دانشکار

{ یک یادگیری کارساز }

بوتکمپ

Artificial Intelligence

MLOps

Session 2

Welcome Back

ML Development Workflow & Version Control

- Recap: MLOps bridges ML research and production
- Today's Focus: Professional ML development practices
- Learning Outcomes:
  - Research → Production mindset shift
  - ML project organization best practices
  - Version control for code, data, and models
  - Reproducibility strategies

Today's Journey

What We'll Cover:

Mindset Shift	Research vs Production thinking
Project Structure	Organization for scale
Version Control	Code, Data, Models
Reproducibility	Reliable results

The Research Mindset Story

Meet Dr. Sarah - Brilliant Researcher

- **Research Mode:**
  - defect_detection_final_v3_REALLY_FINAL.ipyb
  - Hardcoded paths: /Users/sarah/Desktop/phone_images/
  - Copy-paste between cells
  - "It works on my machine!“

- **Production Questions:**
  - "Can others run your code?"
  - "What if we get new data?"
  - "Which model performed best?"
  - "Can we reproduce this?"

Research vs Production Mindset

Two Different Optimizations

Research Mindset	Production Mindset
"Does this work?"	"Does this work reliably at scale?"
Quick & dirty experiments	Clean, documented experiments
Individual exploration	Team collaboration
"Works on my machine"	"Works on any machine"
Discovery focused	Deployment focused

Key Insight: Production practices accelerate research!

Chaotic vs Professional Structure

Project Organization: Chaos vs Clarity

Chaotic Approach	Professional Structure
my_ml_project/ ├── notebook1.ipynb ├── notebook2_copy.ipynb ├── final_model_v2.ipynb ├── data.csv ├── model.pkl └── random_utils.py	phone_defect_detection/ ├── README.md ├── requirements.txt ├── config/ ├── data/ ├── src/ ├── models/ └── notebooks/
❓ Where do I start?	✅ Clear story!

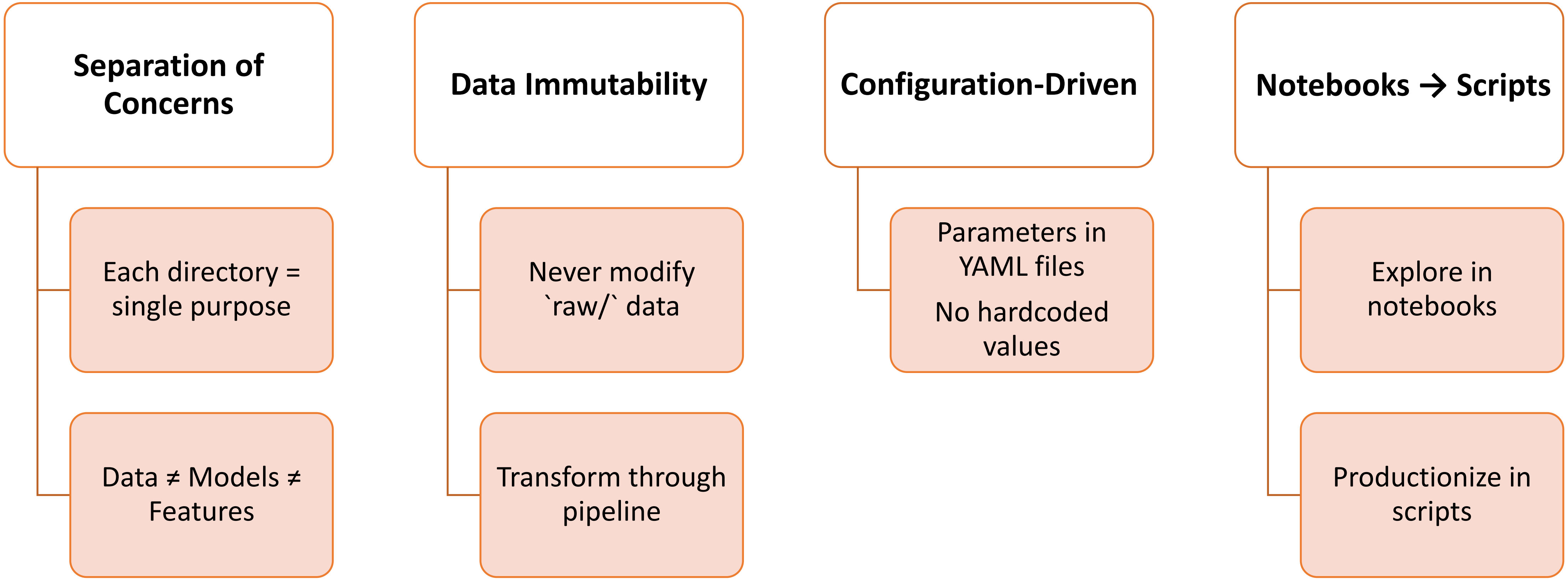
Professional ML Project Structure

VisionaryAI Standard Structure

```
phone_defect_detection/
├── README.md                 # 📖 Project overview
├── requirements.txt          # 📦 Dependencies
├── config/                   # ⚙️ Configuration files
├── data/
│   ├── raw/                  # 🔒 Original, immutable
│   ├── interim/              # 🔄 Intermediate processing
│   └── processed/            # ✅ Final model data
├── src/                      # 💻 Source code
│   ├── data/                 # Data processing
│   ├── features/             # Feature engineering
│   └── models/               # Model training
├── models/                   # 🎯 Trained models
└── notebooks/                # 🔬 Exploration only
```

Key Structure Principles

Four Pillars of ML Project Organization



Separation of Concerns

- Each directory = single purpose
- Data ≠ Models ≠ Features

Data Immutability

- Never modify `raw/` data
- Transform through pipeline

Configuration-Driven

- Parameters in YAML files

  No hardcoded values

Notebooks → Scripts

- Explore in notebooks
- Productionize in scripts

VisionaryAI Project Examples

Structure in Action

Computer Vision:

```
src/data/image_preprocessing.py
src/features/defect_features.py
src/models/cnn_classifier.py
```

NLP:

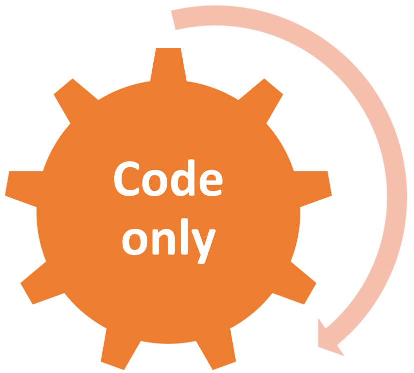
```
src/data/ticket_cleaner.py
src/features/text_vectorizer.py
src/models/ticket_classifier.py
```

Recommender:

```
src/data/user_item_matrix.py
src/features/collaborative_features.py
src/models/recommendation_engine.py
```

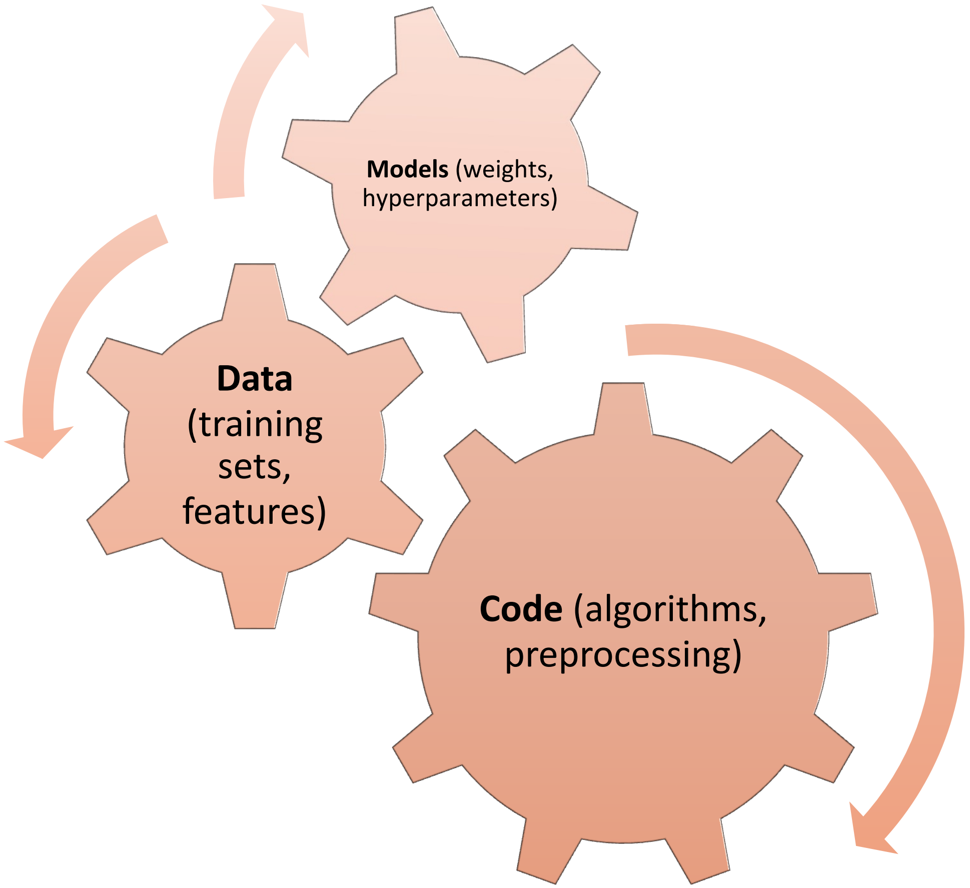
Version Control Complexity

ML's Triple Challenge



Code only

Traditional Software



Models (weights, hyperparameters)

Data (training sets, features)

Code (algorithms, preprocessing)

ML Projects

The Challenge: All three change together!

The Nightmare Scenario

Friday Afternoon Horror Story

- **Questions:**
  - What changed?
  - When did it change?
  - How do we fix it?
- **Reality:**
  - Unclear code changes
  - Multiple "final" models
  - Overwritten notebooks
  - Undocumented hyperparameters
- **Result:** Weekend debugging session

Crisis:

Recommender system performance drops 15%

ML Git Branching Strategy

Branching for Experiments

- **Traditional Branches:**
  - `main` - Production code
  - `feature/user-login` - New features
  - `bugfix/payment-error` - Bug fixes
- **ML Experiment Branches:**
  - `experiment/resnet-architecture` - Model experiments
  - `data/add-validation-set` - Data changes
  - `model/hyperparameter-tuning` - Model optimization

Key: Experiments are temporary, learning is permanent

Collaboration Challenge

VisionaryAI Recommendation Team

Alice:
"Trying collaborative filtering"

Bob:
"Working on content features"

Dave:
"Testing deep learning embeddings"

Carol:
"Improving data preprocessing"

❓ Challenge: How do they collaborate without conflicts?

✅ Solution: Clear branching strategy + shared experiment tracking

Reproducibility Crisis

Why Reproducibility Matters

Scenario 1: Model works in dev, fails in production

- Was it data? Code? Model?
- No reproducibility = No debugging

Scenario 2: Monthly model retraining

- Need consistent quality over time
- Can't reproduce = Can't improve

Scenario 3: Performance claims challenged

- Need to defend results
- No reproducibility = No credibility

ML Reproducibility Challenges

Why ML is Harder to Reproduce

- **Randomness Everywhere**
  - Data splits, weight initialization
  - Training sampling, data augmentation
- **Environment Dependencies**
  - Package versions, GPU drivers
  - OS differences, hardware variations
- **Hidden State**
  - Cached results, checkpoints
  - External APIs, downloaded models
- **Complex Pipelines**
  - Multi-step processes
  - Interdependent components

Reproducibility Solutions

VisionaryAI Reproducibility Checklist:

- ✅ All code version controlled
- ✅ All data versions tracked
- ✅ Random seeds set and documented
- ✅ Dependencies pinned to exact versions
- ✅ Training environment containerized
- ✅ Single-command model training
- ✅ Results reproducible within 2% by others

Standard: Every production model must pass this checklist

Code Demo Preview

Let's See This in Action

Project: Phone Defect Detection System

We'll Demonstrate:

Project Structure Setup:

- Professional organization
- Configuration management

Git Workflow:

- ML-specific branching
- Collaboration strategies

Handling Large Files:

- Data versioning concepts
- Model artifact management

Key Takeaways

Transform Your ML Development

1. **Think Production from Day One** - Clean structure accelerates research
2. **Structure Tells a Story** - Make navigation obvious
3. **Version Control Everything** - Code, data, and models
4. **Reproducibility is Business Critical** - Not just good practice
5. **Collaboration Requires Planning** - Modified Git workflows for ML

Next Session Preview

Coming Up: Environment Management

- **Next Session's Focus:**
  - Virtual environments
  - Containerization with Docker
  - Configuration management
  - Dependency management
- **Question:** Ready to make your ML projects truly reproducible?

THANK YOU

FOR YOUR ATTENTION!