Lab 1 Reading Supplement
Background and Reference for Weekly Tasks

CSCI 4170/6170 — Spring 2026 | Data-Centric ML Pipeline (90–110 minutes/week)

How to use this supplement

This document is a companion to Lab 1. It explains key terms, why each task exists, and what "good enough" looks like for the required Week 1–Week 3 checkpoints. Use it as a quick reference while you work; you do not need to read every section in advance.

Quick navigation

- Week 1 (Parts A–B): datasheet, quick audit, leakage note, single train/test split, one baseline pipeline, one metric + one small artifact
- Week 2 (Parts C–D): SVM model + light tuning, experiment log, calibration (if needed), thresholding under a simple cost model
- Week 3 (Part E): NVIDIA DLI lab progress notebook + short reflection and troubleshooting notes
- Templates: datasheet, leakage-risk note, experiment log column definitions, threshold justification prompts
- Graduate-only extension options (choose one): bootstrap confidence intervals, interpretability, or thresholding under a fairness constraint

Core concepts used throughout Lab 1

1) Data-centric ML pipeline

A data-centric workflow focuses on improving model performance by improving the data and the end-to-end process, not by endlessly trying new model architectures. In this lab, you will document the dataset, check data quality, prevent leakage, and make an explicit decision policy (a threshold or rule).

2) Data leakage

Data leakage occurs when information that would not be available at prediction time is accidentally used during training or evaluation, causing over-optimistic results. Leakage can be obvious (post-outcome fields) or subtle (duplicates across splits, preprocessing fit on the full dataset, target leakage through engineered features).

Common leakage vectors to look for:

- Identifiers or near-identifiers (user_id, transaction_id, patient_id) that correlate with the target
- Timestamps or ordering artifacts (including “future” information) that leak the outcome
- Post-outcome fields (e.g., “claim_paid”, “final_grade”) that are created after the target is known
- Duplicates or near-duplicates that appear in both train and test
- Preprocessing fit on all data (imputation/scaling/encoding performed before the split)

Your Week 1 leakage-risk note should identify 1–2 concrete risks and state how you will prevent them (e.g., drop a column, time split, or deduplicate before splitting).

3) Leakage-safe splits

A split is leakage-safe when the test set represents “future unseen data,” and no information from the test set influences training. In this lab, you will use a single train/test split for simplicity.

Split guidance:

- If rows are time-ordered: use a time-based split (e.g., last 20% as test).
- If not time-ordered: use a random split with a fixed random_state.
- If there are repeated entities (multiple rows per person): a group-based split may be appropriate (optional unless clearly needed).

4) Pipelines and ColumnTransformer

A scikit-learn Pipeline ensures that preprocessing (imputation, scaling, encoding) is fit only on the training data and then applied consistently to validation/test data. Use ColumnTransformer only when you have mixed numeric and categorical features.

Typical preprocessing components:

- Numeric: SimpleImputer(strategy='median'); optionally StandardScaler (especially for SVM/kNN).
- Categorical: SimpleImputer(strategy='most_frequent') + OneHotEncoder(handle_unknown='ignore').
- Model: your chosen baseline or SVM model.

5) Baselines, metrics, and small artifacts

A baseline anchors your expectations. The goal is not to get a high score in Week 1—it is to establish a correct, leakage-safe workflow.

Recommended Week 1 baseline choices:

- Gaussian Naive Bayes (fast; can work well for some high-dimensional problems)

- k-Nearest Neighbors (simple; benefits from scaling)
- Ridge (simple, stable regression baseline)

Primary metrics (choose one and stay consistent across weeks):

- Classification: accuracy (or F1 if classes are imbalanced)
- Regression: MAE (or RMSE)

In Week 1 you will also include one small supporting artifact: a confusion matrix (classification) or a short residual summary (regression).

6) Experiment logging and reproducibility

An experiment log is a lightweight record of what you ran. It supports comparison across runs and makes your work reproducible.

Minimum reproducibility expectations:

- Fix random seeds and record them.
- Keep the train/test split fixed across baseline and SVM comparisons.
- Record model type, key hyperparameters, and metrics for each run.

Week 1 reference (Parts A–B)

Part A — Dataset datasheet + quick data-quality audit

Purpose: make the dataset and task explicit, and identify data quality issues that could mislead evaluation.

A “good enough” datasheet (short) includes:

- Motivation / intended use (1–2 sentences)
- Target definition (what you are predicting and what counts as positive/negative or the numeric target)
- Data source + license/terms (link + short note)
- Feature dictionary (5–10 key features, type, and meaning)
- Limitations/risks (2–3 bullets: missingness, selection bias, label noise, representativeness, etc.)

Quick audit checklist (aim for 10–15 minutes):

- Missingness summary (top columns by % missing)
- Duplicate row check (exact duplicates; note if you remove them)
- Target distribution (class counts or regression summary)
- One bias/ethics note relevant to your dataset (1 short paragraph)

Leakage-risk note (required)

Write a brief note (half-page is enough) identifying 1–2 leakage risks and your prevention plan. Be concrete: name the column(s) or mechanism.

Example structure:

- Risk: column(s)/pattern that could leak the outcome.
- Why it is leakage: how it would be unavailable at prediction time.
- Mitigation: drop/transform the field, change split strategy, or deduplicate.

Part B — Leakage-safe split + one baseline pipeline

Purpose: establish a correct end-to-end training and evaluation workflow.

Required steps:

- Create one train/test split (80/20). Use a time-based split if time-ordered.
- Build one Pipeline with minimal preprocessing fit only on the training set.
- Train one baseline model (GaussianNB, kNN, or Ridge/ElasticNet).
- Report one primary metric on the test set and one supporting artifact (confusion matrix OR residual summary).

Common pitfalls to avoid:

- Preprocessing before splitting (leakage).
- Using the test set for tuning decisions.
- Changing the split between baseline and SVM (comparisons become invalid).

Week 2 reference (Parts C–D)

Part C — Core model: SVM (with light tuning)

Purpose: train a stronger non-tree model and compare fairly against the Week 1 baseline.

SVM family in scikit-learn (choose one):

- LinearSVC or SGDClassifier: linear decision boundary; efficient for many features.
- SVC (kernel): can model non-linear boundaries; slower; requires careful scaling and tuning.

Scaling matters: SVMs and kNN are sensitive to feature scale. If you use StandardScaler, include it inside the Pipeline.

Light tuning guidance (2–3 settings total):

- Try 2–3 values of C (regularization strength), e.g., 0.1, 1, 10.
- If using RBF kernel: try 2 values of gamma (e.g., 'scale' and 0.01).

- Keep everything else fixed; do not run large grid searches in this lab.

Validation options:

- Small validation split from training data (simple).
- 3-fold cross-validation on the training set (more stable).

Experiment Log expectations

Log at least the Week 1 baseline and your SVM variants. Record hyperparameters, seed, metric(s), and timestamp. The log should let a reader identify the best run and reproduce it.

Part D — Decision thresholding under a simple cost model

Purpose: move from “a model score” to an explicit decision rule. In applied ML, the operating threshold is part of the system design.

Calibration (classification): when and why

- Some models output scores that are not calibrated probabilities (especially SVMs).
- If you need probabilities for a cost-based decision, use CalibratedClassifierCV with sigmoid (Platt scaling).
- If you skip calibration, state that you are using uncalibrated scores and justify the choice.

Choosing a threshold (two acceptable approaches):

- Cost matrix: define costs for FP/FN (and optionally TP/TN) and choose the threshold that minimizes expected cost on validation predictions.
- Constraint: choose a threshold to satisfy a requirement such as recall ≥ X or precision ≥ Y.

Required reporting: show results at the default threshold and at your chosen threshold, include a confusion matrix, and write a 3–5 sentence justification.

Week 3 reference (Part E: NVIDIA DLI lab progress)

Purpose: gain hands-on experience with an applied deep learning workflow and practice documenting technical progress and troubleshooting.

What counts as “meaningful progress” in 90–110 minutes:

- Complete roughly 50% or more of the exercises (or a comparable, clearly described portion).
- Capture key outputs (plots, metrics, example predictions) directly in the notebook.
- Record any errors and what you tried to resolve them (even if unresolved).
- Write a short reflection: what you learned and what you would do next with more time.

Suggested documentation template (paste into your notebook)

- Lab selected: [name]

- Exercises completed: [list or %]
- Key results: [2–4 bullets: outputs/metrics/screenshots]
- Issues encountered: [error messages + context]
- Troubleshooting attempted: [what you tried]
- Reflection (5–8 sentences): [what you learned; one next step]

Templates and quick-check tables

Short datasheet template (copy/paste)

Dataset name + link:

License/terms:

Prediction task + target definition:

Intended use / decision context:

Feature dictionary (5–10 key features):

Known limitations/risks (2–3 bullets):

Leakage-risk note template (copy/paste)

Leakage risk #1: [field/mechanism]

Why it could leak: [1–2 sentences]

Mitigation: [drop/split/dedup/etc.]

Leakage risk #2 (optional): ...

Experiment Log (CSV) recommended columns

Use any subset of these, but keep them consistent across runs:

Column	What to record
run_id	Short identifier (e.g., 001, svm_rbf_C1).
timestamp	When you ran it (ISO format is fine).
dataset	Dataset name/version (or file hash).
split_type	random / time-based (and any grouping notes).
seed	Random seed used.
pipeline_notes	Short description of preprocessing steps.

model	Baseline vs SVM variant.
hyperparams	Key hyperparameters (C, gamma, k, alpha, etc.).
metric_primary	Your primary metric on validation/test.
metric_secondary	Optional (e.g., precision/recall).
threshold	Default or chosen threshold (if classification).
notes	Anything notable (leakage fix, feature drop, etc.).

Cost matrix example (classification)

Define costs in a 2x2 table. The absolute numbers are less important than the relative trade-off you are asserting.

	Predicted Negative	Predicted Positive
Actual Negative	TN cost = 0	FP cost = 1
Actual Positive	FN cost = 5	TP cost = 0

Interpretation: here, a false negative is five times as costly as a false positive, so the chosen threshold should prioritize recall.

Graduate-only (6170) extension options (choose ONE)

Option 1: Bootstrap confidence intervals for metrics

Idea: repeatedly resample the test set with replacement, compute the metric each time, and summarize uncertainty with percentile intervals (e.g., 2.5th–97.5th percentiles for a 95% interval).

- Use at least 500 bootstrap resamples (more if fast).
- Report the metric point estimate and the interval.
- Explain what the interval means in one paragraph.

Option 2: Lightweight interpretability (permutation importance + partial dependence)

Permutation importance estimates how much the metric drops when a feature is randomly shuffled. Partial dependence shows the average effect of a feature on the prediction while marginalizing over others.

- Report top 5 features by permutation importance.
- Include 1–2 partial dependence plots for the most important features.
- Discuss at least one failure mode or limitation (e.g., correlated features).

Option 3: Thresholding under a fairness constraint

Define a protected attribute or relevant subgroup variable and choose a threshold subject to a simple constraint, such as equal opportunity (TPR gap <= δ). Then discuss the performance vs. fairness trade-off.

- Clearly define the subgroup variable and why it matters for your task.
- Compute subgroup metrics at the default threshold and at your chosen threshold.
- Explain the trade-off you accepted and why.

Optional references (quick links)

- Pipelines and composite estimators — https://scikitlearn.org/stable/modules/compose.html
- ColumnTransformer — https://scikitlearn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html
- Model evaluation (metrics) — https://scikitlearn.org/stable/modules/model_evaluation.html
- Support Vector Machines — https://scikit-learn.org/stable/modules/svm.html
- Probability calibration — https://scikit-learn.org/stable/modules/calibration.html