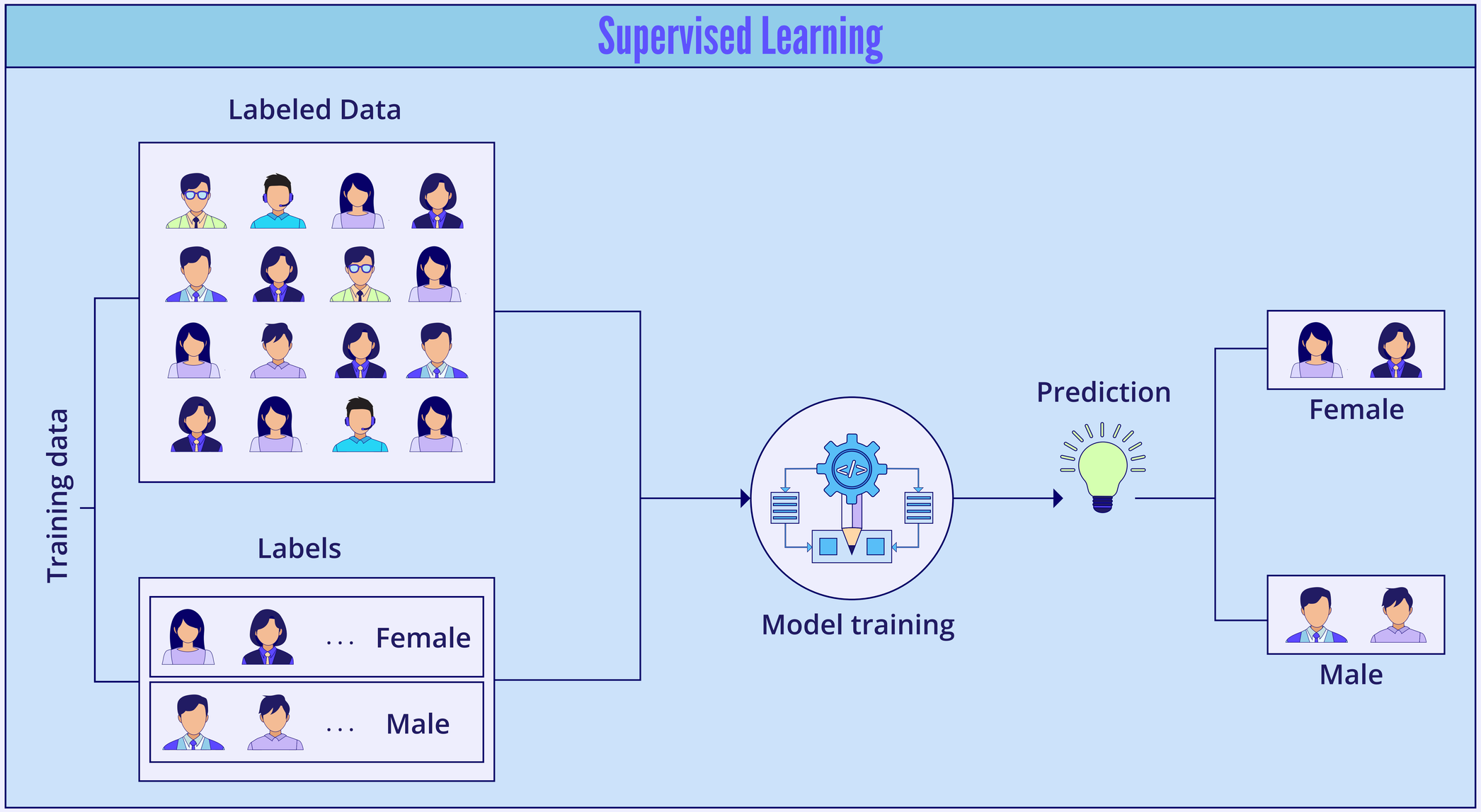
Supervised vs. Unsupervised Learning

Machine learning is a field of artificial intelligence that includes the development of algorithms and statistical models, which enable computer systems to learn and improve based on their experiences. There are two types of machine learning: supervised and unsupervised.

Supervised Learning

The model is trained using a labeled dataset where each data point is associated with a known outcome, allowing it to identify patterns and make predictions for the new and unseen data points



Common languages, libraries, and tools:

Here are some prominent languages and libraries for implementing supervised learning algorithms.
Python: Scikit-learn, TensorFlow, and Keras.
R: Caret and Glmnet.
Matlab: Statistics and Machine Learning Toolbox.

Supervised vs. Unsupervised Learning

educative

Algorithms

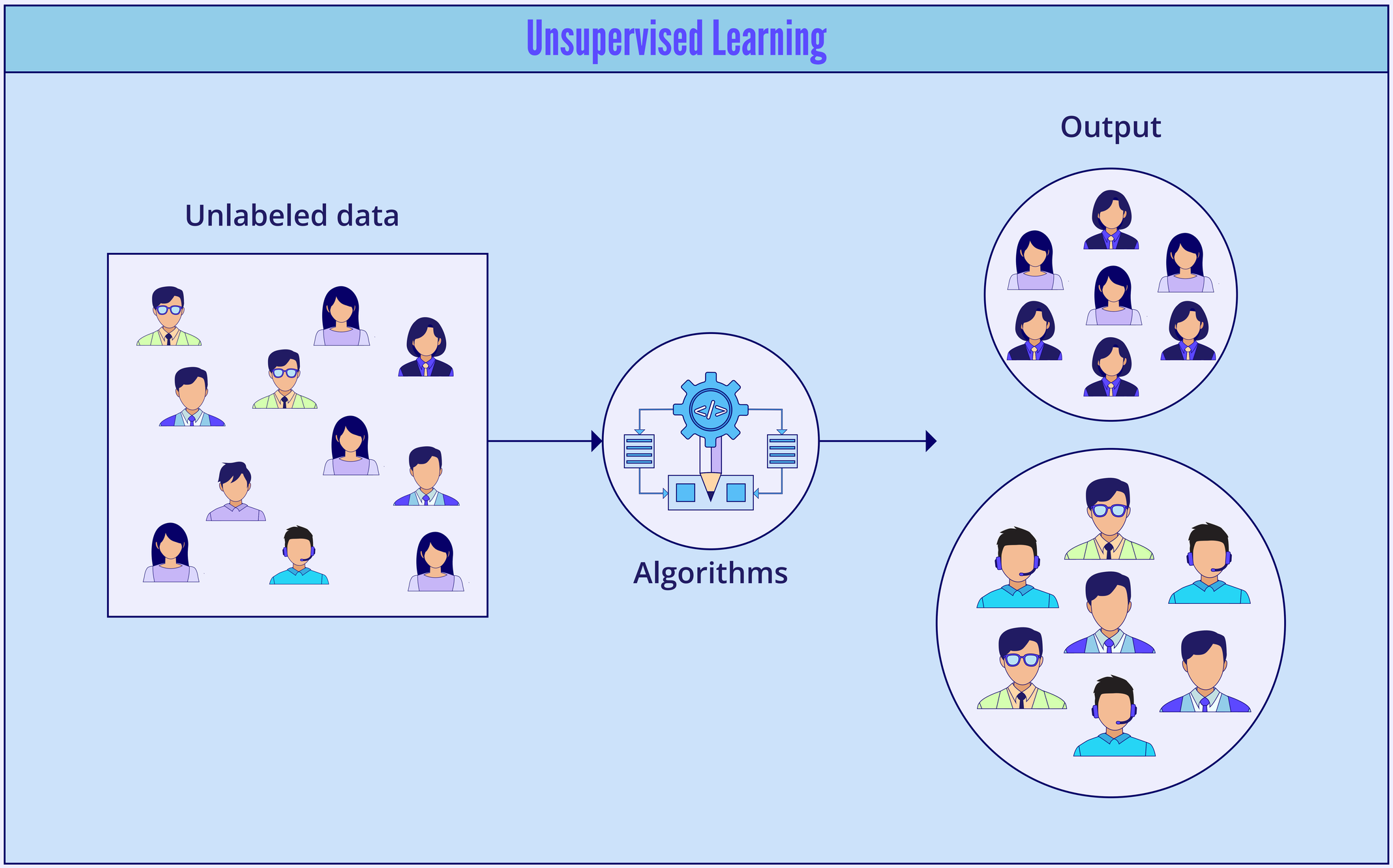
Type	Name	Description	Use Cases	Libraries / Functions	Pros	Cons
Classification	Logistic Regression	Determines the probability that a given input belongs to a specific class	Spam email filtering Customer churn prediction Loan default prediction Ad click prediction	Python: `sklearn.linear_model.LogisticRegression` R package: `glm` Matlab: `fitglm`	Output probabilities provide information about the relevance of features	Assumes a linear decision boundary and does not perform well with nonlinear relationships
	Decision Trees	Partition data recursively to minimize impurity or maximize information gain at each split	Cancer classification Customer segmentation Equipment failure prediction Credit card fraud detection	Python: `sklearn.tree.DecisionTreeClassifier` R package:`rpart` Matlab:`fitctree`	Handles numerical and categorical data, making it easy to understand and interpret	Overfitting is common, and slight differences in data can significantly impact
	Random Forest	Ensemble techniques that create many decision trees during classification training	Insurance claim fraud detection Stock price forecasting Customer lifetime value prediction Disease outbreak prediction	Python: `sklearn.ensemble.RandomForestClassifier` R package: `randomForest` Matlab: `TreeBagger`	Compared to individual decision trees, it reduces overfitting and performs well with high-dimensional data	It is computationally expensive and less interpretable than individual decision trees
	Gradient Boosting Regression	Creates a sequence of decision trees in sequential order, optimizing a differentiable loss function	Tumor growth prediction Demand forecasting Sales forecasting Sales data analysis	Python: `sklearn.ensemble.GradientBoostingRegressor` R package: `gbm` Matlab: `fitensemble`	Frequently produces higher accuracy than random forests and handles missing data effectively	Hyperparameter sensitivity, computational costs, and overfitting are all potential issues
	XGBoost	Optimized implementation of gradient boosting with efficient parallel processing and regularization	Quality control Click-through rate prediction Healthcare outcome prediction Energy consumption prediction	Python: `xgboost.XGBClassifier` R package: `xgboost` Matlab: `fitensemble`	State-of-the-art performance in several machine learning challenges and highly efficient and scalable algorithm	Hyperparameters must be carefully tuned, and computational resources may be needed more than other methods
Regression	Linear Regression	Represents the relationship between dependent and independent variables with a linear equation	House price prediction Stock market trend prediction Financial forecasting Population growth prediction	Python: `sklearn.linear_model.LinearRegression` R package: `lm` Matlab: `fitlm`	Simple and computationally efficient	Assumes linear relationship and is sensitive to outliers
	Ridge Regression	Modified version of linear regression with a penalty term to prevent overfitting	Environmental impact analysis Healthcare resource planning Energy consumption modeling Loan approval prediction	Python: `sklearn.linear_model.Ridge` R package: `glmnet` Matlab: `ridge`	Handles multicollinearity effectively and reduces model complexity	Tuning the regularization parameter is required, and unnecessary features are not eliminated
	Lasso Regression	Regularized variant of linear regression with an L1 penalty to promote sparsity in coefficient estimations	Market trend analysis Price optimization Inventory management Sentiment analysis	Python: `sklearn.linear_model.Lasso` R package: `glmnet` Matlab: `lasso`	Selects features and deals with multicollinearity	Regularization parameters must be tuned to reduce coefficients to zero, which results in feature elimination

Supervised vs. Unsupervised Learning

educative

Unsupervised Learning

It involves training on data without explicit labels, requiring the model to find intrinsic structures or patterns without predefined guidance.



Common languages, libraries, and tools:

Here are some prominent languages and libraries for implementing unsupervised learning algorithms.
Python: Libraries such as scikit-learn, TensorFlow Probability, and PyTorch.
R: Packages like cluster, FactoMineR, and arules.
Matlab: Functions available in the Statistics and Machine Learning Toolbox.

Supervised vs. Unsupervised Learning

educative

Algorithms

Type	Name	Description	Use Cases	Libraries / Functions	Pros	Cons
Clustering	K-means Clustering	Partitioning method that divides data points into k clusters based on centroid similarity	Customer segmentation Image compression Market basket analysis Anomaly detection	Python: `sklearn.cluster.KMeans` R package: `kmeans` (stats package) Matlab: `kmeans` (Statistics and Machine Learning Toolbox)	Simple, intuitive, and computationally efficient	The number of clusters must be identified in advance and is sensitive to the initial cluster centers
	Hierarchical Clustering	Builds a tree of clusters by merging or splitting them based on distance metrics	Gene expression analysis Document clustering Species classification Social network analysis	Python: `scipy.cluster.hierarchy.linkage` R package: `hclust` (stats package) Matlab: `linkage` (Statistics and Machine Learning Toolbox)	Does not require the number of clusters to be provided in advance and includes a dendrogram for visualization	Less scalable than K-means and can be computationally expensive for large datasets
	Gaussian Mixture tModels (GMM)	Probabilistic model representing each cluster with a Gaussian distribution	Image recognition Speech recognition Fraud detection Natural language processing	Python: `sklearn.mixture.GaussianMixture` R package: `Mclust` (mclust package) Matlab: `fitgmdist` (Statistics and Machine Learning Toolbox)	Cluster covariance is flexible and can handle varied cluster shapes	Sensitive to initialization, may converge to local optima, and is computationally expensive for large data sets
Association Rules	Apriori Algorithm	Identifies frequent itemsets in a dataset and extracts association rules based on support and confidence thresholds	Recommender systems Web usage mining Cross-selling analysis Customer behavior analysis	Python: `mlxtend.frequent_patterns.apriori` R package: `arules` (`arules` package) Matlab: Not commonly available	Simple to use, and it efficiently identifies frequent item sets	Computationally expensive for huge datasets but produces a large number of candidate itemsets
	FP-Growth Algorithm	An efficient algorithm for mining frequent itemsets by constructing a compressed dataset representation using a prefix tree (FP-tree)	Association rule mining Customer segmentation Web clickstream analysis Sales transaction analysis	Python: `mlxtend.frequent_patterns.fpgrowth` R package: `fpgrowth` (`arules` package) Matlab: Not commonly available	Efficient, particularly for datasets with a high number of transactions or items	Requires more memory than Apriori and may be slower with dense datasets
	Eclat Algorithm	Association rule mining algorithm that uses transaction identifiers and a bitmap representation to find frequent itemsets	Market basket analysis Recommendation systems Web usage mining Collaborative filtering	Python: Not commonly available. R package: `eclat` (`arules` package) Matlab: Not commonly available	Efficient in identifying common itemsets, especially for big transactional datasets	Limited availability in standard libraries may necessitate specific implementation, which is less common than Apriori or FP-growth
Dimensionality Reduction	Principal Component Analysis (PCA)	Linear transformation technique that projects high-dimensional data into a lower-dimensional subspace while preserving the maximum variance	Data visualization Noise reduction Feature extraction Data compression	Python: `sklearn.decomposition.PCA` R package: `prcomp` (stats package) Matlab: `pca` (Statistics and Machine Learning Toolbox)	Efficient for huge datasets, maintains global structure, and is commonly used for feature extraction	Assumes linear relationships; complex nonlinear systems may not be captured properly
	Independent Component Analysis (ICA)	Separates a multivariate signal into additive, independent components	Speech signal separation Feature extraction from sensor data Blind source separation (BSS) in finance Image denoising	Python: `sklearn.decomposition.FastICA` R package: `fastICA` (fastICA package) Matlab: `fastica` (Statistics and Machine Learning Toolbox)	Assumes less restrictive independence assumptions than PCA, which is useful for source separation	Sensitive to the choice of the number of components and may not perform well when independence assumptions are violated
	Uniform Manifold Approximation and Projection (UMAP)	Nonlinear dimensionality reduction technique that preserves both local and global structure in the data, often used for visualization purposes	Pattern recognition Dimensionality reduction Clustering analysis Embedding high-dimensional data	Python: `umap.UMAP` R package: Not commonly available Matlab: Not commonly available	Preserves complex local and global structure, efficient for large datasets	Requires careful tuning of hyperparameters that may be sensitive to parameter choices

Supervised vs. Unsupervised Learning

educative

Key Takeaways

	Supervised	Unsupervised
Input Data	Labeled	Unlabeled
Training Process	Finding patterns between input features and target labels during training to predict classes or values for similar unseen inputs	Identifying relationships among input instances during training to group them into clusters based on similarity
Output	Known in advance	Unknown, the result may be arbitrary
Use Case	If high-quality labeled data is available	If high-quality labeled data is not available