi Sabo, Bathinda, Punjab, India.*
[2]*Department of Computer Applications, Guru Kashi University, Talwandi Sabo, Bathinda, Punjab, India.*
[3]*Graphic Era Deemed to be University, Dehradun, Uttarakhand, India.*
[4]*Department of CSE, Graphic Era Deemed to be University, Dehradun, Uttarakhand, India.*

[1]*Corresponding Author : mnazmulalam171447@gku.ac.in*

Received: 01 February 2025 Revised: 03 March 2025 Accepted: 04 April 2025 Published: 29 April 2025

***Abstract*** *- We generate enormous amounts of data every day across various fields, such as finance, healthcare, sales, marketing, social media, and industry. State-of-the-art technology leverages this big data to make decisions and gain valuable insights. Machine learning, one of the most advanced and dynamic artificial intelligence techniques, utilizes large datasets to make predictions and develop intelligent applications. Machine learning algorithms enable computers to learn without being explicitly programmed. In this paper, we identify key algorithms and discuss fundamental algorithmic concepts. We explore various categories of machine learning, including supervised, unsupervised, semi-supervised, and reinforcement learning, along with their respective algorithms. Furthermore, we identify advanced machine learning applications across diverse fields. Finally, we discuss the challenges associated with machine learning techniques and potential future directions for developing algorithms and services.*

***Keywords*** *- Machine Learning, Reinforcement learning, Supervised, Unsupervised, Semi-supervised, Neural Network.*

## 1. Introduction

Machine learning involves teaching a machine in such a way that it can generate decisions on its own, enabling the creation of intelligent applications. Combining principles from computer science, data science, artificial intelligence, and statistics, machine learning is an interdisciplinary field that develops algorithms and systems that can recognize patterns in data and make decisions or predictions without explicit programming. It was introduced by Arthur Samuel in 1959. Samuel, a pioneering American computer scientist and AI expert, is renowned for developing early computer checker programs and contributing to the field of machine learning. According to Samuel, machine learning as “the study that enables computers to learn without being explicitly programmed [1, 2].” This groundbreaking concept laid the foundation for contemporary machine-learning techniques. So the question is how they can be taught the machine in such a way that it can generate its own decisions.

The machine is taught in basically four ways such as supervised, semi-supervised, unsupervised, and reinforcement learning. Supervised learning uses input features and target or label data features to train the model. Semi-supervised learning uses both labeled and unlabeled features of data. Reinforcement learning uses action and reward, and unsupervised learning uses unlabeled data. This is how machines learn to predict or classify the task and produce the result. Figure 1 shows the foundation of machine learning, which is capable of learning and making predictions. Figure 2 shows the machine learning-based predictive model. Firstly, it builds a predictive model using algorithms and datasets in the training phase. Secondly, new data is predicted using a predictive model in the testing phase. Figure 3 shows the steps of a machine learning-based problem-solving model. This model serves as a basic framework for machine learning, outlining its operations.

Utilizing large datasets, patterns, and algorithms significantly bolsters the machine's functionality. For optimal results and precision, selecting the appropriate algorithm for the specific task is crucial. Supervised learning excels in tasks involving classification and regression, while unsupervised learning is beneficial for clustering and dimensionality reduction. Semi-supervised learning, which operates on both labeled and unlabeled data, is particularly effective in image labeling and medical diagnosis. Reinforcement learning is good for sequential tasks such as game playing, autonomous vehicles, robotics, etc. Some real-world application examples such as speech recognition, chatGPT, face recognition, cancer detection and robotics all applications have been developed using these learning methods [3, 15]. However, the algorithm's performance and the data's characteristics and nature are correlated. Every algorithm has its own characteristics and performance nature, even though showing similar results on different data. Therefore, understanding the nature and features of the algorithm and its applicability is important to get the best performance of the algorithm. Data preprocessing and feature selection are challenging tasks for researchers and developers. A huge amount of data with high dimensionality requires dimension reduction to reduce the complexity of data.

The selection of features is very important for building a robust machine-learning model. There are many techniques used for feature selection and extraction. After selecting the features, feature extraction is done, and new features are created with lower dimensions. For example, PCA is used to do this task. Data preprocessing and feature selection are important for the model's accuracy, which is described in the following section. The application of machine learning is growing very rapidly. Almost all fields are using this technique to make advanced decisions and solutions. Based on this applicability, it is important to understand these algorithmic behaviors and which one is best suited for what. One of the goals of this paper is to understand the algorithms to implement the right algorithm for the right applications.

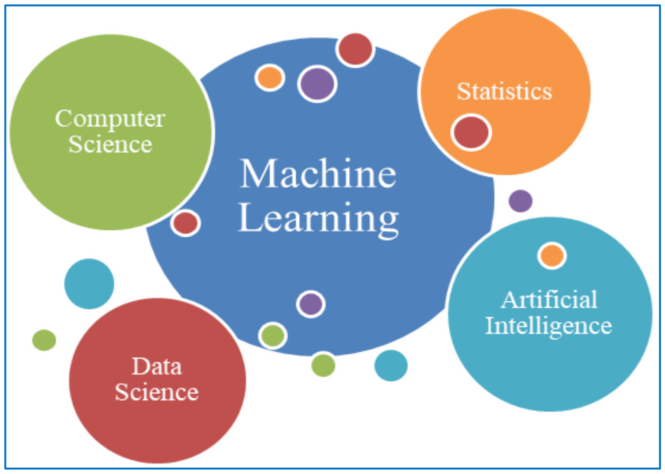


**Fig. 1 Underlying of machine learning**

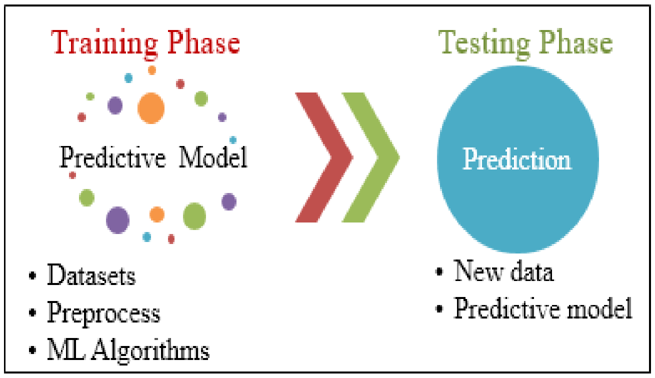


**Fig. 2 Machine learning-based predictive model**

### *1.1. Contributions*

The significant endeavor of this paper is as follows:

1. To identify all categories of algorithms and their subtypes.
2. To explain the fundamental concepts of algorithms based on their working principles.
3. To explain the pseudocode of each algorithm for implementation code.
4. To discuss the different machine learning applications in the real-world problem solution.
5. To consider the research direction for advanced data analysis and services.

### *1.2. Organization*

The rest of the paper is structured as follows: Section 2 introduces various types of machine learning algorithms, covering fundamental concepts and their classifications. Section 3 explains data preprocessing and feature selection techniques. Global trends in machine learning are discussed in Section 4.

Section 5 explores state-of-the-art applications of machine learning. Section 6 addresses the challenges in machine learning and outlines future research directions. Finally, Section 7 concludes the paper with a summary of key findings.

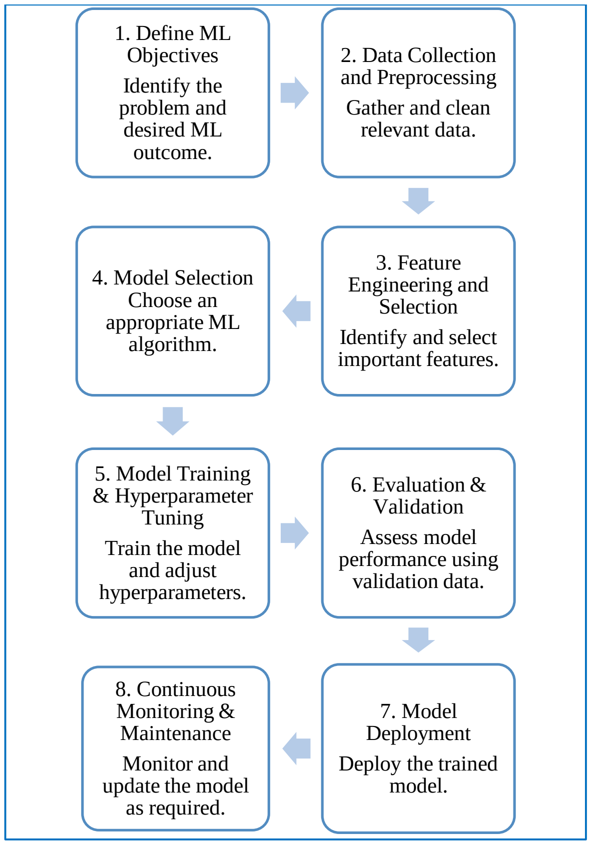

1. Define ML Objectives
Identify the problem and desired ML outcome.
2. Data Collection and Preprocessing
Gather and clean relevant data.
3. Feature Engineering and Selection
Identify and select important features.
4. Model Selection
Choose an appropriate ML algorithm.
5. Model Training & Hyperparameter Tuning
Train the model and adjust hyperparameters.
6. Evaluation & Validation
Assess model performance using validation data.
7. Model Deployment
Deploy the trained model.
8. Continuous Monitoring & Maintenance
Monitor and update the model as required.

**Fig. 3 Machine learning-based problem-solving model**

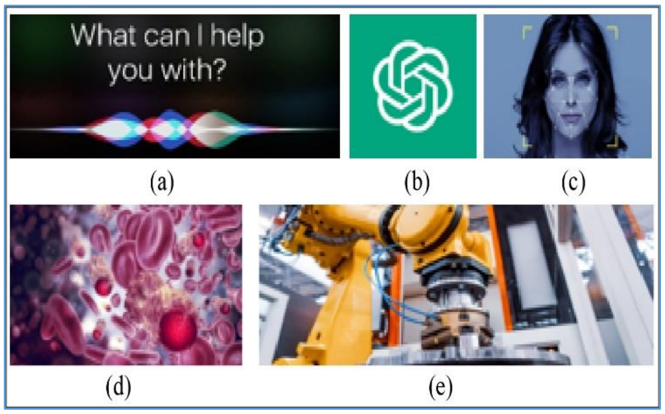


**Fig. 4 Real-world applications of machine learning, (a) Speech processing, (b) ChatGPT, (c) Face recognition, (d) Cancer detection, and (e) ML in robotics.**

## 2. Machine Learning Techniques

There are mainly four categories of machine learning techniques. Supervised, unsupervised, semi-supervised and reinforcement learning, as shown in Figure 5. In this section, we discuss each type of technique that is used in real-world applications. Figure 6 shows the list of algorithms of all categories used in real-world problem-solving.

### *2.1. Supervised Learning*

Supervised learning is a basic machine learning model that divides the learning process into two stages: training and testing [4].

Samples from training data are used as input during the training process so that the learner or learning algorithm can build the learning model. The learning model creates predictions for the test or production data during testing by utilizing the execution engine. The output of the learning model, also known as “tagged data” or “labeled data,” provides the final prediction or classified data. To train the model, input-output pairs are used, denoted as ($x_i$, $y_i$). The input in these pairs is denoted by $x_i$, and the matching intended output is denoted by $y_i$ [13].

In order to accurately predict future unseen data, the primary goal of supervised learning is to establish a mapping between input and output [5]. To evaluate the algorithm's predictive accuracy, the difference between expected and actual labels is measured using a loss function [6]. The degree of the model's departure from the true values is represented by the loss function L (yˆ, y), where y stands for the actual label and yˆ for the predicted output.

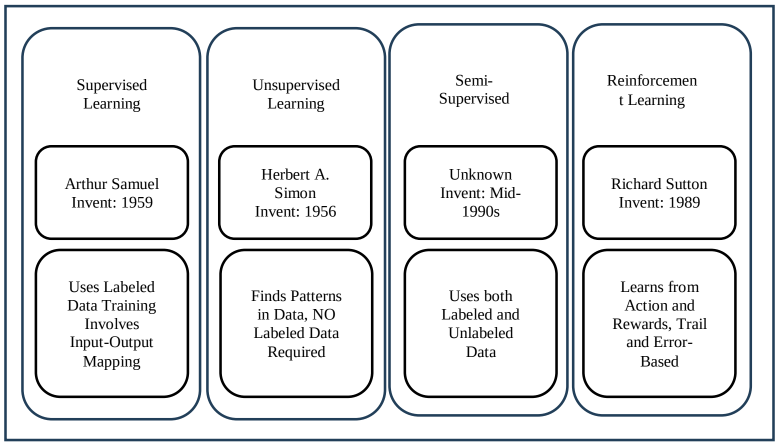
The algorithm iteratively modifies its parameters during the learning process to reduce the difference between its predictions and the actual labels [7]. When the model’s error, measured by the loss function, falls below a reasonable level, it is considered sufficiently trained to generalize and make accurate predictions for new, unseen data instances.

Figure 7 displays a schema of the SL model's environment. After obtaining the data xi as input, the learner generates an output $yˆ_i$. By computing the loss function L ($yˆ_i$, $y_i$), the output is compared to the true value yi. After that, iterations are made. SL excels at tasks involving regression and classification [8, 22, 23].

The loss function L (yˆ, y) illustrates the degree to which the model departs from the true values, where y represents the actual label and yˆ the predicted output. The algorithm iteratively adjusts its parameters throughout the learning process to minimize discrepancies between its predictions and the actual labels [7].

When the error, as indicated by the loss function, reaches a threshold that can be tolerated and the model can generalize and produce precise predictions for novel, unseen data instances, it is said to be sufficiently trained.

Figure 7 shows a schema of the SL model’s configuration. After obtaining the data xi as input, the learner produces an output $yˆ_i$. The output and the actual value of $y_i$ are compared using the loss function L ($yˆ_i$, $y_i$), which is computed. The process is then carried out once more [8, 22, 23].


Supervised Learning
Unsupervised Learning
Semi-Supervised
Reinforcement Learning
Arthur Samuel Invent: 1959
Herbert A. Simon Invent: 1956
Unknown Invent: Mid-1990s
Richard Sutton Invent: 1989
Uses Labeled Data Training Involves Input-Output Mapping
Finds Patterns in Data, NO Labeled Data Required
Uses both Labeled and Unlabeled Data
Learns from Action and Rewards, Trail and Error-Based
**Fig. 5 Primary categories of machine learning algorithm**

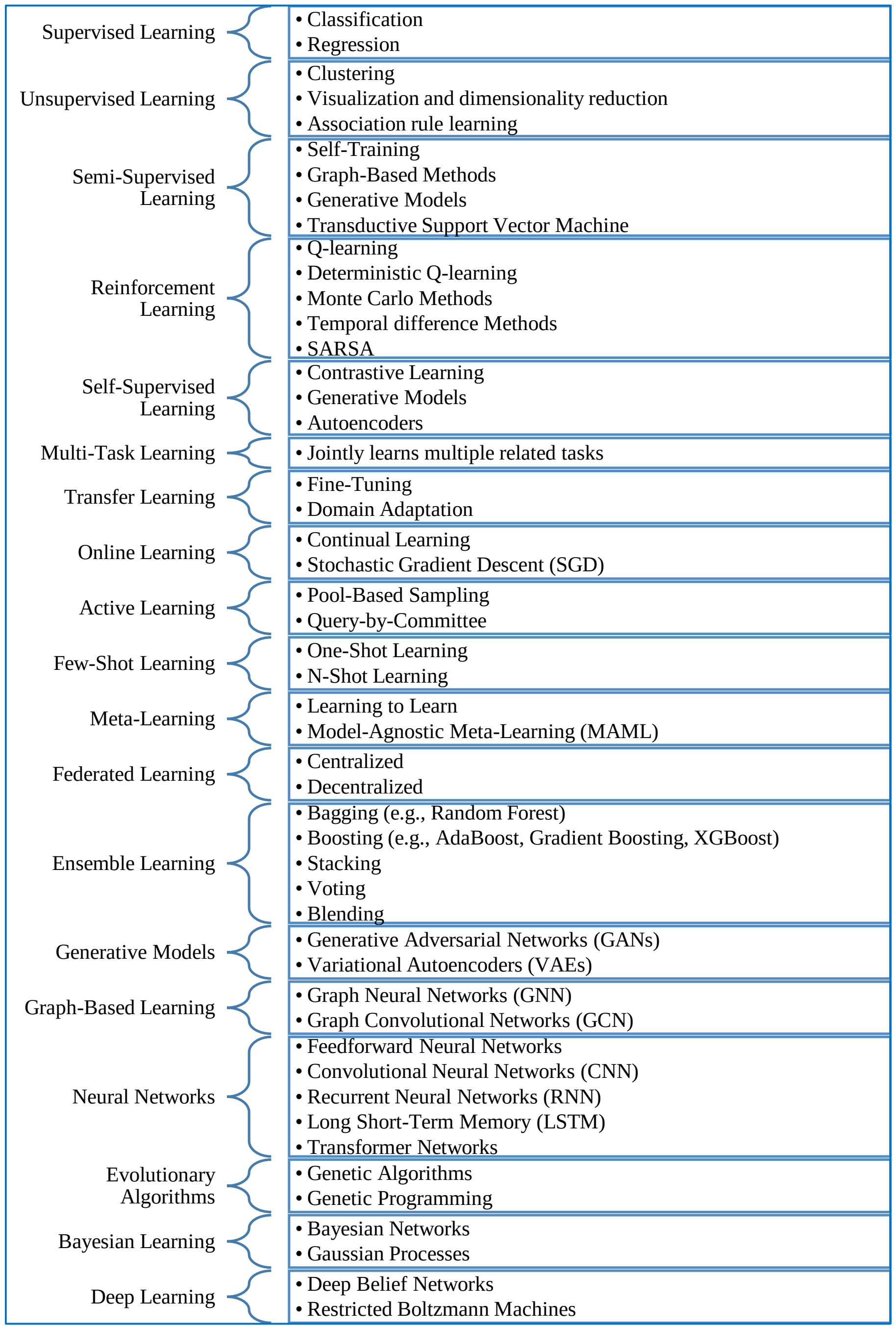


**Fig. 6 Taxonomy of machine learning techniques**

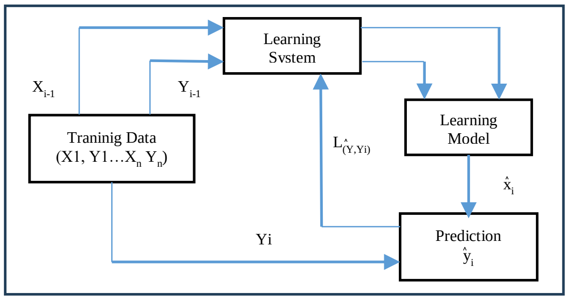


**Fig. 7 Supervised learning model**

## *2.2. Supervised Learning Algorithms and Tasks*

Supervised learning algorithm uses datasets to train the model. The datasets carry both input and correspondence output. The input is the features or attributes of the datasets, and the output is the labels or classes of the datasets [9]. Supervised learning is ideal for classification and regression analysis [10].

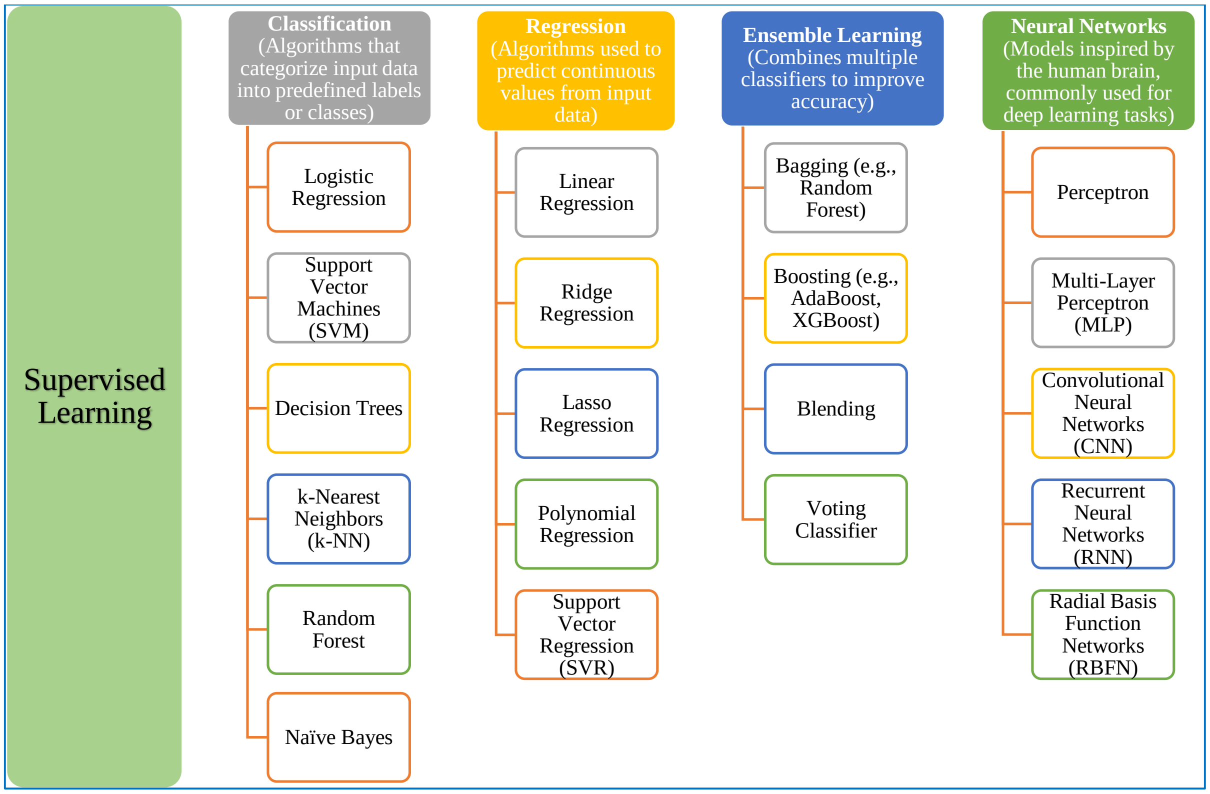

Supervised Learning
Classification (Algorithms that categorize input data into predefined labels or classes)
Logistic Regression
Support Vector Machines (SVM)
Decision Trees
k-Nearest Neighbors (k-NN)
Random Forest
Naïve Bayes
Regression (Algorithms used to predict continuous values from input data)
Linear Regression
Ridge Regression
Lasso Regression
Polynomial Regression
Support Vector Regression (SVR)
Ensemble Learning (Combines multiple classifiers to improve accuracy)
Bagging (e.g., Random Forest)
Boosting (e.g., AdaBoost, XGBoost)
Blending
Voting Classifier
Neural Networks (Models inspired by the human brain, commonly used for deep learning tasks)
Perceptron
Multi-Layer Perceptron (MLP)
Convolutional Neural Networks (CNN)
Recurrent Neural Networks (RNN)
Radial Basis Function Networks (RBFN)

**Fig. 8 Taxonomy of supervised learning based on their task and nature**

### *2.2.1. Decision Tree*

In supervised machine learning, decision trees are a widely used technique for tasks involving regression and binary classification [11-13]. It has the ability to forecast outcomes and contrast them with the real ones. The fact that decision trees are simple to understand and manage, which is often a preferred option, is one of their alluring features. A decision tree is a visual representation of the decision-making process that looks like a tree, for example, in a situation where we wish to forecast if someone will visit the beach. We can use three predictors for this purpose.

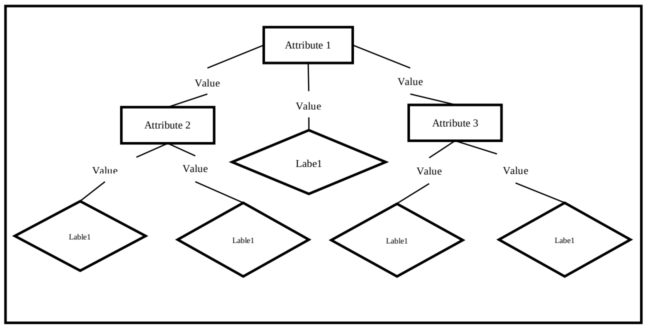


**Fig. 9 Decision tree architecture**

The first predictor, called “sky”, records the weather conditions as either cloudy, rainy, or sunny. The second predictor, “weekend” indicates whether it is a weekend or not. The third predictor, “wind speed,” captures whether the wind is high or low. Finally, we create a result variable named “Maria goes to the beach” which represents the decision. The answer will only be either yes or no because this is a binary classification problem.

Decision Tree Algorithm

| | | |
|---|---|---|
| Input | : | A training dataset S, containing feature vectors x and their corresponding class labels y. Splitting criterion, IG, Gini. |
| Output | : | A decision tree, internal nodes as features and leaf nodes as labels. |
| Step 1 | : | Start with a training data set, S, which includes features and their classifications. |
| Step 2 | : | Determine the best feature of the dataset: BestAttribute = Best feature from S based on some criteria. |
| Step 3 | : | Split S into subsets based on the possible values of the BestAttribute. |
| Step 4 | : | Create a decision tree node for BestAttribute. |

```
Step 5 : For each subset of S:
    If the subset can no longer be classified:
    - Create a leaf node with the class label.
      Else:
    - Recursively repeat steps 2-5 on the subset.
Step 6 : Return the decision tree with nodes and leaf nodes.
```

### *2.2.2. k-Nearest Neighbor*

One well-liked supervised machine learning technique for multiclass classification is k-Nearest Neighbor [3, 14]. This instance-based machine learning algorithm is also known as lazy learning. Lazy learning is demonstrated by the rote classifier, which memorizes the entire training set and only classifies a test instance if its attributes correspond to one of the training examples. Each test instance computes the distance or similarity between z = (x´, y´), and all training examples (x, y)∈ D have to determine the list of nearest neighbors from D to $D_z$. Such calculations can be expensive if the number of training examples is large. Once the list of nearest neighbors is obtained, the test instance is classified based on the majority class of its nearest neighbors.

*Majority Voting*

$$y' = \underset{v}{argmax} \sum_{(x_i,y_i)\in D_z} 1(v = y_i) \quad (1)$$

Where v is a class label, yi is a neighbor's class label, and 1(.) is an indicator function that, if its argument is true, returns 1; otherwise, it returns 0. The number of nearest neighbors in the Nearest Neighbor algorithm is k, and the training set consists of D examples. The Euclidean distance function can be used to determine similarity or distance. In the majority voting system, the classification of each neighbor has the same effect. The algorithm is, therefore, sensitive to the choice of k. To lessen the impact of k, each nearest neighbor should weigh the influence of xi based on its distance: wi=1/d (x´, xi)2. Training examples that are far from z, therefore, have less of an impact on classification than those that are close to z. A distance-weighted voting scheme is used to determine class labels.

*Distance-Weighted Voting*

$$y' = \underset{v}{argmax} \sum_{(x_i,y_i)\in D_z} w_i . 1(v = y_i) \quad (2)$$

This algorithm is used in many areas for its simplicity and effectiveness in solving classification and regression [15]. There are some common areas like pattern recognition, face recognition, handwriting recognition, image classification, recommender systems, text mining, document classification, banking and finance, anomaly detection, geographical and geospatial applications, genomics and bioinformatics, market segmentation, speech recognition have used this method effectively. What is the most likely label for c? For the solution, find k nearest neighbors of c. Then, extract the maximum label as the label of c. Let k = 3, the 3 nearest points to c are a, a, and o. Therefore, the most likely label for c is a.

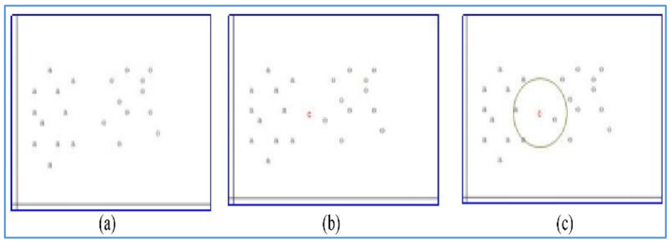


**Fig. 10 k-Nearest Neighbor example 1, (b) k-Nearest Neighbor example 2, and (c) k-Nearest Neighbor example 3.**

```
KNN Algorithm
Algorithm KNN (x_new, K, D):
Input:   - x_new: the new data point to classify or predict
         - K: the number of nearest neighbors
         - D: the training dataset containing (x_i, y_i) pairs
Output:  - y_new: the predicted class (for classification) or
predicted value (for regression)
Step 1 : Compute distances
         For each data point (x_i, y_i) in D:
             Calculate distance d (x_new, x_i) using a distance
             metric (e.g., Euclidean distance)
Step 2 : Select K nearest neighbors
         Sort the distances in ascending order
         Select the K smallest distances and their
         corresponding labels y_i
Step 3 : Make a prediction
         If the problem is classification:
           - Create a frequency count of the labels of the K
             nearest neighbors
           - Assign the label with the highest frequency as
             the predicted label y_new
         Else, if the problem is regression:
           - Compute the mean of the labels (y_i values) of
             the K nearest neighbors
           - Assign this mean value as the predicted value
             y_new
Step 4 (Optional): Apply distance-weighted voting
         If classification with distance-weighted voting:
           - For each of the K nearest neighbors, compute the
             weight as w_i = 1 / (d (x_new, x_i) + ε)
           - For each class C_k, sum the weighted votes for
             C_k
           - Assign the class with the highest weighted vote
             as y_new
      Return y_new as the prediction.
   End Algorithm
```

### *2.2.3. Support Vector Machines*

SVM is a supervised learning algorithm used for binary classification [16]. In this algorithm, the support vector is a data point that resides on the margin and is used to determine the boundary and classify the new data. Margins are the distance between the support vectors and hyperplane in each class. In SVMs, a large margin is better than a small margin. The decision boundary or hyperplane, also called the separation line, separates the data points into two classes. SVM seeks to determine which hyperplane in a multi-dimensional space best divides the data points into distinct classes [17, 18]. SVM algorithms are given below.

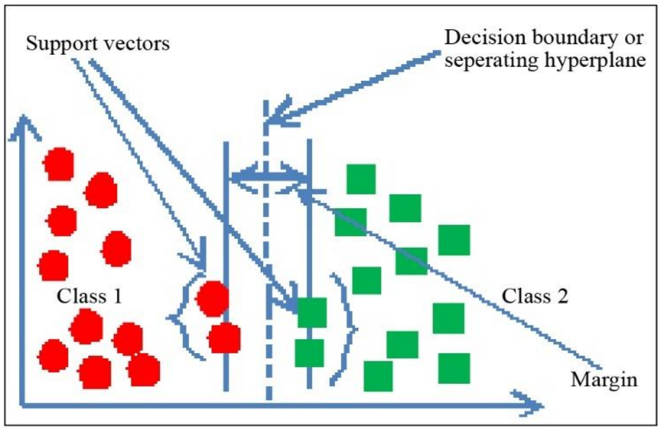


**Fig. 11 Support vector machines**

```
Support Vector Machines (SVMs) Algorithm
Algorithm SVM (X, Y, C, kernel):
Input:   - X: training data points (features)
           - Y: labels for the training data
           - C: regularization parameter
           - kernel: the kernel function (e.g., linear, polynomial,
             RBF)
Output:  - w, b: weights and bias for the separating hyperplane
Step 1 :  Define the optimization problem
           - Minimize the objective function (hinge loss +
regularization term)
           - Subject to the constraint that each data point is
classified correctly with a margin
Step 2 :  Solve the quadratic optimization problem
           - Use a solver (e.g., Sequential Minimal
Optimization) to find the optimal w and b
Step 3 :  Make prediction for a new data point x_new
          - Compute the decision function f(x_new) = w^T *
            x_new + b
           - If f(x_new) >= 0, assign the label y_new = +1
           - Else, assign the label y_new = -1
        Return y_new as the predicted class
    End Algorithm
```

### *2.2.4. Linear Regression Algorithm*

Linear regression is a technique for predictive modeling used in statistics and machine learning. It is one of the best statistical techniques widely used in various fields for modeling the relationship between variables. This technique is used in engineering, management, biological science, social science, chemical science, economics and many more. The common linear regression models are simple linear regression, which uses exactly one regressor or predictor, and multiple linear regression, which uses more than one predictor [19-21].

In simple linear regression, the equation is expressed as:

$$y = \beta 0 + \beta 1x + \varepsilon \qquad (3)$$

Where $y$ is the dependent variable, $x$ as the independent variable, $\beta_0$ denoted as intercept, $\beta 1$ denoted as slop and constant, and $\varepsilon$ represents the random error term.

And multiple linear regression, which contains multiple regressors such as $x_{1,}$ $x_{2}$ , $x_{3}$ …$x_{k;}$ therefore, the equation is expressed as:

$$y = \beta 0 + \beta 1x1 + \beta 2x2 + \ldots + \beta kxk + \varepsilon \qquad (4)$$

The residual is a crucial statistical tool in regression analysis used to evaluate how well the model fits the data. It can be applied to increase the model's accuracy.

The residuals, or the vertical separations between the actual data points and the predicted values on the regression line, are shown in Figure 10 by the green dashed lines.

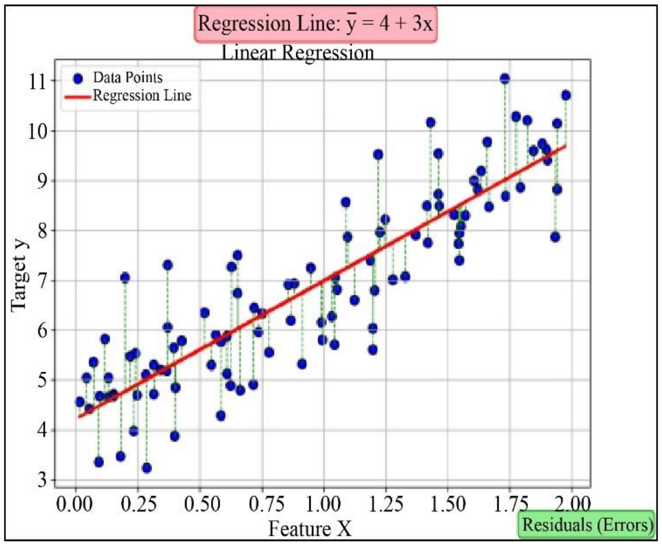


**Fig. 12 Linear regression with residual error**

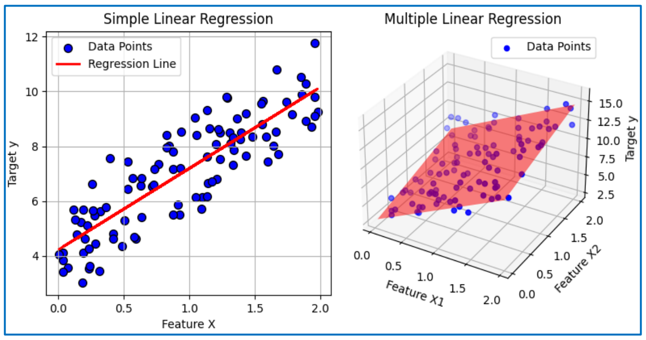


**Fig. 13 The Simple linear regression model uses a single feature and the target variable, and the multiple regression model uses two features and the target variable**

```
Linear Regression Algorithm
Algorithm LinearRegression(X, Y):
Input:    - X: training data points (features)
          - Y: corresponding labels (real values)
Output: - w, b: the parameters for the linear regression model

Step 1 :  Define the linear model
          - y_pred = w^T * X + b
Step 2 :  Define the cost function
          - Cost function = Mean Squared Error (MSE) between
            y_pred and Y
Step 3 :  Minimize the cost function
          - Use gradient descent or closed-form solution to
            compute w and b that minimize the MSE
Step 4 :  Make prediction for a new data point x_new
          - Compute y_new = w^T * x_new + b
    Return y_new as the predicted value
    End Algorithm
```

*2.2.5. Random Forest Algorithm*

Multiple decision trees are used in the Random Forest algorithm, an ensemble learning technique, to make predictions. It creates multiple subsets by randomly sampling data with replacement using bootstrapping techniques [24, 25]. A statistical technique called bootstrap allows the creation of multiple samples from a single dataset. A subset of random features is used to train each decision tree in the forest. Then, decision trees are built for every subset by splitting the nodes followed by feature criteria. Finally, aggregate the votes of each tree and select the majority voting class for classification and averages of all trees for regression. The architecture of a random forest tree is given in Figure 14.

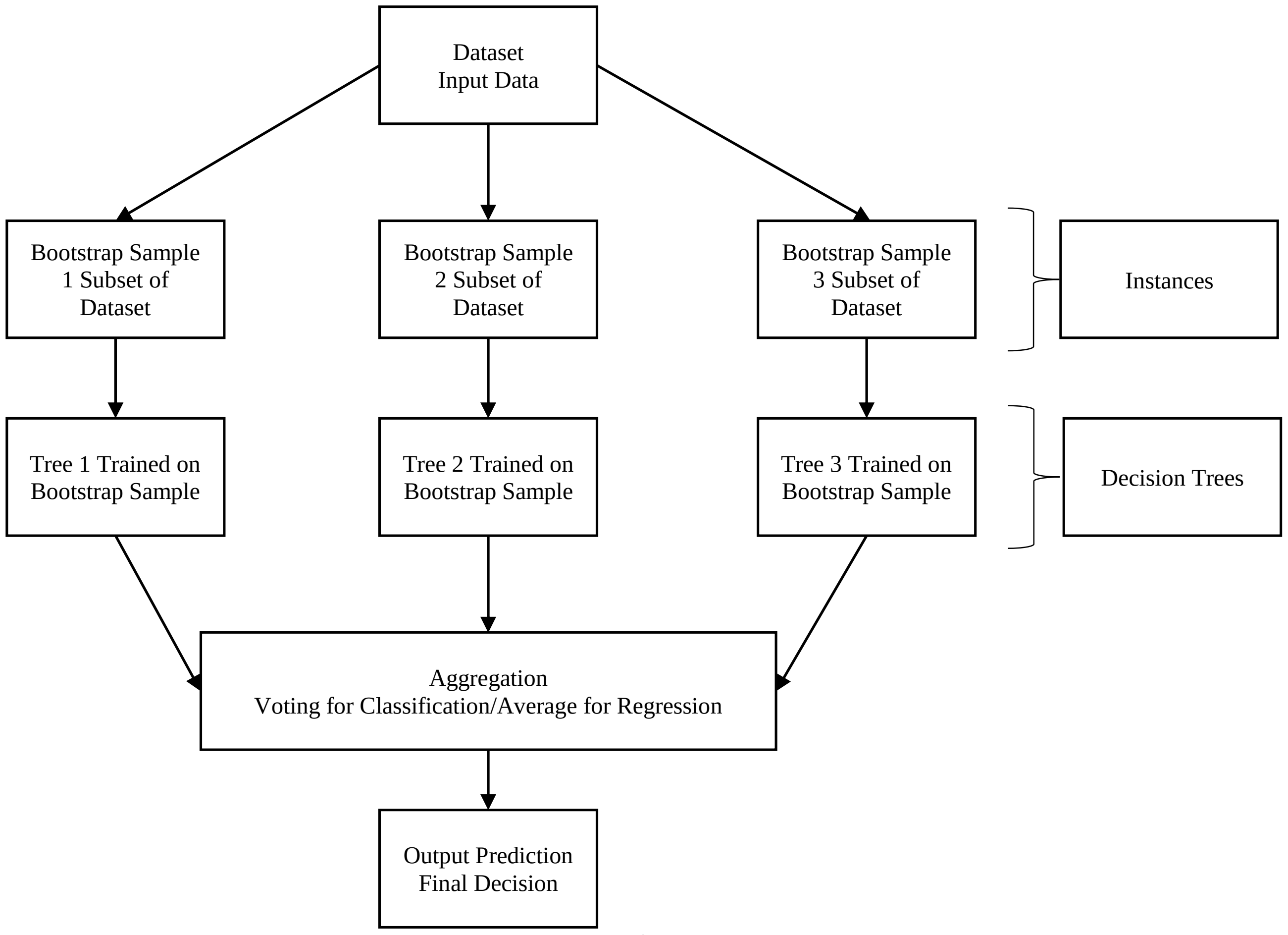

Dataset
Input Data
Bootstrap Sample 1 Subset of Dataset
Bootstrap Sample 2 Subset of Dataset
Bootstrap Sample 3 Subset of Dataset
Instances
Tree 1 Trained on Bootstrap Sample
Tree 2 Trained on Bootstrap Sample
Tree 3 Trained on Bootstrap Sample
Decision Trees
Aggregation
Voting for Classification/Average for Regression
Output Prediction
Final Decision

**Fig. 14 Random forest architecture**

```
Random Forest Algorithm
Algorithm RandomForest (X, Y, num_trees):
Input:   - X: training data points (features)
         - Y: corresponding labels (classification or regression)
         - num_trees: the number of decision trees to create
Output:  - Forest: a collection of decision trees
Step 1 :  Create num_trees decision trees
          For each tree:
              - Select a random subset of the data points from X
(bootstrap sampling)
```

```
            - Randomly select a subset of features to split the data at each node
            - Build a decision tree by recursively splitting the data based on the selected features

Step 2 : Make prediction for a new data point x_new
        - For classification:
            - Let each decision tree in the forest predict the class of x_new
            - Use majority voting to assign the final predicted class y_new
        - For regression:
            - Let each tree predict a value for x_new
            - Compute the average of all predicted values as y_new
    Return y_new as the final prediction
  End Algorithm
```

### *2.3. Unsupervised Learning*

This is known as unsupervised learning because, unlike supervised learning, there are no supervisors or correct answers. Algorithms use their own creativity in identifying and presenting the interesting structure of the data.

Algorithms for unsupervised learning learn very few features from the data. When new data is added, its class is determined using the previously learned features. Clustering and feature reduction are its main uses. We have shown the taxonomy of unsupervised learning in Figure 17.

### *2.4. Unsupervised Learning Algorithms and Tasks*

Unsupervised learning algorithms identify hidden patterns and structures in unlabeled data, making them essential for exploratory data analysis, anomaly detection, and clustering. Algorithms such as K-Means, GMMs, and Hierarchical Clustering partition data into meaningful clusters, while PCA and t-SNE perform dimensionality reduction to enhance feature representation.

Advanced techniques like Autoencoders and Generative Adversarial Networks (GANs) leverage deep learning to encode latent structures and generate synthetic data. These algorithms power diverse applications, including fraud detection, customer segmentation, and feature learning in complex, high-dimensional datasets.

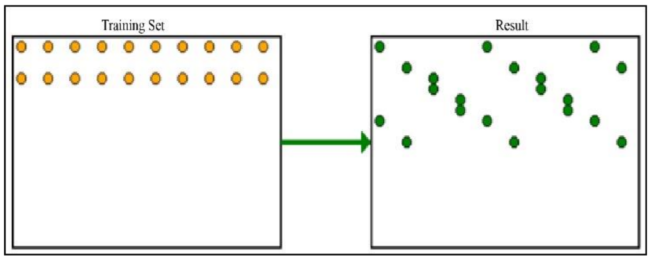


**Fig. 15 Unsupervised learning**

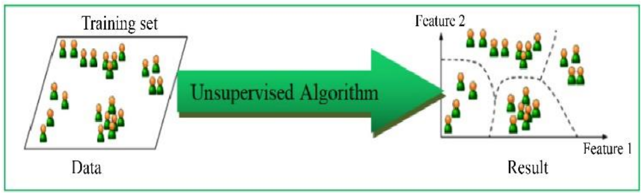


**Fig. 16 Unsupervised algorithm (clustering)**

| | |
|---|---|
| Clustering Analysis | • k-Means Clustering<br>• Hierarchical Clustering<br>• Approximate Nearest Neighbor Search<br>• Gaussian Mixture Models<br>• Hierarchical Agglomerative Clustering |
| Dimensionality Reduction | • Principal Component Analysis<br>• Autoencoders<br>• t-SNE<br>• Self-Organizing Maps |
| Anomaly Detection | • Isolation Forest<br>• One-Class SVM<br>• Local Outlier Factor |
| Feature Embedding | • t-Distributed Stochastic Neighbor Embedding<br>• Multidimensional Scaling<br>• Uniform Manifold Approximation and Projection<br>• Isomap Embedding<br>• Locally Linear Embedding (LLE)<br>• Manifold Learning |
| Pattern Discovery | • Frequent Pattern Mining<br>• Sequence Pattern Mining<br>• Graph Pattern Mining |
| Text Analysis | • Hierarchical Topic Modeling<br>• Word Embedding<br>• Latent Dirichlet Allocation<br>• Language Modeling<br>• Text Summarization |

**Fig. 17 Taxonomy of unsupervised machine learning algorithms based on their tasks**

#### *2.4.1. k-Means Clustering*

According to Lloyd (1957) and McQueen (1967), k-Means is a partitional clustering program. K-Means clustering is used to divide the given data into k clusters. Every cluster has a centroid, which is its central point. Convergence occurs when a cluster ceases to evolve [26, 27]. The k-Means algorithm is used when the user inputs “k.”. The algorithm below explains how to apply k-Means to datasets for clustering. The provided equation is used to determine the Euclidean distance.

$$[(x, y), (a, b)] \equiv \sqrt{(x-a)^2 + (y-b)^2} \tag{5}$$

```
k-Means Algorithm
Algorithm KMeans (X, K, max_iterations):
Input:    - X: dataset containing n data points
          - K: the number of clusters
          - max_iterations: the maximum number of iterations to perform
Output:   - Centroids: The K cluster centroids
          - Cluster assignments: a label for each data point indicating its cluster
Step 1 :  Initialize K centroids randomly from the dataset.
Step 2 :  Repeat until convergence or until max_iterations is reached:
              - For each data point x_i in X:
                  - Assign x_i to the nearest centroid based on distance (e.g., Euclidean).
              - For each cluster:
                  - Recalculate the centroid as the mean of all data points assigned to that cluster.
Step 3 :  Return the final centroids and cluster assignments.
    End Algorithm
```

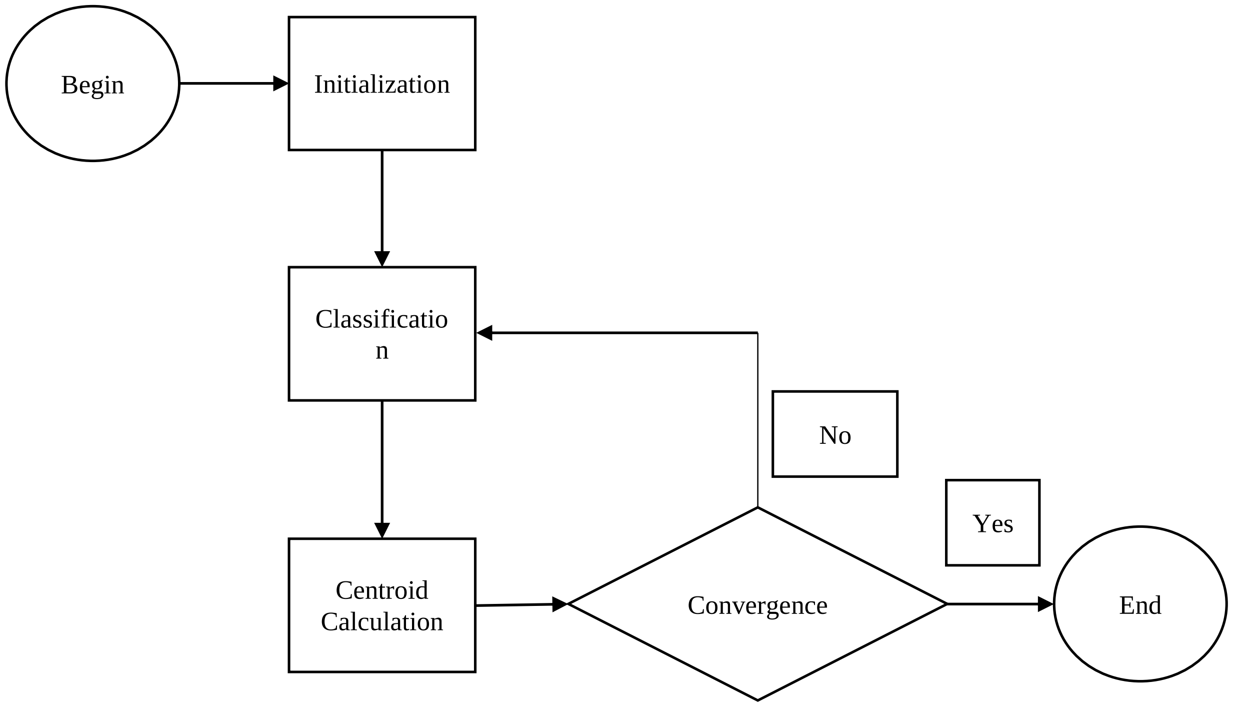


**Fig. 18 Standard k-means algorithm**

### *2.4.2. PCA*

Dimensionality reduction, principal components, eigenvectors and eigenvalues are the key concepts of PCA. By employing principal components to preserve as much variability as possible, PCA minimizes the number of variables in a dataset. Regarding how much variance they capture, principal components are the most effective and arranged orthogonally [13, 28].

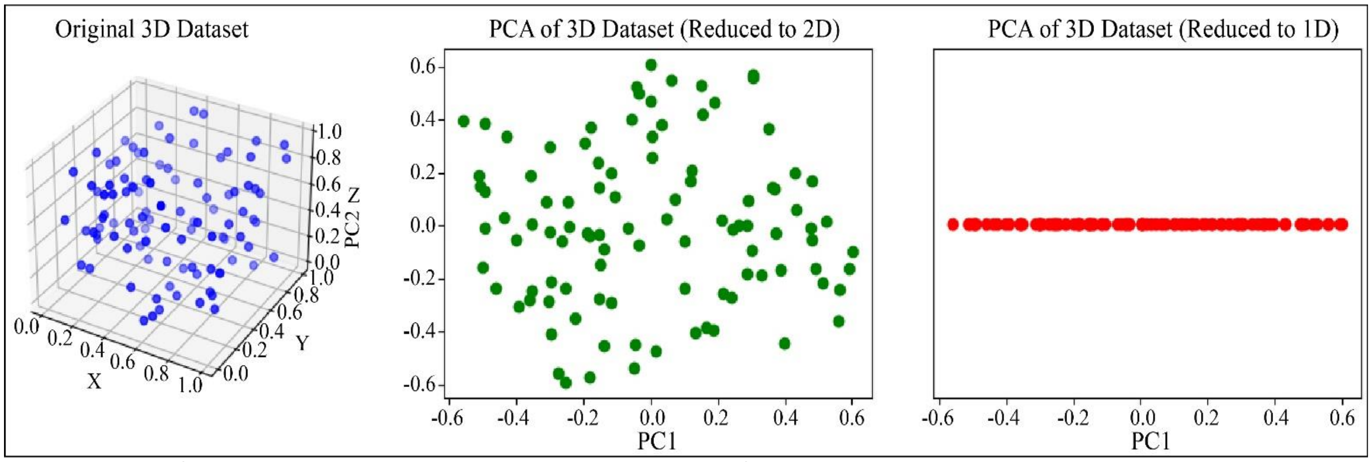


**Fig. 19 Example of PCA original 3D dataset reduced to 2D and 1D, (a) Original features in 3D space, (b) PCA in 2D space, and (c) PCA in 1D space.**

The algorithm that follows explains how PCA operates. Standardizing the data and calculating the mean and deviation to scale the data to contribute equally are the first steps in using the dataset in PCA. Compute the covariance matrix next to display the relationship between the various variables. Subsequently, determine the covariance matrix's eigenvalues and eigenvectors to determine the variance magnitudes in each direction and the directions through eigenvectors [29].

When selecting the principal components, the higher eigenvalue determines which eigenvectors are the top choices. The more variance that component explains, the higher the eigenvalue. Finally, the data is transformed into new dimensional space formed by the selected principle components, resulting in a reduced dataset. Figure 19 describes PCA's pictorially 3D, 2D, and 1D space. This plot shows the original data in three dimensions (X, Y, Z). The 3D dataset was reduced to a 2D dataset after reducing its dimensionality from 3D to 2D using PCA. The new components are PC1 and PC2. PCA of the 3D Dataset Reduced to 1D, and this plot shows the dataset after reducing its dimensionality from 3D to 1D using PCA. Only the First Principal Component (PC1).

```
PCA Algorithm
Algorithm PCA (X, n_components):
Input:  - X: dataset with n samples and m features
        - n_components: the number of principal components
          to retain
Output: - Transformed data: X projected onto the principal
          components
        - Principal components: the directions of maximum
          variance
Step 1 : Normalize the dataset X by subtracting the mean of
         each feature.
Step 2 : Compute the covariance matrix of the normalized
         dataset.
Step 3 : Compute the eigenvalues and eigenvectors of the
         covariance matrix.
Step 4 : Select the top n_components eigenvectors
         corresponding to the largest eigenvalues.
Step 5 : Project the dataset X onto the selected
         eigenvectors.
Step 6 : Return the transformed data and the principal
         components.
         End Algorithm
```

*2.4.3. Autoencoder*

The two main components of an autoencoder are an encoder and a decoder. While the encoder's task is to compress the input data while minimizing information loss, the decoder's task is to reconstruct the original data using the compressed representation. Autoencoders are used extensively in various fields, including data compression, anomaly detection, dimensionality reduction, image denoising, feature extraction, and image colorization [30].

The input layer, encoder hidden layer, latent space (compressed data), decoder hidden layer, and output layer are the five main components of a basic autoencoder architecture. Notably, in order to ensure compression and prevent simple data replication, the number of neurons in the output layer must match that in the input layer. Mathematically, the process can be described as:

Where the input vector is denoted by x, the encoder function by f(x), the compressed representation by z, the decoder function by g(z), and the reconstructed output by x′. A more sophisticated variation, deep autoencoders, contains several hidden layers in both the encoder and the decoder. These layers must mirror each other, with the same number of neurons on both sides. Different types of autoencoders are described here.

$$\text{Input } x \to \underset{\text{Encoder}}{\overbrace{f(x) \to}} z \to \underset{\text{Decoder}}{\overbrace{g(z) \to}} x' \text{ Reconstructed output}$$

**Fig. 20 Autoencoder**

```
Autoencoder Algorithm
Algorithm Autoencoder (X, encoder_layers, decoder_layers,
learning_rate, epochs):
Input:
    - X: dataset with n samples
    - encoder_layers: architecture of the encoder (e.g.,
      number of hidden layers)
    - decoder_layers: architecture of the decoder
    - learning_rate: the learning rate for optimization
    - epochs: number of training epochs
Output:
    - Compressed representation (latent space)
    - Reconstructed data
Step 1 : Initialize the encoder and decoder networks with the
         given layers.
Step 2 : Perform forward propagation:
         - Pass the input X through the encoder to get the
           compressed representation.
         - Pass the compressed representation through the
           decoder to reconstruct X.
Step 3 : Compute the reconstruction loss (e.g., Mean
         Squared Error).
Step 4 : Perform backpropagation to update weights using
         gradient descent.
Step 5 : Repeat steps 2-4 for the specified number of epochs.
Step 6 : Return the compressed representation and
         reconstructed data.
    End Algorithm
```

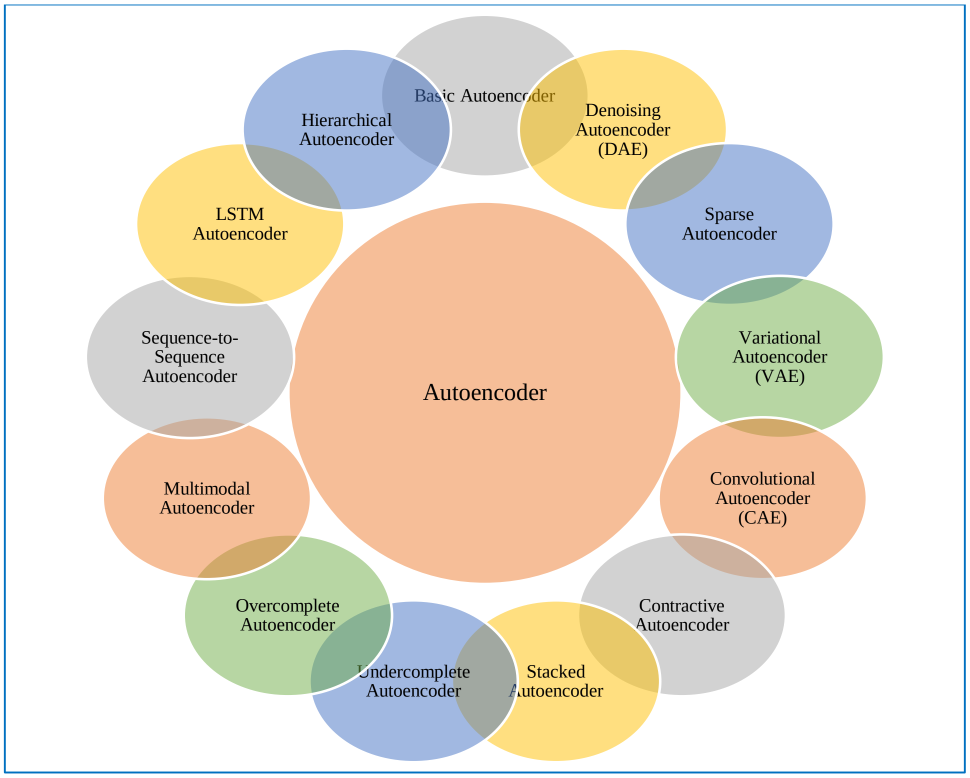


**Fig. 21 Various autoencoders**

**Table 1. Various autoencoders tasks and characteristics**

| Method | Task | Characteristics | References |
|---|---|---|---|
| Basic Autoencoder | Compress and reconstruct data | Simple architecture with encoder and decoder. Used for dimensionality reduction and feature learning. | [30] |
| Denoising Autoencoder (DAE) | Remove noise from input data | Trained with corrupted input and learns to recover the original clean data. Useful for noise reduction and data preprocessing. | [31] |
| Sparse Autoencoder | Feature extraction | Encourages sparsity in hidden layers, where only a small subset of neurons activates. Typically used for feature extraction and learning sparse representations. | [32] |
| Variational Autoencoder (VAE) | Generate new data samples | Assumes latent variables follow a Gaussian distribution. A generative model is used for sampling new data similar to the training set, enabling probabilistic modeling. | [33] |
| Convolutional Autoencoder (CAE) | Capture spatial hierarchies in image data | Incorporates convolutional layers, making it suitable for image data. Effective for capturing spatial dependencies in images. | [34] |
| Contractive Autoencoder | Enforce robustness to small changes in input | Adds a penalty term to the loss function to minimize the derivative of the hidden representations. Promotes robustness to small input variations. | [35] |
| Stacked Autoencoder | Create deep networks for complex data representations | Multiple layers of autoencoders are stacked. Each layer's output is the input to the next, enabling the learning of more complex data representations. | [36] |
| Undercomplete Autoencoder | Compress data into a smaller representation | The latent space is smaller than the input, enforcing data compression. Used for efficient data representation. | [37] |
| Overcomplete Autoencoder | Capture complex features | Has more hidden units than input dimensions. Regularization is required to avoid learning the identity function and ensure meaningful feature extraction. | [38] |

| | | | |
|---|---|---|---|
| Multimodal Autoencoder | Combine different modalities of data (e.g., image, text) | Trained on multiple modalities to create a shared latent space. Useful for cross-domain data representation. | [39] |
| Sequence-to-Sequence Autoencoder | Model sequential data like time series or text | The input and output are sequences. Used for tasks like time series prediction, machine translation, and other sequence-based tasks. | [40] |
| LSTM Autoencoder | Capture temporal dependencies in sequential data | Uses LSTM units to handle time dependencies. Often applied in time series data or sequential modeling tasks. | [41] |
| Hierarchical Autoencoder | Extract information at different levels of granularity | Input is broken down into hierarchical layers, allowing for multilevel data representation. Effective for hierarchical data structures. | [42] |

### *2.4.4. Isolation Forest*

To isolate observations, Isolation Forest randomly selects a feature and then randomly selects a split value between the feature's minimum and maximum values. This makes the algorithm especially useful for anomaly detection. Here's an overview of how the algorithm works:

- Isolation Trees: The algorithm constructs an ensemble of isolation trees for a given dataset. Each tree is created by recursively splitting the data points until each point is isolated in a leaf node.
- Path Length: The algorithm determines the path length for each data point, which is the separation between the root node and the leaf node in each tree where the point ends.
- Anomaly Score: The average path length across all trees is the anomaly score for a given data point. Due to their rapid isolation, points with shorter average path lengths are regarded as anomalies. The anomaly score s (x, n) for a data point x in a forest of n trees is computed as follows.

$$s(x,n) = 2^{-\frac{E(h(x)}{c(n)}} \quad (6)$$

Where E(h(x) is the average path length of *x* and *c(n)* is the average path length of unsuccessful searches in a Binary Search Tree, defined as:

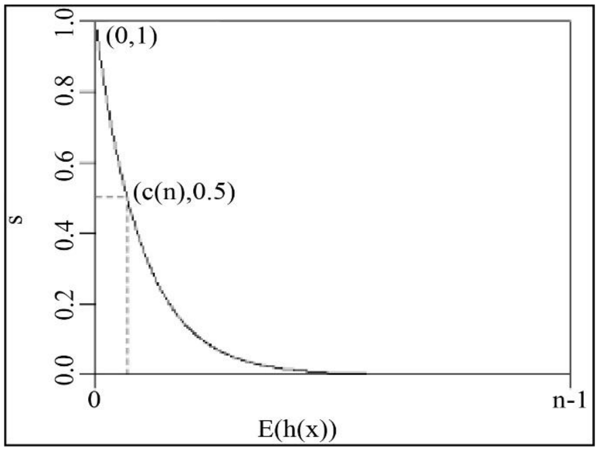


**Fig. 22 Illustrates the relationship between the expected path length, E(h(x)), and the anomaly score, s**

Here, c(n) represents the average path length defined in Equation 7 [43].

$$c(n) = 2H(n-1) - (2(n-1)/n) \quad (7)$$

With H(i) being the i-th harmonic number, we can calculate it using the following:

$$H(i) = ln(i) + \gamma \ (Euler's\ constant\ \gamma \approx 0.577) \quad (8)$$

In Equation (6):

- When E(h(x)) → c(n), s → 0.5;
- When E(h(x)) → 0, s → 1; and
- When E(h(x)) → n − 1, s → 0.

When the expected path length E(h(x)) matches the average path length c(n), the anomaly score s is equal to 0.5, regardless of the value of n. s is monotonic to h(x). Figure 22 illustrates the relationship between E(h(x)) and s, and the following conditions are applied where $0 < s \leq 1$ for $0 < h(x) \leq n - 1$. Using the anomaly scores, we are able to make the following assessment:

- If instances return s very close to 1, then they are definitely anomalies,
- If instances have a much smaller than 0.5, then they are quite safe to be regarded as normal instances and
- If all the instances return ≈ 0.5, then the entire sample does not really have any distinct anomaly.

```
Isolation Forest algorithm
Algorithm Isolation_Forest (X, n_trees, max_samples)
Input:
    X: Dataset with n samples
    n_trees: Number of isolation trees to create
    max_samples: Maximum number of samples to use in each tree
Output:
    Anomaly scores for each data point in X
Step 1 : Initialize a list to store isolation trees
    trees ← []
```

```
Step 2 : Build n_trees isolation trees
    for i = 1 to n_trees do
        // Randomly select max_samples from X
        sample_X ← Random_Sample (X, max_samples)
        // Build an isolation tree using sample_X
        tree ← Build_Tree(sample_X)
        // Add the tree to the list of trees
        trees. append (tree)
    end for
Step 3 : Compute the path length for each data point in X
    path_lengths ← []
    for each data_point in X do
        total_path_length ← 0
        for each tree in trees do
            path_length←Calculate_Path_Length(tree, data_point)
            total_path_length←total_path_length+path_length
        end for
        // Average path length across all trees
        avg_path_length ← total_path_length / n_trees
        path_lengths.append (avg_path_length)
    end for
Step 4 : Calculate the anomaly score for each data point
    anomaly_scores ← []
    for each avg_path_length in path_lengths do
        score← Calculate_Anomaly_Score(avg_path_length)
        anomaly_scores. Append (score)
    end for
Step 5 : Return anomaly_scores
End Algorithm
```

### *2.4.5. One Class SVM*

The One-Class Support Vector Machine (OCSVM) is a specialized type of support vector machine used for anomaly detection. It learns a decision function that can classify new data as either similar to or different from the training data, making it suitable for novelty detection [44, 45]. The following is the working procedure:

- Kernel Function: In OCSVM, the kernel function K(x,y) plays a crucial role by mapping the input data into a higher-dimensional space. This transformation makes separating normal data from anomalies easier by defining a clear boundary. Commonly used kernels include the Radial Basis Function (RBF), polynomial, and linear kernels, which help capture complex patterns in the data.
- Decision Function: $f(x) = w.\phi(x) - \rho$ is the definition of the decision function, where w denotes the weight vector, ρ the offset, and ϕ(x) is the feature map.
- Optimization Issue: The issue with OCSVM optimization is:

$$min_{\omega,\rho,\xi_i \frac{1}{2}}||w||^2 + \frac{1}{\nu l}\sum_{i=1}^{l}\xi_i - \rho \tag{9}$$

Subject to: $(w.\phi(x_i)) \geq \rho - \xi i,\ \xi_i \geq 0,\ i=1,\ldots,l$

Where ξi are slack variables that deal with data points that are misclassified and fall inside the margin, and $\nu$ is a parameter that controls the trade-off between maximizing the margin and minimizing the number of anomalies.

- Kernel trick: Finding the best boundary in a high-dimensional space to distinguish between normal and outlier data points allows for detecting anomalies.

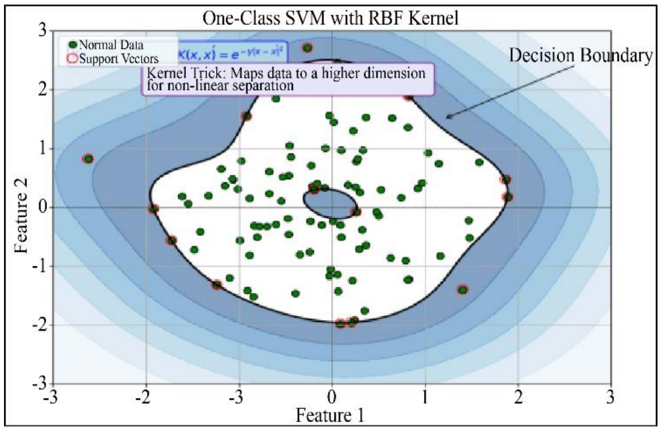


**Fig. 23 OC-SVM, plotting the decision boundary, normal data points and support vectors**

```
One Class-SVM algorithm
Algorithm One Class SVM (X, kernel, nu):
Input:
    - X: dataset with n samples
    - kernel: kernel function (e.g., linear, RBF)
    - nu: an upper bound on the fraction of anomalies
Output:
    - Anomaly score for each data point
Step 1 : Train the SVM on the dataset X using the specified
         kernel and nu.
Step 2 : For each data point x_i in X:
    - Compute the decision function value f(x_i).
    - If f(x_i) < 0, classify x_i as an outlier.
Step 3 : Return the anomaly score for each data point based
         on the decision function.
End Algorithm
```

### *2.4.6. Local Outlier Factor*

The LOF algorithm measures the local density deviation of a data point relative to its neighbors. By comparing a point's density to that of its surrounding neighbors, LOF can identify points that have a significantly lower density than their neighbors as potential anomalies [46]. The following is the working procedure.

- k-Distance: For every point p, the distance to its k-th nearest neighbor is computed
- Reachability Distance: The reachability distance between a point p and a point o is determined by:
- Reach-dist $_k$ (p,o) = max{k-distance (o), d(p,o)}

- Local Reachability Density(LRD): The inverse of the average reachability distance determined by the k-nearest neighbors is the local reachability density of point p.

$$lrd_k(p) = \left(\frac{\sum_{o \in N_{k(p)}} reach\text{-}dist_k(p,o)}{|N_k(p)|}\right)^{-1} \tag{10}$$

LOF Score: A point's local reachability density is divided by the local reachability density of its k-nearest neighbors to determine its average LOF score, which is:

$$LOF_k(p) = \left(\frac{\sum_{o \in N_{k(p)}} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|}\right) \tag{11}$$

- Anomalies: To find anomalies, LOF calculates a data point's local density deviation from its neighbors.

```
Local Outlier Factor Algorithm
Algorithm LOF (data, k):
Input:
    data: A set of data points
    k: The number of nearest neighbors
Output:
    LOF scores for each data point
Step 1 : For each point p in data:
    1.1. Calculate the k-distance of p
        k-distance(p) = distance to the k-th nearest neighbor of p
Step 2 : For each point p in data:
    2.1. Initialize a list reachability_distances
    2.2. For each point o in the k-nearest neighbors of p:
        2.2.1. Calculate the reachability distance
            Reach-distk(p, o) = max(k-distance(o), distance(p, o))
        2.2.2. Append Reach-distk(p, o) to reachability_distances
    2.3. Calculate the local reachability density (LRD) of p
        LRD(p) = 1 / (average of reachability_distances)
Step 3 : For each point p in data:
    3.1. Initialize sum_LRD_neighbors = 0
    3.2. For each point o in the k-nearest neighbors of p:
        3.2.1. Calculate the local reachability density of neighbor o
            LRD(o) = 1 / (average of reachability distances to k-neighbors of o)
        3.2.2. sum_LRD_neighbors += LRD(o)
    3.3. Calculate the LOF score for p
        LOF(p) = (sum_LRD_neighbors / k) / LRD(p)
Step 4 : Return the LOF scores for all points in the data
```

### *2.4.7. t-SNE Algorithm*

t-SNE algorithm is used to reduce high-dimensional data to a lower-dimensional space for visualization while preserving relationships between data points [47]. There are several advantages of t-SNE over PCA, such as working with similar local data points close to each other in the lower dimensional space rather than global variance. It can handle non-linear data, whereas PCA uses a linear method. It is good for complex and high-dimensional data. It can create better visualization than PCA because it can visualize in 2D or 3D by grouping similar points together while separating dissimilar points, but PCA spreads the variances across the components linearly. Another important thing is that t-SNE has a parameter called perplexity that allows the user to balance between local and global aspects of the data [48]. This makes t-SNE flexible and suitable for different kinds of datasets. The t-SNE algorithm is given below.

```
t-SNE Algorithm
Algorithm t-SNE (X, perplexity, learning_rate, iterations):
Input:
    - X: dataset with n samples (high-dimensional data)
    - perplexity: balance between local and global aspects of the data
    - learning_rate: step size for optimization
    - iterations: number of iterations to run
Output:
    - Low-dimensional representation of the data (2D or 3D)
Step 1 : Compute pairwise affinities between data points in the high-dimensional space:
    1.1 For each data point, compute the probability distribution of its neighbors.
    1.2 Use perplexity to determine the similarity between points.
Step 2 : Initialize the low-dimensional map (e.g., 2D or 3D) randomly:
    2.1 Generate random starting positions for the n samples in the lower-dimensional space.
Step 3 : Minimize the Kullback-Leibler (KL) divergence between the high-dimensional and low-dimensional distributions using gradient descent:
    3.1 For each iteration:
        - Calculate the similarity of points in the lower-dimensional space.
        - Compute the gradient of the KL divergence between high-dimensional and low-dimensional similarities.
        - Update the positions of the points in the low-dimensional map based on the gradient and learning rate.
Step 4 : Repeat Step 3 for the specified number of iterations.
Step 5 : Return the final low-dimensional representation of the data.
    End Algorithm
```

### *2.4.8. Association Rule Mining*

Association rule mining is a data mining technique utilized to uncover intriguing relationships or associations among items or variables within a dataset. It identifies frequent patterns and dependencies among data items, which can be categorized into items-individual entities like products in a store, words in a document, or items in a basket-and transactions, which represent collections of items that capture a customer's purchase, a user's web page visit, or any event involving items.

Key metrics in this process include support, a measure of how frequently an item set (a combination of items) appears in the dataset, reflecting the popularity or occurrence of the item set, and confidence, which assesses the strength of the association between two item sets by indicating how often one item set is found alongside another. An association rule is expressed in the format "If {Antecedent} then {Consequent}," linking two item sets with a specified support and confidence level. Frequent item sets are those that meet a minimum support threshold and are considered significant for generating association rules.

The Apriori algorithm is a widely used method for association rule mining, employing a level-wise approach to discover frequent item sets by iteratively pruning infrequent ones based on the Apriori property, which states that if an item set is frequent, all its subsets must also be frequent.

For instance, in a retail dataset with customer transactions, frequent item sets may indicate that customers who purchase milk and bread are also likely to buy eggs, leading to the association rule: "If {milk, bread} then {eggs}" with corresponding support and confidence values. Association rule mining has diverse applications across various domains, including market basket analysis to identify product associations in retail sales, recommender systems for suggesting related items or content, healthcare for discovering patterns in patient diagnoses and treatments, and web usage mining to analyze user behavior for targeted marketing strategies [49, 50]. Based on the association rules, we discover the following. Customers who buy bread are likely to buy milk (66.67% confidence), customers who buy milk are likely to buy bread (50% confidence), and customers who buy bread are somewhat likely to buy Eggs (33.33% confidence).

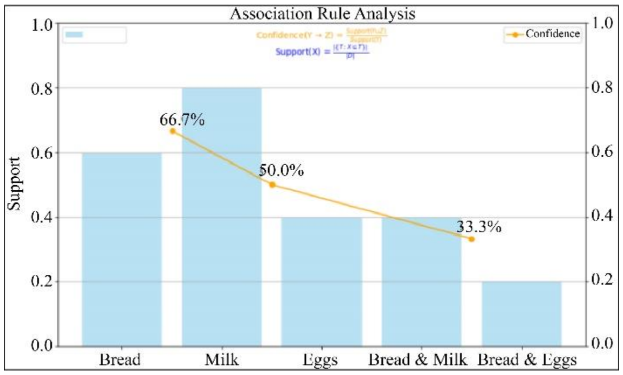


**Fig. 24 Association rule mining analysis**

**Table 2. Sales datasets**

| Transaction ID | Items Purchased |
|---|---|
| 1 | Bread, Milk, Eggs |
| 2 | Milk, Eggs |
| 3 | Bread, Diapers, Beer |
| 4 | Bread, Milk, Diapers |
| 5 | Milk, Beer |

```
Association Rule Mining Algorithm (Apriori)
Algorithm Apriori (T, min_support, min_confidence):
Input:
    - T: transactional dataset
    - min_support: minimum support threshold
    - min_confidence: minimum confidence threshold
Output:
    - Frequent itemsets and strong association rules
Step 1 : Find all frequent itemsets:
    - Generate candidate itemsets and prune those below the min support.
Step 2 : Generate strong association rules:
    - For each frequent itemset, generate rules.
    - Compute confidence for each rule and retain those with confidence >= min_confidence.
Step 3 : Return the frequent itemsets and strong association rules.
  End Algorithm
```

### *2.5. Semi-Supervised*

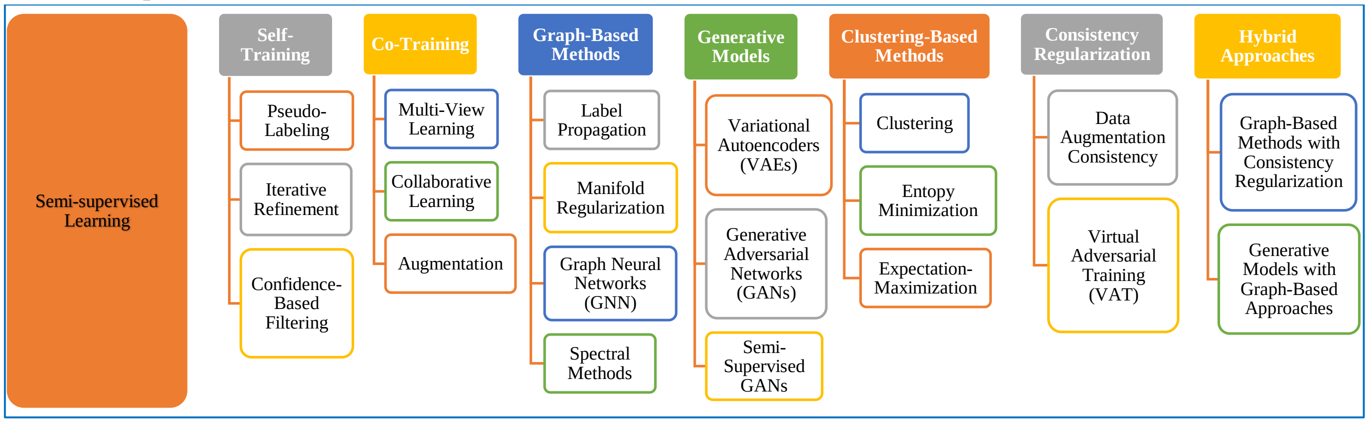


**Fig. 25 Taxonomy of semi-supervised learning**

There is an algorithm that can handle training data that is partially labeled, unlabeled, and partially labeled. We refer to this type of learning as semi-supervised learning. Supervised and unsupervised learning are typically combined to create semi-supervised learning. Google Photos service, for instance. The same person can be automatically identified once you upload a family photo here. Techniques for semi-supervised learning include self-training, graph-based approaches, generative models, and transductive support vector machines. A detailed semi-supervised taxonomy is given in Figure 25.

### *2.6. Semi-Supervised Learning Algorithms and Tasks*

SSL is an advanced machine learning paradigm that leverages a small set of labeled data alongside a large volume of unlabeled data to enhance model performance. It bridges the gap between supervised and unsupervised learning by incorporating techniques such as self-training, consistency regularization, and graph-based methods to efficiently infer patterns from unlabeled data. SSL is particularly effective in domains where labeled data is scarce or expensive to obtain, such as medical diagnosis, speech recognition, and anomaly detection. Key algorithms include pseudo-labeling, entropy minimization, and contrastive learning, which iteratively refine decision boundaries while maintaining robustness against overfitting. The primary objective of SSL tasks is to improve generalization by utilizing the intrinsic structure of data, ultimately reducing dependency on extensive human annotations.

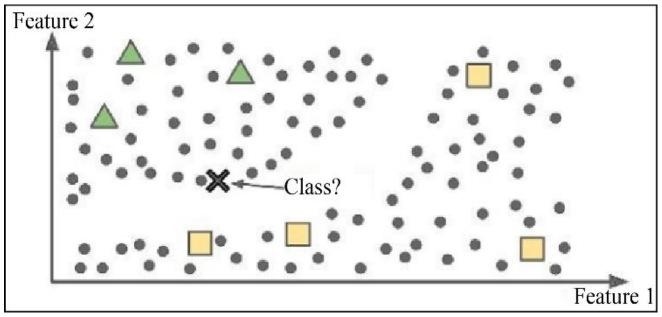


**Fig. 26 Semi-supervised learning [10]**

#### *2.6.1. Self-Training*

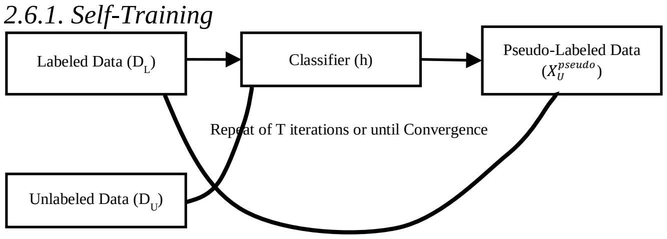

Labeled Data ($D_L$)
Classifier (h)
Pseudo-Labeled Data ($X_U^{pseudo}$)
Repeat of T iterations or until Convergence
Unlabeled Data ($D_U$)

**Fig. 27 Self-training semi-supervised learning process**

A common method is self-training, in which a model is first trained on labeled data and then uses the unlabeled data to make predictions. The model is then iteratively retrained by treating the most confident predictions as additional labeled examples [51]. It is helpful because, in email spam filters, an initial model is typically trained using a small set of labeled emails (spam and non-spam). The model then labels large volumes of unlabeled emails with high confidence. These confidently predicted labels are added to the training data to iteratively improve the spam filter's accuracy. The working process begins by using a labeled dataset $D_L$ to train a classifier $h$. Once trained, the classifier also processes an unlabeled dataset $D_U$, assigning pseudo-labels to the unlabeled examples based on its predictions.

These pseudo-labeled samples $X_U{}^{pseudo}$ are then added back to the labeled dataset $D_L$, expanding the training data and enabling the classifier to iteratively improve with larger, albeit partially pseudo-labeled datasets that include both human-labeled data and machine-generated pseudo-labeled data. This cycle of training and pseudo-labeling continues over multiple iterations until a predefined stopping criterion is reached, such as a maximum number of iterations $T$ or a convergence criterion, indicating stable performance.

#### *2.6.2. Graph-Based Methods*

Graph-based methods use data points as nodes and their similarities as edges, constructing a graph where labeled and unlabeled data interact to propagate label information across the graph [52]. For example, in social network analysis, in social networks like Facebook or LinkedIn, graph-based semi-supervised learning is used to infer user attributes (e.g., interests, job titles) by modeling the network of user connections. Labeled user profiles can propagate information to unlabeled profiles based on network structure, improving predictions of interests or job roles.

#### *2.6.3. Generative Models*

Generative models attempt to model the joint distribution of the input features and labels, utilizing unlabeled data to better capture the underlying structure of the data [53]. For example, in medical imaging and medical diagnosis, generative models like Variational Autoencoders (VAEs) can be applied to tasks such as classifying types of tumors. With limited labeled MRI scans of tumors, the generative model can learn the distribution of both labeled and unlabeled images to generate new examples, improving the model's ability to differentiate between benign and malignant tumors.

#### *2.6.4. Transductive Support Vector Machine*

Transductive Support Vector Machines (TSVMs) are extensions of Support Vector Machines (SVMs) tailored for semi-supervised learning. Unlike traditional SVMs, which rely solely on labeled data, TSVMs utilize both labeled and unlabeled data to enhance classification accuracy. They work by finding a hyperplane that separates the class while considering the distribution of the unlabeled data to define a more effective decision boundary. TSVMs aim to maximize the margin between classes using information from the entire dataset. Being transductive, TSVMs focus on the specific test set available during training, unlike inductive models that generalize to unseen data. Despite their computational intensity, TSVMs have proven useful in areas like text and image classification, offering improved accuracy and robustness in scenarios where labeled data is hard to find. We can categorise it as a hybrid approach because it bridges the gap between supervised SVM and semi-supervised by incorporating unlabeled data.

### *2.7. Reinforcement*

In reinforcement learning, the machine gradually updates some of its programs. But, the program automatically understands when to stop updating. That is, it realizes that the program may terminate on its own if it proceeds further, and then it starts to slow down on its own. A program moves forward when it understands that the state of the program is well and stops when it understands the danger. Reinforcement learning is the term for this kind of learning [1]. For example, consider an intelligent agent on which a reinforcement learning algorithm is applied. Agents can observe their surroundings, choose and carry out actions, and either receive rewards or penalties for receiving unfavorable ones. After that, it must figure out for itself the best course of action to maximize rewards over time. This is referred to as the principle in reinforcement learning. What an agent should do in a particular circumstance is specified by a policy. In other words, the trainer can offer a positive reward if the agent is trained to play a game. The game offers no reward in any other way, a positive reward when it is won and a negative reward when lost [2].

### *2.8. Reinforcement Learning Algorithms and Tasks*

RL algorithms, such as Q-learning, DQN, Policy Gradient Methods, and PPO, leverage techniques like value iteration, policy optimization, and experience replay to enhance learning efficiency and stability. Tasks in RL span from robotics control and autonomous navigation to algorithmic trading and real-time strategy games, where an agent must balance exploration-exploitation trade-offs to maximize cumulative rewards.

Advanced RL frameworks integrate deep learning, model-based planning, and meta-learning to adapt to complex, high-dimensional environments with minimal supervision, pushing the boundaries of AI-driven decision-making. An excellent illustration of reinforcement learning is the AlphaGo program from DeepMind. It garnered media attention when it defeated world go champion Lee Seidl in March 2016. After analyzing millions of games and playing numerous games against itself, it discovered its winning formula. However, during the match against the champion, it ceased to learn. AlphaGo merely put the lessons it had learned into practice. Additionally, a lot of robots learn to move by using reinforcement learning algorithms. Q-learning, deterministic Q-learning, Monte Carlo methods, temporal difference methods, and SARSA are examples of reinforcement learning techniques frequently employed in various applications. A detailed taxonomy of RL is provided in Figure 29.

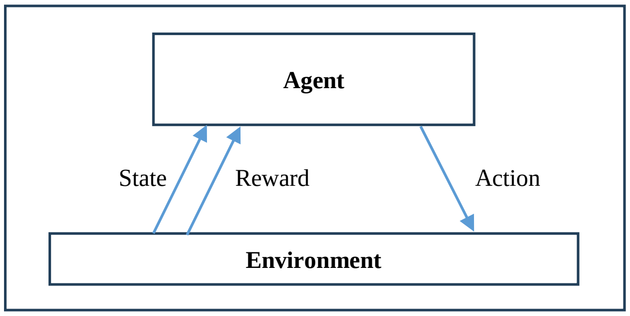
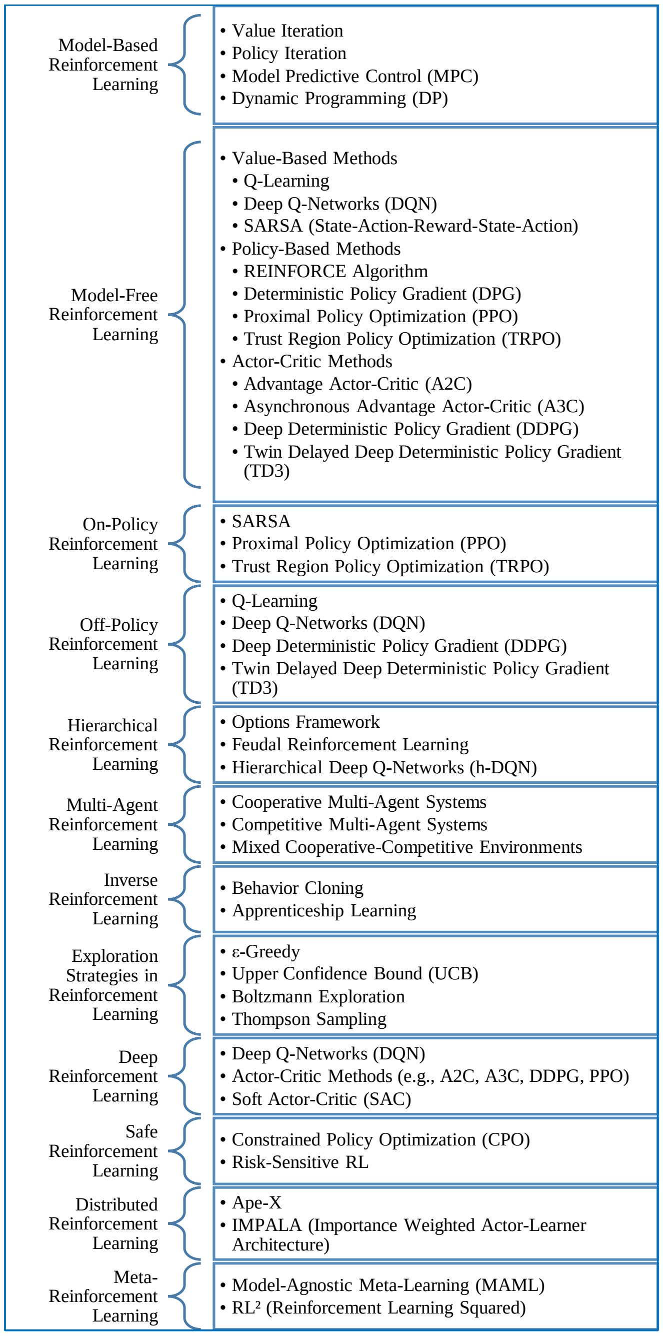


**Fig. 28 Reinforcement learning [1]**



Model-Based Reinforcement Learning
- Value Iteration
- Policy Iteration
- Model Predictive Control (MPC)
- Dynamic Programming (DP)

Model-Free Reinforcement Learning
- Value-Based Methods
  - Q-Learning
  - Deep Q-Networks (DQN)
  - SARSA (State-Action-Reward-State-Action)
- Policy-Based Methods
  - REINFORCE Algorithm
  - Deterministic Policy Gradient (DPG)
  - Proximal Policy Optimization (PPO)
  - Trust Region Policy Optimization (TRPO)
- Actor-Critic Methods
  - Advantage Actor-Critic (A2C)
  - Asynchronous Advantage Actor-Critic (A3C)
  - Deep Deterministic Policy Gradient (DDPG)
  - Twin Delayed Deep Deterministic Policy Gradient (TD3)

On-Policy Reinforcement Learning
- SARSA
- Proximal Policy Optimization (PPO)
- Trust Region Policy Optimization (TRPO)

Off-Policy Reinforcement Learning
- Q-Learning
- Deep Q-Networks (DQN)
- Deep Deterministic Policy Gradient (DDPG)
- Twin Delayed Deep Deterministic Policy Gradient (TD3)

Hierarchical Reinforcement Learning
- Options Framework
- Feudal Reinforcement Learning
- Hierarchical Deep Q-Networks (h-DQN)

Multi-Agent Reinforcement Learning
- Cooperative Multi-Agent Systems
- Competitive Multi-Agent Systems
- Mixed Cooperative-Competitive Environments

Inverse Reinforcement Learning
- Behavior Cloning
- Apprenticeship Learning

Exploration Strategies in Reinforcement Learning
- ε-Greedy
- Upper Confidence Bound (UCB)
- Boltzmann Exploration
- Thompson Sampling

Deep Reinforcement Learning
- Deep Q-Networks (DQN)
- Actor-Critic Methods (e.g., A2C, A3C, DDPG, PPO)
- Soft Actor-Critic (SAC)

Safe Reinforcement Learning
- Constrained Policy Optimization (CPO)
- Risk-Sensitive RL

Distributed Reinforcement Learning
- Ape-X
- IMPALA (Importance Weighted Actor-Learner Architecture)

Meta-Reinforcement Learning
- Model-Agnostic Meta-Learning (MAML)
- $RL^2$ (Reinforcement Learning Squared)

**Fig. 29 Taxonomy of reinforcement learning**

#### *2.8.1. Q-Learning (QL)*

Finding the optimal strategy for a given Markov Decision Process (MDP) is the aim of the model-free reinforcement learning algorithm Q-Learning. It learns by iteratively updating the values of state-action pairs, or Q-values, based on the observed rewards and transitions. Q-Learning uses a table called a Q-table to store and update the Q-values. The Q-values represent the anticipated future rewards for carrying out a particular action in a particular state. Through iterative interactions with the environment and Q-value updates, Q-Learning can converge to an optimal policy that maximizes the cumulative reward.

Q-values, also known as action values, are a fundamental concept in reinforcement learning. In the context of Q-Learning and related algorithms, Q-values represent the expected cumulative reward an agent can obtain by carrying out a specific action in a specific state. Formally, the expected total of future rewards the agent will receive by starting in state s, acting in a, and then following a particular policy is known as the Q-value Q (s, a) for a given state-action pair (s, a). The following is a common recursive definition of the Q-value.

$$Q(s, a) = R(s, a) + \gamma * \max(Q(s', a')) \tag{12}$$

Where:

- The immediate reward received when acting in state s. is denoted by R (s, a),
- The discount factor, γ (gamma), establishes the relative importance of future rewards compared to immediate rewards. After action a in state s. s' is the next state reached. max (Q (s', a')) represents the maximum Q-value among all possible actions a' in state s'

By iteratively updating the Q-values based on observed rewards and transitions, reinforcement learning algorithms aim to find the optimal policy that maximizes the cumulative reward. The agent uses the Q-values to make action-selection decisions, often employing exploration-exploitation trade-offs to balance between trying out new actions and exploiting the knowledge gained so far.

The Q-values are typically stored in a table called the Q-table in discrete state and action spaces, while in continuous spaces, they can be approximated using function approximators like neural networks in algorithms such as Deep Q-Networks (DQN). For example, in optimal route planning, suppose there is a delivery robot that needs to navigate through a complex maze-like environment to deliver packages efficiently. Then, Q-Learning can be used to find the optimal path for the robot by updating Q-values based on rewards (e.g., reaching the destination) and penalties (e.g., hitting obstacles). The robot explores the environment, gradually learning the best actions to take in each state to reach the destination quickly.

#### *2.8.2. Deterministic Q-Learning (DQL)*

Deterministic Q-Learning is a variant of the standard Q-Learning algorithm designed specifically for environments with continuous action spaces. Traditional Q-Learning, which works well for discrete actions, selects the action based on the maximum Q-value for each state. However, finding the maximum Q-value in continuous action spaces is not easy. To address this problem, DQL approximates the Q-values using a function approximator, such as a deep neural network. After receiving the state as input, the neural network produces the Q-values for each possible course of action. The action with the highest Q-value is selected for each state during training.

#### *2.8.3. Monte-Carlo Methods (MCM)*

Monte Carlo methods are a model-free, value-based class of reinforcement learning algorithms that learn by averaging the rewards obtained from complete environmental interaction episodes. Unlike Q-Learning, which updates the values based on each step, Monte-Carlo methods wait until the end of an episode to update the Q-values. The basic idea is to estimate a state-action pair's expected return (cumulative reward) by averaging the actual returns observed across multiple episodes. This approach is particularly useful when the dynamics of the environment are unknown or when the agent can only receive rewards at the end of an episode.

#### *2.8.4. Temporal Difference Methods (TDM)*

Temporal Difference Methods are a class of reinforcement learning algorithms that update the Q-values based on the observed rewards and the estimated value of the subsequent state. Unlike Monte-Carlo methods, which wait until the end of an episode, Temporal Difference methods update the Q-values after each time step. The update is based on the current estimate of the Q-value and the target value, which combines the immediate reward with the estimated Q-value of the subsequent state. This technique allows Temporal Difference methods to learn online by progressively updating the Q-values as the agent interacts with the environment.

#### *2.8.5. SARSA*

SARSA is an on-policy reinforcement learning algorithm that stands for "State-Action-Reward-State-Action.". It is similar to Q-Learning even though its update rule is different. When updating the Q-values in Sarsa, the following factors are considered: the current state, the action taken in that state, the reward received, the subsequent state, and the next action chosen in compliance with the policy. The update rule considers both the observed reward and the estimated Q-value of the next state-action pair. Sarsa is particularly well-suited for environments with stochastic transitions because it can adjust its policy in real time and learn directly from interactions.

### ***2.9. Ensemble Learning***

Ensemble learning is a machine learning fusion technique shown in Figure 30 that combines knowledge from several learning models to enable more precise and effective decision-making. Ensemble machine learning and statistics approaches combine several learning algorithms to produce better predictive results than anyone learning algorithm alone [54]. Ensemble methods, which train multiple learners and then combine them for use, boosting, and bagging as representatives, are one type of advanced learning technique [55]. An ensemble approach is typically far more accurate than a single learner, and it has already produced impressive results in a number of real-world tasks.

The ensemble method teaches multiple students how to solve the same problem. Ensemble learning is distinct from traditional learning approaches that seek to generate a learner from training data. It is also referred to as committee-based learning or learning multiple classifier systems. Bias, variance, and noise are the primary causes of error in learning models [56]. Integrated machine learning techniques ensure the accuracy and stability of machine learning algorithms, which reduce these error-inducing factors.

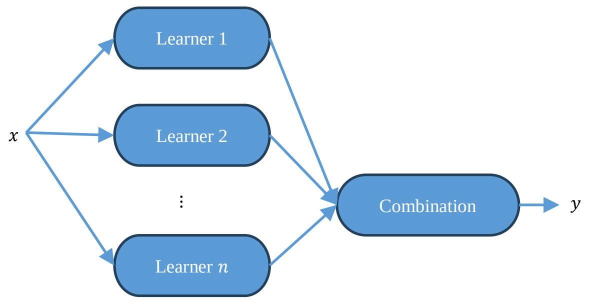


**Fig. 30 Ensemble learning architecture [11]**

Ensemble learning is like this scenario where an individual wants to buy a laptop. Instead of directly purchasing the laptop recommended by the salesperson at the showroom, the person takes a more comprehensive approach to make an informed decision.

He seeks opinions from friends, family, and colleagues, researches various portals to explore different laptop models, and visits review sites to gather more insights. In essence, he gathers multiple perspectives and reviews before concluding. This approach of considering diverse opinions and reviews to make a better decision can be linked to the concept of ensemble learning. Ensemble learning methods can be classified into three primary categories [57, 58].

### *2.9.1. Bagging*

Bagging involves training multiple decision trees on different subsets of the same dataset and combining their predictions through averaging. The term “Bagging” is derived from “Bootstrap AGGregatING,” emphasizing the use of bootstrapping and aggregation. Some examples of bagging methods include Bagged Decision Trees (canonical bagging), Random Forest, and Extra Trees.

```
Bagging Algorithm
1. Initialize empty list of models: models = []
2. for i = 1 to n_estimators do
3.     Create bootstrap sample from the training dataset
4.     Train a base model (e.g., Decision Tree) on the
           bootstrap sample
5.     Add the trained model to the models list
6. end for
7. Initialize empty list of predictions: all_predictions = []
8. for each test sample in test_set do
9.     Initialize empty list for sample_predictions =[]
10.    for each model in models, do
11.        prediction = model.predict(test_sample)
12.        Append prediction to sample_predictions
13.    end for
14.    final_prediction = majority_vote(sample_predictions)
 # For classification
       or
       final_prediction = average(sample_predictions)  # For
           regression
15.    Append final_prediction to all_predictions
16. end for
17. return all_predictions
```

### *2.9.2. Stacking*

Training various model types on the same dataset and using other models to Figure out how to best combine their predictions is known as stacking. Stacking techniques include super ensembles, blending, and stacked models.

### *2.9.3. Boosting*

Iteratively adding ensemble members that alter the predictions of earlier models and produce a weighted average of those predictions is known as “boosting”. A family of algorithms known as “boosting” has the ability to turn weak learners into strong ones. Examples: Stochastic Gradient Boosting, Gradient Boosting Machines, and AdaBoost (canonical boosting).

## ***2.10. Neural Network***

An artificial neural network or neural network is a set of algorithms created to detect hidden patterns and relationships within data, operating in a way that mimics the human brain [59]. The Taxonomy of NN is in Figure 31. These networks are made up of interconnected neurons, whether natural or synthetic. Neural networks can adapt to changing inputs, delivering the most effective results without modifying the output parameters. It can give optimal solutions to nonlinear types of problems. Rooted in artificial intelligence, neural networks are becoming increasingly popular in designing trading systems due to their flexibility and learning abilities. It uses three or more layers. The input layer receives input, one or more hidden layers process it, and the output layer generates the output from the preceding layer. Weight, summing, and activation functions are used in neural networks to predict or classify data based on input. Throughout the learning process, weight is used to assess each input feature's significance. It modifies the impact of each input on the neuron's output and regulates the influence of features. The more significant features are given more weight in order to increase their influence. In order to reduce the error between the expected and actual outputs, it is also utilized for learning and adaptation through the backpropagation process. The weighted sum of each neuron's inputs is calculated by the summing function.

To provide the model greater flexibility in fitting the data for precise prediction, a bias term is applied in the summing function prior to applying the activation function. The neural network is made non-linear by applying the activation function.

No matter how many layers the model has, it would act like a linear regression model without the activation function. It converts the summing function's output into a value that can either be used as the final output or passed to the following layer. The Figure below lists the various kinds of activation functions. These are some of the most widely used activation functions in deep learning and machine learning, each with unique properties and uses [60-62].

- Rectified Linear Unit (ReLU): The function outputs 0 for negative inputs and the input itself for positive values.
- Softmax: Often used in classification, showing probabilities over a set of classes.
- Sigmoid: A squashing function that maps any input to a range between 0 and 1.
- Linear: A direct linear mapping.
- Tanh: Maps inputs to a range between -1 and 1, showing an "S"-shaped curve like the sigmoid but centered at zero.
- Softplus: A smooth approximation of ReLU that outputs positive values for all inputs and is differentiable across the entire range.

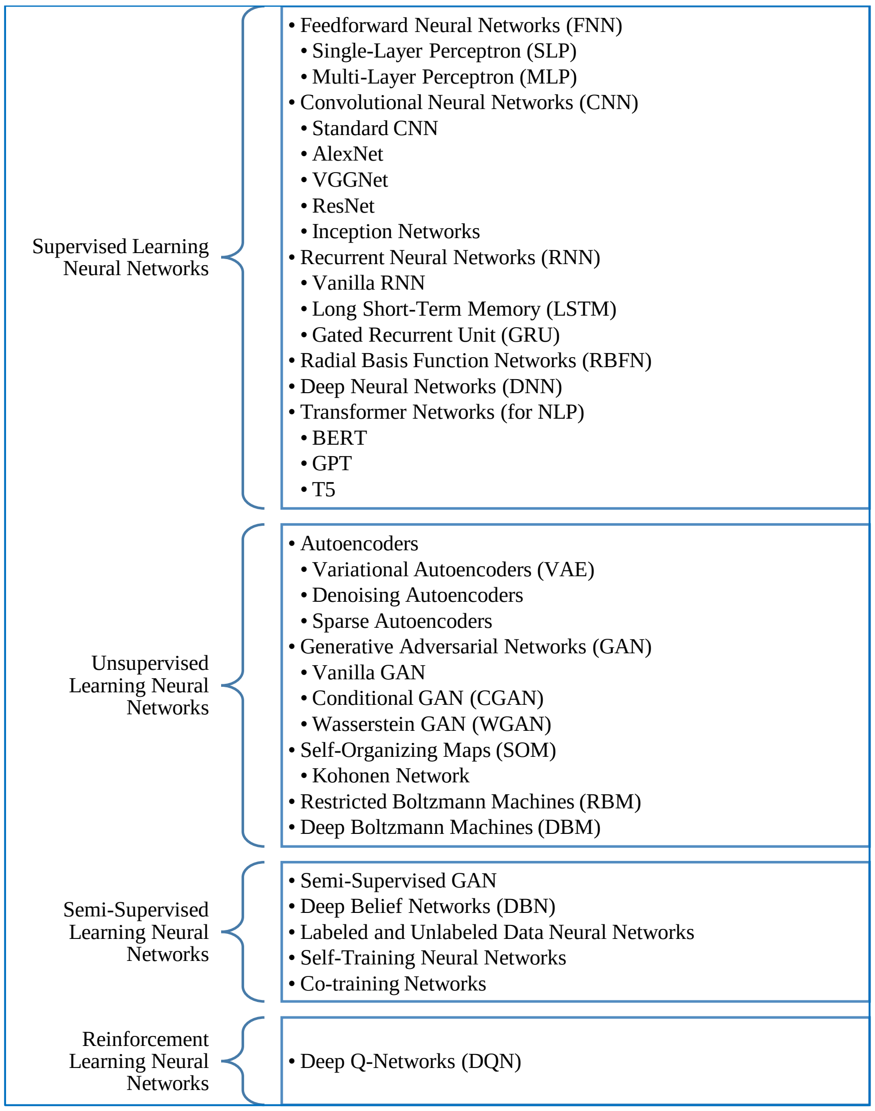

Supervised Learning Neural Networks
• Feedforward Neural Networks (FNN)
• Single-Layer Perceptron (SLP)
• Multi-Layer Perceptron (MLP)
• Convolutional Neural Networks (CNN)
• Standard CNN
• AlexNet
• VGGNet
• ResNet
• Inception Networks
• Recurrent Neural Networks (RNN)
• Vanilla RNN
• Long Short-Term Memory (LSTM)
• Gated Recurrent Unit (GRU)
• Radial Basis Function Networks (RBFN)
• Deep Neural Networks (DNN)
• Transformer Networks (for NLP)
• BERT
• GPT
• T5
Unsupervised Learning Neural Networks
• Autoencoders
• Variational Autoencoders (VAE)
• Denoising Autoencoders
• Sparse Autoencoders
• Generative Adversarial Networks (GAN)
• Vanilla GAN
• Conditional GAN (CGAN)
• Wasserstein GAN (WGAN)
• Self-Organizing Maps (SOM)
• Kohonen Network
• Restricted Boltzmann Machines (RBM)
• Deep Boltzmann Machines (DBM)
Semi-Supervised Learning Neural Networks
• Semi-Supervised GAN
• Deep Belief Networks (DBN)
• Labeled and Unlabeled Data Neural Networks
• Self-Training Neural Networks
• Co-training Networks
Reinforcement Learning Neural Networks
• Deep Q-Networks (DQN)

**Fig. 31 Taxonomy of neural network**

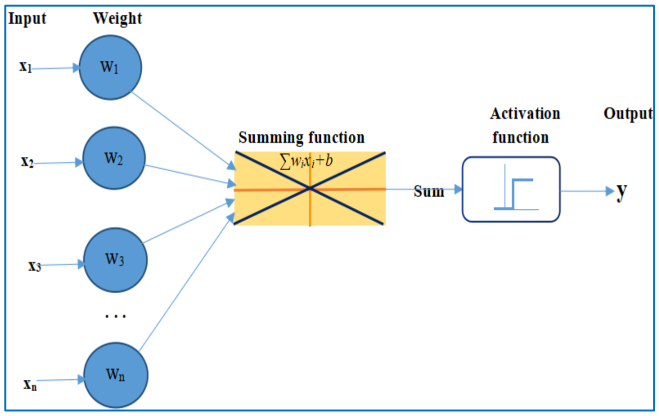


**Fig. 32 Neural network architecture**

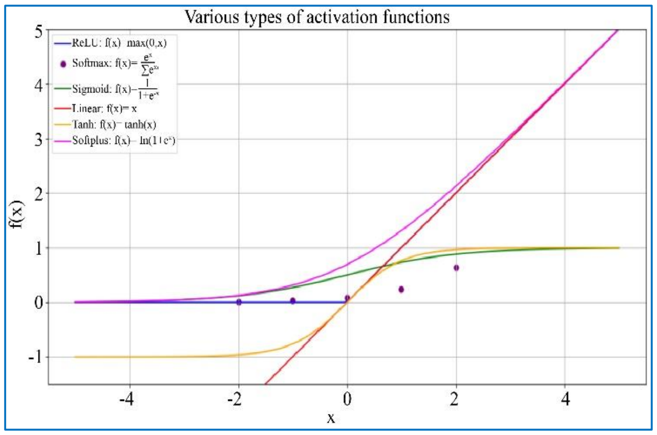


**Fig. 33 Various activation functions in NN**

*2.10.1. Supervised Neural Network*

The supervised learning policy is the foundation of the Supervised Neural Network (SNN). It gains knowledge from labeled data, which ensures that every input data set produces accurate results. Using a training set, the network is trained to translate input data into the appropriate output. If there is a discrepancy between the expected and actual outputs, the neural network's parameters are adjusted and fed into the network once more. Feed Forward Networks (FNNs), which transfer data from input to output in a single direction, are used by supervised neural networks. It is good for classification and regression. Image classification is a typical example of SNN. FNN, CNN, RNN, LSTM networks, and DBNs are the basic types of SNN, and all use feed-forward network strategies.

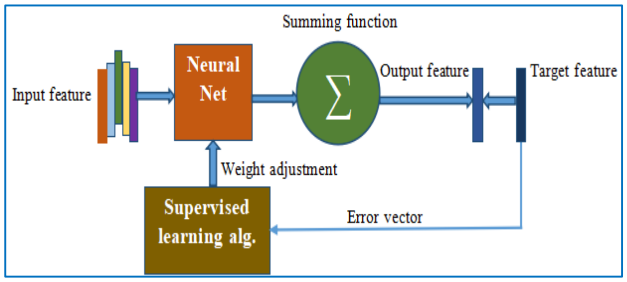


**Fig. 34 Supervised neural network**

*2.10.2. Unsupervised Neural Network*

Unsupervised neural networks learn from input data based on its structure and patterns without the need for label data. Neural networks create groups based on correlations with similar data. It is primarily employed for dimensionality reduction, association, and clustering. Common UNN types include autoencoders, SOMs, and GANs.

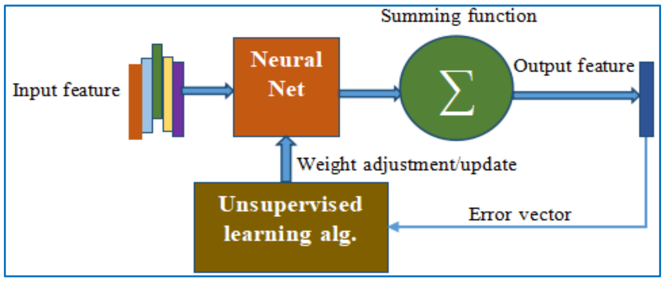


**Fig. 35 Unsupervised neural network**

*2.10.3. Reinforcement Neural Network*

This is a goal-oriented model, aiming to maximize cumulative rewards over time. In this learning, the correct output is not provided; instead, it learns from the environment through the process of trial and error by receiving feedback in the form of rewards and penalties. Reinforcement learning can be used as a standalone algorithm in the traditional form; however, when neural network function is used in reinforcement learning, it is called a reinforcement neural network.

It is also called deep reinforcement learning, using a neural network system. The detailed description and Figure 36 are given in the section on reinforcement learning. Q-learning, Deep Q-Networks, Policy Gradient Methods, and Actor-Critic Methods are examples of reinforcement neural network models [1, 10].

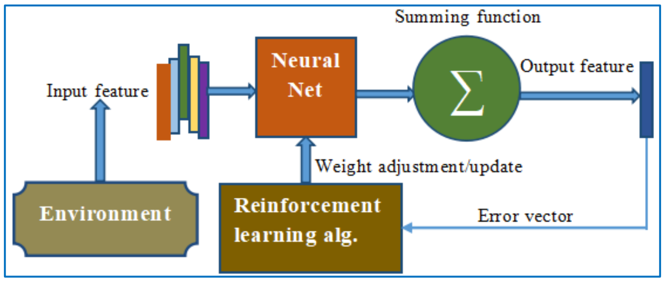


**Fig. 36 Reinforcement neural network**

*2.10.4. Convolutional Neural Network*

CNN, also known as Convolutional Networks or simply Convnet, are widely used. Convolutional networks, which are multilayer perceptions, are appropriate for pattern classification. The most widely used deep learning architecture at the moment is this one. Deep learning has garnered attention recently due to its enormous popularity and efficacy.

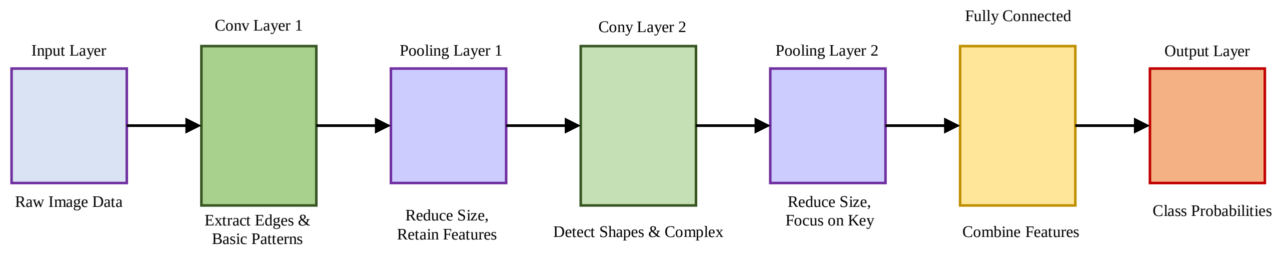


**Fig. 37 CNN architecture**

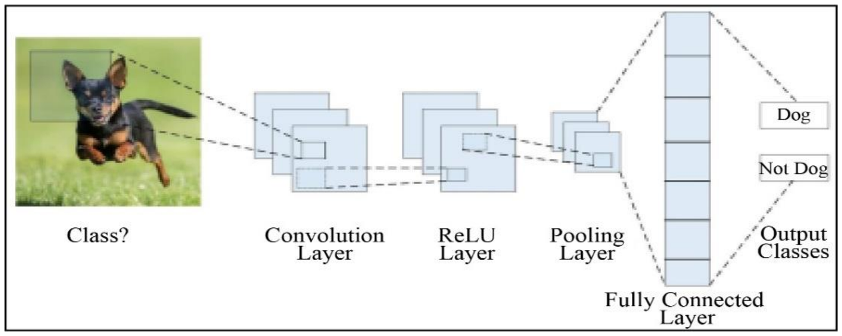


**Fig. 38 Image classification using CNN [20]**

Since AlexNet's launch in 2012, interest in convolutional networks has increased dramatically. The researchers upgraded from an 8-layer AlexNet to a 152-layer ResNet move-on in just three years [63, 64]. Convolutional networks are currently used to model image-related issues. It has been effectively used in image processing, face recognition, natural language processing, recommender systems, and other fields. Its primary benefit is its ability to recognize significant features automatically without human oversight. For instance, it automatically learns distinct features for each class when presented with numerous photos of dogs and cats.

The development of these networks is motivated by neurobiological principles, which date back to Hubel and Wiesel's groundbreaking research on the orientation-selective neurons in the visual cortex of cats. Yann LeCun and others were the first to invent it in the 1980s. One of the earliest real-world uses of CNNs for handwritten digit recognition was LeCun's work, specifically the LeNet-5 architecture. LeCun and his group used an older concept based on ideas from David H. Hubel and Torsten Wiesel, who were awarded the 1981 Nobel Prize in Physiology or Medicine for their groundbreaking 1968 paper. They looked into the visual cortex of animals and discovered links between smaller areas of the visual field and the activity of a tiny but distinct brain area. With greater degrees of translation, scaling, skewing, and other distortions, it is specially made to detect two-dimensional shapes invariantly. The working policy of the four main components of the CNN above are as follows:

- Input Layer: Like other ANNs, the input layer of CNN stores the pixel values of the input image, which are then sent into the network for processing.
- Convolutional Layer: This layer computes the output of neurons attached to particular, localized areas of the input image in order to identify features. In order to create feature maps that emphasize patterns like edges or textures, each neuron computes the scalar product between its set of weights, a kernel made by random numbers with discrete values and the corresponding region of the input. Then, to add non-linearity, the Rectified Linear Unit (ReLU) activation function-which is frequently employed in CNNs-is applied element-by-element, converting negative values to zero while maintaining positive values. The main function of the ReLU layer is to provide nonlinear transformation to CNN for capturing complex patterns of images [20]. Different types of convolution and its applicability is explained in Table 1.
- Pooling Layer: By down-sampling, the pooling layer shrinks the feature maps' spatial dimensions. By reducing the number of calculations and parameters, this procedure helps to avoid overfitting and increases computational efficiency. Usually, pooling keeps the most important aspects while eliminating the less crucial ones. Three types of pooling methods are commonly used in this layer: min, max, and global average pooling methods.
- Fully-Connected Layer: The fully-connected layers receive the output from the convolutional and pooling layers and produce class scores based on extracted features, much like the structure found in conventional ANNs. The input image is then classified using these scores. Between these layers, ReLU activation functions are frequently used to advance. There are various types of convolution. Each convolution type plays a unique role in different applications, leveraging specialized patterns and data structures to optimize performance for specific tasks.

**Table 3. Different types of convolution**

| Type | Dimension | Purpose | Applications |
|---|---|---|---|
| 1D Convolution | 1D | Sequential patterns | Time-series, audio, NLP |
| 2D Convolution | 2D | Spatial patterns in images | Image processing, CNNs |
| 3D Convolution | 3D | Spatiotemporal data | Video, medical imaging |
| Transpose Convolution | 2D/3D | Upsampling, image reconstruction | GANs, image segmentation |
| Dilated Convolution | 2D/3D | Wider receptive fields | Semantic segmentation, contextual learning |
| Separable Convolution | 2D/3D | Efficient computation | Lightweight models (e.g., MobileNets) |
| Grouped Convolution | 2D/3D | Parallel feature learning | ResNeXt, efficient deep learning |
| Circular Convolution | 1D/2D | Periodic data patterns | Signal processing with periodicity |

```
Convolutional Neural Network (CNN) Algorithm
Algorithm CNN (X, Y, num_epochs, learning_rate):
Input:
    - X: training data points (image data)
    - Y: corresponding labels (classification or regression)
    - num_epochs: number of training iterations
    - learning_rate: learning rate for optimization
Output:
    - Trained model (weights and biases)
Step 1 : Initialize the CNN architecture
        - Define convolutional layers, activation functions (e.g.,
          ReLU), pooling layers, and fully connected layers
Step 2 : Forward propagation
        - Pass input X through the layers:
        - Convolution -> Activation -> Pooling -> Fully
          Connected -> Output layer
        - Compute the predicted output y_pred for each sample
          in X
Step 3 : Compute the loss
        - Use a loss function (e.g., cross-entropy for
          classification or MSE for regression) to measure the
          difference between y_pred and Y
Step 4 : Backward propagation
        - Use backpropagation to compute gradients of the loss
          with respect to the model parameters (weights and
          biases)
Step 5 : Update the model parameters
        - Use an optimization algorithm (e.g., Stochastic
          Gradient Descent) to update weights and biases
Step 6 : Repeat steps 2-5 for num_epochs iterations
Step 7 : Make prediction for a new data point x_new
        - Pass x_new through the trained CNN to get the
          predicted class or value y_new
  Return y_new as the prediction
End Algorithm
```

**Table 4. Different ML algorithms and its application with pros and cons**

| Algorithm | Year of Invention | Inventor | Application | Pros | Cons | Ref. |
|---|---|---|---|---|---|---|
| Linear Regression | 1805 | Adrien-Marie Legendre | Predicting sales, risk assessment | Simple, interpretable, fast to train. | Sensitive to outliers, assumes linearity. | [19, 20] |
| PCA | 1933 | Karl Pearson | Image compression, exploratory data analysis | Reduces dimensionality, helps with visualization, and eliminates multicollinearity. | Loss of information assumes linearity. | [28, 29] |
| ANN | 1943 (concept), 1986 | Warren McCulloch, Walter Pitts, Geoffrey Hinton | Image recognition, speech recognition, game AI | Can model complex patterns, adaptable for a variety of tasks. | It requires large datasets, is computationally expensive, and is difficult to interpret. | [59-62] |
| KNN | 1951 | Evelyn Fix, Joseph Hodges | Recommendation systems, classification tasks | Simple to understand, no training required. | Computationally expensive, sensitive to irrelevant features and noise. | [14,15] |
| k-Means Clustering | 1957 | Stuart Lloyd | Market segmentation, image compression | Simple, fast, and works well for simple clusters. | Struggles with complex datasets, sensitive to initialization and outliers. | [26, 27] |
| Logistic Regression | 1958 | David Cox | Binary classification tasks, medical | Works well for binary classification, interpretable. | Assumes linear decision boundary struggles with large or non-linear | [21] |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | diagnosis | | datasets. | |
| Naive Bayes | 2nd half of the 18th century | Thomas Bayes (concept), others later developed. | Spam detection, text classification | Simple, fast, works well with high-dimensional data, interpretable. | Assumes feature independence may struggle with complex datasets. | [1] |
| Decision Trees | 1963 | Ross Quinlan | Credit scoring, customer segmentation | Easy to interpret, requires little data preprocessing, and handles both numerical and categorical data. | Prone to overfitting, sensitive to small variations in data. | [1, 9] |
| Recurrent Neural Networks (RNN) | 1980 | John Hopfield | Natural language processing, time series analysis | Effective for sequential data, captures time dependencies. | Prone to vanishing/exploding gradient problems, struggles with long-term dependencies. | [2, 65] |
| Autoencoders | 1980s | Geoff Hinton, Yann LeCun, others | Dimensionality reduction, denoising data | Effective for unsupervised feature learning dimensionality reduction. | Prone to overfitting, difficult to interpret, requires careful tuning. | [30] |
| Reinforcement Learning (Q-Learning) | 1989 | Chris Watkins | Game playing, robotic control | Good for sequential decision-making problems and learn optimal policies through trial and error. | Slow to converge, can be unstable, and requires a large number of interactions with the environment. | [1] |
| Support Vector Machine (SVM) | 1992 | Vladimir Vapnik | Text classification, image recognition | Effective in high-dimensional spaces, robust to overfitting with proper tuning. | It can be slow, difficult to tune, and doesn’t work well with noisy data. | [2, 18] |
| Random Forest | 1995 | Tin Kam Ho | Fraud detection, feature selection | Reduces overfitting compared to decision trees and handles large datasets well. | Slower in making predictions, difficult to interpret individual trees. | [25] |
| AdaBoost | 1996 | Yoav Freund, Robert Schapire | Face detection, web search ranking | Good accuracy, especially for simple classifiers, reduces bias. | Sensitive to noisy data and outliers, it requires good weak classifiers. | [57] |
| DBSCAN | 1996 | Martin Ester, Hans-Peter Kriegel | Geospatial data analysis, anomaly detection | It can find clusters of arbitrary shapes that are robust to outliers. | Struggles with varying densities of clusters, sensitive to the choice of parameters. | [1] |
| Gradient Boosting Machines (GBM) | 1997 | Jerome H. Friedman | Customer churn prediction, risk modeling | High accuracy, performs well with structured/tabular data. | Prone to overfitting, sensitive to noisy data, slow to train. | [57] |
| Long Short-Term Memory (LSTM) | 1997 | Sepp Hochreiter, J□rgen Schmidhuber | Language modeling, speech recognition | It solves the vanishing gradient problem and is effective for long-term sequential data. | Computationally expensive, slower to train, and difficult to interpret. | [2] |
| CNN | 1998 | Yann LeCun | Image and video recognition, medical image analysis | Excellent for image processing tasks, automatically extracts features. | It requires large amounts of data and computation and is prone to overfitting on small datasets. | [63] |

| | | | | | | |
|---|---|---|---|---|---|---|
| LOF | 2000 | Markus M. Breunig, Hans-Peter Kriegel | Fraud detection, network security | It detects anomalies by considering local density and is adaptive to clusters. | Sensitive to the choice of parameters, struggles with very large datasets. | [46] |
| t-SNE | 2008 | Geoffrey Hinton, Laurens van der Maaten | Visualizing high-dimensional data | Great for visualizing high-dimensional data. | Computationally expensive, it doesn't preserve global structure well. | [47] |
| iForest | 2008 | Fei Tony Liu, Kai Ming Ting, Zhi-Hua Zhou | Anomaly detection, fraud detection. | Effective for anomaly detection, handles high-dimensional datasets, and scales well. | It can struggle with low anomaly contamination and may require parameter tuning. | [43] |
| Generative Adversarial Networks (GANs) | 2014 | Ian Goodfellow | Image generation, data augmentation. | Effective for data generation tasks (images, text), can model complex distributions. | Difficult to train, prone to mode collapse, requires large datasets. | [53] |
| XGBoost | 2014 | Tianqi Chen, Carlos Guestrin | Risk prediction and recommendation systems. | Highly accurate, efficient, works well for large datasets, and handles missing data. | Computationally expensive, can overfit without proper tuning, and less interpretable than simpler models. | [56] |
| Transformer Networks | 2017 | Ashish Vaswani et al. | Natural language processing, machine translation. | State-of-the-art NLP tasks capture long-range dependencies well. | Very computationally expensive, requires large datasets, and is prone to overfitting on small datasets. | [98] |
| LightGBM | 2017 | Microsoft Research | Large-scale machine learning tasks and recommendation systems. | Faster training than XGBoost handles large datasets with low memory usage. | It can be sensitive to overfitting and is not as interpretable, especially on small datasets. | [99] |
| CatBoost | 2017 | Yandex | Classification tasks, particularly with categorical features. | Works well with categorical features and requires less tuning than other gradient-boosting methods. | Computationally intensive, less interpretable. | [99] |

## 3. Data Preprocessing and Feature Selection

Data preprocessing is essential for preparing data to build an effective model. Handling missing values, label encoding, normalization, and standardization techniques are used for data preprocessing. However, the usability of techniques depends on the nature of the data that will be processed. Another crucial phase in data preprocessing, feature selection, concentrates on the most informative variables to improve model performance. The Chi-Square Test, which assesses feature independence from the target variable, is best suited for categorical features with a categorical target, whereas the correlation coefficient for numerical features aids in determining linear relationships between features and the target. Since it identifies features with significant variance across target classes, Analysis of Variance (ANOVA) is helpful for numerical features with a categorical target. Mutual Information captures non-linear dependencies with the target variable and performs well for both continuous and categorical features. The iterative processes of forward selection and backward elimination add features according to how well they contribute to model accuracy. In contrast, backward elimination iteratively eliminates the least important features to enhance model performance. Based on model training, Recursive Feature Elimination (RFE) selects and ranks features by recursively eliminating the least significant ones. Lasso Regression is useful for high-dimensional data because it reduces the coefficients of less significant features to zero. Tree-based techniques that rank features according to their significance in lowering impurity in decision nodes include Random Forest and Gradient Boosting. Linear Discriminant Analysis (LDA) is best suited for supervised dimensionality reduction when class separability is crucial, whereas Principal Component Analysis (PCA) converts features into uncorrelated components, which is especially helpful with highly correlated features. Genetic algorithms allow the evolutionary search for the best feature combinations, making them ideal for large or complex datasets. By streamlining feature sets, these techniques enhance computational efficiency and model performance [11-13].

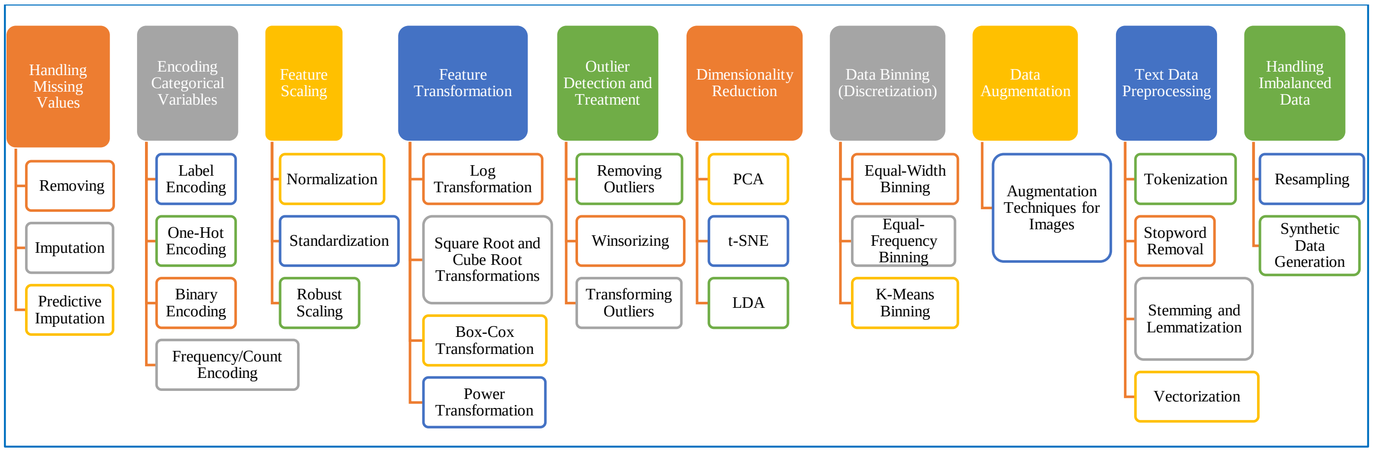


**Fig. 39 Data preprocessing techniques**

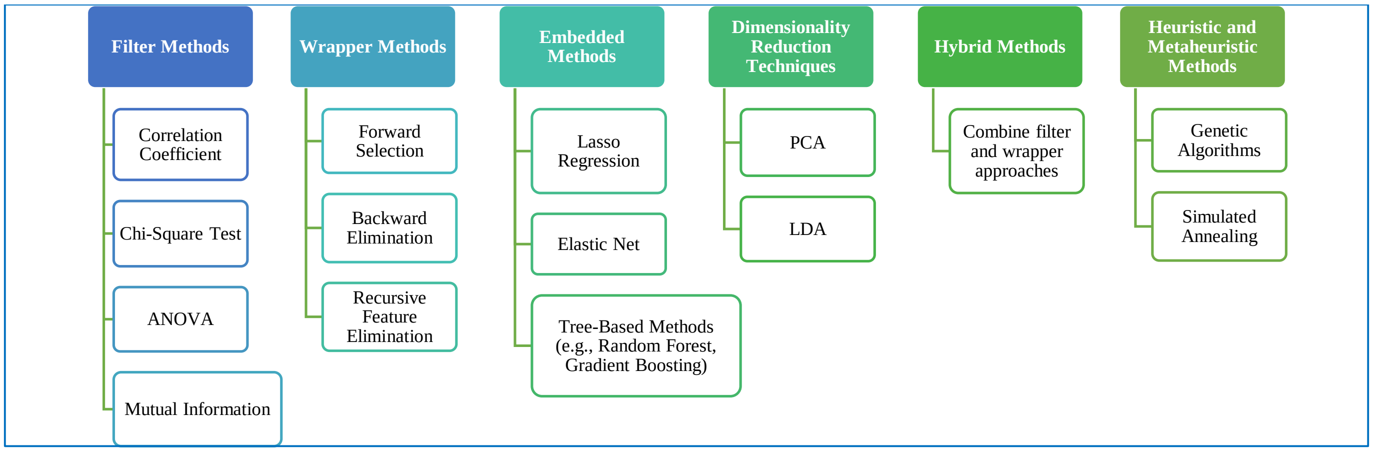


**Fig. 40 Feature selection techniques**

## 4. The Global Trend in Machine Learning

Machine learning is a global trend. We can gauge the popularity of each learning method by analyzing the number of published research papers in major databases like IEEE, Springer, and Scopus over the years. Tracking how often these terms have been searched globally on Google over time. Analyzing GitHub to observe the growth of repositories or projects related to these topics. Check surveys and reports (such as those by Kaggle or Stack Overflow) to see which techniques are most used by professionals in AI/ML.

Looking into references in major AI frameworks like TensorFlow, PyTorch, or Scikit-learn documentation. The use of reinforcement learning has increased recently along with supervised learning is in higher rank and popularity of 90% recently; unsupervised learning has recently stable and popularity of 80%; semi-supervised learning is lower than unsupervised learning, and its index is 70% recently [65, 66]. The summary of the Machine learning popularity index is given in Figure 41.

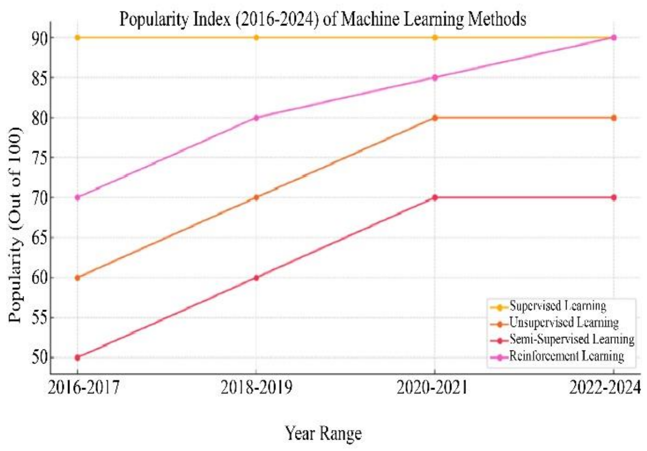


**Fig. 41 Machine learning popularity index**

## 5. Machine Learning Applications

ML is revolutionizing many industries by empowering sophisticated systems to examine data, spot trends, and formulate well-informed forecasts. In the healthcare sector, ML assists in diagnosing illnesses by identifying abnormalities in medical images and forecasting patient outcomes. Financial institutions employ ML to identify real-time fraudulent activities, evaluate credit risks, and deliver customized banking experiences. In retail and e-commerce, ML enhances personalized recommendations and streamlines supply chain management. Additionally, ML models interpret complex datasets in environmental science to predict weather patterns and monitor climate changes. By automating decisions and offering valuable insights, ML accelerates innovation and operational efficiency, driving the evolution toward a data-centric world. Figure 42 shows evidence that the medicine and healthcare sector are the most promising areas where ML technology has the largest share at 35%, the engineering sector showing considerable interest, making upto 20%, the Financial sector and other fields occupy 15%, showing a balanced distribution, the agriculture sector is showing notable investment at 10% and 5% specialized nanotechnology [65-69]. Based on this statistic, some common applications are explained in the following section.

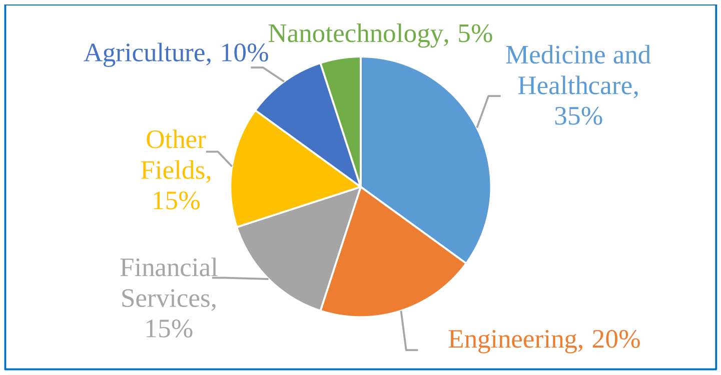


**Fig. 42 Percentage of machine learning application area**

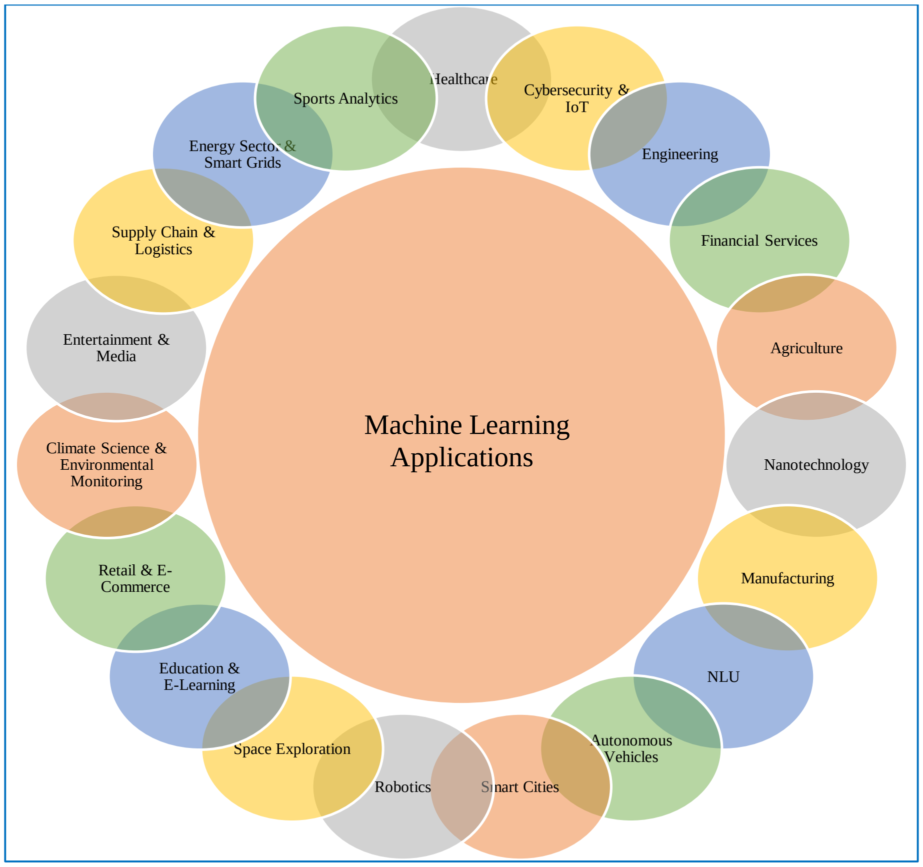


**Fig. 43 Applications of machine learning**

### 5.1. Healthcare

Machine learning applications have surged in the last few years, especially in predictive models for diseases such as heart disease, diabetes, and, more recently, COVID-19 [70, 71]. Deep learning and federated learning have also been prominent, particularly for image classification and medical diagnostics, personalized treatment, and drug discovery. Machine learning can predict and diagnose various diseases, including neurodegenerative disorders like Alzheimer's disease and Parkinson's disease, as well as serious mental health disorders such as psychosis and depression.

ML is used in genetic research to identify mutations linked to diseases, predict genetic risks, and understand complex genetic interactions. NLP is used for medical records, enabling information extraction from unstructured data. ML improved robotic surgery and assistance in real-time. It is estimated that every year robotic surgery is growing 18% globally. Remote patient monitoring and telehealth improve care access and reduce hospital visits. ML helps to reduce operation costs and waiting time by optimizing healthcare operations [72].

### 5.2. Cybersecurity & IoT

From 2020 onwards, there has been significant interest in applying machine learning to network anomaly detection, fraud detection, and IoT data processing, especially with the growing number of connected devices. ML is utilized in threat detection and intrusion prevention by analyzing network traffic [73]. ML can classify and detect malware by analyzing code structure and attack patterns. Phishing detection using machine learning is a promising application that is available and widely used.

Ml algorithms are used to manage authentication protocols to authenticate IoT devices. ML techniques, including anomaly detection, identify unusual traffic patterns or abnormal device behaviors within IoT networks, which are often targeted due to their limited security.ML models predict when IoT devices will likely fail, enabling proactive maintenance and reducing downtime, which is critical for uninterrupted network security.ML models can detect fraudulent activities within IoT ecosystems by analyzing user behavior, transaction patterns, and device activity. ML algorithms help identify and shut down botnets by detecting command-and-control traffic patterns.

### 5.3. Engineering

Especially in bioengineering, AI assists in designing medical devices and optimizing infrastructure projects. Moreover, ML is applied in predictive maintenance, quality control and inspection, building and structural health monitoring, energy optimization, smart manufacturing, robotics and automation systems, aerospace engineering design and optimization [74], NLP for engineering documents, civil engineering [75] and urban planning using ML by creating data-driven approaches that lead to safer, more efficient, and environmentally friendly urban environments [76]. ML is used in environmental engineering and climate prediction. In environmental engineering, ML helps monitor and manage resources like water, air, and soil by analyzing data from sensors, satellites, and other sources [78] to identify patterns, predict contamination events, and improve pollution control. Climate prediction assesses the impacts of climate change on ecosystems.

ML techniques are applied in mechanical engineering to enhance manufacturing efficiency. ML in chemical engineering [79] by enhancing process optimization, accelerating materials discovery, and improving safety. In materials science [77], ML expedites the discovery of new chemicals and materials by predicting the properties of compounds before they are synthesized. This approach significantly shortens the development cycle for new materials in fields like pharmaceuticals, polymers, and battery technology.

### 5.4. Financial Services

Machine learning helps in fraud detection, algorithmic trading, and personalized financial services. By using machine learning techniques, we can identify financial fraud. Making quick decisions for loan approval by calculating credit score. It offers real-time solutions to make complex decisions. ML can save time and increase productivity. The customer gets 24/7 services through chatbot and virtual assistant, and it reduces operational costs by the one-time implementation. Process automation reduces time and cost and improves productivity [80]. Network security is a vital issue in financial institutions, and ML can play a significant role in securing the system. Money laundering techniques can be prevented by ML-enabled monitoring [81].

### 5.5. Agriculture

AI optimizes crop monitoring, yield prediction, disease detection, weed detection, livestock production, species recognition, soil management, water management, and sustainable farming practices. Weed is called one of the most important enemies of crop production. ML can be used for weed detection and management. ML can detect weeds with the help of sensors and then develop tools and robots to destroy them [82]. Crop quality identification for accurate pricing. ML can be used to identify the features related to crop quality. Different ML techniques are applied to observe the features and identify the quality.

ML is used for species recognition. It can identify and classify the plant species automatically. It can reduce the human efforts and time. Farming complexes can use ML techniques to manage livestock. Sensors will be attached to the animals and observe their behavior and food habits. ML is also used to optimize livestock production. Agricultural main resources such as water and soil play a significant role and can be monitored and managed by ML techniques. ML applications are adjusting water level, monitoring soil conditions, and weather prediction.

### *5.6. Nanotechnology*

ML helps advance materials research and the development of nanoscale devices. ML can support automation in nanoscale manufacturing processes, improving efficiency, reducing costs, and ensuring precision. As ML algorithms become more adept at predicting biological responses, they could be used to design personalized Nano medicines tailored to an individual's genetic profile, enhancing treatment efficacy.

Quantum computing, combined with ML, is expected to revolutionize nanotechnology by enabling more accurate simulations and predictions for molecular design and materials properties at the quantum level. This technology is rapidly growing in some specialized sectors like clean energy, pharmaceuticals, and semiconductors [83, 84].

### *5.7. Manufacturing*

In order to increase productivity and cut expenses, industry 4.0 uses AI for predictive maintenance, quality control, defect detection, process and supply chain optimization, product design and development, energy management, work safety and monitoring, and robotic automation. An emerging field is self-optimizing, in which machines learn from production data and modify parameters independently. ML is used in conjunction with digital twins, or virtual copies of physical assets, to simulate and optimize processes in real-time [85].

An integrated supply chain with machine learning is more resilient and responsive to interruptions. Real-time monitoring and predictive analytics are two applications of machine learning that are expanding quickly. The strong use of ML techniques is seen in the transition to IIoT, cloud computing, additive manufacturing, and augmented and virtual reality [86].

### *5.8. Natural Language Understanding*

This application is widely used. Translation of language, voice-to-text, text-to-voice conversion, speech recognition, voice recognition, and sentiment analysis are all examples of NLP applications [13].

Computers can understand human voices and translate them into text in real time and vice versa. Sentiment analysis is another popular ML application. Users give product reviews on blogs, forums, and social media such as Facebook, Zomato, Cars.com, etc. Based on these reviews, businesses and brands can recommend and improve their product [91]. Moreover, it also can analyze emotions such as happy, very happy, sad, angry, interest, not interest, etc.

### *5.9. Some Other Applications*

Machine learning has become crucial for personalized content, predictive analytics, and customer segmentation. Over half of organizations are now using ML for these purposes. ML helps in optimizing power grids [89], predicting energy demand, and enhancing the efficiency of renewable energy systems such as wind and solar power. ML is critical in autonomous driving systems, traffic management, route optimization, and vehicle predictive maintenance. Machine learning with IoT is a promising application for monitoring and management of smart cities and transportation. AI-driven ML systems help in personalized marketing, customer service chatbots, inventory management, and demand forecasting.

ML applications in education include personalized learning systems [90], grading automation, and intelligent tutoring systems. ML in recommendation systems is a very popular application nowadays. It is mostly used in interactive web environments, such as content recommendation and personalized shopping, which are widely used in e-commerce business [91]. Businesses can analyze customers' shopping behavior, and they can suggest product recommendations. Recommend movies based on users watching behavior are widely used [92].

Video and audio editing, gaming, and generating creative content such as music [93] or visual art [94] are also gaining popularity and being widely used. ML is used in surveillance, military robotics, and strategic decision-making. Moreover, ML assists in recruitment processes, employee performance tracking, and human capital management.

## 6. Challenges and Research Directions

Machine learning systems are deployed in many areas and offer opportunities in our everyday life. Despite these new opportunities, there are still many challenges in machine learning. In model generalization to unseen data, there is an underfitting and overfitting issue, specifically in deep learning models in noisy data, resulting in poor performance [95].

Imbalanced datasets with limited labeled data hinder the effectiveness of supervised learning. Bias is another challenge due to the underrepresented of minority classes. The deep learning model requires more interpretable, human-understandable rationales in decision-making. The vast datasets and complex machine learning models require more efficient algorithms, hardware acceleration techniques and quantum-inspired approaches to increase the scalability and computational efficiency [96].

Lack of adaptation to a dynamic environment requires continual learning algorithms to update and evolve over time without requiring retraining. Data collection from diverse sources such as medical, agriculture, cybersecurity, IoT, and industrial sources is not candid, although collecting such useful data is crucial for further analysis and decision-making. There should be concrete data collection methods to investigate real-world data. We produce huge amounts of data, which is called big data, but this data comes in many forms, such as structured, semi-structured, and unstructured. Data can be poor quality.

It may contain missing values, unnecessary values, etc. It is a big challenge for machine learning to process and make unique formats. There are no unified methods to generalize this data. Therefore, we need concrete data preprocessing methods to use machine learning algorithms effectively in a particular domain. Selecting the proper algorithm for diverse datasets is another challenge. It can be a single-model or multi-model disagreement.

Multi-model disagreement in machine learning is a phenomenon where different models trained on the same dataset provide diverse predictions for a given input. It is not because of the algorithm but because of the features of the datasets. Therefore, the results produced from this model can be inaccurate, unexpected and ambiguous for the application. Nevertheless, the combined model could be prominent for latent work in future [97].

One must handle both the datasets and algorithms to get the best result by implementing machine learning in the application. It needs to consider the algorithm based on the nature and characteristics of the datasets before implementing the algorithm. Insufficient and poor-quality data are ultimately useless or have little value for the application. Datasets should have sufficient information to get insights for making decisions. Therefore, handling the datasets properly and fitting the appropriate algorithm can make robust machine learning-based intelligent applications. Nevertheless, the inter-disciplinary cooperation and continuous investigation and improvement of algorithms will be well-positioned to sustain its revolutionary influence in the field of artificial and machine learning.

## 7. Conclusion

In our study, we have conducted an extensive overview of machine learning and neural networks, a vital part of deep learning. We have explained the major algorithms that are most applicable in various fields. Understanding the fundamental concepts and their applicability will help the education and research community, practitioners, and research scholars gain insights into each algorithm, which has been discussed here. The popularity of the ML index shows that supervised learning is the most popular based on use and applicability.

Moreover, we have discussed various applications and their suitable algorithms. Different techniques, such as dataset preprocessing and feature selection, have also been discussed. Various activation functions in neural networks have also been discussed to clarify their applicability in various experiments. Nevertheless, we identified challenges and future directions to make burly and highly accurate with biased-free intelligent applications.

## References

[1] Tom M. Mitchell, *Machine Learning*, McGraw-Hill, 2017. [Publisher Link]

[2] Aurélien Géron, *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, O'Reilly Media, Inc., 2019. [Google Scholar] [Publisher Link]

[3] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, *Introduction to Data Mining*, Pearson Education Limited, 2019. [Google Scholar] [Publisher Link]

[4] Elaine Rich, Kevin Knight, and Shivashankar B. Nair, *Artificial Intelligence*, 3rd ed., TATA McGraw-Hill, 2009. [Publisher Link]

[5] Harry Henderson, *Artificial Intelligence*, Milestones in Discovery and Invention, Infobase Publishing, 2007. [Google Scholar] [Publisher Link]

[6] Mahdi Rezaei, and Reinhard Klette, *Computer Vision for Driver Assistance*, Springer International Publishing, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[7] Taiwo Oladipupo Ayodele, *Types of Machine Learning Algorithms*, New Advances in Machine Learning, pp. 19-48, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[8] Zhi-Hua Zhou, *Ensemble Methods: Foundations and Algorithms*, CRC Press, 2012. [Google Scholar] [Publisher Link]

[9] Rahul Saxena, How Decision Tree Algorithm Works, Dataaspirant, 2017. [Online]. Available: https://dataaspirant.com/how-decision-tree-algorithm-works/

[10] Aurélien Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*, O'Reilly Media, Inc., 2022. [Google Scholar] [Publisher Link]

[11] Antonio Gulli, and Sujit Pal, *Deep Learning with Keras: Implementing Deep Learning Models and Neural Networks with the Power of Python*, Packt Publishing Ltd, 2017. [Google Scholar] [Publisher Link]

[12] Tom M. Mitchell, *The Discipline of Machine Learning*, Carnegie Mellon University, School of Computer Science, Machine Learning Department, 2006. [Google Scholar] [Publisher Link]

[13] Iqbal H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," *SN Computer Science*, vol. 2, no. 3, pp. 1-21, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Diah Puspitasari et al., "Heart Disease: Application of the K-Nearest Neighbor (KNN) Method," *International Information and Engineering Technology Association*, vol. 29, no. 4, pp. 1275-1281, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[15] Gongde Guo et al., "KNN Model-Based Approach in Classification," *OTM Confederated International Conferences*, '*On the Move to Meaningful Internet Systems*', Catania, Italy, vol. 1, pp. 986-996, 2003. [CrossRef] [Google Scholar] [Publisher Link]

[16] V. Kecman, *Support Vector Machines-An Introduction*, Support Vector Machines: Theory and Applications, Springer, Berlin, Heidelberg, pp. 1-47, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[17] Dustin Boswell, "Introduction to Support Vector Machines," *Department of Computer Science and Engineering, University of California San Diego*, vol. 11, pp. 16-17, 2002. [Google Scholar]

[18] Vikramaditya Jakkula, "Tutorial on Support Vector Machine (SVM)," *School of EECS, Washington State University*, vol. 37, no. 2-5, 2006. [Google Scholar]

[19] Kecheng Qu, "Research on Linear Regression Algorithm," *2nd International Conference on Physics, Computing and Mathematical*, vol. 395, pp. 1-6, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[20] Supichaya Sunthornjittanon, "*Linear Regression Analysis on Net Income of an Agrochemical Company in Thailand*," Portland State University, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[21] Riccardo Trinchero, and Flavio Canavero, "Machine Learning Regression Techniques for the Modeling of Complex Systems: An Overview," *IEEE Electromagnetic Compatibility Magazine*, vol. 10, no. 4, pp. 71-79, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[22] Shruthi H. Shetty et al., *Supervised Machine Learning: Algorithms and Applications*, Fundamentals and Methods of Machine and Deep Learning: Algorithms, Tools and Applications, pp. 1-16, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[23] F.Y. Osisanwo et al., "Supervised Machine Learning Algorithms: Classification and Comparison," *International Journal of Computer Trends and Technology*, vol. 48, no. 3, pp. 128-138, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[24] S.B. Kotsiantis, *Supervised Machine Learning: A Review of Classification Techniques*, Frontiers in Artificial Intelligence and Applications, Emerging Artificial Intelligence Applications in Computer Engineering, vol. 160, pp. 3-24, 2007. [Google Scholar] [Publisher Link]

[25] José-Luis Solorio-Ramírez et al., "Random Forest Algorithm for the Classification of Spectral Data of Astronomical Objects," *Algorithms*, vol. 16, no. 6, pp. 1-16, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[26] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam, "The K-Means Algorithm: A Comprehensive Survey and Performance Evaluation," *Electronics*, vol. 9, no. 8, pp. 1-12, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[27] Xin Jin, and Jiawei Han, "K-Means Clustering," *Encyclopedia of Machine Learning*, pp. 563-564, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[28] Jonathon Shlens, "A Tutorial on Principal Component Analysis," *arXiv Preprint*, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[29] Andrzej Maćkiewicz, and Waldemar Ratajczak, "Principal Components Analysis (PCA)," *Computers & Geosciences*, vol. 19, no. 3, pp. 303-342, 1993. [CrossRef] [Google Scholar] [Publisher Link]

[30] Shuangshuang Chen, and Wei Guo, "Auto-Encoders in Deep Learning-A Review with New Perspectives," *Mathematics*, vol. 11, no. 8, pp. 1-54, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[31] Pascal Vincent et al., "Extracting and Composing Robust Features with Denoising Autoencoders," *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, pp. 1096-1103, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[32] Andrew Ng, "Sparse Autoencoder," *CS294A Lecture Notes*, 2011. [Google Scholar] [Publisher Link]

[33] Diederik P. Kingma, and Max Welling, "Auto-Encoding Variational Bayes," *arXiv Preprint*, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[34] Jonathan Masci et al., "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction," *International Conference on Artificial Neural Networks*, Espoo, Finland, pp. 52-59, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[35] Salah Rifai et al., "Contractive Auto-Encoders: Explicit Invariance During Feature Extraction," *Proceedings of the 28th International Conference on International Conference on Machine Learning*, Bellevue, Washington, USA, pp. 833-840, 2011. [Google Scholar] [Publisher Link]

[36] Pascal Vincent et al., "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371-3408, 2010. [Google Scholar] [Publisher Link]

[37] Yoshua Bengio, Aaron Courville, and Pascal Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798-1828, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[38] Bruno A. Olshausen, and David J. Field, "Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1?," *Vision Research*, vol. 37, no. 23, pp. 3311-3325, 1997. [CrossRef] [Google Scholar] [Publisher Link]

[39] Nitish Srivastava, and Ruslan Salakhutdinov, "Multimodal Learning with Deep Boltzmann Machines," *Advances in Neural Information Processing Systems*, vol. 25, pp. 2222-2230, 2012. [Google Scholar] [Publisher Link]

[40] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le, "Sequence to Sequence Learning with Neural Networks," *Advances in Neural Information Processing Systems*, vol. 27, pp. 3104-3112, 2014. [Google Scholar] [Publisher Link]

[41] Kyunghyun Cho et al., “Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation,” *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, pp. 1724-1734, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[42] Arash Vahdat, and Jan Kautz, “NVAE: A Deep Hierarchical Variational Autoencoder,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 19667-19679, 2020. [Google Scholar] [Publisher Link]

[43] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou, “Isolation Forest,” *8th IEEE International Conference on Data Mining*, Pisa, Italy, pp. 413-422, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[44] Y. Wang, J. Wong, and A. Miner, “Anomaly Intrusion Detection Using One-Class SVM,” *Proceedings from the 5th Annual IEEE SMC Information Assurance Workshop*, pp. 358-364, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[45] Kun-Lun Li et al., “Improving One-Class SVM for Anomaly Detection,” *Proceedings of the International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, Xi'an, vol. 5, pp. 3077-3081, 2003. [CrossRef] [Google Scholar] [Publisher Link]

[46] Markus M. Breunig et al., “LOF: Identifying Density-Based Local Outliers,” *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Dallas Texas, USA, pp. 93-104, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[47] Laurens Van Der Maaten, and Geoffrey Hinton, “Visualizing Data Using T-SNE,” *Journal of Machine Learning Research*, vol. 9, pp. 2579-2605, 2008. [Google Scholar] [Publisher Link]

[48] Martin Wattenberg, Fernanda Viégas, and Ian Johnson, “How to Use t-SNE Effectively,” *Distill*, vol. 1, no. 10, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[49] Haoyu Xie, “Research and Case Analysis of Apriori Algorithm Based on Mining Frequent Itemsets,” *Open Journal of Social Sciences*, vol. 9, no. 4, pp. 458-463, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[50] Hongfei Xu et al., “Research on an Improved Association Rule Mining Algorithm,” *IEEE International Conference on Power Data Science*, Taizhou, China, pp. 37-42, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[51] Massih-Reza Amini et al., “Self-Training: A Survey,” *arXiv Preprint*, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[52] Zixing Song et al., “Graph-Based Semi-Supervised Learning: A Comprehensive Review,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 11, pp. 8174-8194, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[53] L. Ruthotto, and E. Haber, “An Introduction to Deep Generative Modeling,” *GAMM-Mitteilungen*, vol. 44, no. 2, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[54] Xibin Dong et al., “A Survey on Ensemble Learning,” *Frontiers of Computer Science*, vol. 14, pp. 241-258, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[55] Thomas G. Dietterich, “Ensemble Methods in Machine Learning,” *International Workshop on Multiple Classifier Systems*, Cagliari, Italy, vol. 1, pp. 1-15, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[56] David Opitz, and Richard Maclin, “Popular Ensemble Methods: An Empirical Study,” *Journal of Artificial Intelligence Research*, vol. 11, pp. 169-198, 1999. [CrossRef] [Google Scholar] [Publisher Link]

[57] Ying Zhou, Thomas A. Mazzuchi, and Shahram Sarkani, “M-Adaboost-A Based Ensemble System for Network Intrusion Detection,” *Expert Systems with Applications*, vol. 162, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[58] Batta Mahesh, “Machine Learning Algorithms-A Review,” *International Journal of Science and Research*, vol. 9, no. 1, pp. 381-386, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[59] A.D. Dongare, R.R. Kharde, and Amit D. Kachare, “Introduction to Artificial Neural Network,” *International Journal of Engineering and Innovative Technology*, vol. 2, no. 1, pp. 189-194, 2012. [Google Scholar]

[60] Yu-chen Wu, and Jun-wen Feng, “Development and Application of Artificial Neural Network,” *Wireless Personal Communications*, vol. 102, no. 2, pp. 1645-1656, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[61] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi, “Understanding of A Convolutional Neural Network,” *International Conference on Engineering and Technology*, Antalya, Turkey, pp. 1-6, 2017. [CrossRef] [Publisher Link]

[62] Keiron O'Shea, and Ryan Nash, “An Introduction to Convolutional Neural Networks,” *arXiv Preprint*, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[63] Laith Alzubaidi et al., “Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions,” *Journal of Big Data*, vol. 8, no. 1, pp. 1-74, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[64] Anjar Wanto, Yuhandri Yuhandri, and Okfalisa Okfalisa, “RetMobileNet: A New Deep Learning Approach for Multi-Class Eye Disease Identification,” *Artificial Intelligence Review*, vol. 38, no. 4, pp. 1055-1067, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[65] Yoshua Bengio, “Learning Deep Architectures for AI,” *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1-127, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[66] Raffaele Pugliese, Stefano Regondi, and Riccardo Marini, “Machine Learning-Based Approach: Global Trends, Research Directions, and Regulatory Standpoints,” *Data Science and Management*, vol. 4, pp. 19-29, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[67] S. Reema Sree, S.B. Vyshnavi, and N. Jayapandian, "Real-World Application of Machine Learning and Deep Learning," *International Conference on Smart Systems and Inventive Technology*, Tirunelveli, India, pp. 1069-1073, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[68] Vineet Chaoji, Rajeev Rastogi, and Gourav Roy, "Machine Learning in the Real World," *Proceedings of the VLDB Endowment*, vol. 9, no. 13, pp. 1597-1600, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[69] George Tzanis et al., "Modern Applications of Machine Learning," *Proceedings of the 1st Annual SEERC Doctoral Student Conference*, vol. 1, no. 1. pp. 1-10, 2006. [Google Scholar]

[70] Shahid Tufail et al., "Advancements and Challenges in Machine Learning: A Comprehensive Review of Models, Libraries, Applications, And Algorithms," *Electronics*, vol. 12, no. 8, pp. 1-43, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[71] Hafsa Habehh, and Suril Gohel, "Machine Learning in Healthcare," *Current Genomics*, vol. 22, no. 4, pp. 291-300, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[72] Qi An et al., "A Comprehensive Review on Machine Learning in Healthcare Industry: Classification, Restrictions, Opportunities and Challenges," *Sensors*, vol. 23, no. 9, pp. 1-21, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[73] Yang Lu, and Li Da Xu, "Internet of Things (IoT) Cybersecurity Research: A Review of Current Research Topics," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2103-2115, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[74] Yoram Reich, and S.V. Barai, "Evaluating Machine Learning Models for Engineering Problems," *Artificial Intelligence in Engineering*, vol. 13, no. 3, pp. 257-272, 1999. [CrossRef] [Google Scholar] [Publisher Link]

[75] Yoram Reich, "Machine Learning Techniques for Civil Engineering Problems," *Computer-Aided Civil and Infrastructure Engineering*, vol. 12, no. 4, pp. 295-310, 1997. [CrossRef] [Google Scholar] [Publisher Link]

[76] Huu-Tai Thai, "Machine Learning for Structural Engineering: A State-of-the-Art Review," *Structures*, vol. 38, pp. 448-491, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[77] Guannan Huang et al., "Application of Machine Learning in Material Synthesis and Property Prediction," *Materials*, vol. 16, no. 17, pp. 1-30, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[78] Sabrina C. Shen et al., "Computational Design and Manufacturing of Sustainable Materials through First-Principles and Materiomics," *Chemical Reviews*, vol. 123, no. 5, pp. 2242-2275, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[79] Baiyu Lu, "The Application of Machine Learning in Chemical Engineering: A Literature Review," *Proceedings of the 9th International Conference on Humanities and Social Science Research*, Atlantis Press, pp. 57-66, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[80] Periklis Gogas, and Theophilos Papadimitriou, "Machine Learning in Economics and Finance," *Computational Economics*, vol. 57, no. 1, pp. 1-4, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[81] Komal et al., *Opportunities and Challenges of AI/ML in Finance*, The Impact of AI Innovation on Financial Sectors in the Era of Industry 5.0, pp. 238-260, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[82] Konstantinos G. Liakos et al., "Machine Learning in Agriculture: A Review," *Sensors*, vol. 18, no. 8, pp. 1-29, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[83] Wenxiang Liu et al., "Applications of Machine Learning in Computational Nanotechnology," *Nanotechnology*, vol. 33, no. 16, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[84] Avnish Pareek et al., "Nanotechnology for Green Applications: How Far on the Anvil of Machine Learning!," *Biobased Nanotechnology for Green Applications*, pp. 1-38, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[85] Rahul Rai et al., "Machine Learning in Manufacturing and Industry 4.0 Applications," *International Journal of Production Research*, vol. 59, no. 16, pp. 4773-4778, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[86] Tingting Chen et al., "Machine Learning in Manufacturing Towards Industry 4.0: From 'For Now' to 'Four-Know'," *Applied Sciences*, vol. 13, no. 3, pp. 1-32, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[87] Massimo Bertolini et al., "Machine Learning for Industrial Applications: A Comprehensive Literature Review," *Expert Systems with Applications*, vol. 175, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[88] Thorsten Wuest et al., "Machine Learning in Manufacturing: Advantages, Challenges, and Applications," *Production & Manufacturing Research*, vol. 4, no. 1, pp. 23-45, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[89] Sam Yang, Bjorn Vaagensmith, and Deepika Patra, "Power Grid Contingency Analysis with Machine Learning: A Brief Survey and Prospects," *Resilience Week*, Salt Lake City, UT, USA, pp. 119-125, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[90] Kuldeep Singh Kaswan, Jagjit Singh Dhatterwal, and Rudra Pratap Ojha, *AI in Personalized Learning*, Advances in Technological Innovations in Higher Education, CRC Press, pp. 103-117, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[91] Md. Zahurul Haque, "E-Commerce Product Recommendation System Based on ML Algorithms," *arXiv Preprint*, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[92] Badrul Sarwar et al., "Analysis of Recommendation Algorithms for E-Commerce," *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pp. 158-167, 2000. [CrossRef] [Google Scholar] [Publisher Link]

[93] Yizhao Ni et al., “An End-to-End Machine Learning System for Harmonic Analysis of Music,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 6, pp. 1771-1783, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[94] Iria Santos et al., “Artificial Neural Networks and Deep Learning in the Visual Arts: A Review,” *Neural Computing and Applications*, vol. 33, pp. 121-157, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[95] Ravil I. Mukhamediev et al., “Review of Artificial Intelligence and Machine Learning Technologies: Classification, Restrictions, Opportunities and Challenges,” *Mathematics*, vol. 10, no. 15, pp. 1-25, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[96] Enrico Barbierato, and Alice Gatti, “The Challenges of Machine Learning: A Critical Review,” *Electronics*, vol. 13, no. 2, pp. 1-30, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[97] Eric Horvitz, “Machine Learning, Reasoning, and Intelligence in Daily Life: Directions and Challenges,” *Proceedings of*, Microsoft, vol. 360, 2006. [Google Scholar]

[98] Niki Parmar et al., “Image Transformer,” *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, PMLR 80, pp. 4055-4064, 2018. [Google Scholar] [Publisher Link]

[99] Jung Min Ahn, Jungwook Kim, and Kyunghyun Kim, “Ensemble Machine Learning of Gradient Boosting (Xgboost, Lightgbm, Catboost) and Attention-Based CNN-LSTM for Harmful Algal Blooms Forecasting,” *Toxins*, vol. 15, no. 10, pp. 1-15, 2023. [CrossRef] [Google Scholar] [Publisher Link]