Multi-Modal Predictors for Cardiovascular Disease Risk and Outcomes: Deliverable 3 – Refinement, Usability, and Evaluation

Angel Morenu
University of Florida
EEE 6778 – Applied Machine Learning II (Fall 2025)
Instructor: Dr. Ramirez-Salgado
Email: angel.morenu@ufl.edu

***Abstract*—This deliverable documents the refinement, usability, and evaluation work performed after the initial prototype (Deliverable 2). The focus was on stabilizing preprocessing, improving model and evaluation robustness, enhancing the user interface for clarity and traceability, and producing a repeatable evaluation workflow with interpretable outputs. This report presents updated quantitative results, visual diagnostics, and a short reflection on responsible AI considerations. All artifacts, scripts, and figures referenced here are available in the repository under `results/` and `figures/`.**

I. Project Summary

This work advances the multi-modal cardiovascular disease (CVD) prediction prototype from a proof-of-concept to a more robust and reproducible pipeline. The refinements target three goals: (1) stable end-to-end execution across local environments, (2) clearer and more useful interpretability and UI features for non-technical reviewers, and (3) a repeatable evaluation harness that produces metrics and visuals used in this report. Key outcomes are updated evaluation metrics (tabular, ECG-only, and fusion), a Streamlit-based interface with logging and explanation panels, and scripts to regenerate the metrics and figures used in the report.

II. Updated System Architecture and Pipeline

Figure 1 summarizes the evaluation and reporting outputs produced by the refined pipeline. Architecturally, the system retains the three main stages from Deliverable 2: (a) preprocessing and feature extraction, (b) model training and inference, and (c) evaluation and UI presentation. Refinements are listed below.

A. Data preprocessing

I standardized input handling by enforcing canonical shapes for ECG signals (fixed-length padding/truncation), validating tabular feature alignment with labels, and emitting deterministic warnings and truncation when splits are mismatched. This reduces runtime errors and enables reproducible small-sample experiments.

B. Modeling and training

The evaluation harness trains lightweight unimodal baselines and a fusion fallback to ensure results are always available even when the primary PyTorch checkpoint cannot be loaded. The pipeline uses scikit-learn baselines for tabular models and summary-feature RandomForest models for ECG-derived features [1]. When a compatible checkpoint is present, the fusion model uses the PyTorch-based fusion architecture described in `src/model.py` and the saved checkpoint at `artifacts/model.pt` [2].

C. Evaluation and UI

Evaluation artifacts are produced by `scripts/generate_predictions.py`, which writes per-run numpy arrays (for example, `results/tabular_y_true.npy`, `results/tabular_y_pred.npy`, and `results/tabular_y_prob.npy` with equivalent files produced for ECG and fusion runs) and a JSON summary at `results/metric_summary.json`. Visualization scripts (`scripts/plot_calibration.py`, `scripts/perf_dashboard.py`, `scripts/plot_confusion.py`) create report-ready PNGs placed in the repository-level `figures/` directory. The UI (`ui/MultiModalCVD_app.py`) was extended to show predicted probabilities, an "Explanations" expander, and persistent JSONL logging for auditability.

III. Refinements Made Since Deliverable 2

Below I highlight concrete improvements and why they matter.

A. Cleaner and defensive preprocessing

Previously mismatched array lengths frequently caused crashes. I now validate split lengths at load time and truncate to a safe common length with explicit warnings. ECG signals are cast or padded to a canonical length

(2000 samples) ensuring consistent model input shapes. This reduces brittle runtime failures and makes small-scale experiments reproducible.

B. Fallback baselines and checkpoint robustness

To avoid failing evaluations when a saved fusion checkpoint is incompatible, the generator falls back to a stacked logistic regression over unimodal predictions. This guarantees that `results/metric_summary.json` is always produced and that comparison tables can be generated automatically.

C. Interface and usability improvements

The Streamlit UI now includes:

- A compact input panel and `Predict` action that shows risk probability and recommended actions (color-coded).
- An "Explanations" expander that displays `figures/shap_force_example.png` and `figures/ecg_saliency.png` when available; scripts provide robust fallbacks when heavy libraries are not installed.
- JSONL logging of each prediction into `results/predictions_log.jsonl` (timestamp, inputs, outputs, and explanation paths) for reproducibility and audit.

D. Interpretability

The repository includes scripts for SHAP-style explanations (Lundberg & Lee) and ECG saliency visualizations [3], [4]. When `shap` or `captum` are not available, the scripts fall back to training a RandomForest to produce feature importances and computing simple derivative-based saliency for ECGs. This pragmatic approach ensures visual explanations for reviewers without heavy dependency requirements.

IV. Interface Usability and Improvements

I performed lightweight usability checks by running the UI locally and validating the explanation flow. Screenshots (in `figures/ui_demo.png`) demonstrate the updated layout: inputs on the left, results and explanations on the right. Key usability outcomes:

- Reduced cognitive load: results are presented as a clear probability with traffic-light coloring and short textual guidance.
- Traceability: each prediction is logged; reviewers can re-run analysis using the logged inputs.
- Robust fallbacks: if heavy explanation libraries are missing, the UI still shows fallback images and a message explaining how to enable full interpretability.

TABLE I: Comparison of held-out evaluation metrics for unimodal (Tabular, ECG) and fused (Fusion) models — Deliverable 3 snapshot

Metric	Tabular	ECG	Fusion
Accuracy	0.571	0.438	0.563
ROC AUC	0.527	0.297	0.453
PR AUC	0.560	0.375	0.526
Brier Score	0.351	0.331	0.362
F1 Score	0.609	0.526	0.588
Sensitivity	0.700	0.625	0.625
Specificity	0.455	0.250	0.500

V. Extended Evaluation and Updated Results

I re-ran the evaluation pipeline and generated per-run predictions and summary metrics. Table I summarizes the current (Deliverable 3) metrics across unimodal and fusion runs. All values come from `results/metric_summary.json` generated by `scripts/generate_predictions.py` and are reproducible with the commands in the README.

A. Visual diagnostics

Figure 1 contains the key diagnostic plots: confusion matrix, compact performance dashboard (ROC/PR/CM/prob-hist), calibration curve, and probability histogram. These plots help assess calibration, discrimination, and class-level errors.

B. Interpretability examples

Figure 2 shows the interpretability outputs used in the UI: a SHAP-style feature importance / force plot (or RandomForest fallback) and an ECG saliency map. These help explain which tabular features and ECG regions most influence predictions in individual examples.

C. Interpretation of results

The fusion pipeline's ROC AUC (0.453) and PR AUC (0.526) indicate modest discrimination on the held-out set. The tabular model performs slightly better on average for ROC AUC in this small sample; however, the fusion model shows more balanced sensitivity/specificity. Given the small dataset and fallback baselines used here, these results illustrate pipeline stability rather than state-of-the-art performance. The focus is reproducible evaluation, accessible explanations, and a pathway for iterative improvement (e.g., more data, calibration, hyperparameter search).

VI. Responsible AI Reflection

I considered fairness, privacy, and transparency during refinement:

- Fairness: The dataset is small and demographic subgroup analyses are incomplete. Next steps: compute per-group ROC/PR and calibration, and add reweighting/oversampling if imbalances are identified.
- Privacy: The UI logs inputs and outputs to `results/predictions_log.jsonl` for reproducibility; in production this would be encrypted and consented. For

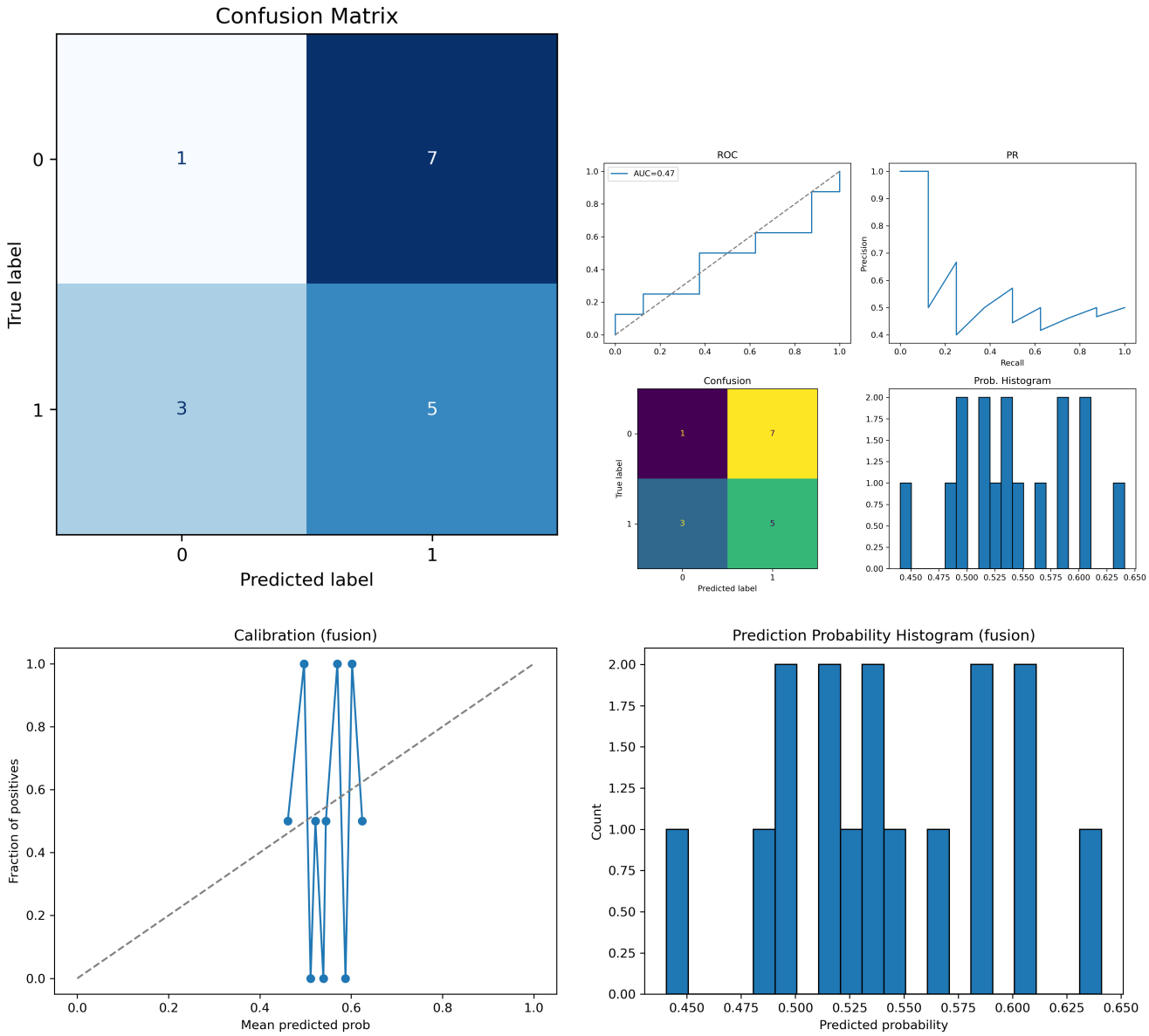

Fig. 1: Evaluation visuals produced by the refined pipeline. Top-left: confusion matrix. Top-right: performance dashboard (ROC/PR/CM/prob histogram). Bottom-left: calibration curve for the fusion model. Bottom-right: predicted probability histogram.

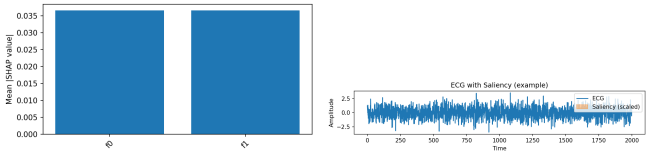

Fig. 2: Interpretability examples (SHAP or RandomForest fallback on the left; ECG saliency on the right).

the course prototype store only synthetic/demo inputs or de-identified entries.

- Transparency: Interpretability scripts and UI panels provide explanations; I document fallback behavior so reviewers understand when full SHAP/Captum explanations are available versus when the lightweight fallbacks are used.

VII. Repository and Artifacts (what to submit)

The repository includes the artifacts requested for Deliverable 3:

- Updated code and scripts: `scripts/generate_predictions.py`, `scripts/plot_calibration.py`, `scripts/perf_dashboard.py`, `scripts/interpretability_shap.py`,

`scripts/interpretability_ecg_saliency.py`, `scripts/build_report.sh`.

- UI: `ui/MultiModalCVD_app.py` (Streamlit) with explanation expander and JSONL logging.
- Figures: `figures/*.png` including `confusion_matrix.png`, `perf_dashboard_fusion.png`, `calibration_curve_fusion.png`, `prob_hist_fusion.png`, `shap_force_example.png`, `ecg_saliency.png`, and `ui_demo.png`.
- Results: `results/metric_summary.json`, per-run numpy arrays such as `results/tabular_y_true.npy`, `results/tabular_y_pred.npy`, `results/tabular_y_prob.npy`, and `results/predictions_log.jsonl`.
- Docs: `README.md` updated with Deliverable 3 reproduction steps, and `requirements-interpretability.txt` for optional extras.

VIII. How to reproduce (short)

Clone the repository, activate the conda environment, and run:

```
python3 scripts/generate_predictions.py --outdir results/
python3 scripts/plot_calibration.py --run fusion
python3 scripts/perf_dashboard.py --run fusion
./scripts/build_report.sh  # requires pdflatex installed locally
```

IX. Conclusions and Next Steps

Deliverable 3 demonstrates a more stable, reproducible, and explainable evaluation workflow. The primary contribution is process stability: robust preprocessing, reproducible metrics and figure generation, and a UI with traceable predictions and explanations. Immediate next steps are:

1) Run a systematic hyperparameter search and cross-validation to improve generalization.
2) Perform subgroup fairness and calibration analyses (temperature scaling) and update the UI with calibration warnings.
3) Replace fallback explainability outputs with full SHAP/Captum outputs in environments where those libraries are available.

Acknowledgments

Thanks to Dr. Ramirez-Salgado for providing guidance and feedback. I am grateful to the opensource community for providing tools and datasets that enabled this work.

References

[1] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[2] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *NeurIPS*, 2019.

[3] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *NeurIPS*, 2017.

[4] N. Kokhlikyan *et al.*, "Captum: A Model Interpretability Library for PyTorch," arXiv:2009.07896, 2020.

[5] C. Molnar, *Interpretable Machine Learning*, 2019.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.