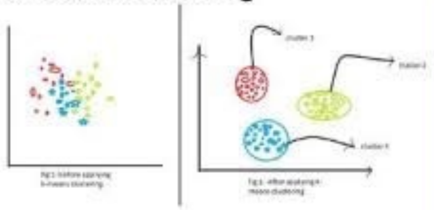
Cheatsheet
Machine Learning Algorithms

Algorithm	Type	Best Use Case	Key Formula or Logic	Assumptions	Pros	Cons	When Not to Use	Real-World Example
Linear Regression	Supervised	Predicting continuous values	Y = b0 + b1X + b2X2 + ...	Linearity, independence	Simple, interpretable, fast	Sensitive to outliers, non-linear limits	Data with strong non-linearity	House price prediction
Logistic Regression	Supervised	Binary classification	P = 1 / (1 + e^-(b0 + b1X + ...))	Log-odds linearity	Probabilistic, interpretable	Weak with non-linear boundaries	Data is highly non-linear	Spam detection
Decision Tree	Supervised	Classification and regression	Recursive binary split	None	Easy to interpret	Overfitting, unstable	Noisy or complex datasets	Loan default prediction
Random Forest	Supervised	Ensemble accuracy	Bagging plus averaging trees	Tree independence	High accuracy, robust	Slower, less interpretable	Need real-time results	Fraud detection
Gradient Boosting	Supervised	High-performance modeling	Additive trees minimizing loss	Sequential dependency	State-of-the-art accuracy	Overfitting, needs tuning	When interpretability matters	Credit scoring
SVM	Supervised	Max-margin classification	Maximize margin using kernel trick	Separability, scaling	Works in high dimensions	Slow on large data	Large noisy datasets	Facial recognition
KNN	Supervised	Few-shot classification	Distance-based majority vote	Feature scaling	Simple, no training phase	Slow, noise sensitive	High-dimensional noisy data	Recommender systems
Naive Bayes	Supervised	Text classification	Bayes theorem plus feature independence	Independent features	Fast, good with text data	Fails with correlated features	Feature dependency present	Sentiment analysis
K-Means	Unsupervised	Customer segmentation	Minimize intra-cluster distance	Spherical, equal clusters	Fast, easy to implement	Needs K, sensitive to scale	Non-spherical data	Customer segmentation
Hierarchical Clustering	Unsupervised	Data structure understanding	Nested dendrogram	Distance metric	No need for K, visual	Memory and compute intensive	Very large datasets	Gene expression analysis
PCA	Dimensionality Reduction	Reducing feature dimensionality	Eigenvectors of covariance matrix	Large variance important	Noise reduction, speed-up	Hard to interpret	All features are important	Image compression
Neural Networks (MLP)	Supervised	Complex pattern modeling	Weighted sums plus activation functions	Enough data, scaling	Non-linear learning power	Needs large data and tuning	Small data, low compute	Image classification
CNN	Supervised	Image, video, spatial data	Convolution plus pooling layers	Grid-like spatial data	Excellent for images	High resource demand	Sequence or text data	Self-driving vision
RNN	Supervised	Sequence modeling	Feedback loops over time	Sequential structure	Time-series and text ready	Vanishing gradient	Long sequences	Stock prediction
Transformer (BERT, GPT)	Supervised or Self-supervised	NLP tasks, chat, translation	Attention mechanism plus position encoding	Large training data	Long context, fast	Heavy compute, large model	Small projects	ChatGPT, translation tools
Autoencoders	Unsupervised	Compression and anomaly detection	Encoder-decoder plus reconstruction loss	Symmetric network	Effective denoising	Can overfit, black-box	When no compression needed	Fraud detection
DBSCAN	Unsupervised	Arbitrary shape clustering	Density-based region growing	Cluster density	Noise tolerant, shape-flexible	Fails on varying density	Sparse high-dimensional data	Geo-spatial clustering

Analytics Vidhya

Top 9 Descriptive Models

K-means clustering

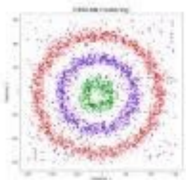


What it does: partitions data into K compact clusters around centroids.
Use when: clusters are roughly spherical, you want speed at scale.
Watch out: sensitive to K and outliers.

Hierarchical agglomerative clustering

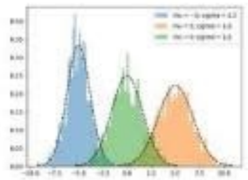
What it does: builds a tree (dendrogram) of groups from bottom up.
Use when: you want multi-level structure and don't know K upfront.
Watch out: can be slow on large datasets; distance/linkage choices matter.

DBSCAN



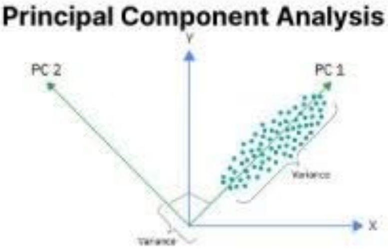
What it does: finds dense regions and labels low-density points as noise.
Use when: clusters are arbitrary shapes and you want outlier detection.
Watch out: epsilon/minPts tuning is dataset-specific.

Gaussian Mixture Models (GMM)



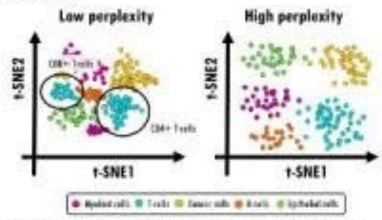
What it does: soft clusters via a mixture of Gaussians with EM.
Use when: overlapping clusters and you want membership probabilities.
Watch out: assumes Gaussian components; can overfit without care.

Principal Component Analysis



What it does: linear dimensionality reduction along maximum variance axes.
Use when: you need fast compression, denoising, or interpretable components.
Watch out: linear only; scale your features.

t-SNE

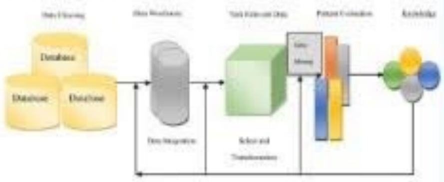


What it does: non-linear embedding that preserves local neighborhoods for visualization.
Use when: 2D/3D plots of complex manifolds.
Watch out: purely for visualization; perplexity and random seed change the look.

UMAP

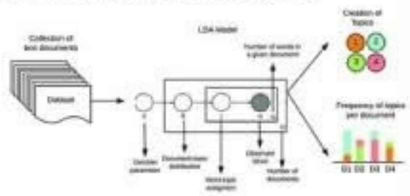
What it does: faster manifold learning, preserves global+local structure better than t-SNE in many cases.
Use when: you need quick, stable embeddings for exploration.
Watch out: several hyperparameters; still primarily for visualization.

Association rule mining



What it does: uncovers "items that occur together" patterns like A ⇒ B.
Use when: market baskets, click bundles, co-occurrence analysis.
Watch out: support/confidence alone can be misleading; use lift/conviction too.

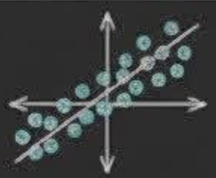
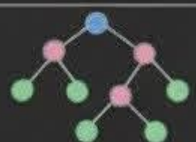
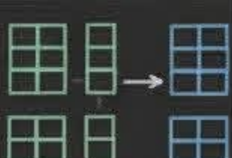
Latent Dirichlet Allocation



What it does: topic modeling documents as mixtures of latent topics.
Use when: summarizing large text corpora or building interpretable themes.
Watch out: number of topics and priors matter; preprocessing is crucial.

Time Complexity of 10 Most Popular ML Algorithms

MD ISMAIL SOJAL

		Training	*Inference*
	Linear Regression (OLS)	$O(nm^2 + m^3)$	$O(m)$
	Linear Regression (SGD)	$O(n_{epoch}nm)$	$O(m)$
	Logistic Regression (Binary)	$O(n_{epoch}nm)$	$O(m)$
	Logistic Regression (Multiclass OvR)	$O(n_{epoch}nmc)$	$O(mc)$
	Decision Tree	$O(n \cdot log(n) \cdot m)$ $O(n^2 \cdot m)$* Worst case	$O(d_{tree})$
	Random Forest Classifier	$O(n_{trees} \cdot n \cdot log(n) \cdot m)$	$O(n_{trees} \cdot d_{tree})$
	Support Vector Machines (SVMs)	$O(n^2m + n^3)$	$O(m \cdot n_{SV})$
	k-Nearest Neighbors	—	$O(nm)$
$P(B\|A) = \frac{P(B \cap A)}{P(A)}$	Naive Bayes	$O(nm)$	$O(mc)$
	Principal Component Analysis (PCA)	$O(nm^2 + m^3)$	—
	t-SNE	$O(n^2m)$	—
	KMeans Clustering	$O(iknm)$	??

n: samples | m: dimensions | n_{epoch}: epochs | c: classes | d_{tree}: depth

n_{SV}: Support vectors | k: clusters | i: iterations