ML Algorithms - the classical approach

Maciej Kępa

Roadmap

Data Community

01

AI/ML architecture - how it all fits together

Understand the full ML stack from data sources to production, including roles, cloud reference architectures, layers, tools, and a simple ETL + model demo.

02

Data Preparation - practical foundations

Learn practical techniques for building ML data pipelines, cleaning data, validating datasets, and preparing a dataset in PySpark

03

Feature Engineering – the art of extracting value from data

Discover how to create effective features, handle different types of data, apply encoding and normalization, monitor feature drift, and track features with Spark + MLflow.

04

ML Algorithms - the classical approach

Get familiar with core algorithms like regression, trees, and gradient boosting, understand classification, regression, clustering, their pros and cons, and try MLlib demos in Databricks.

05

Model Training in Practice

Learn how to split data, perform hyperparameter tuning, scale training, and train models with code and metrics using MLflow.

06

Deep Learning - leveling up

Understand neural network basics, choose between TensorFlow and PyTorch, leverage GPU/TPU scaling, and build a simple CNN with visualization.

07

ML Pipelines - automation and CI/CD

Build repeatable ML workflows with orchestration tools, CI/CD pipelines, model registry/versioning, and a Databricks pipeline demo.

08

MLOps - manage your ML solution

Explore ML lifecycle management with tracking, model monitoring, data drift handling, canary deployments, A/B testing, and monitoring demos.

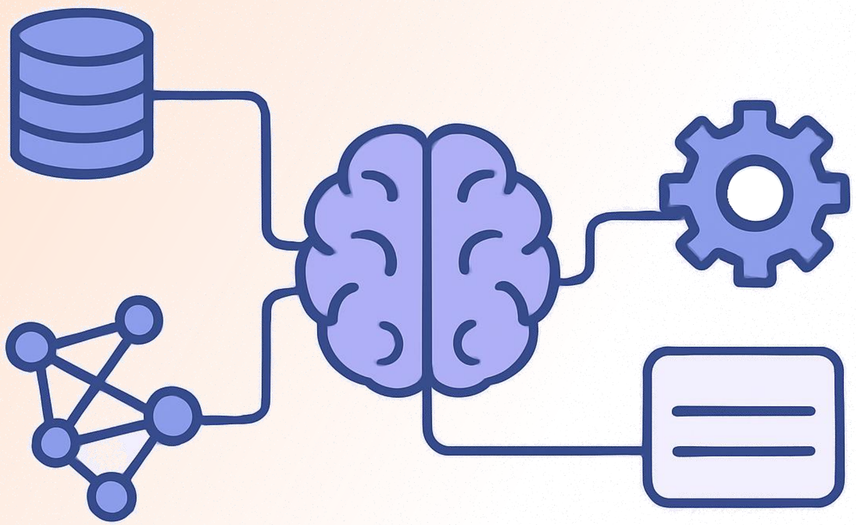
09

Generative AI and LLMs - the new wave of technology

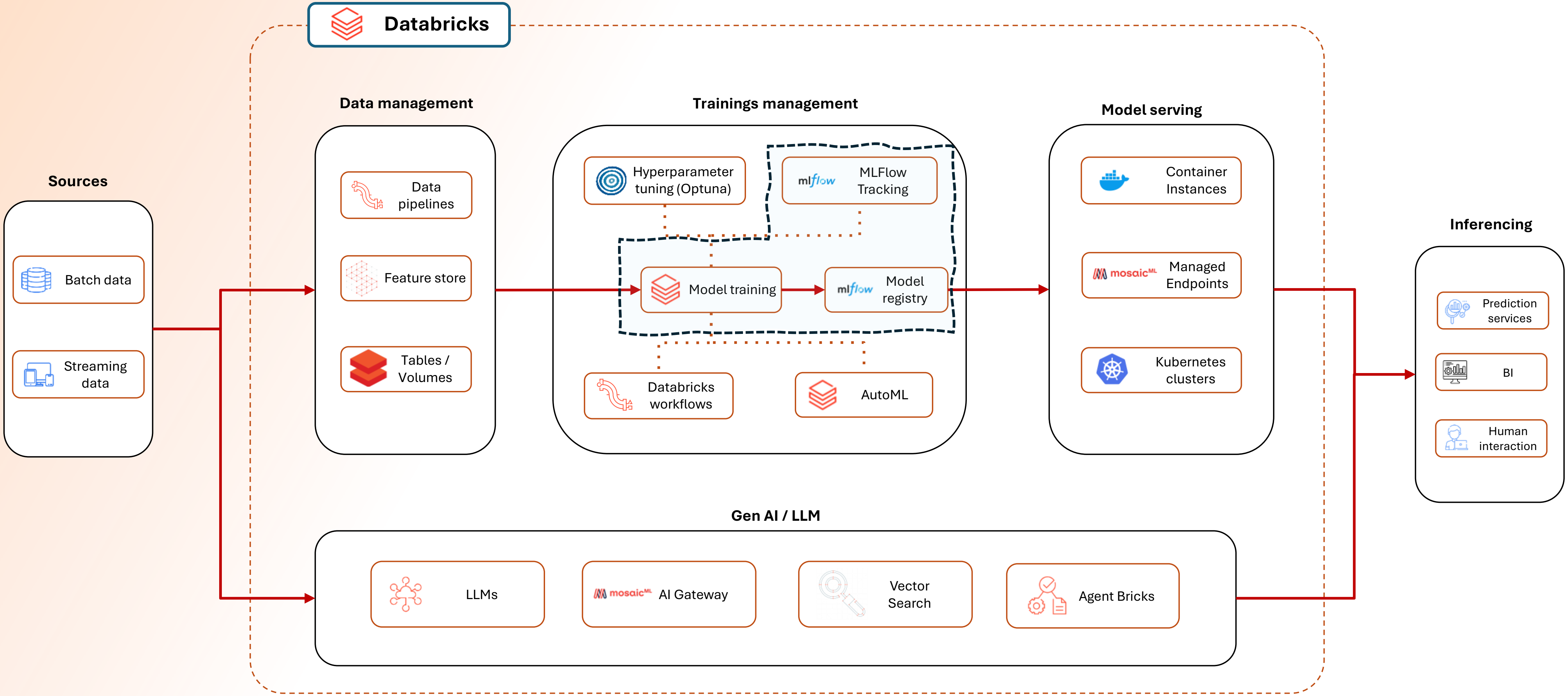
Learn how LLMs differ from classical ML, understand transformers, RAG, AI agent orchestration, and try a simple RAG demo in Azure OpenAI.

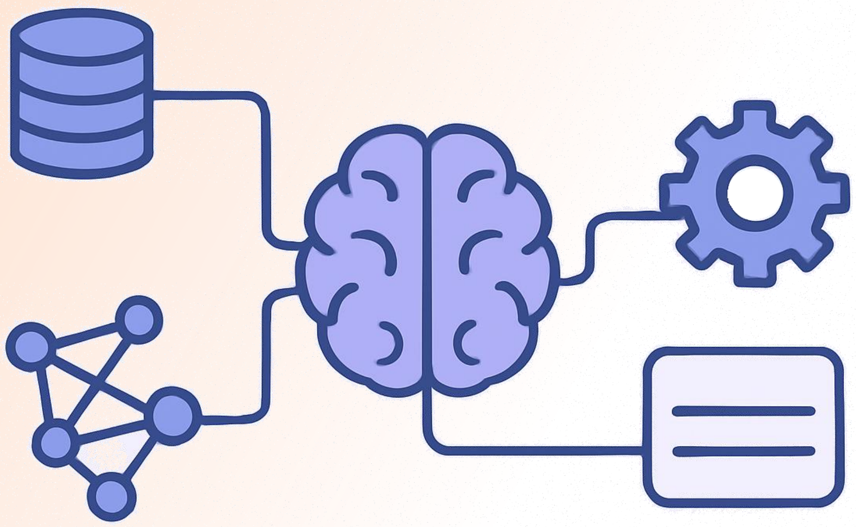
Agenda

1. Introduction
2. Machine learning tasks and algorithms
3. Choosing the right approach
4. Workshop: linear regression
5. Workshop: logistic regression
6. Workshop: gradient boosting
7. Workshop: clustering
8. Best practices



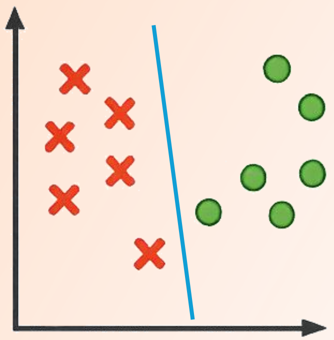
Introduction

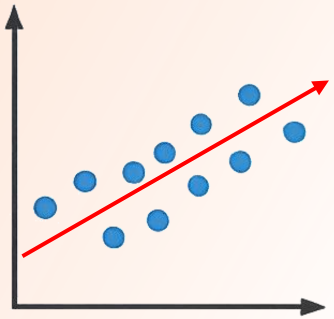

Databricks
Sources
Batch data
Streaming data
Data management
Data pipelines
Feature store
Tables / Volumes
Trainings management
Hyperparameter tuning (Optuna)
mlflow MLFlow Tracking
Model training
mlflow Model registry
Databricks workflows
AutoML
Model serving
Container Instances
mosaicML Managed Endpoints
Kubernetes clusters
Inferencing
Prediction services
BI
Human interaction
Gen AI / LLM
LLMs
mosaicML AI Gateway
Vector Search
Agent Bricks

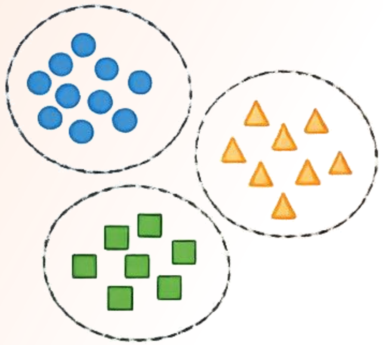


Machine learning tasks

Machine learning tasks







Supervised Learning

Learn from labeled data to make predictions on new, unseen examples.

- **Classification:** Predicting categories (spam detection, disease diagnosis, customer churn)
- **Regression:** Predicting continuous values (house prices, sales forecasts, temperature)

Unsupervised Learning

Discover patterns in unlabeled data without predefined outcomes.

- **Clustering:** Grouping similar items (customer segmentation, document organization, anomaly detection)

Linear and Logistic Regression

Simple yet powerful algorithms that form the foundation of many ML applications. These models assume a linear relationship between input features and the target variable.

Linear Regression

Use case: Predicting continuous outcomes like revenue, temperature, or stock prices

How it works: Fits a straight line through data points to minimize prediction error

Logistic Regression

Use case: Binary classification tasks like email spam filtering or loan approval decisions

How it works: Transforms linear outputs into probabilities between 0 and 1

Linear models traits

Data Community

Key Assumptions

- Features have a **linear relationship** with the target
- Observations are **independent** of each other
- Errors follow a **normal distribution**
- Features show **minimal multicollinearity**

When to Use

- You need **interpretable results** for stakeholders
- Dataset is **relatively small**
- Relationships appear **mostly linear**
- **Fast training** is a priority

⚠️ Linear models struggle with complex, non-linear patterns and interactions between features.

For such cases, consider tree-based or advanced methods.

Decision Trees and Tree Ensembles

Decision Trees are intuitive, flow-chart like models that simulate human decision-making processes, making them highly interpretable and easy to understand.

How They Work

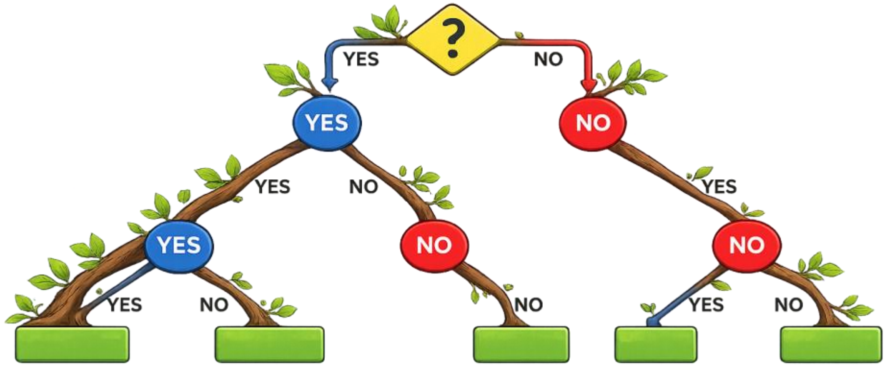
Data is recursively split into subsets based on feature values, forming branches until a decision (leaf node) is reached. Each split aims to maximize the homogeneity of the results.

Interpretability

Their visual, hierarchical structure allows direct insight into how predictions are made, ideal for explaining complex outcomes to non-technical stakeholders.

Versatility

Can handle both numerical and categorical data, and are used for both classification (e.g., predicting customer churn) and regression (e.g., predicting house prices) tasks.



Gradient Boosting

State-of-the-art ensemble technique that builds trees sequentially, where each new tree corrects errors made by previous trees. Dominates ML competitions and production systems.

1

Build Initial Model

Start with a simple prediction (often the mean)

2

Calculate Residuals

Measure errors between predictions and actual values

3

Train New Tree

Fit a tree to predict the residuals

4

Update Model

Add new tree to ensemble with learning rate scaling

Tree-based models traits

Key Assumptions

- Data can be split into **meaningful subsets**
- **Relationships** are mostly captured by feature thresholds
- Features contain enough signal without heavy preprocessing
- **Sufficient data** to avoid overfitting

When to Use

- You need **interpretable results** for stakeholders
- **Minimal** feature engineering is preferred
- For handling **complex, non-linear** relationships
- **Fast training** is a priority

⚠️ Tree-based methods can overfit easily, especially with noisy data or very deep trees, so pruning or ensembles are often needed.

Clustering

Clustering algorithms organize unlabeled data into meaningful groups, revealing inherent patterns and structures without prior knowledge. They are fundamental for exploratory data analysis.

K-Means Clustering

Use case: Large datasets with known K, cases like customer segmentation or image compression

How it works: Partitions data into a predefined number (K) of spherical clusters based on proximity to centroids

Hierarchical Clustering

Use case: When K is unknown or we need to vizualize relationships, for cases like biological taxonomy

How it works: Builds a tree-like hierarchy of nested clusters, allowing for exploration at various granularity levels

DBSCAN

Use case: For variable cluster shapes and noise detection - in anomaly detection or geospatial data analysis

How it works: Discovers density-connected clusters of arbitrary shapes and effectively identifies outliers

Clustering model traits

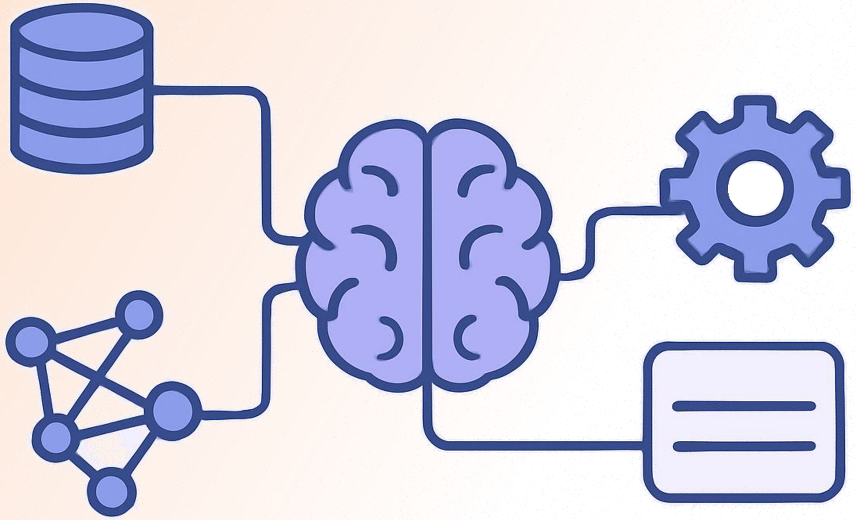
Key Assumptions

- Clusters have some meaningful structure
- The chosen **distance metric** reflects meaningful similarity
- The **number of clusters** (if required) is roughly correct.
- Features are **scaled** appropriately

When to Use

- You want to find **hidden groupings** in unlabeled data
- Data has natural separations or patterns
- **Dimensionality** is not extremely high (or reduced via PCA)
- Exploratory analysis or segmentation is needed

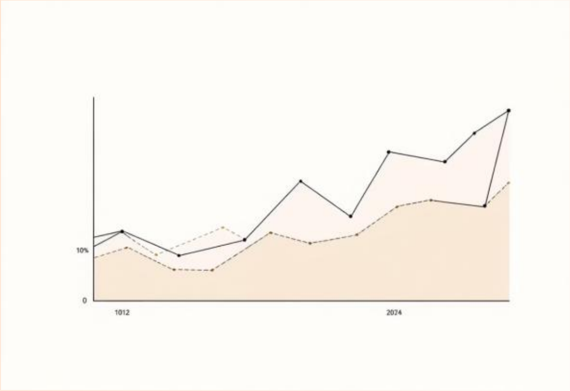
⚠️ Clustering results depend heavily on distance metrics, scaling, and chosen number of clusters; misconfiguration can give misleading groupings.

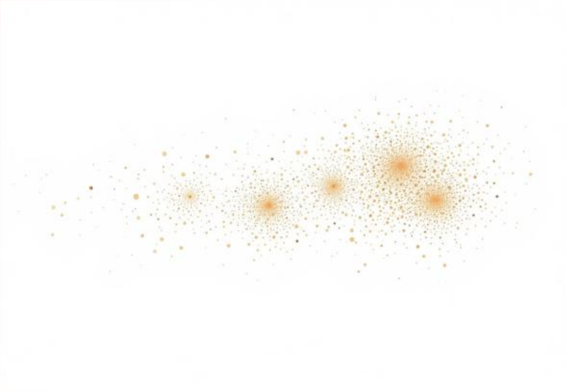


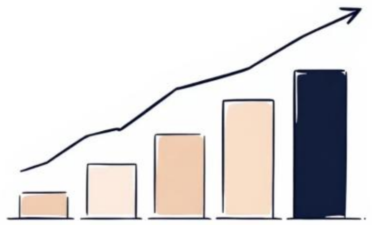
Choosing the right approach

Choosing the right approach

Algorithm selection isn't just a technical decision; it's a strategic business one. It requires balancing technical performance with real-world constraints and organizational goals.







Linear Models

Ideal for scenarios demanding high interpretability and quick deployment, especially in regulated industries.

Suited for small datasets, linear relationships, and when fast inference is critical for business operations.

Clustering

Essential for understanding customer behavior, market segmentation, and anomaly detection when data is unlabeled.

Facilitates exploratory analysis to uncover hidden patterns that drive new business insights.

Decision Trees

Offers clear, human-understandable rules, making them excellent for quick prototyping and stakeholder communication where transparency is key.

Effective for moderate datasets with categorical features.

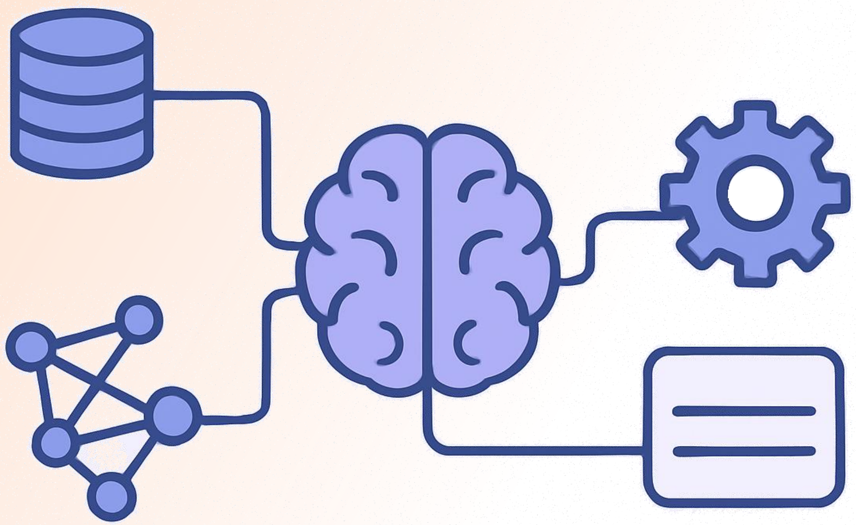
Gradient Boosting

The go-to for maximizing predictive accuracy in critical applications and competitions, particularly with structured data.

Requires careful hyperparameter tuning, but the performance gains often justify the investment for high-impact business outcomes.

Pro tip: Prioritize business value (ROI) and stakeholder understanding. Start with simpler models to establish a baseline and faster time-to-market, then incrementally introduce complexity only when justified by significant performance gains, considering your computational budget, operational constraints, and long-term maintenance.

DEMO



Summary

Takeaways

Understand Your Data and Problem: match the algorithm to your data characteristics (e.g., linear, non-linear) and the specific task (e.g., classification, regression).

Start Simple, Iterate Incrementally: begin with interpretable models like linear regression. Only add complexity with tree-based or boosting methods if performance gains justify it.

Balance Performance and Interpretability: evaluate models not just on accuracy, but also on training time, inference speed, and the ability to explain their predictions to stakeholders.

Thank you!

Contact:

https://www.linkedin.com/in/maciej-kepa

https://github.com/maciejkepa/ai-ml-in-practice