Mastering the End-to-End Machine Learning Model Building Process: Best Practices and Pitfalls

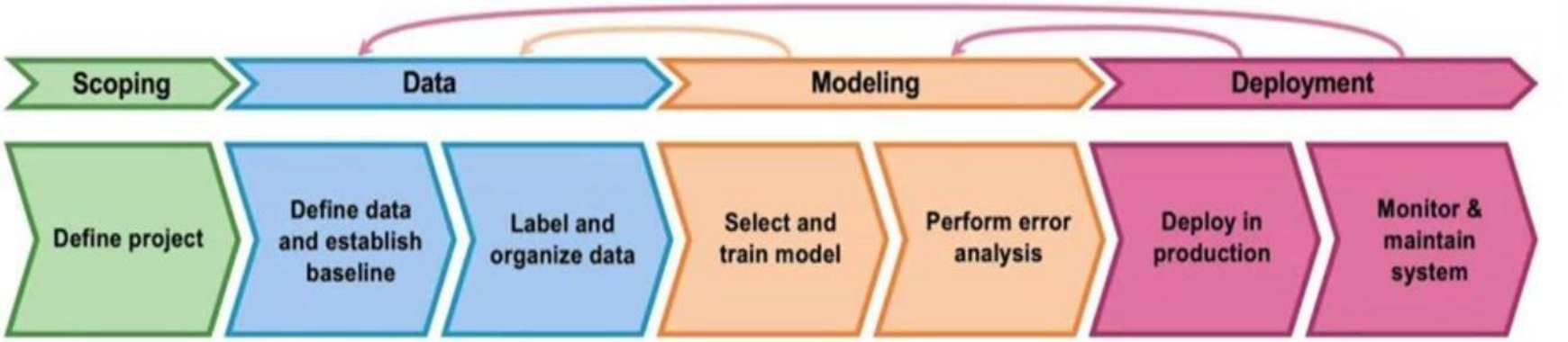
Paril Ghori

Senior Data Scientist

Agenda

1. ML Lifecycle Overview
2. Data Ingestion & Preprocessing
3. Data Preprocessing & Cleaning
4. Feature Engineering
5. Model Selection
6. Model Training & Hyperparameter Tuning
7. Evaluation & Validation
8. Deployment
9. Monitoring
10. Conclusions
11. Case Studies

ML Lifecycle Overview



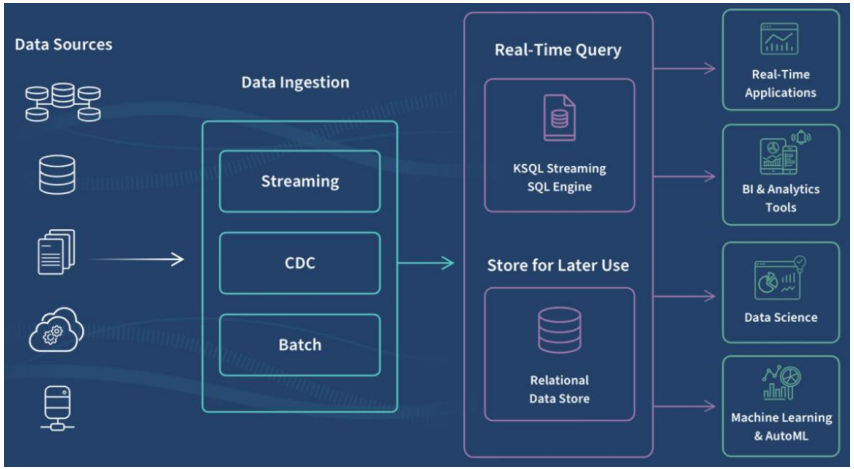
Data Ingestion & Collection

Best Practices

- ✓ Understand Requirements
- ✓ Gather Sufficient, Representative Data
- ✓ Data Versioning & Documentation
- ✓ Privacy and Compliance

Common Pitfalls

- ✗ Biased or Unrepresentative Data
- ✗ Insufficient Quantity of Data
- ✗ “Garbage In” (Poor Quality Data)
- ✗ Changing Data Definitions



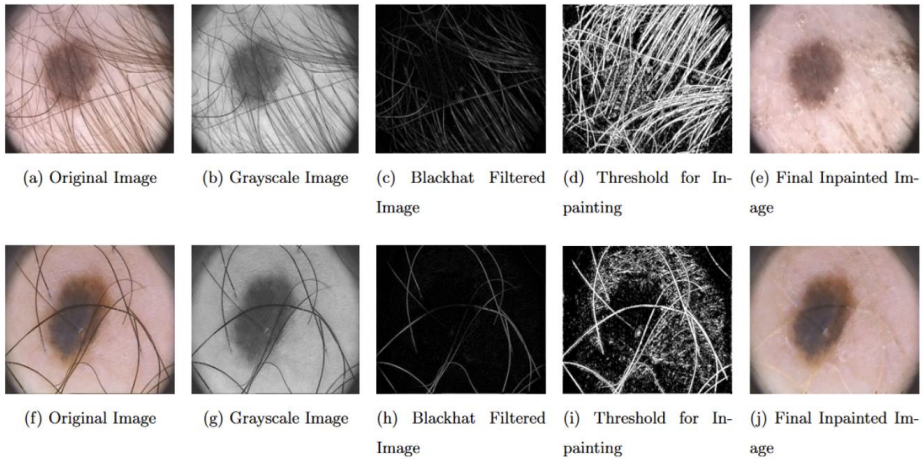
Data Preprocessing & Cleaning

Best Practices

- ✓ Exploratory Data Analysis (EDA)
- ✓ Cleaning & Imputation
- ✓ Automate Preprocessing Pipelines
- ✓ Feature Encoding

Common Pitfalls

- ✗ Target Leakage in Preprocessing
- ✗ Inconsistent Processing
- ✗ Over-cleaning or Censoring Data
- ✗ Not Normalizing When Needed



(a) Original Image (b) Grayscale Image (c) Blackhat Filtered Image (d) Threshold for Inpainting (e) Final Inpainted Image

(f) Original Image (g) Grayscale Image (h) Blackhat Filtered Image (i) Threshold for Inpainting (j) Final Inpainted Image

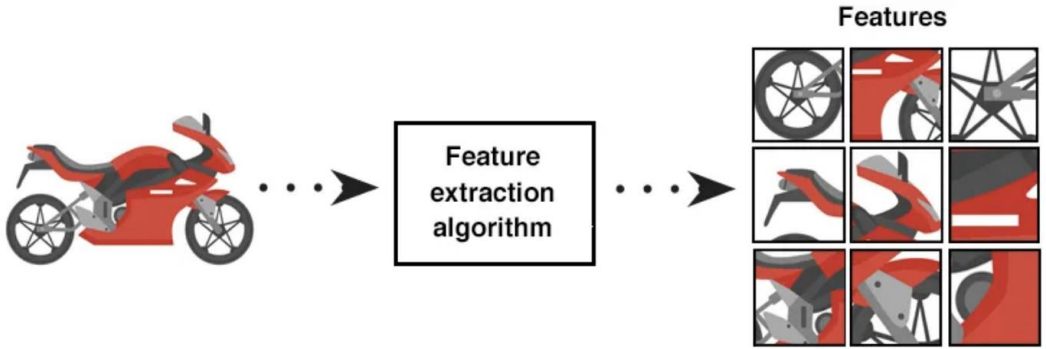
Feature Engineering

Best Practices

- ✓ Leverage Domain Knowledge
- ✓ Keep It Relevant and Simple
- ✓ Encode and Transform Properly
- ✓ Use Feature Store (for Scale)

Common Pitfalls

- ✗ Too Many Features (Curse of Dimensionality)
- ✗ Irrelevant Features
- ✗ Mis-encoded Categoricals
- ✗ Feature Leakage



Model Selection

Best Practices

- ✓ Start Simple (Baseline Models)
- ✓ Compare Multiple Algorithms
- ✓ Balance Complexity and Performance
- ✓ Cross-Validation for Reliable Choice

Common Pitfalls

- ✗ Ignoring a Simple Solution
- ✗ Not Considering Data/Model Fit
- ✗ No Regard for Interpretability or Business Constraints
- ✗ Overlooking Ensemble Downsides

project	experiment	base_model	best val acc	epochs	base lr	optimizer	lr scheduler	loss func
plant_Pathology	resnet152_v1_without_transform	resnet152_v1	0.820841	20	0.01	sgd	steplr	softmaxcrossentropy
plant_Pathology	resnet152_v1_with_transform	resnet152_v1	0.906764	20	0.01	sgd	steplr	softmaxcrossentropy
plant_Pathology	resnext101_64x4d_with_transform	resnext101_64x4d	0.875686	20	0.01	sgd	steplr	softmaxcrossentropy
plant_Pathology	resnext101_64x4d_without_transform	resnext101_64x4d	0.723949	20	0.01	sgd	steplr	softmaxcrossentropy
plant_Pathology	densenet201	densenet201	0.753199	20	0.01	sgd	multisteplr	softmaxcrossentropy
plant_Pathology	mobilenet1.0	mobilenet1.0	0.749543	20	0.01	sgd	steplr	softmaxcrossentropy
plant_Pathology	mobilenetv3_large	mobilenetv3_large	0.872029	20	0.01	sgd	steplr	softmaxcrossentropy
plant_Pathology	squeezenet1.1	squeezenet1.1	0.826325	20	0.001	sgd	steplr	softmaxcrossentropy
plant_Pathology	vgg19_bn	vgg19_bn	0.738574	20	0.001	sgd	steplr	softmaxcrossentropy

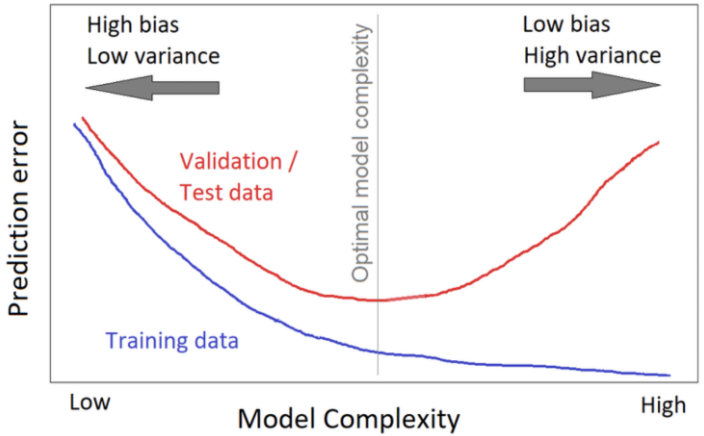
Model Training & Hyperparameter Tuning

Best Practices

- ✓ Monitor Overfitting/Underfitting
- ✓ Train/Validation/Test Split
- ✓ Track and Limit Experiments
- ✓ Systematic Search
- ✓ Automate Where Possible

Common Pitfalls

- ✗ Data Leakage During Training
- ✗ No Reproducibility
- ✗ Tuning on Test Data
- ✗ Tuning Too Many Parameters at Once
- ✗ Ignoring Hyperparameter Interactions



Model Evaluation

Best Practices

- ✓ Choose the Right Metrics
- ✓ Cross-Validate for Confidence
- ✓ Test on an Independent Dataset
- ✓ Evaluate Under Realistic Conditions

Common Pitfalls

- ✗ Overemphasis on a Single Metric
- ✗ Evaluating on Seen Data
- ✗ No Statistical Significance Analysis
- ✗ Ignoring Qualitative Evaluation

Metric	Goal	Ideal Value	Importance
Precision	Correct positive predictions	High	Crucial when the cost of false positives is high or when minimizing false detections is desired.
Recall	Identify all positive instances	High	Essential when missing positive cases is costly or when detecting all positive instances is vital.
F1 Score	Balanced performance	High	Useful when dealing with imbalanced datasets or when false positives and false negatives have different costs.
AUC	Overall classification performance	High	Important for assessing the model's performance across various classification thresholds and when comparing different models.

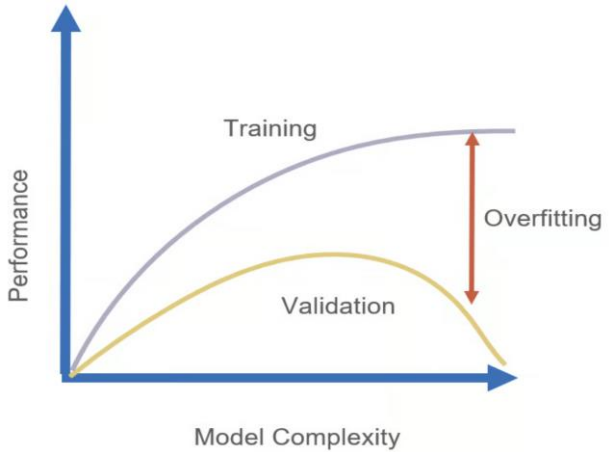
Model Validation

Best Practices

- ✓ Hold-Out Validation (Final Test)
- ✓ Simulate Production Environment
- ✓ Performance and Stress Testing
- ✓ Domain Validation & Stakeholder Approval
- ✓ Compliance and Bias Checks

Common Pitfalls

- ✗ Assuming Training Conditions Hold
- ✗ Poor Model Versioning
- ✗ Validation in Isolation
- ✗ Ignoring Edge Cases



Deployment

Best Practices

- ✓ Automate Deployment (MLOps)
- ✓ Scalability and Redundancy
- ✓ Monitoring Hooks
- ✓ Model Versioning & A/B Testing
- ✓ Rollback Plan

Common Pitfalls

- ✗ Environment Mismatch
- ✗ Missing Preprocessing in Production
- ✗ No Model Registry or Management
- ✗ Ignoring System Constraints
- ✗ Security and Privacy Oversights

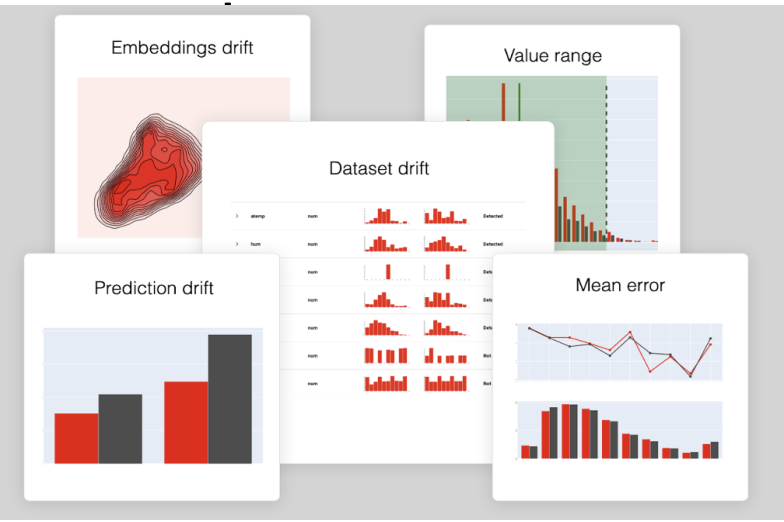
Monitoring & Maintenance

Best Practices

- ✓ Track Performance Over Time
- ✓ Data Drift & Concept Drift Detection
- ✓ Regular Retraining Schedule
- ✓ A/B Testing Model Updates

Common Pitfalls

- ✗ Over-Reliance on the Model
- ✗ Alert Fatigue or No Alerts
- ✗ Not Logging Predictions



Conclusion

1. **Data Quality is Everything:** Poor image quality and labeling errors are the most common root causes of failure. Invest in representative, high-quality datasets upfront.
2. **Start Simple, Scale Wisely:** A baseline CNN or pre-trained model often performs surprisingly well. Complexity ≠ performance. Build up only when justified.
3. **Preprocessing & Pipelines Should Mirror Reality:** What works in the lab may fail in production. Simulate real-world noise, lighting, and occlusions during training.
4. **Monitoring Doesn't End at Deployment:** Track model drift, false positives, and environment changes. Visual models are particularly vulnerable to concept drift.
5. **Stakeholder Trust Depends on Explainability:** Visual explanations (e.g., Grad-CAM) help bridge the gap between model predictions and human understanding, especially in safety-critical applications.

Case Study

📸 Case Study: Concept Drift in Computer Vision – Amazon Go Checkout Failures

Amazon Go stores rely on a complex computer vision system to enable “Just Walk Out” technology, allowing customers to pick up items and leave without manual checkout. Cameras and sensors identify products selected by shoppers and automatically charge their account.

In some locations, particularly during store rollouts, customers began reporting **checkout failures** — either being charged for wrong items or missing charges entirely. These failures created trust issues and operational bottlenecks.

🔍 What went wrong?

The core issue was **visual concept drift**. As new store layouts, lighting conditions, seasonal product packaging, and customer behavior (e.g., heavy coats in winter, reusable bags) changed over time, the trained vision system struggled to generalize.

For example, a sandwich wrapped in bright yellow summer packaging was easily recognized — but its winter version with darker, reflective wrapping caused misclassification. Additionally, shadows or hand occlusion in dimmer aisles led to dropped detections.

The model had been trained on a fixed dataset representing only ideal store conditions. Without continuous updates or adaptive learning, its performance degraded silently — a classic case of **unmonitored concept drift in real-world environments**.

🛠️ Lesson:

Even high-performing CV models can fail in dynamic retail environments if not actively monitored and retrained. Incorporating edge feedback loops, drift detection, and synthetic data augmentation can prevent costly blind spots.

Resource

- **ML Lifecycle details**
  https://www.datacamp.com/blog/machine-learning-lifecycle-explained
- **Data Ingestion Case Study**
  https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G/?pStoreID=hpepp%27%5B0%5D#:~:text=That%20is%20because%20Amazon%27s%20computer,rs%2F2OfPWoD
- **MLOps and Deployment**
  https://ml-ops.org/content/mlops-principles
- **Concept Drift Detection**
  https://medium.com/@thomaslambart/concept-drift-detection-an-overview-d087feea9676
- **Monitoring ML Models in Production**
  https://neptune.ai/blog/how-to-monitor-your-models-in-production-guide

Some Books

- *"Designing Machine Learning Systems"* by Chip Huyen (Covers best practices for deployment, monitoring, and lifecycle management)
- " Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists " by Alice Zheng and Amanda Casari *(Deep dive into best practices for feature engineering.)*
- *" Building Machine Learning Powered Applications: Going from Idea to Product" by Emmanuel Ameisen (End-to-end guide for taking ML projects to production, excellent for lifecycle discussions.)*