Managing Machine Learning Projects

⚠ How to identify users ?

- Listen to users
  - interviews, focus groups
  - identify pain points
  - How are they solving today?
- observe them in context
  - field observation, shadowing
  - identify latent problems

⚠ what is validation process ?

1. formulate a hypothesis about the solution
2. test the hypothesis with the users
3. Analyze your learnings
4. decide to continue or pivot
5. refine hypothesis & repeat

⚠ Machine Learning adds business value for your users

- automation
- prediction
- Personalization

⚠ Heuristics are methods of solving problems using a simplified set of rules based on past experience

⚠ Heuristics vs ML

Heuristics	Machine Learning
easier to create and maintain	often better performing
minimal computation cost	can evolve with re-training
high interpretability	suitable for a wider range of problems

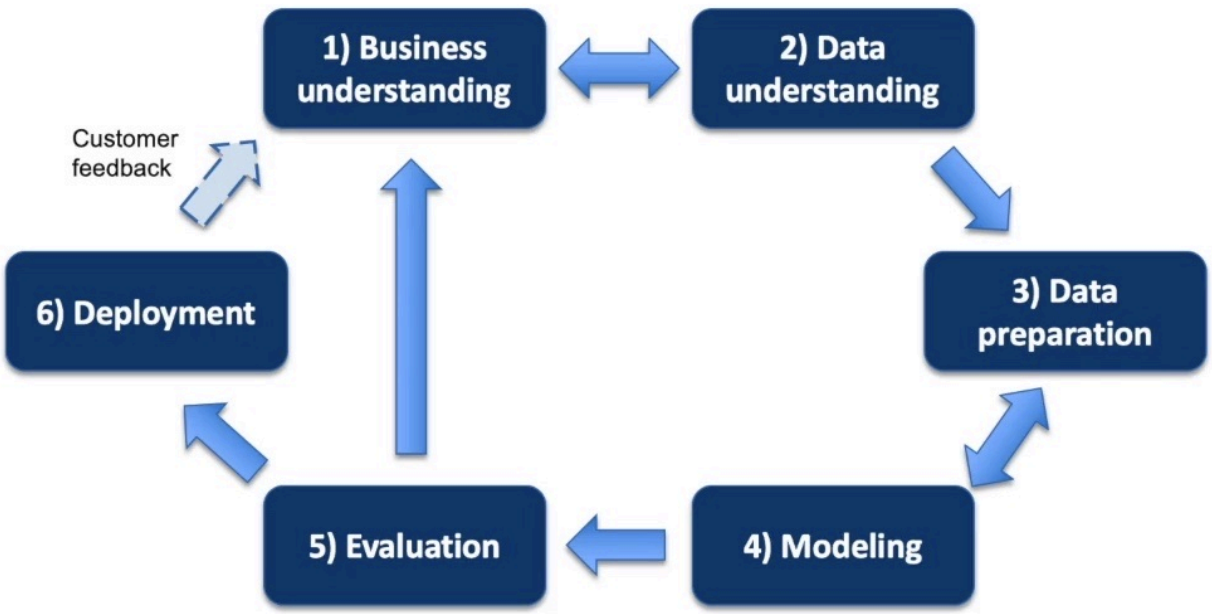
⚠ Relative to normal software projects, ML projects

- require a broader set of skills / team
- have higher technical risk
- Are harder to show progress
- require more ongoing support

⚠ Challenges of ML projects

- probabilistic rather than deterministic
- higher technical risk
- much more upfront work required
- often require change management

⚠ CRISP-DM process



⚠ Business Understanding

1.1 Define the problem	1.2 Define success	1.3 Identify factors
• Target user • Write the problem statement • Why it matters • How is it solved today? • Gaps in current state	• Quantify the expected business impact • Identify constraints • Translate impact into metrics – outcome & output metrics • Define success targets for metrics	• Gather domain expertise • Identify potentially relevant factors

⚠ Data Understanding

2.1 Gather data	2.2 Validate data	2.3 Explore the data
• Identify data sources for each factor • Label data • Create features	• Quality control data • Resolve data issues – missing, erroneous, outliers	• Statistical analysis and visualization • Dimensionality reduction • Identify relationships & patterns

⚠ Data Preparation

3.1 Split data	3.2 Determine feature set	3.3 Prepare for modeling
• Split data for training and test	• Feature engineering • Feature selection	• Encoding categorical features • Scale/standardize data • Resolve class imbalance

⚠ Modeling

4.1 Model selection	4.2 Model tuning
• Evaluate algorithms via cross-validation • Documentation and versioning	• Hyperparameter optimization • Documentation and versioning • Model re-training

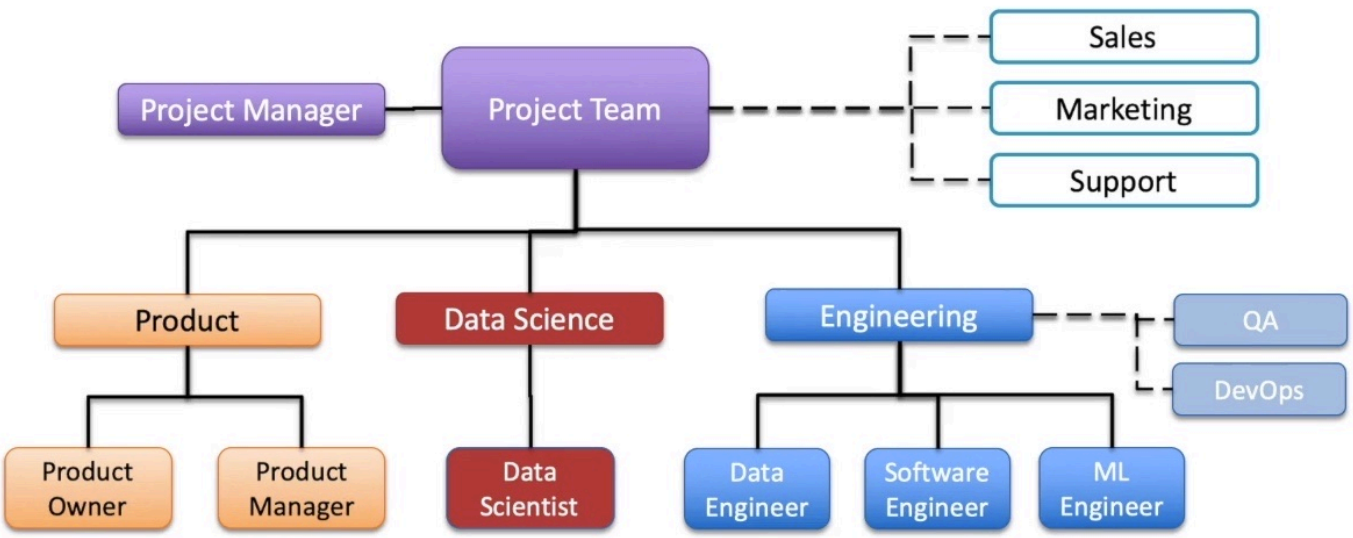
⚠ Evaluation

5.1 Evaluate results	5.2 Test solution
• Model scoring on test set • Interpretation of model outputs and performance	• Software unit & integration tests • Model testing – unit tests, directional expectation • User tests

⚠ Deployment

6.1 Deploy	6.2 Monitor
• API framework • Product integration • Scaling infrastructure • Security • Software deployment process	• Model performance monitoring • Model retraining

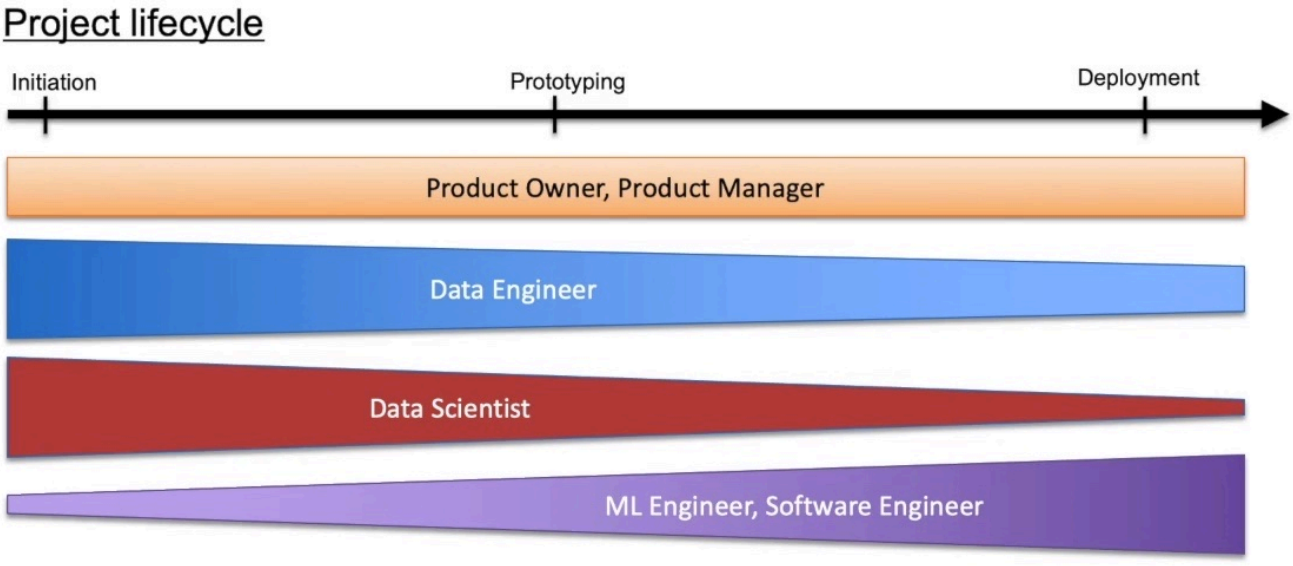
⚠ Typical team roles



⚠ Data Scientist vs. ML Engineer

Data Scientist	ML Engineer / MLOps
statistical / data science background plus programming skills & domain expertise	computer science or eng. background plus ML training
gather, process & derive insights from data	develop production data pipelines and ML system
determining of ML approach and prototyping	work with software engineering & Devops on model integration and deployment

⚠ Involvement over project cycle



⚠ Agile approach to ML

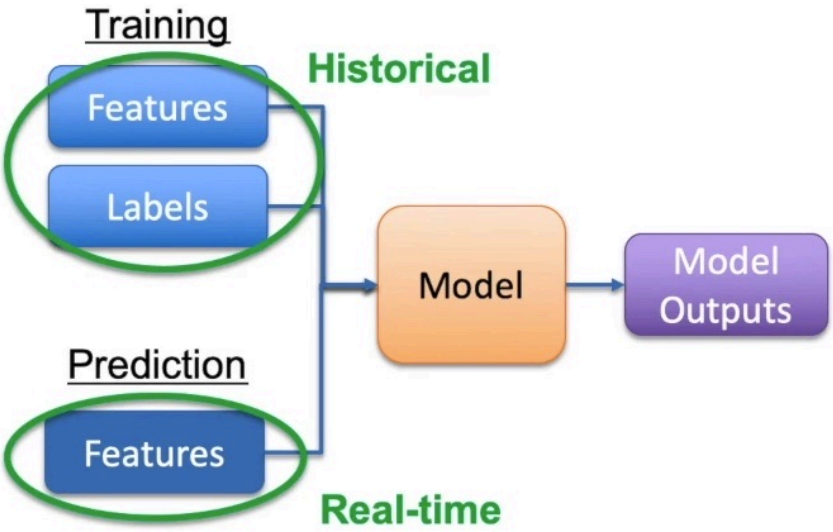
↳ sequence of iterative experiments

- explore a hypothesis
- build it, using more of CRISP-DM each time
- observe it in action, get feedback
- analyze results and repeat

⚠ Out come metrics Vs output metrics

Outcome metrics	output metrics
refers to the desired business impact on your organization or for your customer	refers to the desired output from the model Δ will set after setting the desired outcome
Stated in terms of the expected impact ($)	measured in terms of a model performance metric
does not contain model performance metrics or other technical metrics	typically not communicated to the customer

⚠ Model training and evaluation



⚠ factors that influence data requirements

- number of features
- complexity of feature-target relationships
- data quality (missing and noisy data)
- desired model performance

⚠ Data can have several issues

- missing data
- anomalous data
- incorrectly mapped data

⚠ types of missing data

	Missing Completely at Random	Missing at Random	Missing Not at Random
Description	No pattern in missing data or association to values of other attributes	Probability of missing-ness relates to another feature of the data	Probability of missing-ness relates to values of the feature itself
Example	Power outages of sensors	Males are less likely to answer survey questions about depression	Purchased item ratings skew towards people who hated the product
Potential for Bias	Low	High	High

⚠ Dealing with missing data

- drop (remove)
- flag (treat as special value of feature)
- replace (with mean value / median)
- backfill or forward-fill
- infer it

⚠ outliers

↳ points which fall far from the rest, either in a feature value or the target value

⚠ Exploratory data analysis (EDA)

↳ helps us catch issues in our data & understand it better

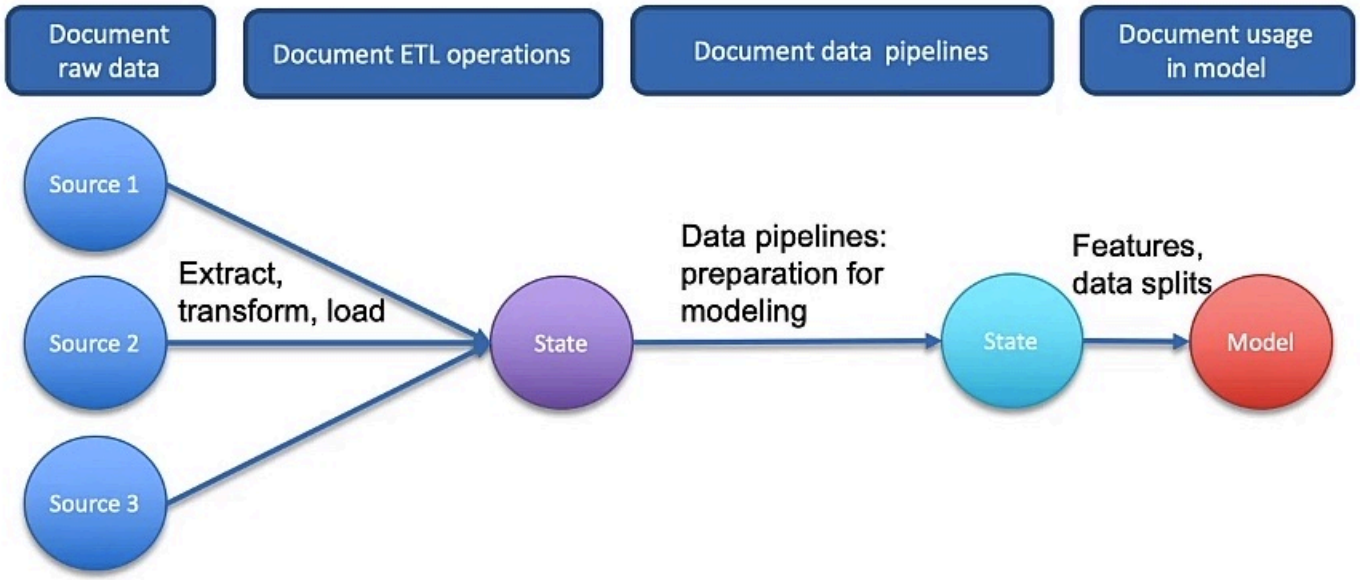
⚠ Feature Selection Methods

	Filter Methods	Wrapper Methods	Embedded Methods
Description	• Statistical tests which rely on characteristics of the data only	• Train model on subsets of features	• Extracts features which contribute most to training of a model
Pros & Cons	• Computationally inexpensive • Often used before modeling to remove irrelevant features	• Computationally very expensive • Often unfeasible for real-world modeling	• Leverage model training with minimal additional computation

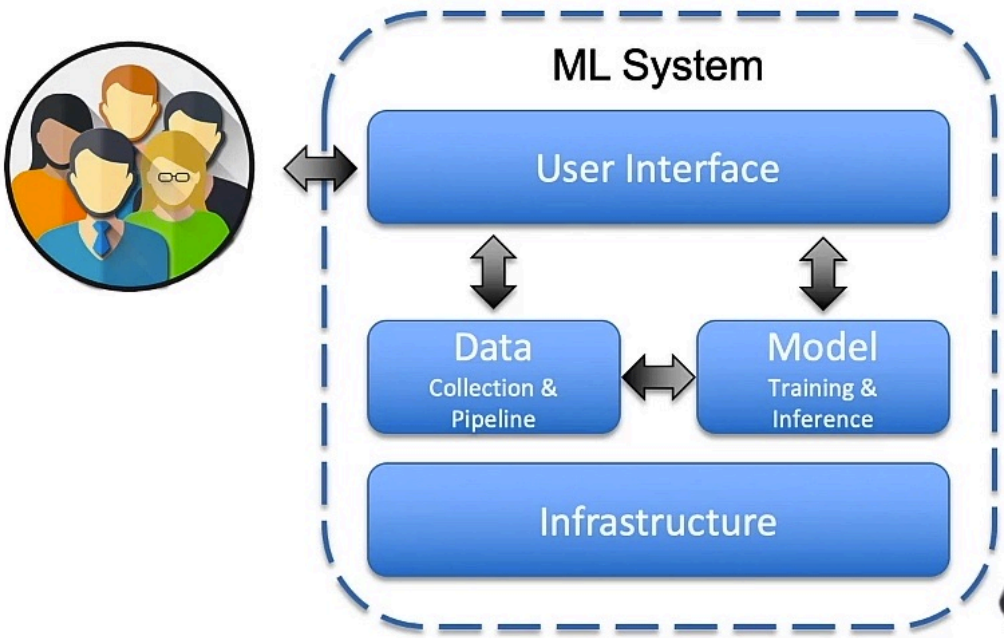
⚠ Reproducibility

- ability to reproduce results is a major issue in ML projects
- is important because
  - helps debug future issues
  - employees can leave
  - handoffs between teams
  - peer reviews establish credibility
- best practices
  - documentation
  - data lineage (tracking data from source to consumption)
  - model & code versioning

⚠ Data lineage



⚠ what is a ML system?



⚠ There are several system design decisions which impact the choice of technology

① Cloud vs Edge ② offline learning vs online learning

③ Batch predictions vs online predictions

⚠ Cloud vs Edge

	Cloud ML	Edge ML
Description	Computations done on cloud and result delivered to end device	Computations done directly on device (phone, sensor, etc)
Requirements	Network connectivity	Sufficient compute power, memory
Benefits	High throughput	Low latency, privacy, no need for connectivity
Examples	• Chatbots • Demand prediction	• Quality control • Autonomous driving

⚠ offline vs online models

	Scheduled	Real-time
Model re-training	Offline learning	Online learning
Prediction	Batch prediction	Online prediction

	Offline learning	Online learning
Description	Model re-training done on a schedule (weeks/months) using datapoints in many iterations	Continual re-training as new data arrives (mins/hours) using each new datapoint once
Benefits	• Easier to implement in production • Easier to evaluate	• Handles big data • Real-time adaptation to changing environment
Challenges	• Slower to adapt to changes in environment or data distribution	• Harder to implement & evaluate performance
Examples	• Most current applications	• Flagging spam in social media

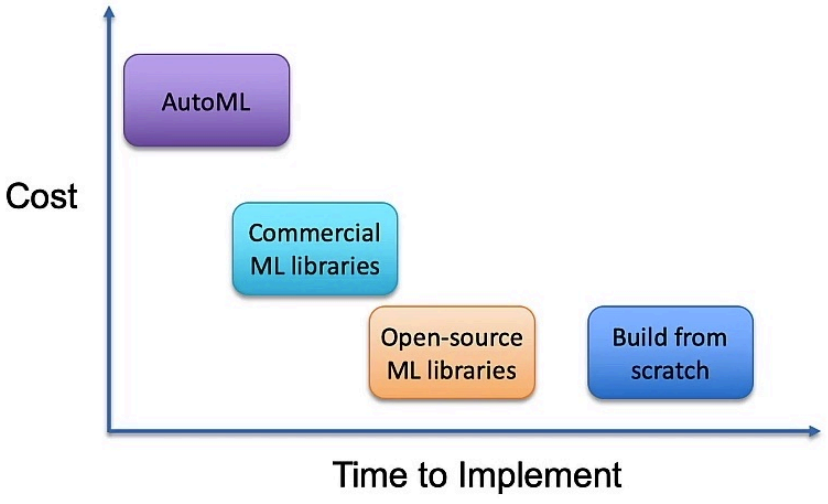
⚠ Batch vs online prediction

	Batch prediction	Online prediction
Description	Generate predictions on batch of observations on a recurring schedule	Real-time predictions generated upon request
Benefits	• Leverage more efficient operations and technologies • Easier monitoring of drift	• Predictions available immediately
Challenges	• Predictions not immediately available for new data	• Minimizing latency • Monitoring of model drift
Examples	• Recommendation systems • Demand prediction	• Translation app • Autonomous vehicles

⚠ Working with Big Data has unique challenges

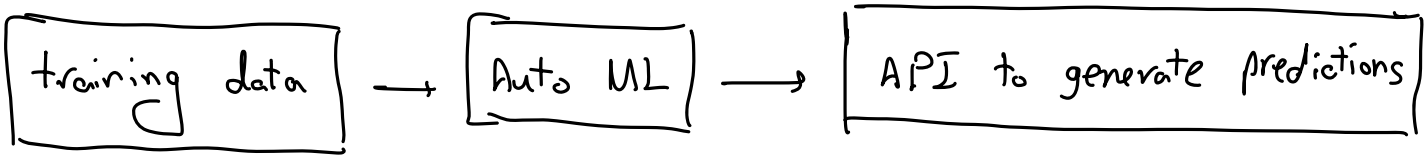
- storage & processing
- exploration
- modeling

⚠ ML technology options



⚠ What is Auto ML?

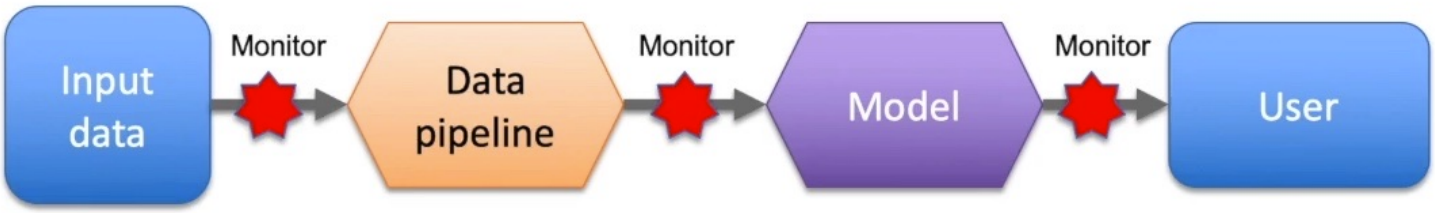
↳ enables developers with limited ML expertise to quickly build models with little / no code



⚠ ML system failure

- training - serving skew
  - ↳ mismatch between the training data and the input data while in production
- excessive latency
- data drift
  - ↳ model is trained on a static training set, but the environment changes over time
- concept drift
  - ↳ distributions of data may stay same, but the patterns that the model learned no longer apply

⚠ ML system monitoring



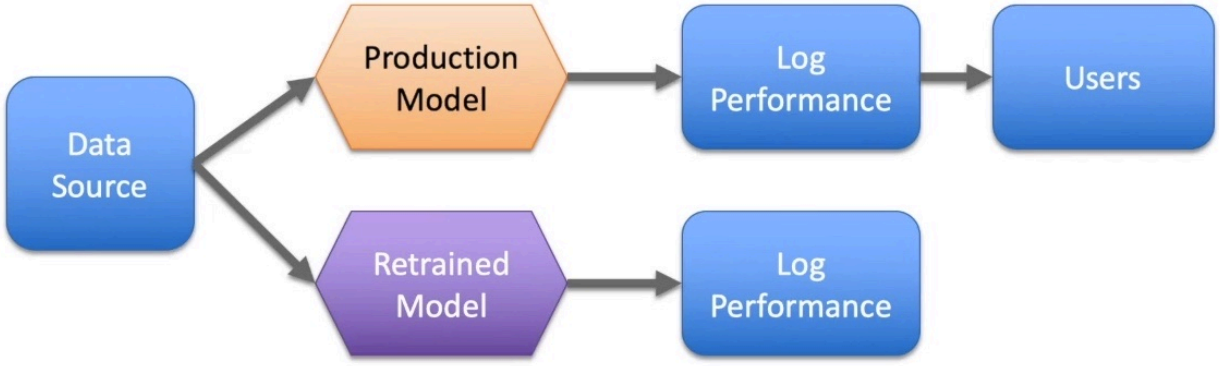
⚠ Retraining vs. updating models

Retraining	Updating
Using data collected since previous training to re-train model weights	Uses new data to re-do the modeling process
enables model to adapt to changes to environment	Allows for adjustments to model form
can be done on a schedule or triggered	Can lead to identification of higher quality model

⚠ why version models ?

- evaluate performance across iterations
- track dependencies - ensure reproducibility
- facilitates collaboration
- build in rollback capability
- enable testing multiple models

⚠ shadow releasing



If retrained model performance exceeds production model, move retrained model into production.

⚠ champion - challenger testing