Pitfalls and Challenges of ML-Powered Applications

Emmanuel Ameisen

What is an ML platform?

ML Platforms

“The ML Platforms track focuses on the practice of moving machine learning systems from **development** to **deployment**, and the next-generation **tooling** that makes this an **organic process**.”

A (humble) user review of ML platforms

- Led dozens of projects using and building such platforms at Insight Data Science
- Been on multiple teams using these tools

Plan

1. What are ML platforms (now)
2. Offline data management
3. Model performance validation
4. Model deployment
5. Monitoring and alerting
6. Error preemption

Platforms have a wide surface area

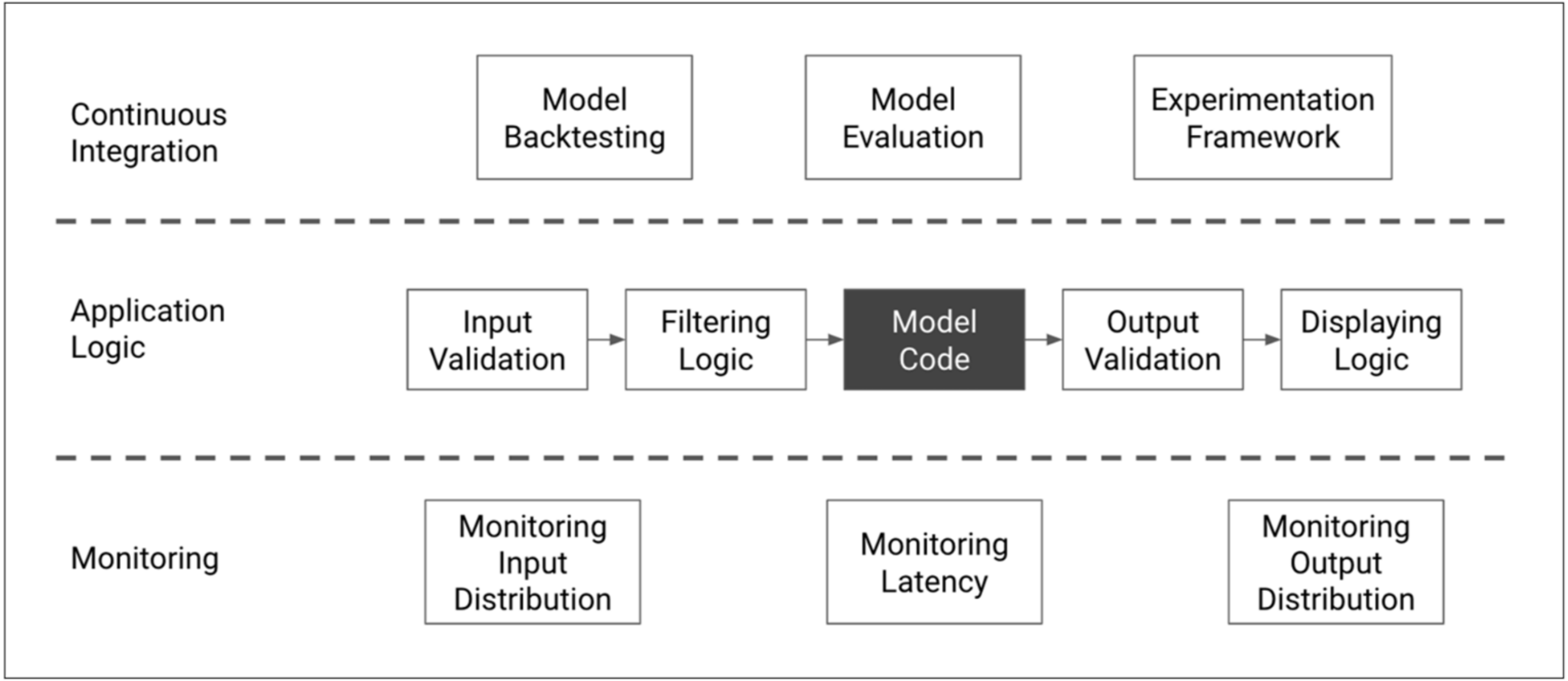
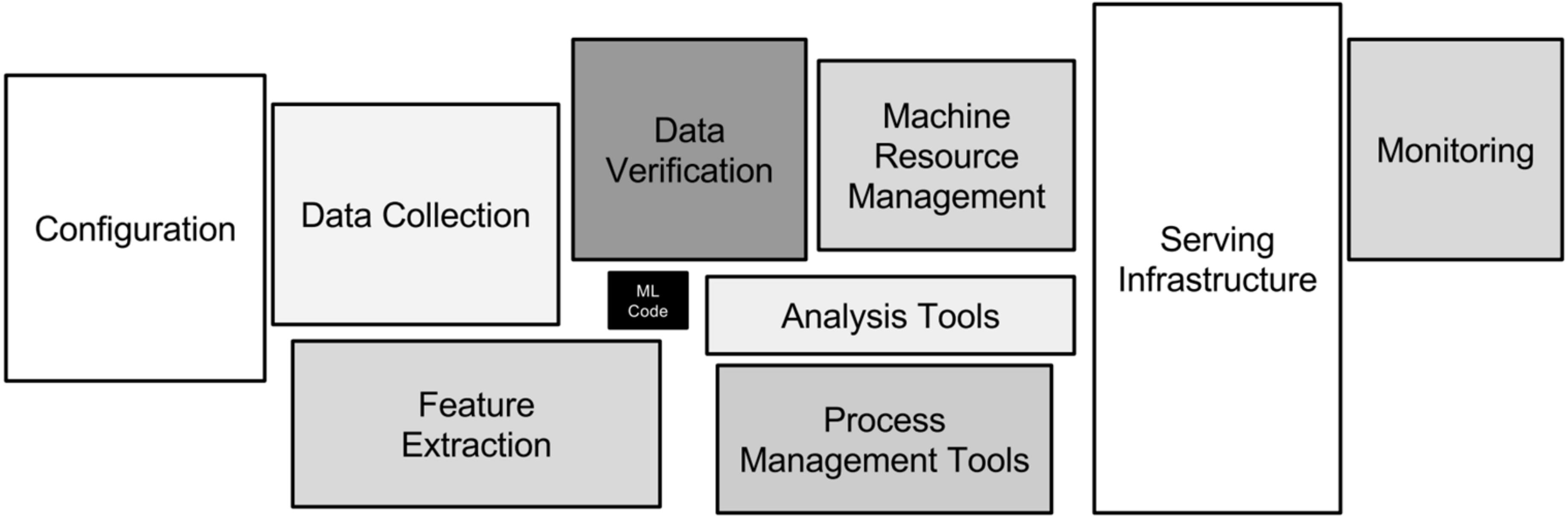


Illustration from "Building ML Powered Applications"

Platforms have a wide surface area



Hidden Technical Debt in Machine Learning Systems

ML products aren't about the ML

- Most of the surface area of an ML application is not the model
  - Most of the problems emerge outside of the model
  - It is easier to fix the system than the model

Offline data management

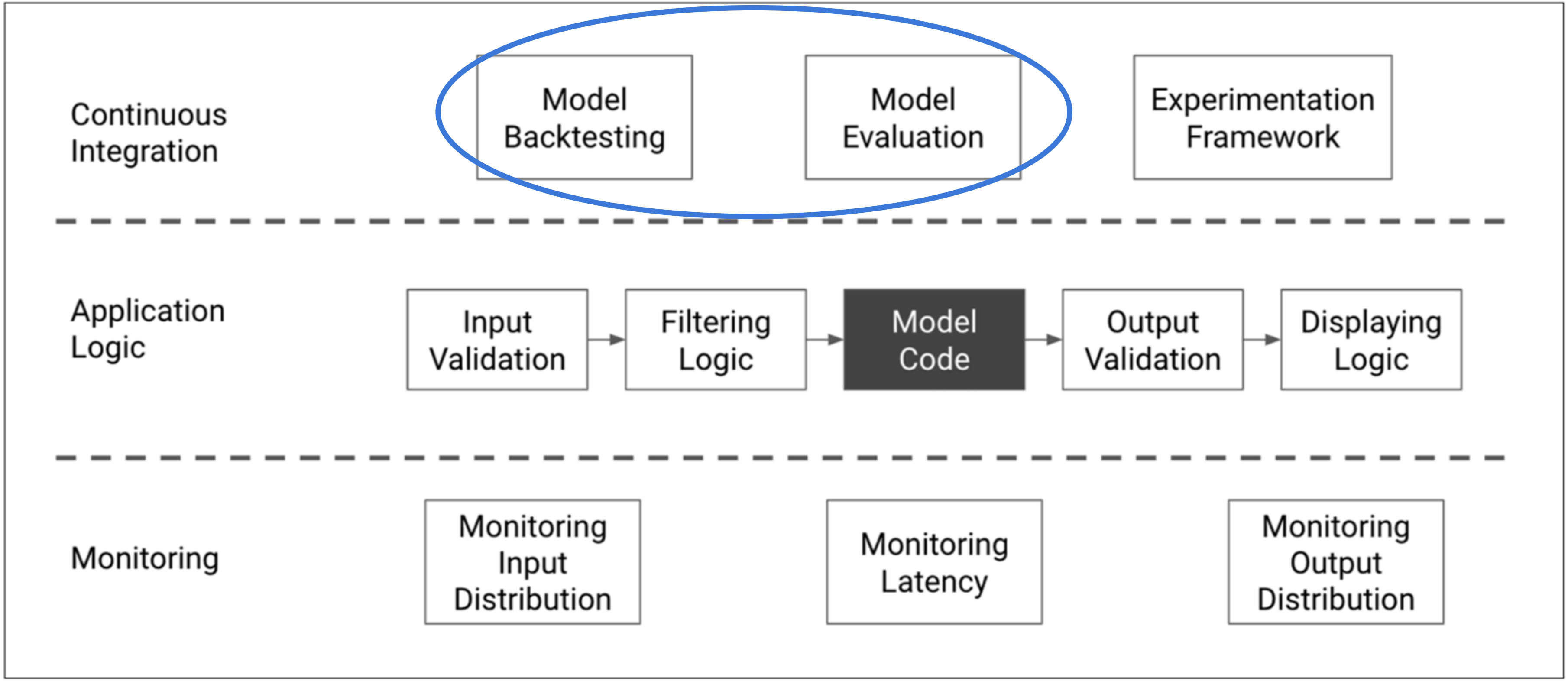


Illustration from "Building ML Powered Applications"

The dataset is a main part of the model

- Each model should be tied to a dataset
  - Feature list and date range
  - Feature versions
  - Ideally with sufficient information to train an identical model

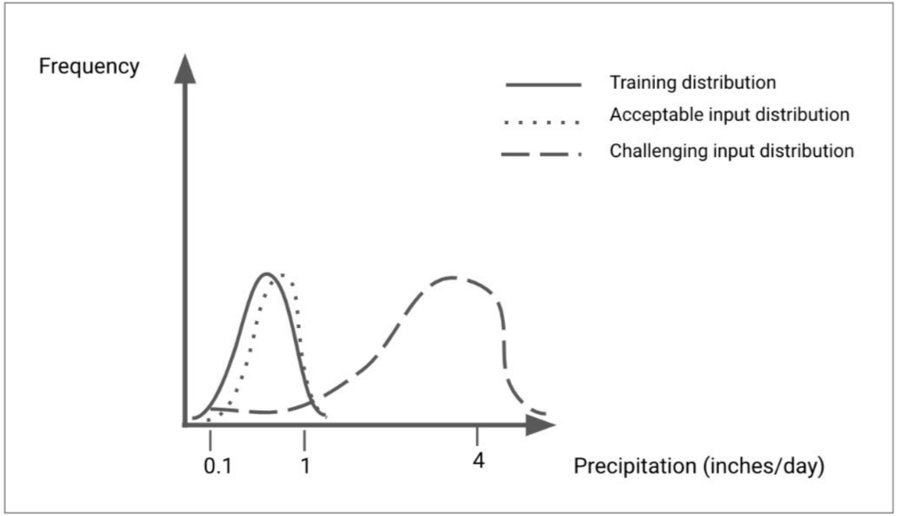


Illustration from "Building ML Powered Applications"

The challenge of generating data

- Adding new features should be as quick as possible
  - Joining with other existing features
  - Generating new derived features
  - Capturing new events to derive feature from

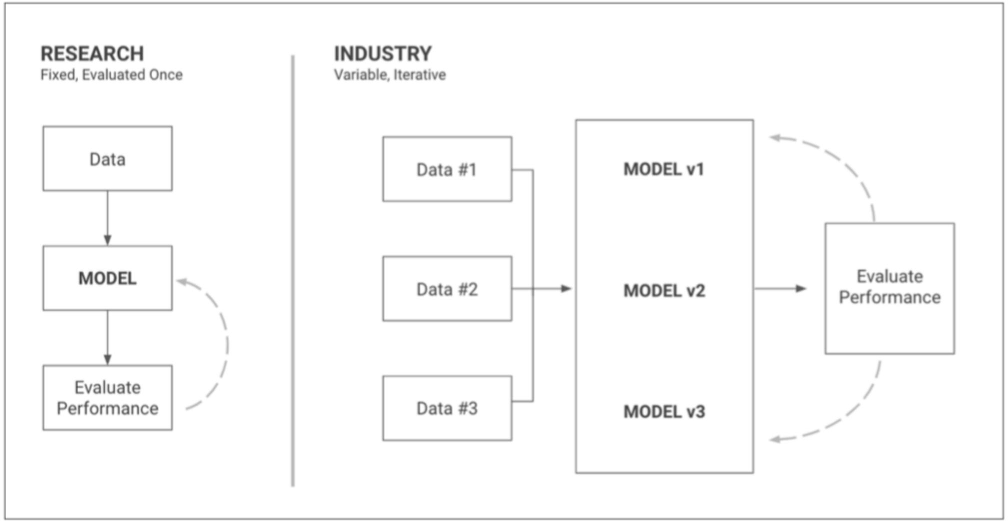


Illustration from "Building ML Powered Applications"

Feature stores and feature sharing

Stitch
Fix
Style
Shuffle

Storing arbitrary features

- Storing and sharing vectorized learned representations
- Enabling vector indexing (search applications)

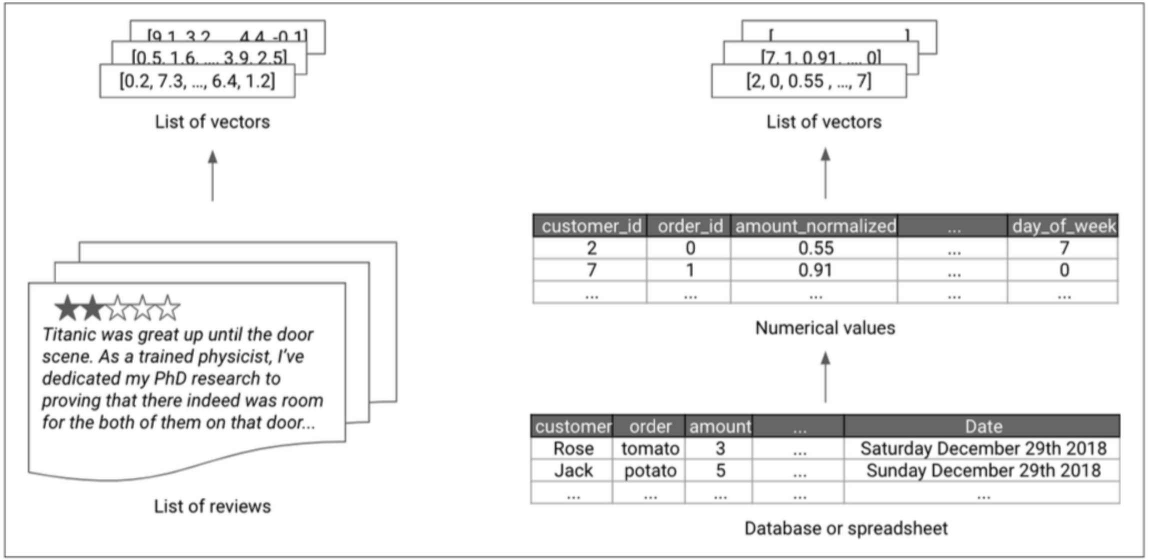


Illustration from "Building ML Powered Applications"

Data storage for ML

- Fixed dataset vs dataset as a feature
- To be reproducible, a model needs to be tied to the data it was trained on
- Feature stores can create model lift by encouraging feature sharing

Model performance validation

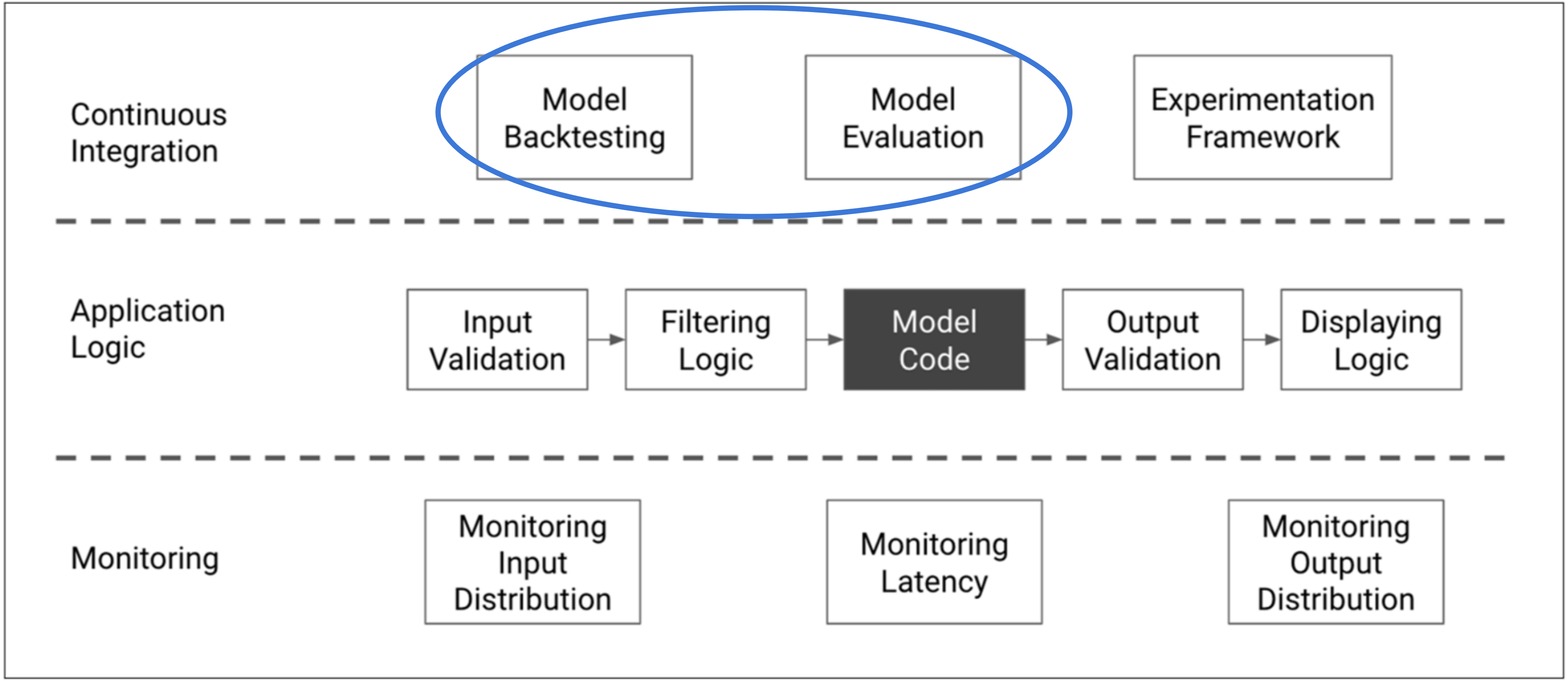


Illustration from "Building ML Powered Applications"

It is often not provable that your model will work

Regular Software

- ✓ No crash
- ✓ Tests passed
  - ✓ Unit tests
  - ✓ Integration tests
  - ✓ Regression tests

80% Confident in quality of application

ML Software

- ✓ No crash
- ✓ Tests passed
  - ✓ Unit tests
  - ✓ Integration tests
  - ✓ Regression tests
  - ✓ Distribution tests
  - ✓ Model back tests
- ✓ Accuracy, Precision, Recall

20% Confident in success of application

Illustration from "Building ML Powered Applications"

Prod data storage creates data leakage

- It is almost impossible to prevent time traveling when using prod data directly
- This data leakage leads to models being wrong in subtle ways
- Lambda architectures (Zipline, Semblance) address this with event streams

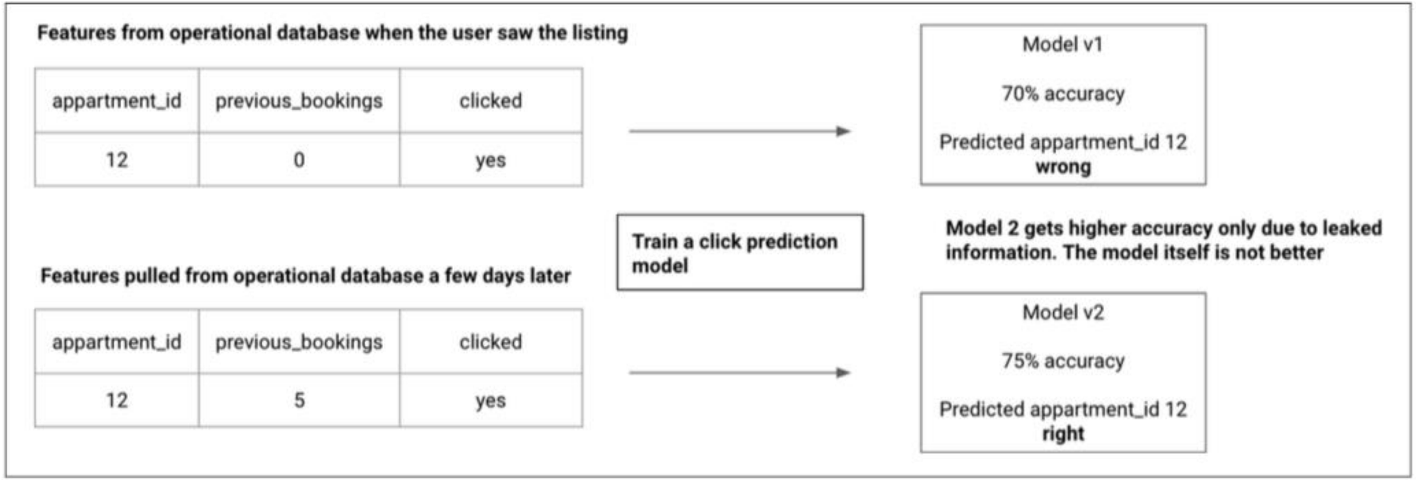


Illustration from "Building ML Powered Applications"

No system can protect a user from themselves

"I separated the data using a random split"

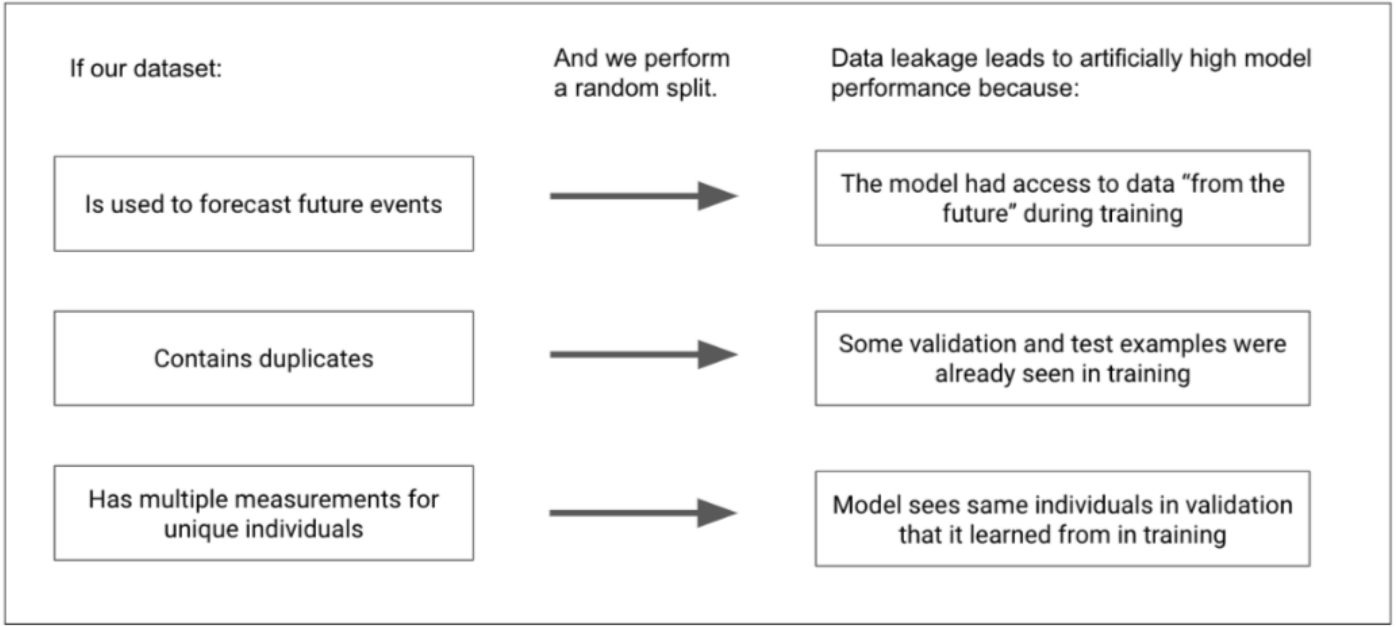


Illustration from "Building ML Powered Applications"

Moving organically along the accuracy/risk scale

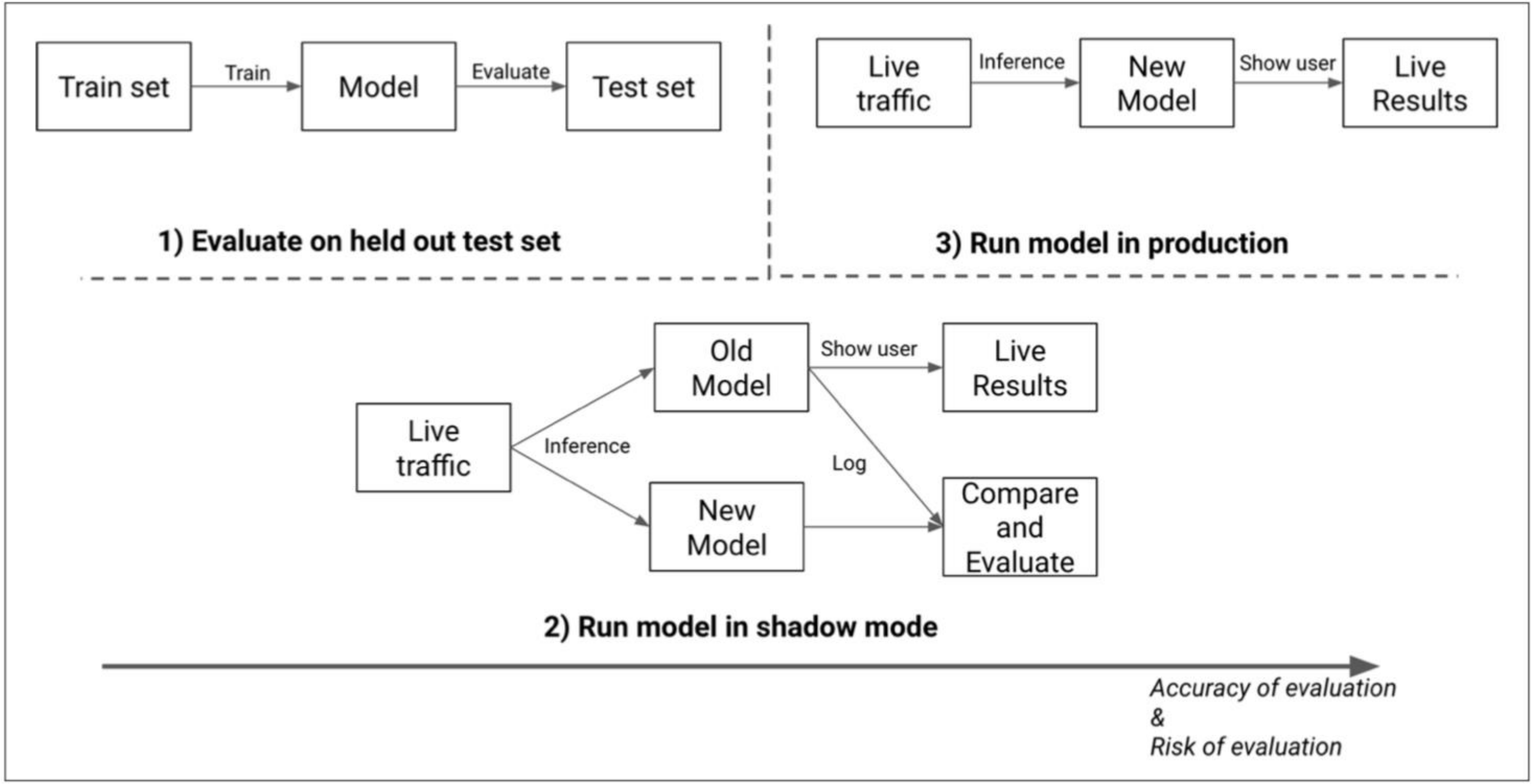


Illustration from "Building ML Powered Applications"

~~Eliminate~~ Reduce the impact of data leakage

- Simple (ideally accurate) backtesting
- Shadow scoring
- Safe deployment
  - Gradual rollout
  - Easy rollbacks

Model deployment

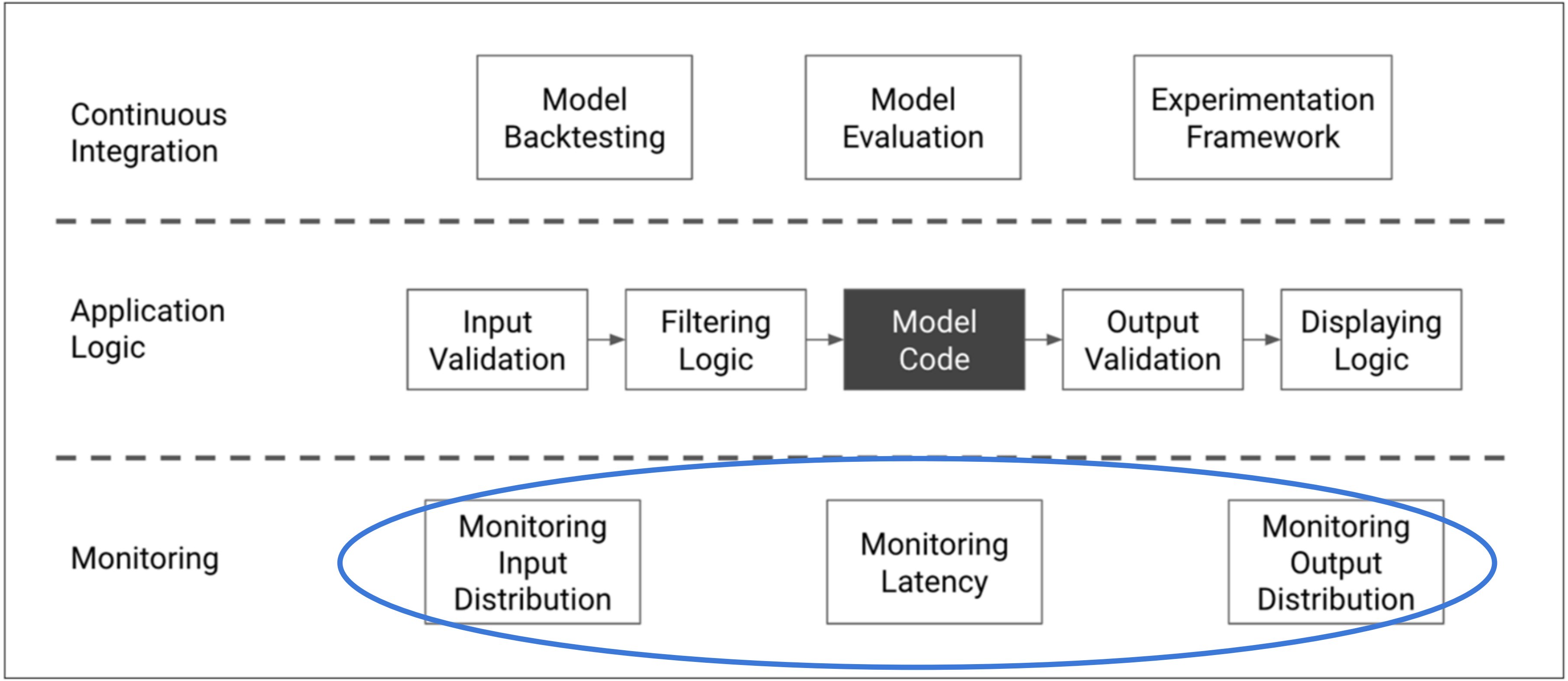


Illustration from "Building ML Powered Applications"

Model deployment is too hard for humans

"You just serialized the train model and load it in a flask app"

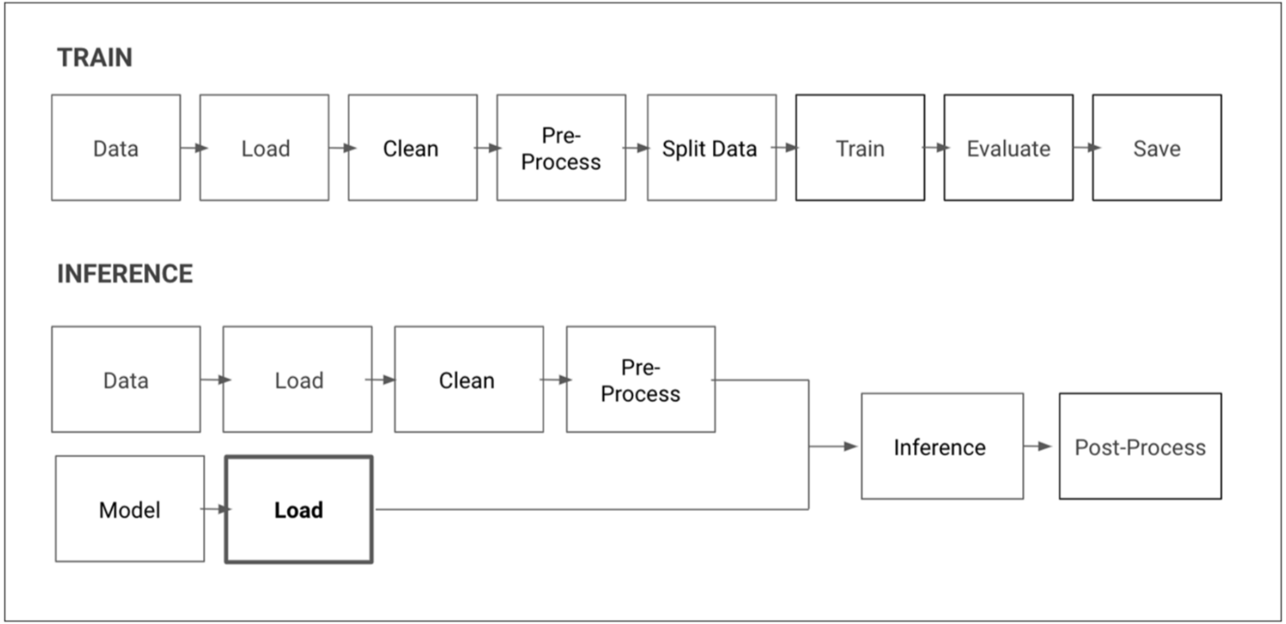


Illustration from "Building ML Powered Applications"

Model deployment with versioning

- Which date was the model trained on
- What version of the data?
- Did we filter out users from these regions?

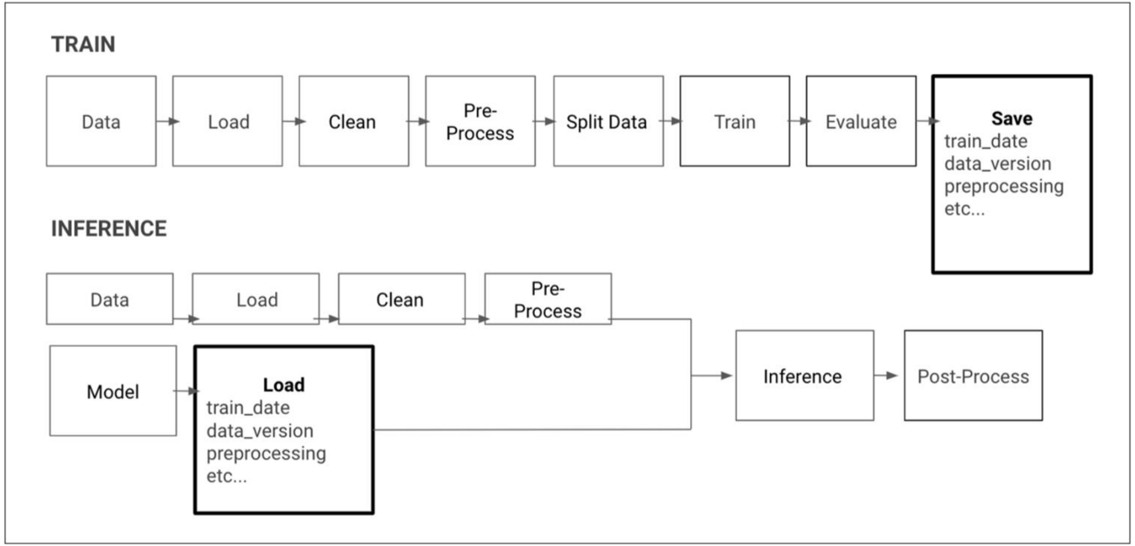


Illustration from "Building ML Powered Applications"

Model deployment with (more) versioning

- Which version of the app was this model trained on?
- Can we serve different models based on app version?

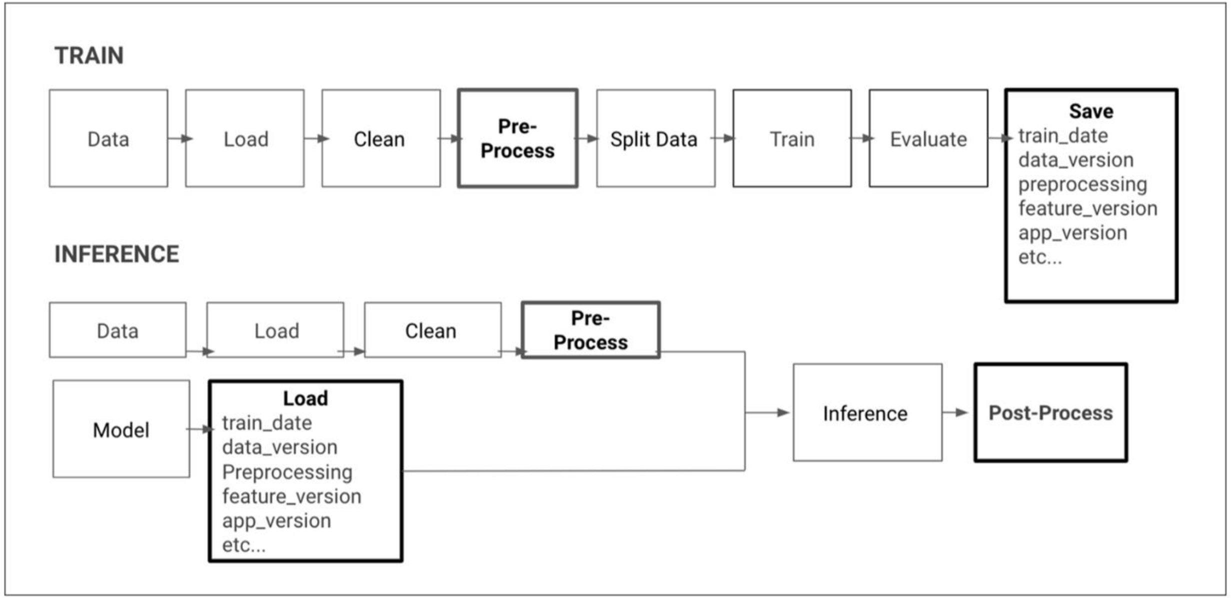


Illustration from "Building ML Powered Applications"

The fight against model staleness

- Most models go stale
- Retraining, validating, and deploying a model is toilsome

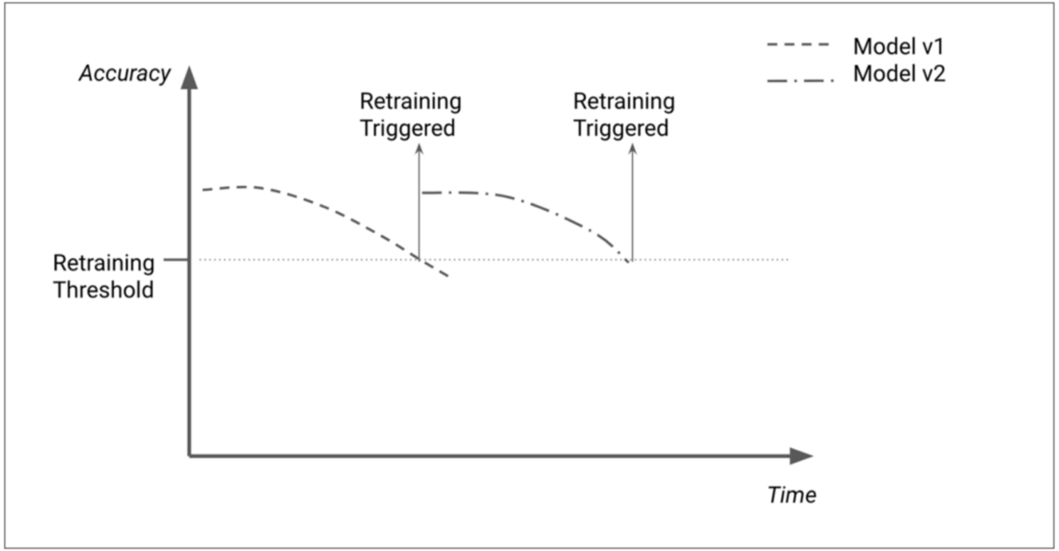


Illustration from "Building ML Powered Applications"

Helping humans deploy models

- Versioning of
  - Data
  - Model
  - Application
- Automatic redeployment
  - Determination of ideal interval
  - Automatic rollout and alerting

Monitoring and alerting

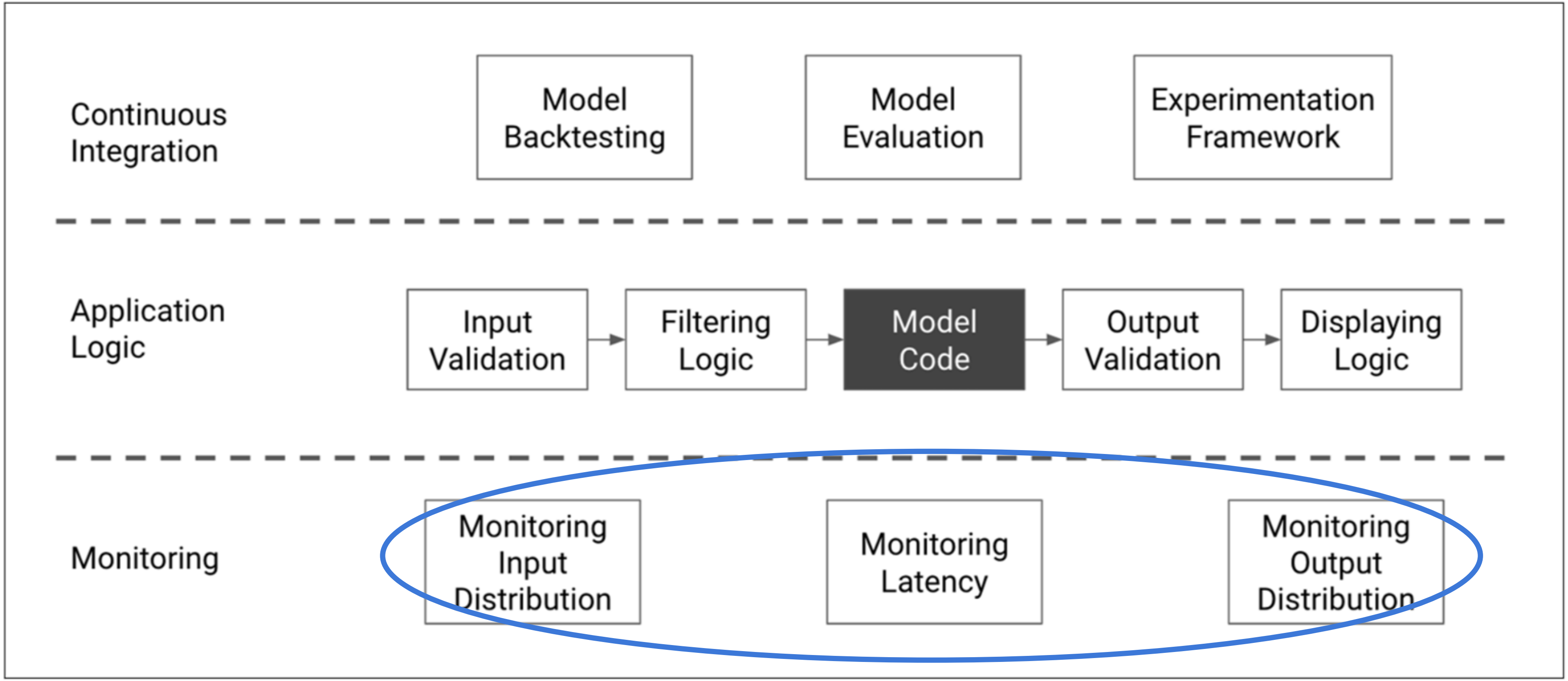


Illustration from "Building ML Powered Applications"

Shine a light on this data

- Bugs will happen
- Debugging models without seeing data is **very** hard
- True for training and inference

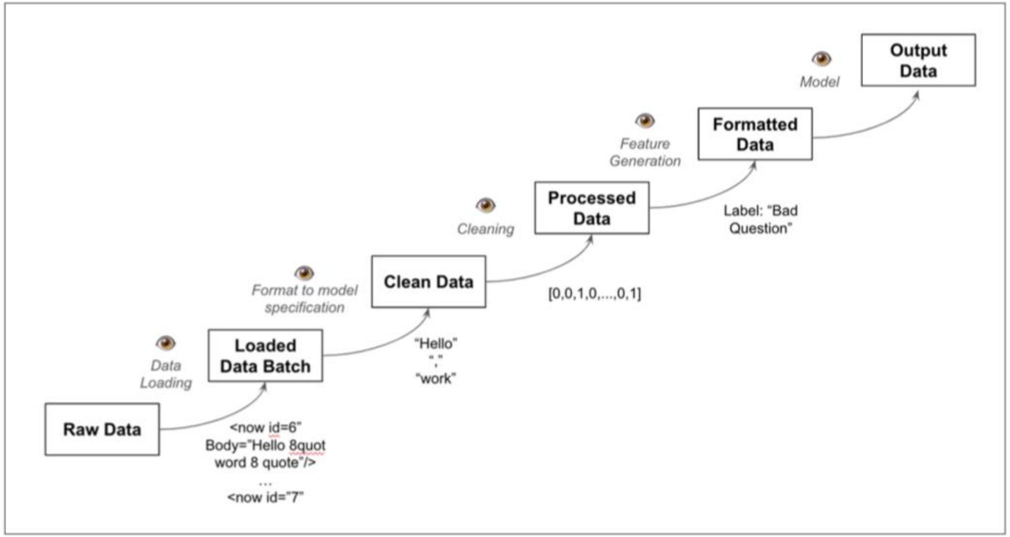


Illustration from "Building ML Powered Applications"

Alerts and testing in production

- Automated alerts can help catch simple issues
- Make it easy to tune thresholds to dial in false positives and negatives
  - Alert fatigue is real

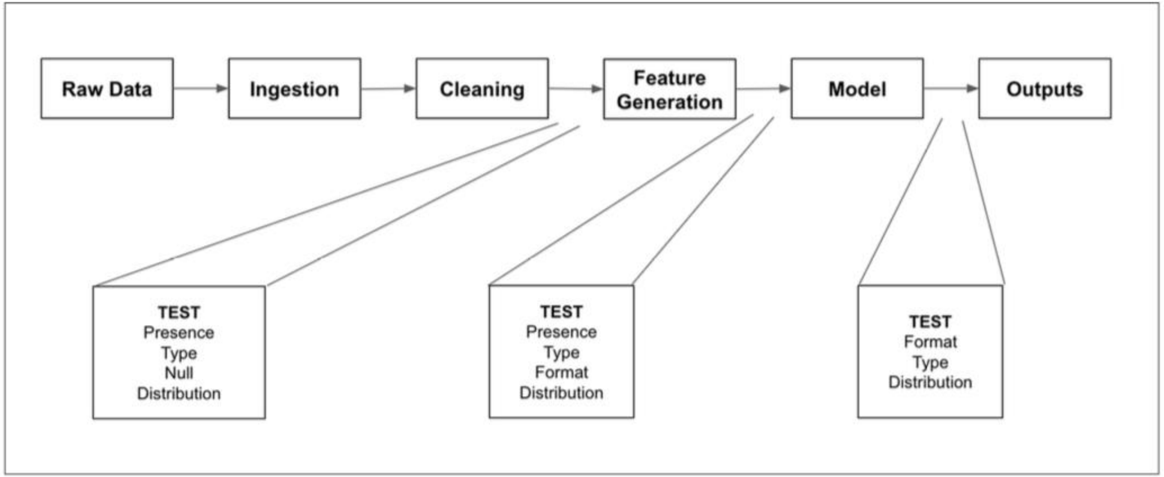


Illustration from "Building ML Powered Applications"

Error preemption

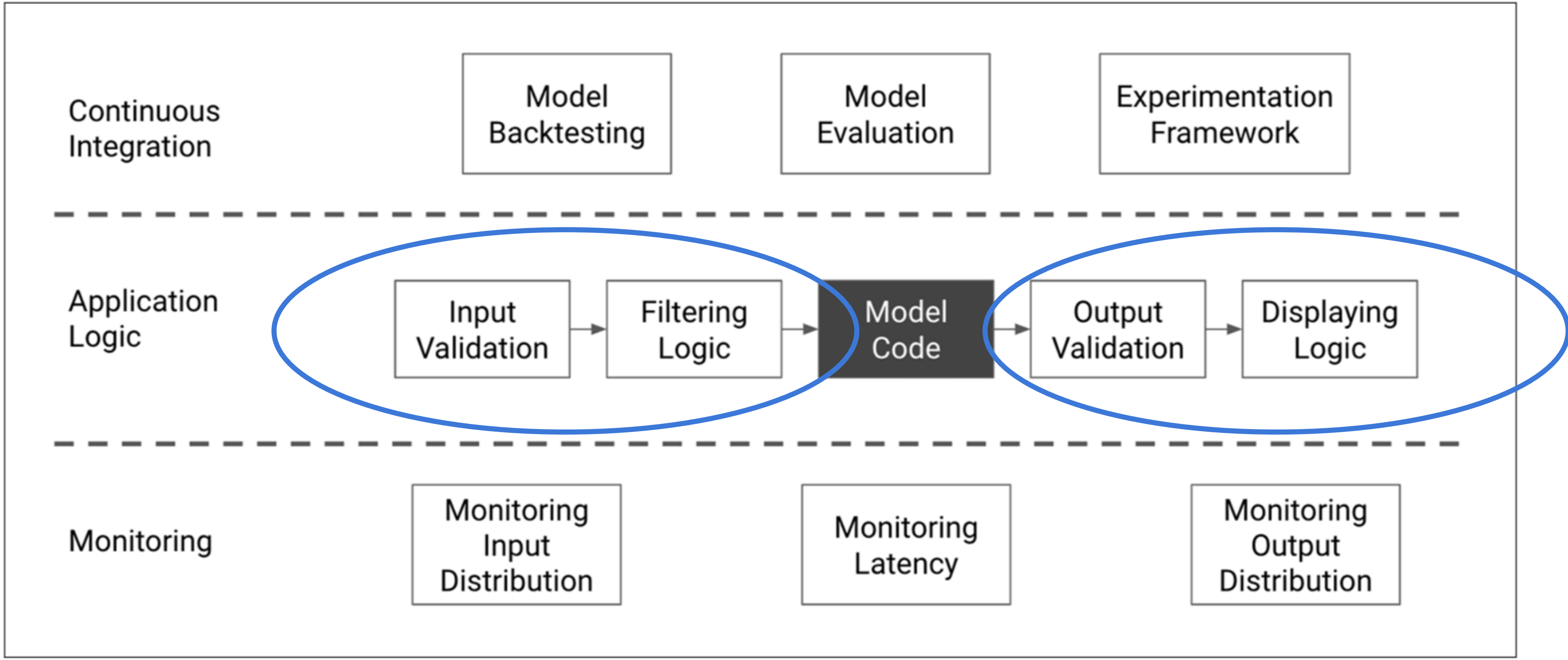


Illustration from "Building ML Powered Applications"

The wrong kind of robustness

- If the data is in the right shape, a model will make a prediction
- How can we know if the model is winging it?

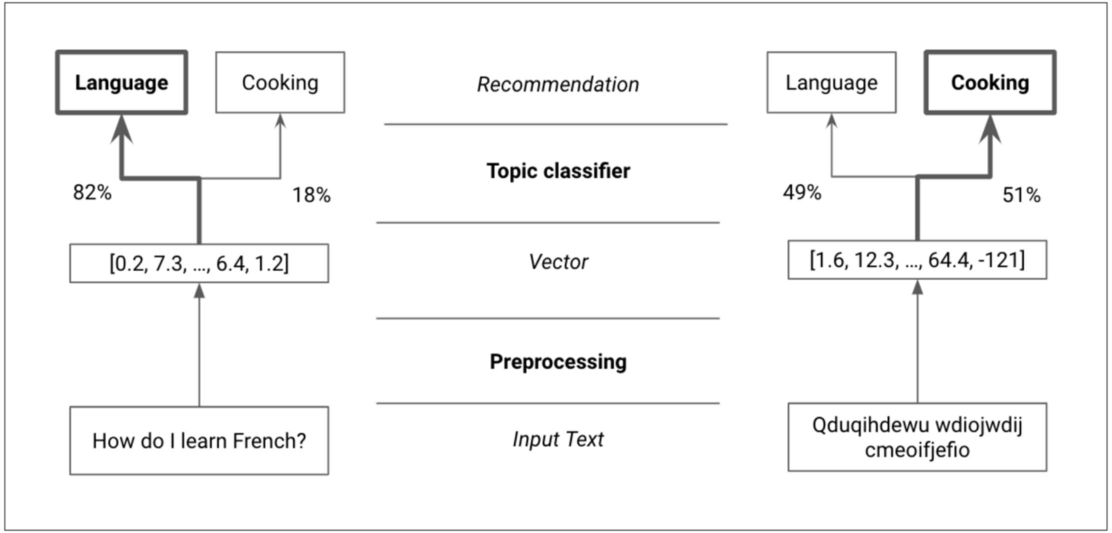


Illustration from “Building ML Powered Applications”

Branching logic for input checks

- Make it easy to check the inputs to a model and branch off
  - Presence checks
  - Statistical and range checks
  - Model confidence checks

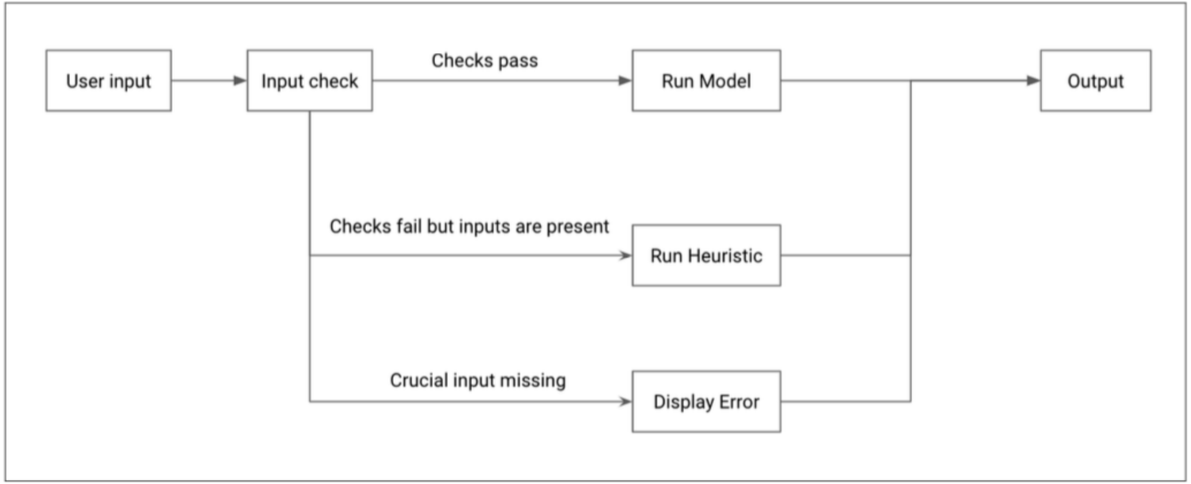


Illustration from "Building ML Powered Applications"

Anomaly detection to support models

- If you can detect some type of anomalies, don't even run models on them
    - Caveat: you may want to eventually train your model so it is self sufficient

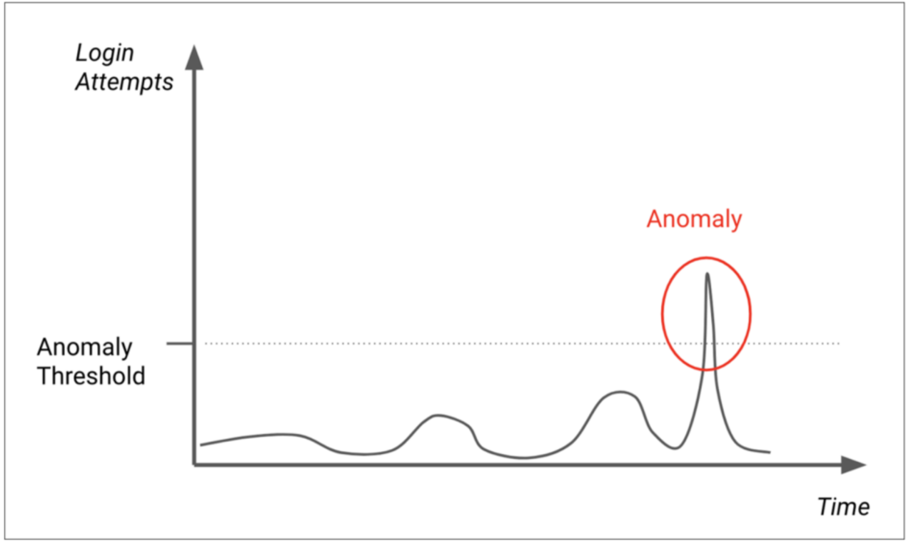


Illustration from "Building ML Powered Applications"

Adding a filtering model

- Use a simpler model to filter inputs
- Used by Google Smart Reply to decide whether to propose an answer

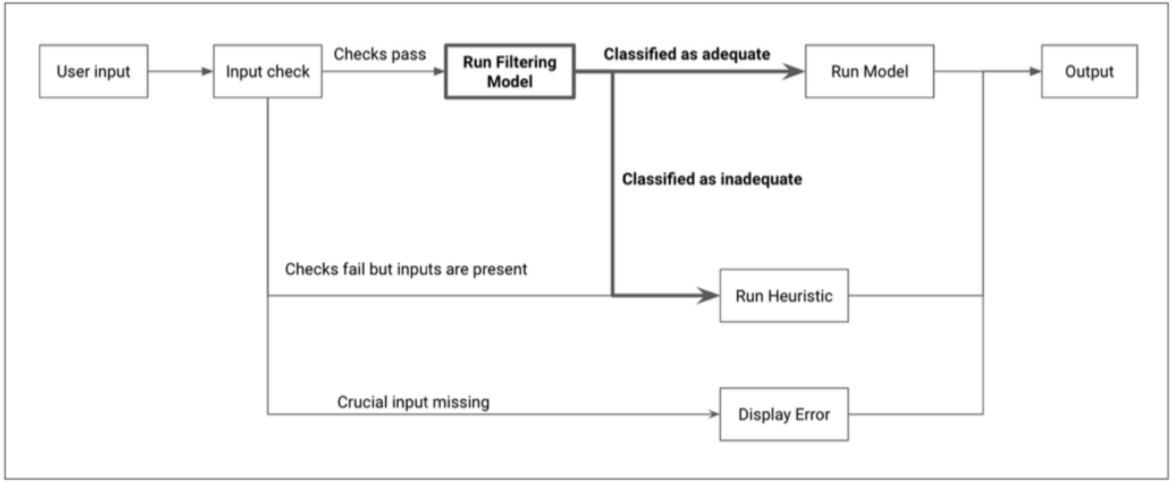


Illustration from "Building ML Powered Applications"

Models will make errors: build for it

- Heuristic fallback
- Input and model confidence checks
- Anomaly detection
- Filtering model

Poorly summarized takeaways

1. What are ML platforms
    - Tools to help manage inevitable model failures
2. Offline data management
    - Creating and combining features should be easier than trying new models
3. Model performance validation
    - Helping prevent time-traveling is valuable, but only to a certain extent
4. Model deployment
    - Infrastructure could handle re-deploying models automatically and assisting with versioning
5. Monitoring and alerting
    - Enable inspection at different parts of the supply chain, and provide tunable alerts
6. Error preemption
    - Enable input and output checks to plan for said inevitable model failures

Thank you

For more lessons learned and tips for building ML apps:

Find the first chapter at mlpowered.com/book/

Reach out to me @mlpowered