AI System Design & Data Engineering

The Data-Centric Foundation

In the realm of artificial intelligence, the adage "Garbage In, Garbage Out" holds paramount importance. The efficacy of any intelligent system is intrinsically limited by the quality, structure, and representativeness of its foundational data. This lecture explores how meticulous data engineering underpins robust AI architectures, ensuring models deliver reliable, unbiased insights for real-world applications.

Systematic Pipeline for Robust AI Design

A structured pipeline forms the backbone of effective AI system development, guiding from conceptualisation to operationalisation. This methodical approach mitigates risks, enhances reproducibility, and aligns technical solutions with strategic objectives.

01

1. Problem Formulation

Articulate the business objective and reframe it as a precise machine learning task, such as binary classification for customer churn prediction or regression for sales forecasting.

02

2. Data Acquisition & Ingestion

Gather data from heterogeneous sources, including relational databases, external APIs, and application logs, ensuring seamless integration and initial validation.

03

3. Data Preprocessing & Exploration

Scrutinise datasets for patterns, anomalies, and quality issues through cleaning, transformation, and exploratory analysis—the core focus of this discourse.

04

4. Model Training & Experimentation

Select algorithms, conduct training runs, and optimise hyperparameters using techniques like grid search or Bayesian optimisation for peak performance.

05

5. Model Evaluation

Assess performance on unseen validation sets with metrics tailored to the domain, such as precision-recall curves for imbalanced datasets.

06

6. System Deployment

Integrate the model into production environments via RESTful APIs, batch processing scripts, or edge devices for scalable inference.

07

7. Monitoring & Maintenance

Implement MLOps practices for ongoing surveillance of model drift, performance degradation, and automated retraining to sustain accuracy.

Data Preprocessing: Cleaning & Feature Engineering

The objective of data cleaning is to forge a pristine, representative dataset optimised for model ingestion, thereby elevating predictive accuracy and reliability.

Handling Missing Values

Employ strategies like listwise deletion for minor gaps, mean or median imputation for numerical data, k-nearest neighbours for contextual filling, or predictive modelling to infer absent entries, preserving dataset integrity.

Outlier Detection

Identify anomalies using interquartile range (IQR) or Z-score thresholds, augmented by domain expertise; mitigate via capping at percentiles, logarithmic transformations, or selective removal to avert distortion in model learning.

Data Type & Format Standardisation

Enforce uniformity across datasets, such as ISO date formats and consistent categorical encodings (e.g., UTF-8 strings), to facilitate seamless processing and prevent parsing errors.

Rationale: Unresolved noise and inconsistencies propagate biases, impairing generalisation and destabilising training dynamics in AI systems.

Data Preprocessing: Cleaning & Feature Engineering

Feature engineering transforms raw data into insightful variables, amplifying the model's capacity to discern patterns and relationships.

1

Feature Creation

Synthesise novel attributes from raw inputs, such as computing age from birth dates, extracting weekday from timestamps, or aggregating transaction volumes over time windows to enrich contextual understanding.

2

Categorical Encoding

- One-Hot Encoding: Ideal for nominal variables like product categories, generating binary vectors to avoid ordinal assumptions.
- Label/Ordinal Encoding: Suited for ranked data, such as education levels (e.g., 1=secondary, 2=bachelor's), preserving inherent hierarchies.

3

Transformations

Apply non-linear adjustments like logarithmic scaling for skewed distributions or polynomial expansions to capture interactions, stabilising variance and enhancing feature discriminability.

Goal: Deliver the model a curated set of salient features that maximise task-specific learning efficiency and interpretability.

Data Preprocessing: Scaling & Dimensionality Reduction

Scaling normalises feature magnitudes, crucial for algorithms reliant on distances or gradients, such as support vector machines, clustering methods, and deep neural networks.

Normalisation (Min-Max Scaling)

Rescales features to a bounded interval, typically [0, 1], via the formula: X_scaled = (X - X_min) / (X_max - X_min). This preserves relative proportions but proves vulnerable to outliers, which can compress the effective range.

Suitable for image pixel data or when zero-to-one scaling aligns with model assumptions.

Standardisation (Z-Score Normalisation)

Centres features around a mean of 0 with unit standard deviation: X_scaled = (X - μ) / σ. Robust to outliers, it facilitates convergence in gradient descent and equalises influence across variables.

Preferred for Gaussian-assuming algorithms like linear regression or principal component analysis.

Key Insight: Unscaled features can skew optimisation, favouring high-magnitude variables and yielding suboptimal convergence.

Data Preprocessing: Scaling & Dimensionality Reduction

Principal Component Analysis (PCA) offers an unsupervised method to condense high-dimensional data while retaining essential variance.

Reduced Space

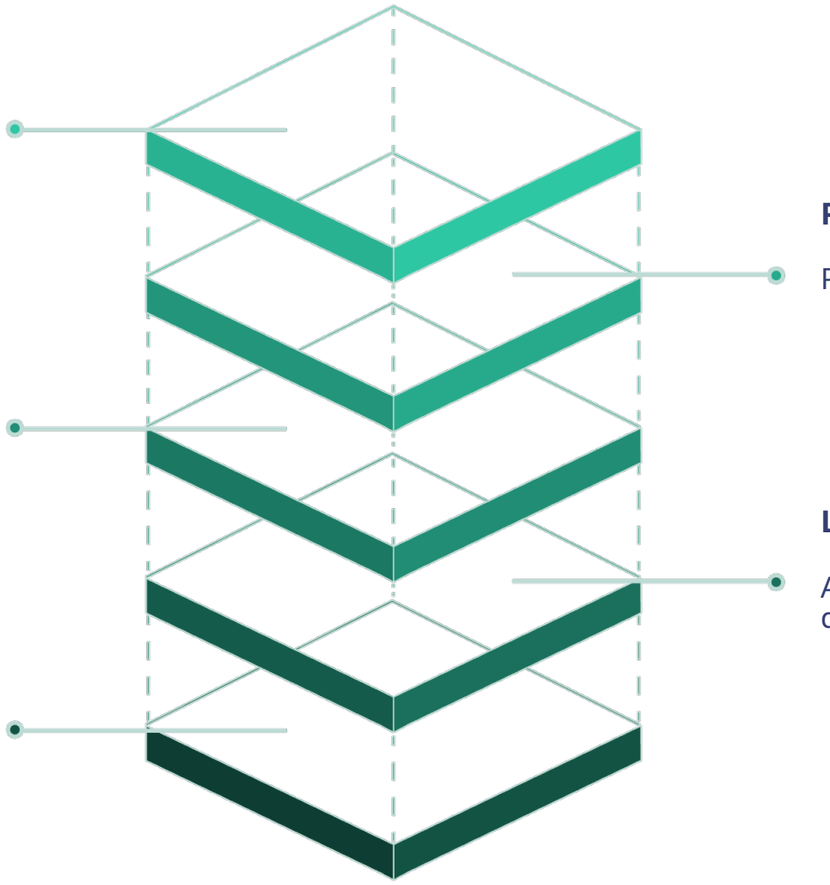
Lower-dimensional representation with max variance

Principal Axes

Orthogonal components ordered by variance

Original Space

High-dimensional data with original axes



Projection

Project data onto top components

Linear Transform

Apply covariance and eigen decomposition

Objective: Derive orthogonal principal components via eigen decomposition of the data covariance matrix, prioritising directions of maximal variance to compress features effectively.

Benefits: Alleviates computational burdens and the curse of dimensionality; prunes noisy dimensions to curb overfitting. However, it sacrifices interpretability, as components represent linear amalgamations of originals.

Trade-off: Balance retained variance (e.g., 95% threshold) against dimensionality to optimise model performance without excessive information loss.

Addressing Dataset Imbalance

Real-world datasets often suffer from class imbalance, skewing models towards prevalent classes and undermining minority detection—prevalent in fraud detection or rare disease diagnosis.

Oversampling Techniques

Augment minority instances: Random oversampling duplicates samples, risking overfitting; SMOTE generates synthetic examples through nearest-neighbour interpolation, fostering diversity and generalisation.

Undersampling Approaches

Trim majority class via random removal, though this hazards discarding critical patterns; informed variants like Tomek links target borderline instances for more balanced excision.

Algorithmic Adjustments

Incorporate cost-sensitive learning, assigning elevated penalties to minority misclassifications, or utilise ensemble methods like balanced random forests to inherently address skew.

Evaluation: Favour precision, recall, F1-score, and area under the precision-recall curve over accuracy to gauge true minority performance.

Data Processing Paradigms: Real-time vs. Batch

Selecting between batch and real-time processing shapes AI system architecture, balancing latency, throughput, and complexity for diverse applications.

Batch Processing

Processes finite, voluminous datasets in periodic batches. Offers high throughput but incurs delays (hours to days). Ideal for model training, ETL pipelines, and historical analytics using frameworks like Apache Spark or Hadoop.

Real-time (Stream) Processing

Handles infinite data streams with sub-second latency. Prioritises immediacy over volume, suiting fraud alerts, dynamic recommendations, or IoT monitoring via tools such as Apache Kafka, Flink, or Storm.

Hybrid Architectures: Lambda integrates batch for accuracy and streams for speed; Kappa unifies via a single streaming layer, streamlining maintenance for evolving AI systems.