From Student Project → Real ML Pipeline (Complete Reference)

This document is a production-grade ML pipeline reference. Every folder, file, and rule is explained. Nothing is skipped.

1. Complete ML Project Tree (With Comments)

```
mlproject/                          # Project root (never rely on where code is executed)
■
■■■ artifacts/                      # All outputs produced by the pipeline
■   ■
■   ■■■ run_YYYY-MM-DD_HH-MM-SS/    # One isolated folder per pipeline run
■       ■
■       ■■■ data/                   # Data produced/used during the run
■       ■   ■■■ raw.csv             # Raw ingested data
■       ■   ■■■ train.csv           # Train split
■       ■   ■■■ test.csv            # Test/validation split
■       ■
■       ■■■ model/                  # Trained models
■       ■   ■■■ model.pkl           # Serialized ML model
■       ■   ■■■ encoder.pkl         # Encoders / preprocessors
■       ■
■       ■■■ metrics/                # Evaluation outputs
■           ■■■ metrics.json        # Accuracy, RMSE, etc.
■           ■■■ plots/              # Confusion matrix, ROC, feature importance
■
■■■ logs/                           # Logs explain *why* something happened
■   ■
■   ■■■ run_YYYY-MM-DD_HH-MM-SS/    # Same RUN_ID as artifacts
■       ■■■ app.log                 # Central application log
■
■■■ src/                            # All source code lives here
■   ■
■   ■■■ config/                     # Configuration & environment-agnostic setup
■   ■   ■■■ paths.py                # Single source of truth for all paths
■   ■   ■■■ params.yaml             # Hyperparameters, feature lists, thresholds
■   ■
■   ■■■ pipelines/                  # Orchestration layer (controls execution)
■   ■   ■■■ training.py             # End-to-end training pipeline
■   ■   ■■■ evaluation.py           # Evaluation-only pipeline
■   ■
■   ■■■ components/                 # Pure logic, reusable building blocks
■   ■   ■■■ ingestion.py            # Reads raw data and saves artifacts
■   ■   ■■■ validation.py           # Schema checks, missing values, ranges
■   ■   ■■■ training.py             # Model training logic only
■   ■   ■■■ evaluation.py           # Metrics calculation only
■   ■
■   ■■■ logger.py                   # Centralized logging configuration
■
■■■ README.md                       # Project documentation & usage instructions
```

2. paths.py – Centralized Path Standardization

```
import os
from datetime import datetime
```

```
# Unique ID for every pipeline execution
RUN_ID = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")

# Project root anchored to code location, NOT execution directory
PROJECT_ROOT = os.path.abspath(
    os.path.join(os.path.dirname(__file__), "..", "..")
)

# Run-specific root folders
ARTIFACTS_DIR = os.path.join(PROJECT_ROOT, "artifacts", f"run_{RUN_ID}")
LOGS_DIR = os.path.join(PROJECT_ROOT, "logs", f"run_{RUN_ID}")

# Artifact subfolders
DATA_DIR = os.path.join(ARTIFACTS_DIR, "data")
MODEL_DIR = os.path.join(ARTIFACTS_DIR, "model")
METRICS_DIR = os.path.join(ARTIFACTS_DIR, "metrics")

# Create all directories once at startup
for path in [ARTIFACTS_DIR, LOGS_DIR, DATA_DIR, MODEL_DIR, METRICS_DIR]:
    os.makedirs(path, exist_ok=True)
```

3. logger.py – Centralized Logging

```
from src.config.paths import LOGS_DIR
import logging
import os

LOG_FILE_PATH = os.path.join(LOGS_DIR, "app.log")

logging.basicConfig(
    filename=LOG_FILE_PATH,
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s"
)

logger = logging.getLogger(__name__)
```

4. Component Example (No Environment Logic)

```
from src.config.paths import DATA_DIR
import pandas as pd
import os

def ingest_data(source_path: str):
    raw_data_path = os.path.join(DATA_DIR, "raw.csv")
    df = pd.read_csv(source_path)
    df.to_csv(raw_data_path, index=False)
    return raw_data_path
```

5. Pipeline Example (Orchestration Only)

```
from src.components.ingestion import ingest_data
from src.logger import logger

def run_training_pipeline():
    logger.info("Training pipeline started")
    data_path = ingest_data("data/source.csv")
    logger.info(f"Data ingested at {data_path}")
    logger.info("Training pipeline completed successfully")
```

6. Non-Negotiable Rules

• Never use os.getcwd()
• Never hardcode paths inside components
• One pipeline run = one RUN_ID
• Pipelines orchestrate, components execute
• Logs explain WHY, artifacts show WHAT
• paths.py is the single source of truth