Machine Learning

AI Professionals Bootcamp | Week 3

2025-12-28

GenAI policy reminder (Week 3)

You may use Generative AI **only** for clarifying questions.

- ✅ Allowed: definitions, “what does this error mean?”, reading docs, concept explanations
- ❌ Not allowed: generating your solution code, debugging by copy-paste, writing full functions

> ⚠️ **Warning**
>
> This week is a graded repo. If GenAI writes your training/predict code, you will not build the skill.

Announcements / admin

- This week you will ship a **baseline ML system** (train + evaluate + batch predict)
- **Offline-first:** your repo must run without internet
- Daily habit: **commit + push** (at least 1 commit/day)
- Hidden tests reward: determinism (seeded) + helpful failures (clear errors)

> **Note**
>
> Capstone teams + project ideas are finalized by end of **Week 5 (Jan 15, 2026)**.

Day 1: ML in one picture + your dataset contract

Goal: understand the supervised learning loop and write a clear **dataset contract** for the Week 3 project.

Bootcamp • SDAIA Academy

Today’s Flow

- **Session 1 (60m):** ML in one picture (supervised learning basics)
- *Asr Prayer (20m)*
- **Session 2 (60m):** From feature table → dataset contract (X/y, IDs, leakage)
- *Maghrib Prayer (20m)*
- **Session 3 (60m):** Repo + CLI orientation (what you will run all week)
- *Isha Prayer (20m)*
- **Hands-on (120m):** Generate sample data + draft your model card + Git push

Learning Objectives

By the end of today, you can:

- Explain **supervised learning** in 1 sentence
- Distinguish **training** vs **inference** (what columns are available)
- Identify **features vs target vs ID** columns in a table
- Run `ml-baseline --help` and `ml-baseline make-sample-data`
- Create `reports/model_card.md` with a draft **dataset contract**

Warm-up: connect Week 2 → Week 3 (5 minutes)

In pairs, answer:

1. In Week 2, what did “**one row**” represent in your feature table?
2. Name **one ID column** you would keep to join predictions back.
3. Where should a feature table live in a repo?

Checkpoint: you can say “one row = ***”, “ID =*** ”, “path = `data/processed/...`”.

Week 3 project (in plain English)

By Friday, your repo can:

- read a feature table from `data/processed/features.<csv|parquet>`
- train a baseline model and save a **run folder** under `models/runs/<run_id>/`
- batch predict on a new file and write `outputs/preds.csv`
- explain what you did in `reports/model_card.md` + `reports/eval_summary.md`

Today we only build the foundation: ML basics + dataset contract + “hello CLI”.

End-state demo (what you'll run later this week)

Train

```
uv run ml-baseline train --target is_high_value
```

Predict

```
uv run ml-baseline predict --run latest \
  --input data/processed/features.csv \
  --output outputs/preds.csv
```

Note

Don't worry about the options yet. This week is: **define** → **split** → **baseline** → **train** → **evaluate** → **save** → **predict** → **report**.

Session 1

ML in one picture (supervised learning basics)

Session 1 objectives

- Define **features** and **target**
- Explain the supervised learning loop at a high level
- Tell the difference between training time and inference time

What is supervised machine learning?

You have examples where the **answer is known**.

- **Input (features):** information you have *before* a decision
- **Output (target):** the label/value you want to predict
- A model learns a rule: **X** → **y**

> **Tip**
>
> In this bootcamp, “ML” means: “turn a table into a reliable prediction pipeline.”

One picture: the supervised learning loop

```
Feature table (X + y)
   |
   | split (simulate “new data”)
   v
Train split  ——> fit model ——> save model artifacts
Holdout split —> evaluate  ——> metrics + tables

New file (X only) —> load model —> predict —> preds.csv
```

Today: we focus on understanding the pieces (not the math).

Features vs target (X vs y)

Remember - **X (features):** what you know at prediction time - **y (target):** what you want the model to output - You must decide this *before* training

Example table

user_id	country	n_orders	total_amount	is_high_value
u001	US	8	92.0	1
u002	GB	2	18.5	0
u003	CA	5	63.2	0

Classification vs regression (quick intuition)

- **Classification:** predict a category (e.g., 0/1, “spam/not spam”)
- **Regression:** predict a number (e.g., price, demand)

> **Note**
>
> In the sample data, `is_high_value` is **classification** (0 or 1).

Training vs inference

During **training**: - you have **features + target** - you can measure “how good” the model is (evaluation)

During **inference** (real use): - you have **features only** - the model produces **predictions**

> ⚠ **Warning**
>
> If your inference file contains the target column, you are probably leaking information.

Micro-exercise: label X, y, and IDs (5 minutes)

Look at the table below.

user_id	country	n_orders	total_amount	is_high_value
u010	US	6	71.0	0

1. Which column is the **target (y)**?
2. Which columns are **features (X)**?
3. Which column is an **ID passthrough**?

Checkpoint: you can answer in 3 short phrases: “y = …”, “X = …”, “ID = …”.

Solution (example)

- **y (target):** `is_high_value`
- **ID passthrough:** `user_id`
- **X (features):** `country`, `n_orders`, `total_amount`

(Some projects may include more ID columns, like `customer_id`, `order_id`, etc.)

Quick Check

Question: At inference time, do we have the target column?

Answer: No. Inference input should be **X only** (plus optional ID columns).

Session 1 recap

- Supervised ML learns **X** → **y** from labeled examples
- Training has **X + y**; inference has **X only**
- Your first job is to define the problem clearly (target + unit of analysis)

Asr break

20 minutes

When you return: be ready to explain your unit of analysis in 1 sentence.

Session 2

From feature table → dataset contract (X/y, IDs, leakage)

Session 2 objectives

- Define **unit of analysis** (what one row means)
- Separate **features vs target vs IDs** in a real dataset
- Understand **leakage** as “cheating” and avoid it
- Start a dataset contract in a model card

What is a feature table? (Week 2 recap)

A feature table is a dataset designed for modeling.

- **One row = one prediction**
- Columns are “things you know” (features), plus (sometimes) the target
- Stored under: `data/processed/`

> **Tip**
>
> Week 3 starts from the feature table. We do **not** start from raw data.

Unit of analysis (UoA)

The unit of analysis answers:

> “What does **one row** represent?”

Examples: - one row = one user - one row = one transaction - one row = one day per store

> **Note**
>
> If you change the unit of analysis, you changed the problem.

The 3 column types you must name

1. **Target (y):** what you want to predict
2. **Features (X):** what the model is allowed to use
3. **ID passthrough:** kept for joining predictions back (not used as signal)

Today you'll write these down. Later you'll enforce them in a schema file.

Why keep ID columns?

ID passthrough columns are useful because:

- you need to **join predictions** back to real entities
- you can debug mistakes (“which row was wrong?”)
- you can group results (later) for error analysis

> ⚠ **Warning**
>
> IDs are often unique. If you train on them directly, models can “memorize” instead of learning.

Leakage (cheating) — plain English

Leakage means your features contain information you would **not** have at prediction time.

- “future” information
- post-outcome information
- direct copies/encodings of the target

> **Tip**
>
> Leakage can make metrics look amazing… and then fail in real life.

Micro-exercise: spot the leakage risk (6 minutes)

Which feature is **leakage** if you're predicting "will the user be high value next month?"

a. `n_orders_last_30_days`

b. `total_spend_next_30_days`

c. `country`

Now answer: *why*?

Checkpoint: you can explain your choice in 1 sentence.

Solution (leakage exercise)

B is leakage because it uses information from the **future** (next 30 days).

- A and C could be valid features (they exist before the future happens)

The dataset contract you'll write today

In `reports/model_card.md`, write:

- **Unit of analysis** (one row = …)
- **Target** name + what it means
- **ID passthrough columns**
- **Features allowed at inference**
- **Forbidden columns** (target + obvious leakage fields)
- A first guess of your **primary metric** (we'll refine later)

Quick Check

Question: Why do we keep ID columns if we don’t train on them?

Answer: So we can join predictions back, debug errors, and report results per entity.

Session 2 recap

- A feature table is “one row = one prediction”
- You must name: **target, features, IDs**
- Leakage = using information you won’t have at prediction time

Maghrib break

20 minutes

When you return: open the Week 3 repo and run `ml-baseline --help`.

Session 3

Repo + CLI orientation (what you will run all week)

Session 3 objectives

- Understand the Week 3 repo layout (only the important folders)
- Run the CLI help and generate sample data
- Know where outputs should be written

Repo tour (what matters this week)

```
week3-ml-baseline-system/
  data/processed/         # your features table lives here
  src/ml_baseline/        # library + CLI code
  models/                 # saved runs + registry
  reports/                # model_card.md + eval_summary.md
  tests/                  # unit tests (visible)
```

The CLI is your interface

Why we use a CLI:

- one command = a repeatable action (train / predict)
- graders (and teammates) can run it exactly
- easier to debug than “mystery notebook state”

> **Tip**
>
> If `ml-baseline --help` works, your project is already more shippable.

Commands you'll use today (Mac vs Windows)

macOS/Linux

```
cd week3-ml-baseline-system
uv sync
uv run ml-baseline --help
uv run ml-baseline make-sample-data
```

Windows PowerShell

```
cd week3-ml-baseline-system
uv sync
uv run ml-baseline --help
uv run ml-baseline make-sample-data
```

If `pytest` is missing later: `uv sync --extra dev`

Micro-exercise: read CLI help (5 minutes)

1. Run: `uv run ml-baseline --help`
2. Find:
   - the command name that creates sample data
   - one option that belongs to `train`
3. Don’t understand an option? Ignore it for now.

Checkpoint: you can point to `make-sample-data` in the help output.

Solution: what to notice in help output

- Commands are listed (e.g., `make-sample-data`, `train`, `predict`)
- Help text tells you what each command does
- Options (`--target`, `--seed`, …) are the “knobs” you’ll use later

Checkpoint

Raise your hand when:

- `uv run ml-baseline --help` works
- `data/processed/features.*` exists after `make-sample-data`

Session 3 recap

- The repo is organized to ship: `data/` → `models/` → `reports/`
- The CLI is the “front door” to your system
- Today’s minimum is: help works + sample data + model card draft

Isha break

20 minutes

When you return: start Hands-on Task 1 (setup) immediately.

Hands-on

Generate sample data + draft your dataset contract + Git push

Hands-on success criteria (today)

Minimum ✅

- `uv run ml-baseline --help` works
- `uv run ml-baseline make-sample-data` writes `data/processed/features.*`
- `reports/model_card.md` exists and is filled (draft)
- 1+ commit pushed to GitHub

Optional ⭐

Project layout (top levels)

```
week3-ml-baseline-system/
  data/processed/
  models/
  outputs/
  reports/
  src/
  tests/
```

Task 1 — Setup the Week 3 repo (15 minutes)

- Open the repo folder: `week3-ml-baseline-system/`
- Run `uv sync`
- Verify the CLI runs: `uv run ml-baseline --help`

Checkpoint: the help text prints and exit code is 0.

Hint: common setup issues

> ⚠ **Warning**
>
> If a command fails, don’t panic—read the error carefully.

- “command not found: uv” → install `uv` (Week 1 skill)
- “No such file or directory” → you are in the wrong folder
- “python version” errors → use Python 3.11+

Solution (Task 1)

macOS/Linux

```
cd week3-ml-baseline-system
uv sync
uv run ml-baseline --help
```

Windows PowerShell

```
cd week3-ml-baseline-system
uv sync
uv run ml-baseline --help
```

Task 2 — Generate sample data (15 minutes)

- Run: `uv run ml-baseline make-sample-data`
- Confirm the file exists:
  - `data/processed/features.csv` (or `features.parquet`)

Checkpoint: you can open the file and see columns.

Solution (Task 2)

macOS/Linux

```
uv run ml-baseline make-sample-data
ls -la data/processed
head -n 5 data/processed/features.csv
```

Windows PowerShell

```
uv run ml-baseline make-sample-data
dir data/processed
Get-Content data/processed/features.csv -TotalCount 5
```

If your file is `features.parquet`, that's fine (it means Parquet is supported on your machine).

Task 3 — Inspect the columns (15 minutes)

- List the columns in `features.*`
- Write down:
  - **target**
  - **ID passthrough**
  - **features**

Checkpoint: you can say: “y = …, ID = …, X = …”.

Solution (Task 3)

For the sample feature table:

- **Target (y):** `is_high_value`
- **ID passthrough:** `user_id`
- **Features (X):** `country`, `n_orders`, `avg_amount`, `total_amount`

Task 4 — Draft your model card (25 minutes)

- Create: `reports/model_card.md`
- Fill in the dataset contract:
  - unit of analysis
  - target meaning
  - features + IDs
  - forbidden columns (at least the target)
- Keep it short and clear (you will improve it on Day 5)

Checkpoint: your model card has no blank placeholders.

Model card template (copy/paste)

```
# Model Card — Week 3 (Draft)

## 1) What is the prediction?
- **Target (y):** `__________`
- **Unit of analysis:** one row = __________
- **Decision supported:** __________

## 2) Data contract (inference)
- **ID passthrough columns:** __________
- **Required feature columns (X):** __________
- **Forbidden columns:** target + leakage fields

## 3) Evaluation plan (fill on Day 2-3)
- Split strategy: __________
- Primary metric: __________
```

Task 5 — Git checkpoint (10 minutes)

- `git status`
- Commit with a clear message (example below)
- Push to GitHub

Checkpoint: your new commit is visible online.

Solution (Task 5)

```
git status
git add reports/model_card.md data/processed/ || true
git commit -m "w3d1: sample data + model card draft"
git push
```

Tip

Small commits are a superpower. Don’t wait until the end of the week.

Vibe coding (safe version)

1. Write the plan in 5 bullets (no code yet)
2. Implement the smallest piece
3. Run → break → read error → fix
4. Commit
5. Repeat

> ⚠ **Warning**
>
> Do not ask GenAI to write your solution code. Ask it to explain concepts or errors.

Debug playbook

When stuck:

- Re-run with `--help` and confirm you're using the repo environment (`uv run ...`)
- Confirm you are in the repo root (you should see `pyproject.toml`)
- Read the error top-to-bottom (file + line + message)
- Make one small change, then re-run

Stretch goals (optional)

⭐ If you finish early:

- Install Parquet support: `uv sync --extra parquet`
- Generate `features.parquet` and open it
- Start thinking about your real dataset's target + unit of analysis

Exit Ticket

In 1–2 sentences each:

1. What is the difference between **features** and **target**?
2. Why is leakage dangerous?
3. What is your unit of analysis for the sample dataset?

What to do after class (Day 1 assignment)

Due: before Day 2 (Dec 29, 2025)

1. Ensure these commands work:
   - `uv run ml-baseline --help`
   - `uv run ml-baseline make-sample-data`
2. Commit + push your `reports/model_card.md` draft

Deliverable: GitHub repo link with your Day 1 commit(s).

> **Tip**
>
> Add 1–2 commits with clear messages. Don’t wait until the last minute.

Thank You!

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority