Reflective Journal – Module 3: ML Concepts & Workflow

Coming into this lab, the full machine learning workflow felt like a long checklist that I had to "get through." By the end, it made sense as a coherent system where each step sets up the next one to succeed. The turning point was seeing how small, early choices (like data exploration and feature selection) directly shaped downstream model performance and the quality of my interpretations.

What clicked about learning types

The difference between supervised and unsupervised learning finally clicked when I started thinking of supervised learning as "training with an answer key." In our wine dataset, the class label guided the model to connect chemical features to a known target. In contrast, unsupervised methods would have tried to invent structure without labels—which is powerful for discovery but not what we needed for classification. Reinforcement learning felt conceptually different: there is no fixed answer key, only feedback loops and rewards, which made me appreciate why it's used in games and robotics rather than static tabular problems like this one.

EDA changed how I think about modeling

I used to view EDA as a formality; now I see it as the risk-reduction phase of ML. The class distribution and "no missing values" check immediately lowered my uncertainty about data quality. More importantly, the correlation heatmap gave me an early signal about linear relationships among features. That preview explained why Logistic Regression ultimately beat a shallow Decision Tree. The heatmap also taught me a classic lesson: correlation is not causation. Some features moved together (e.g., alcohol with others), but that doesn't prove a cause–effect relationship—only that linear separation might be feasible.

From "fit a model" to "reason about generalization"

I've read about overfitting before, but the concept didn't truly land until I compared two models and examined the confusion matrix. The logistic model generalized better (≈88.9% accuracy) and produced fewer off-diagonal errors than the decision tree. The matrix exposed a consistent pain point: confusions between class_1 and class_2. That single visual taught me more than accuracy ever could. I started asking "*which* mistakes is the model making and *why*?"—often a sign that classes are less separable in the chosen feature space or that I need better preprocessing.

What was hard—and how I worked through it

The hardest part was interpreting metrics beyond accuracy. Precision, recall, and F1 felt abstract until I tied them to the confusion matrix boxes: rows as truth, columns as predictions, diagonal

as “wins,” and off-diagonal as “costly mistakes.” I also struggled with how much feature engineering was “enough.” Testing a simple trio (alcohol, color_intensity, proline) lowered accuracy versus using more features, which taught me that minimalism can backfire if I drop informative signals. My strategy going forward is: start simple, then justify complexity with validation scores and error analysis, not with guesswork.

The scaling “aha”

Another conceptual unlock was recognizing where feature scaling belongs. Logistic Regression assumes features are on comparable scales; trees generally don’t care. That suggests an immediate experiment: apply StandardScaler within a pipeline and re-evaluate. Even before running it, the reasoning makes sense: if some features dominate due to larger numeric ranges, the linear boundary can skew. This realization deepened my respect for the preprocessing → modeling interface.

How this changes my approach

I tend to favor building “cool” models first. This lab convinced me to win with basics: clean data, stratified splits, baseline models, and tight evaluation. I also plan to adopt a habit of model comparison early (at least 2–3 families: linear, tree/ensemble, margin-based like SVM). Just seeing two models side-by-side surfaced learning I would have missed with only one.

Transferring insights to real problems

Two domains immediately come to mind:

- Healthcare classification: predicting heart disease risk from clinical features. The wine exercise maps nicely: labels exist, data are tabular, and confusion costs are asymmetric (false negatives are worse). Here, I’d emphasize calibrated probabilities and threshold tuning, not just raw accuracy.
- Financial operations / fraud detection: many features, evolving patterns, and class imbalance. The workflow from this lab—EDA, baseline models, confusion analysis, then iteration—would transfer directly, with added attention to drift monitoring and human-in-the-loop review.

Questions I want to explore

1. How much does scaling + regularization improve linear models on tabular data? I want to quantify the gain, not just assume it.
2. Where’s the tipping point where tree ensembles (Random Forest, XGBoost) surpass linear models here? My hypothesis: when non-linear interactions matter more than global linear separability.

3. How do I make error costs explicit? Accuracy hides the real-world cost of certain mistakes; I'd like to incorporate cost-sensitive metrics and threshold tuning.
4. Deployment reality: What monitoring is essential to catch data drift and performance decay over time?

My new mental model of the workflow

Before: a linear checklist.
Now: a feedback loop. EDA informs feature choices; feature choices suggest preprocessing; preprocessing shapes which models make sense; evaluation points back to better features or different algorithms. The workflow is less about moving forward and more about tight, purposeful iteration.

Concrete next steps

- Re-run the logistic model with a StandardScaler + regularization pipeline.
- Add SVM and a Random Forest to the comparison, with basic hyperparameter tuning via GridSearchCV.
- Evaluate with macro-averaged F1 and study the class_1 vs class_2 confusions specifically.
- Document findings and keep plots alongside metrics to make decisions explainable.