

VU LEARNING HUB

vulearninghub.xyz

HIGHLIGHTED HANDOUT

CS628

Machine Learning

CS628 by VU Learning HUB

Prepared for VU students • Free study resources

<https://vulearninghub.xyz>

CS628

Machine Learning

Lecture-Wise Study Handout
Virtual University of Pakistan

14 lectures • prepared for exam study 

Table of Contents

1. What Is Machine Learning?
2. Types of Learning
3. Linear Regression and Gradient Descent
4. Classification: Logistic Regression and Metrics
5. Trees, Forests and Boosting
6. SVMs, Kernels and k-Nearest Neighbours
7. Naive Bayes and Text Classification
8. Clustering: K-Means, Hierarchical, DBSCAN
9. Neural Networks and Deep Learning
10. Overfitting, Regularization and Honest Evaluation
11. Features: Engineering, PCA and Selection
12. Anomaly Detection, Recommenders and Time Series
13. Ethics, Fairness and Privacy in ML
14. Deploying ML: Drift, Monitoring and MLOps

About This Course

Machine learning is the science of getting computers to learn from data instead of being explicitly programmed. From spam filters and recommendations to medical diagnosis and self-driving cars, ML systems learn patterns from examples and apply them to new, unseen cases.

This handout covers the full ML pipeline: types of learning, regression and classification, key algorithms (trees, SVM, kNN, Bayes, neural networks), clustering, evaluation, overfitting, and practical topics like feature engineering and ethics. Each lecture gives the intuition, the key formulas in words, examples and exam pointers. 

Lecture 1: What Is Machine Learning?

Learning Objectives

- Define machine learning
- Describe the ML pipeline

Machine learning is defined as algorithms that improve their performance at a task with experience (data). Arthur Samuel's checkers program (1959) coined the term; Tom Mitchell's formal definition — a program learns from experience E with respect to task T and performance measure P if P improves with E — is the standard exam definition. ☰

ML differs from traditional programming: instead of hand-coding rules, we feed data to a learning algorithm that discovers the rules. This suits problems where rules are unknown or too complex to code — recognizing faces, translating languages, predicting prices — but it demands data, and its failures are data failures (biased data → biased models).

The ML pipeline organizes every project: collect data → clean and prepare → choose features → select and train a model → evaluate on unseen data → tune → deploy and monitor. Examiners love this pipeline: most applied questions are answered by locating the problem in the pipeline (is it a data problem, a model problem, or an evaluation problem?).

Key Definitions

Learning (Mitchell's definition) is defined as Mitchell's definition: a program learns if its performance P on task T improves with experience E

Key Points — very important for exams

ML = programs that improve with data (Samuel 1959; Mitchell's T/E/P definition).

Hand-coded rules → learned rules; needs data; biased data gives biased models.

Pipeline: collect → clean → features → train → evaluate → tune → deploy.

Review Questions

- ? How does ML differ from traditional programming?
- ? List the stages of an ML project.

Section Summary

ML learns rules from data; the pipeline — data to deployment — organizes every project.

Exam Pointers

- Define machine learning and describe the ML project pipeline.

Lecture 2: Types of Learning

Learning Objectives

- Distinguish learning paradigms
- Choose paradigms for problems

Supervised learning trains on labelled examples (input → correct output) to predict labels for new inputs: classification (discrete labels: spam/not spam) and regression (continuous values: house prices). It dominates applications but needs expensive labelled data.

Unsupervised learning finds structure in unlabelled data: clustering (grouping customers), dimensionality reduction (compressing features), association rules (market baskets). Reinforcement learning trains agents by reward and punishment — game playing (AlphaGo), robotics — learning policies through trial and error.

Choosing the paradigm is the first modelling decision — very important: labels available and prediction needed → supervised; exploring structure → unsupervised; sequential decisions with rewards → reinforcement. Semi-supervised learning (few labels + much unlabelled data) bridges the gap when labelling is costly.

Key Definitions

Supervised learning is defined as learning a mapping from inputs to known correct outputs using labelled training data

Reinforcement learning is defined as learning by maximizing cumulative reward through trial-and-error interaction with an environment

Key Points — very important for exams

Supervised: labelled data → classification/regression; needs labels.

Unsupervised: unlabelled data → clustering, dimensionality reduction, associations.

Reinforcement: rewards → policies (games, robots); semi-supervised bridges label scarcity.

Review Questions

- ? When would you use clustering instead of classification?
- ? What is semi-supervised learning good for?

Section Summary

Labels decide the paradigm: predict with supervision, explore without it, act for rewards.

Exam Pointers

- Compare supervised, unsupervised and reinforcement learning with two examples each.

Lecture 3: Linear Regression and Gradient Descent

Learning Objectives

- Fit and interpret linear regression
- Explain gradient descent

Linear regression models a continuous target as a weighted sum of features: $y = w \cdot x + b$. The algorithm finds weights minimizing the mean squared error (MSE) between predictions and actual values — via the closed-form normal equation or iterative gradient descent. Assumptions matter: linearity, independent errors, constant variance (homoscedasticity).

Gradient descent is ML's workhorse optimizer: start with random weights, repeatedly step downhill on the loss surface ($w := w - \alpha \cdot \nabla \text{loss}$). The learning rate α is critical — too large diverges, too small crawls; stochastic and mini-batch variants trade noise for speed. The same optimizer trains neural networks later in the course.

Evaluation uses MSE/RMSE and R-squared (fraction of variance explained). Polynomial regression extends the model to curves by adding x^2 , x^3 features — still linear in the weights, a subtle exam point — while regularization (next lectures) controls the resulting flexibility.

Key Definitions

Linear regression is defined as fitting $y = w \cdot x + b$ by minimizing mean squared error

Gradient descent is defined as iterative optimization stepping downhill on the loss: $w := w - \alpha$ times gradient

Key Points — very important for exams

Model: $y = w \cdot x + b$; fit by minimizing MSE (normal equation or gradient descent).

Gradient descent: step against the gradient; learning rate controls convergence.

Metrics: MSE/RMSE, R^2 ; polynomial features add curves while staying linear in weights.

Review Questions

- ? Why is the learning rate critical?
- ? What does $R^2 = 0.85$ mean?

Section Summary

Regression fits lines by minimizing squared error — optimized by walking downhill on the loss.

Exam Pointers

→ Explain linear regression, gradient descent and evaluation metrics.

Lecture 4: Classification: Logistic Regression and Metrics

Learning Objectives

- Build a logistic classifier
- Evaluate classifiers properly

Classification predicts discrete labels. Logistic regression — despite the name — is a classifier: it squeezes a linear score through the sigmoid function into a probability, deciding class by a threshold (usually 0.5). It is simple, interpretable and well-calibrated, making it the baseline every fancier model must beat.

Evaluation needs the confusion matrix: true/false positives/negatives. From it: accuracy (overall correct), precision (of predicted positives, how many truly positive), recall (of actual positives, how many found), and F1 (their harmonic mean). The precision–recall trade-off is central: a spam filter wants precision (never lose real mail); a cancer screen wants recall (never miss a case).

ROC curves and AUC summarize performance across thresholds; imbalanced data (fraud: 0.1% positive) makes accuracy misleading — a model predicting always-negative scores 99.9% — so precision, recall and AUC are the honest metrics. Very important: always match the metric to the mistake's cost.

Key Definitions

Logistic regression is defined as a classifier mapping a linear score through the sigmoid to a class probability

Confusion matrix is defined as the table of true/false positives/negatives underlying classification metrics

Key Points — very important for exams

Logistic regression: sigmoid on a linear score → probability → threshold; strong baseline.

Confusion matrix → accuracy, precision, recall, F1; match metric to error cost.

Imbalanced data kills accuracy's meaning — use precision/recall/AUC; ROC/AUC across thresholds.

Review Questions

- ? When is precision more important than recall?
- ? Why is 99% accuracy meaningless for fraud detection?

Section Summary

Classify with probabilities, and evaluate with the metric that matches the real-world cost of errors.

Exam Pointers

- Explain logistic regression and derive precision, recall, F1 from the confusion matrix.

Lecture 5: Trees, Forests and Boosting

Learning Objectives

- Build decision trees

- Explain ensemble methods

Decision trees split data recursively by the feature that best separates classes — choosing splits by information gain (entropy reduction) or Gini impurity. The result is human-readable: a flowchart of if-then rules. Strengths: interpretability, handling mixed data, no scaling needed. Weakness: deep trees memorize training data — high variance.

Random forests fix this with bagging: train many trees on bootstrapped samples with random feature subsets, then vote. Averaging decorrelates the trees' errors, slashing variance while keeping interpretability via feature importance. Out-of-bag samples give free validation. Forests are the default first-try algorithm for tabular data.

Gradient boosting (XGBoost, LightGBM) builds trees sequentially, each correcting its predecessors' errors — often the winning approach in competitions. The bias–variance story: single trees overfit (variance), boosting reduces bias, forests reduce variance. Exam favourite: explain why forests beat single trees.

Key Definitions

Random forest is defined as an ensemble of trees trained on bootstrapped samples with random feature subsets, voting the result

Gradient boosting is defined as sequential tree ensemble where each tree corrects previous errors

Key Points — very important for exams

Trees: recursive splits by information gain/Gini; interpretable but high-variance.

Random forests: bagged trees vote — variance collapses; feature importance; tabular-data default.

Boosting: sequential error-correction (XGBoost); bias–variance trade-off governs choice.

Review Questions

- ? Why do random forests overfit less than single trees?
- ? Compare bagging and boosting.

Section Summary

Trees decide by splitting; ensembles of trees — voted or boosted — are ML's workhorses.

Exam Pointers

→ Explain decision trees, random forests and boosting; discuss bias–variance.

Lecture 6: SVMs, Kernels and k-Nearest Neighbours

Learning Objectives

- Explain max-margin learning
- Apply kNN properly

Support Vector Machines find the hyperplane maximizing the margin — the gap between the decision boundary and the nearest points of each class (the support vectors). Maximizing the margin is structural risk minimization: the widest separator generalizes best. The C parameter trades margin width against training errors.

The kernel trick handles non-linearity: implicitly map data to high-dimensional space where it becomes linearly separable, computing only inner products via the kernel function (RBF, polynomial). You get non-linear power without the coordinates — elegant and exam-beloved. ☰

k-Nearest Neighbours is the opposite philosophy — lazy learning: store all training data, classify a new point by its k nearest neighbours' majority vote. No training time, but slow prediction, curse of dimensionality, and sensitivity to scaling (normalize features!) and to k (small k: noisy; large k: oversmoothed).

Key Definitions

Support Vector Machine is defined as the classifier maximizing the gap between the decision boundary and the nearest training points

Kernel trick is defined as implicit high-dimensional mapping enabling non-linear SVMs via inner products only

Key Points — very important for exams

SVM: widest-margin hyperplane; support vectors define it; C trades margin vs errors.

Kernel trick: non-linear separation via implicit high-dimensional mapping (RBF).

kNN: lazy, vote of k neighbours; needs scaling; k controls noise vs smoothness.

Review Questions

- ? What are support vectors?
- ? Why must features be scaled for kNN but not for trees?

Section Summary

SVMs maximize the margin (kernels add curves); kNN simply asks the neighbours.

Exam Pointers

→ Explain max-margin classification, the kernel trick, and kNN with its sensitivities.

Lecture 7: Naive Bayes and Text Classification

Learning Objectives

- Build a naive Bayes text classifier
- Explain TF-IDF

Naive Bayes applies Bayes' theorem with a bold assumption: features are conditionally independent given the class. Despite the naivety, it shines on high-dimensional sparse data — especially text — training in one pass with tiny data needs. Spam filtering was its classic triumph.

Text needs vectorization: bag-of-words counts, TF-IDF (term frequency $\times$ inverse document frequency — upweighting distinctive words), and n-grams capturing phrases. Preprocessing (lowercasing, stop-word removal, stemming) feeds the vectors; Laplace smoothing handles unseen words.

The independence assumption is the exam discussion point: violated in practice (good and excellent correlate), yet the classifier ranks well because classification needs correct ordering, not calibrated probabilities. Use naive Bayes as a fast, strong baseline for text — then try the heavier models.

Key Definitions

Naive Bayes is defined as Bayes classifier assuming feature independence given the class — fast and strong for text

TF-IDF is defined as word weighting by frequency in the document discounted by frequency across documents

Key Points — very important for exams

Naive Bayes: Bayes + independence assumption; one-pass training; text champion.

TF-IDF vectorizes text: frequent here, rare elsewhere = distinctive.

Independence is false but harmless for ranking — a great baseline before heavy models.

Review Questions

? Why does naive Bayes work despite its false assumption?

? Compute TF-IDF intuition for the word the.

Section Summary

Count words, weight distinctively, classify with Bayes — simple, fast, surprisingly strong.

Exam Pointers

→ Explain naive Bayes and TF-IDF; discuss the independence assumption.

Lecture 8: Clustering: K-Means, Hierarchical, DBSCAN

Learning Objectives

- Apply clustering algorithms
- Evaluate clusterings

Clustering groups unlabelled points by similarity. K-means: choose k, assign points to nearest centroid, recompute centroids, repeat until stable. Fast and simple — but k must be chosen (elbow method, silhouette scores), results depend on initialization (k-means++ helps), and clusters are assumed

spherical.

Hierarchical clustering builds a dendrogram — agglomerative (merge closest clusters) or divisive — letting the analyst cut at any granularity; no k needed upfront, but $O(n^2)$ cost limits scale. DBSCAN finds arbitrarily shaped clusters via density: core points with enough neighbours grow clusters; the rest are noise — excellent for spatial data with outliers.

Evaluation without labels uses internal measures (silhouette, within-cluster sum of squares) and, when labels exist, external ones (adjusted Rand index). The exam theme: match algorithm to data shape — 
k-means for blobs, DBSCAN for moons and noise.

Key Definitions

K-means is defined as iterative clustering alternating nearest-centroid assignment with centroid recomputation

DBSCAN is defined as density-based clustering that finds arbitrary shapes and labels sparse points as noise

Key Points — very important for exams

K-means: assign → recompute → repeat; needs k ; spherical bias; k-means++ init.

Hierarchical: dendrogram, any granularity, $O(n^2)$; DBSCAN: density-based, arbitrary shapes, noise-handling.

Choose k by elbow/silhouette; match algorithm to cluster shape.

Review Questions

? When does k-means fail? Give a shape example.

? How does DBSCAN define noise?

Section Summary

Group by similarity: k-means for blobs, hierarchies for taxonomies, DBSCAN for shapes and noise.

Exam Pointers

→ Explain k-means, hierarchical clustering and DBSCAN; discuss choosing k .

Lecture 9: Neural Networks and Deep Learning

Learning Objectives

- Describe neural network components
- Explain backpropagation

Neural networks stack layers of neurons: each computes a weighted sum plus bias, passed through a non-linear activation (sigmoid, tanh, ReLU — ReLU's simplicity powers modern deep learning). A network is a composition of these units — and by the universal approximation theorem, even one

hidden layer can approximate any continuous function.

Training uses backpropagation: forward pass computes predictions, the loss is differentiated backward through the layers via the chain rule, and gradient descent updates every weight. Architectures specialize: CNNs (convolutional layers) exploit spatial locality for images; RNNs/LSTMs carry hidden state through sequences for text and time series.

Deep learning's ingredients: big data, GPUs, ReLU/dropout/batch-norm tricks, and architectures matched to data structure. The exam intuition: depth builds hierarchical features — edges → shapes →  objects in vision; characters → words → meaning in language.

Key Definitions

Backpropagation is defined as training neural networks by chain-rule differentiation of loss backward through layers

ReLU is defined as the piecewise-linear activation $\max(0, z)$ enabling deep networks

Key Points — very important for exams

Neurons: weighted sum + non-linearity (ReLU); depth = hierarchical features.

Backpropagation: chain-rule gradients drive gradient descent on all weights.

CNNs for images (locality), RNNs/LSTMs for sequences (memory); data + GPUs + tricks.

Review Questions

? Why are non-linear activations essential?

? Why do CNNs suit images?

Section Summary

Stacked neurons learn hierarchical features; backpropagation teaches them.

Exam Pointers

→ Explain neurons, backpropagation, and CNN/RNN specialization.

Lecture 10: Overfitting, Regularization and Honest Evaluation

Learning Objectives

- Diagnose overfitting
- Apply regularization and validation

Overfitting — memorizing training data, failing on new data — is ML's central pathology. Diagnose with learning curves: training error falling while validation error rises signals overfitting; both high signals underfitting. The bias–variance decomposition explains it: simple models underfit (bias), complex models overfit (variance).

Regularization constrains complexity: L1/Lasso (sparsity — feature selection), L2/Ridge (shrinkage), dropout (random neuron deactivation in nets), early stopping (halt when validation worsens). More data, simpler models and cross-validation are the other remedies.

Honest evaluation requires the train/validation/test split: tune on validation, report once on test — touching test repeatedly leaks information. K-fold cross-validation rotates the validation role for small datasets. Very important: the test set is for final judgment, not for tuning. ☰

Key Definitions

Regularization is defined as penalized fitting (L1/L2) constraining model complexity to reduce overfitting

Key Points — very important for exams

Overfitting: train error ↓, validation error ↑; bias–variance explains under/overfitting.

Remedies: L1/L2, dropout, early stopping, more data, simpler models.

Protocol: tune on validation, judge once on test; k-fold CV for small data.

Review Questions

- ? How do learning curves reveal overfitting?
- ? Why must the test set stay untouched during tuning?

Section Summary

Fit the pattern, not the noise — regularize complexity and evaluate honestly.

Exam Pointers

→ Explain overfitting via bias–variance; describe regularization and proper evaluation protocol.

Lecture 11: Features: Engineering, PCA and Selection

Learning Objectives

- Engineer and encode features
- Apply PCA and selection

Features decide success more than algorithms: garbage in, garbage out. Feature engineering creates informative inputs — dates → day-of-week, text → TF-IDF, categories → one-hot encoding (never ordinal numbers for nominal categories!). Scaling (standardization, normalization) is essential for distance- and gradient-based methods.

Dimensionality reduction compresses features: PCA finds orthogonal directions of maximum variance, keeping the top components — visualizing high-dimensional data and denoising it. The explained-variance ratio guides how many components to keep; the exam intuition is shadow-casting — PCA finds the most informative viewing angles.

Selection prunes features: filter methods (correlation, mutual information), wrapper methods (recursive elimination), embedded methods (Lasso). Fewer features mean faster training, less overfitting and clearer interpretation — a triple win.

Key Definitions

PCA (Principal Component Analysis) is defined as orthogonal projection onto directions of maximum variance for compression and visualization

Key Points — very important for exams

Engineer informative features; one-hot encode nominals; scale for distance/gradient methods.

PCA: max-variance projections compress and denoise; explained variance guides components.

Select features: filters, wrappers, Lasso — faster, leaner, clearer models.

Review Questions

? Why is one-hot better than integer encoding for colours?

? What does PCA's first component represent?

Section Summary

Better features beat fancier algorithms — engineer, scale, compress, select.

Exam Pointers

→ Explain feature engineering, encoding, scaling, PCA and feature selection.

Lecture 12: Anomaly Detection, Recommenders and Time Series

Learning Objectives

- Detect anomalies
- Build recommendation intuition

Anomaly detection finds the unusual: fraud, intrusions, defects. Approaches: statistical (z-scores, Gaussian tails), isolation forests (anomalies isolate in few splits), one-class SVM (boundary around normal data), autoencoders (high reconstruction error = anomaly). The key difficulty is evaluation — anomalies are rare by definition, so precision/recall on careful labels or injected faults.

Recommender systems predict taste: content-based (item features you liked), collaborative filtering (users like you liked), matrix factorization (latent factors via SVD-like decomposition), and hybrids. The cold-start problem (new users/items) and the exploration–exploitation trade-off are the classic challenges.

Time series adds temporal structure: trend, seasonality, autocorrelation — ARIMA models and, increasingly, LSTMs and transformers forecast demand, prices and load. The exam pattern: name the problem structure, then the matching method family.

Key Definitions

Recommender systems is defined as filtering by predicted user preference via content features, user similarity or latent factors

Key Points — very important for exams

Anomalies: statistical, isolation forests, one-class SVM, autoencoders; evaluation is the hard part.

Recommenders: content-based, collaborative, matrix factorization; cold start is the challenge.

Time series: trend/seasonality/autocorrelation → ARIMA, LSTMs; match method to structure.

Review Questions

? Why is evaluating anomaly detectors hard?

? What is the cold-start problem?

Section Summary

Find the rare, predict the tasteful, forecast the temporal — specialized structures need specialized methods.

Exam Pointers

→ Describe anomaly detection and recommender approaches with their challenges.

Lecture 13: Ethics, Fairness and Privacy in ML

Learning Objectives

- Identify ML bias sources
- Apply fairness and privacy measures

ML systems fail socially when they learn society's biases: hiring models trained on historical data discriminate against women, facial recognition underperforms on dark skin, predictive policing amplifies over-policed neighbourhoods. Sources: biased training data, biased labels, unrepresentative sampling, and proxy features (zip code standing in for race).

Responses span the pipeline: audit datasets for representation, choose fairness metrics (demographic parity, equalized odds), constrain models for fairness, and keep humans accountable for consequential decisions. Interpretability tools (feature importance, SHAP values, LIME) open black boxes — required in regulated domains like credit and medicine.

Privacy completes the ethics: models memorize training data (membership inference attacks), so anonymization, differential privacy and data minimization matter. The exam-ready stance: ML ethics is engineering — measure bias, constrain models, protect data — not just philosophy.

Key Definitions

Differential privacy is defined as the practice of adding calibrated noise to computations to protect individual privacy

Key Points — very important for exams

Bias: data, labels, sampling, proxies — hiring, vision and policing cases.

Fairness: audit data, parity/odds metrics, constrained models, human accountability; interpretability (SHAP/LIME).

Privacy: models memorize — anonymize, minimize, differential privacy.

Review Questions

- ? How can zip code become a proxy for race?
- ? What is the privacy risk of memorizing models?

Section Summary

Biased data builds biased models — audit, constrain, interpret and protect privacy by design.

Exam Pointers

→ Discuss bias sources, fairness measures and privacy in ML with examples.

Lecture 14: Deploying ML: Drift, Monitoring and MLOps

Learning Objectives

- Describe deployment challenges
- Design monitoring for ML systems

Deploying ML means leaving the notebook: models are serialized, served behind APIs, and monitored — because the world changes. Data drift (input distributions shift), concept drift (relationships change — a pandemic rewrites demand), and feedback loops (predictions changing behaviour) silently degrade performance; monitoring metrics and retraining pipelines are the remedy.

A/B testing evaluates models in production: route a fraction of traffic to the challenger and compare business metrics, not just ML metrics. MLOps — versioned data, reproducible training, CI/CD for models — brings software discipline to ML systems.

The course closes where it opened: Mitchell's definition demands improving performance with experience — and deployed systems must keep learning, or at least keep being watched. An ML project is done not at deployment but when monitoring, retraining and accountability are running.

Key Definitions

Data/concept drift is defined as degradation of deployed models as input distributions or relationships change over time

Key Points — very important for exams

Deployed models decay: data drift, concept drift, feedback loops — monitor and retrain.

A/B test challengers on business metrics; MLOps = software discipline for ML.

Done = deployed + monitored + retrainable + accountable.

Review Questions

- ? Give an example of concept drift.
- ? What does MLOps add to ML practice?

Section Summary

ML in production is a living system — monitor drift, test challengers, retrain relentlessly.

Exam Pointers

- Explain deployment challenges: drift, monitoring, A/B testing and MLOps.

Glossary of Key Terms

Learning (Mitchell's definition) — Mitchell's definition: a program learns if its performance P on task T improves with experience E

Supervised learning — learning a mapping from inputs to known correct outputs using labelled training data

Reinforcement learning — learning by maximizing cumulative reward through trial-and-error interaction with an environment

Linear regression — fitting $y = w \cdot x + b$ by minimizing mean squared error

Gradient descent — iterative optimization stepping downhill on the loss: $w := w - \alpha$ times gradient

Logistic regression — a classifier mapping a linear score through the sigmoid to a class probability

Confusion matrix — the table of true/false positives/negatives underlying classification metrics

Random forest — an ensemble of trees trained on bootstrapped samples with random feature subsets, voting the result

Gradient boosting — sequential tree ensemble where each tree corrects previous errors

Support Vector Machine — the classifier maximizing the gap between the decision boundary and the nearest training points

Kernel trick — implicit high-dimensional mapping enabling non-linear SVMs via inner products only

Naive Bayes — Bayes classifier assuming feature independence given the class — fast and strong for text

TF-IDF — word weighting by frequency in the document discounted by frequency across documents

K-means — iterative clustering alternating nearest-centroid assignment with centroid recomputation

DBSCAN — density-based clustering that finds arbitrary shapes and labels sparse points as noise

Backpropagation — training neural networks by chain-rule differentiation of loss backward through layers

ReLU — the piecewise-linear activation $\max(0, z)$ enabling deep networks

Regularization — penalized fitting ($L1/L2$) constraining model complexity to reduce overfitting

PCA (Principal Component Analysis) — orthogonal projection onto directions of maximum variance for compression and visualization

Recommender systems — filtering by predicted user preference via content features, user similarity or latent factors

Differential privacy — the practice of adding calibrated noise to computations to protect individual privacy

Data/concept drift — degradation of deployed models as input distributions or relationships change over time