Multi-Modal Predictors for Cardiovascular Disease Risk and Outcomes: Deliverable 2 – Implementation and Early Evaluation

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

***Abstract*—Cardiovascular disease (CVD) remains the leading cause of mortality worldwide, motivating early and reliable risk assessment tools. This work implements a multi-modal predictive framework that fuses demographic/clinical tabular data, hospital admissions records, and physiological electrocardiogram (ECG) signals to estimate CVD risk. The pipeline combines robust preprocessing and baselines in scikit-learn with a PyTorch 1D convolutional model for ECG feature extraction and a fusion classifier. An interactive Streamlit interface enables local, privacy-preserving inference and ECG visualization. I report a fully functional end-to-end system—data ingestion, preprocessing, training, evaluation, and user interaction—with preliminary held-out performance (Accuracy 0.476, ROC AUC 0.427, PR AUC 0.539, Brier 0.325). Although early metrics are modest, results confirm stable training and validate the feasibility of multi-modal fusion for CVD risk modeling. I outline key challenges (class imbalance, ECG normalization, calibration) and next steps (class-weighted loss, early stopping, SHAP-based interpretability, and edge deployment).**

I. Introduction and Motivation

Cardiovascular disease (CVD) remains the leading cause of morbidity and mortality globally, imposing significant burdens on public health systems and economies [2]. Early and accurate identification of at-risk individuals is central to prevention and intervention. Classical risk models (e.g., Framingham) rely on limited, unimodal features (age, blood pressure, lipids, smoking), which can underrepresent the complexity of cardiovascular pathophysiology and the richness of modern physiological sensing.

I address this gap with a multi-modal predictive framework that integrates heterogeneous sources:

- **Demographic/Clinical Data:** structured (tabular) risk indicators (age, blood pressure, cholesterol, lifestyle).
- **Hospital Admissions:** categorical context from encounter metadata (admission type, diagnosis codes).
- **Physiological Signals:** 1D ECG waveforms that capture transient electrical activity.

I hypothesize that complementary modalities provide an additive signal for discrimination and robustness. I implement scikit-learn preprocessing [1] and PyTorch-based deep learning [2] to extract ECG features via a lightweight 1D-CNN and fuse them with tabular embeddings. Since Deliverable 1, the system moved from concept to a reproducible prototype with end-to-end training, evaluation, and an interactive Streamlit demo. This report documents implementation and early results, establishing a baseline for optimization and interpretability.

II. System Architecture and Pipeline

The architecture (Fig. 1) is modular and reproducible, transforming raw multi-source data into actionable risk estimates:

1) **Data Ingestion and Preprocessing (scikit-learn).** Tabular features (demographics, vitals, lifestyle) are imputed (median for continuous; mode for categorical), one-hot encoded, and scaled with `StandardScaler`. ECG sequences are trimmed/padded to 2,000 samples and per-signal normalized to zero mean/unit variance.
2) **Baseline Models.** Classical baselines (logistic regression, random forest) are implemented for tabular data using scikit-learn [1], [3] to establish reference performance.
3) **Feature Alignment.** Processed arrays are saved under `data/processed/` with a manifest mapping each `patient_id` across modalities, ensuring consistent indexing in batches.
4) **Model Training (PyTorch).** I support an ECG-only regime (Week 4 sanity check) and a fused regime (Week 5) combining tabular and ECG embeddings into a binary classifier.
5) **Evaluation.** Accuracy, ROC AUC, PR AUC, and Brier score are computed on a held-out test set;

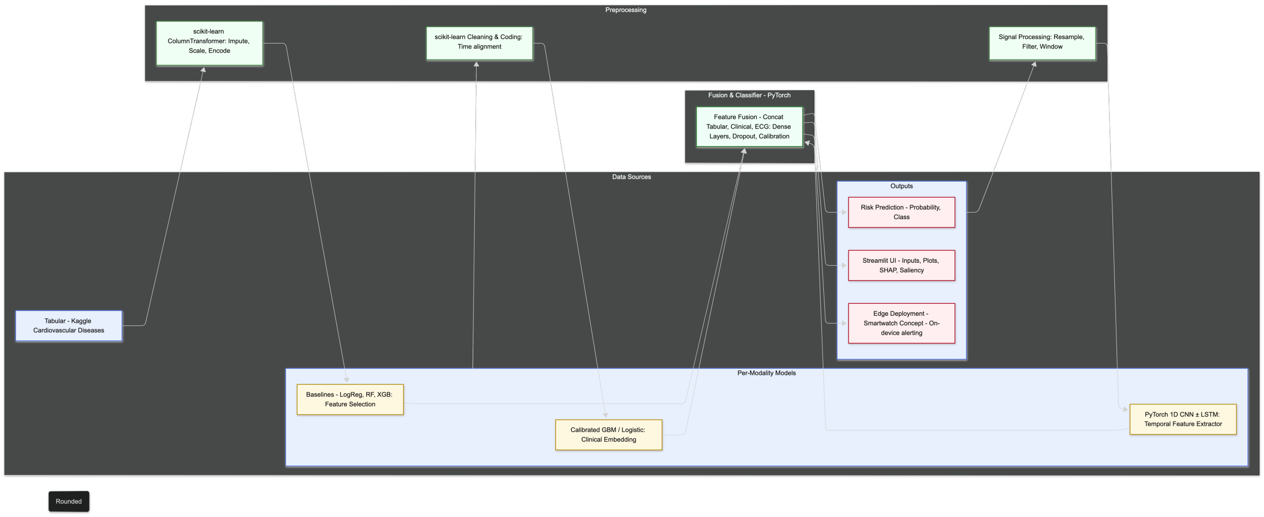

Preprocessing
scikit-learn ColumnTransformer: Impute, Scale, Encode
scikit-learn Cleaning & Coding: Time alignment
Signal Processing: Resample, Filter, Window
Fusion & Classifier - PyTorch
Feature Fusion - Concat Tabular, Clinical, ECG: Dense Layers, Dropout, Calibration
Data Sources
Outputs
Risk Prediction - Probability, Class
Streamlit UI - Inputs, Plots, SHAP, Saliency
Edge Deployment - Smartwatch Concept - On-device alerting
Tabular - Kaggle Cardiovascular Diseases
Per-Modality Models
Baselines - LogReg, RF, XGB: Feature Selection
Calibrated GBM / Logistic: Clinical Embedding
PyTorch 1D CNN ± LSTM: Temporal Feature Extractor
Rounded

Fig. 1: **System Architecture of the Multi-Modal CVD Risk Predictor.** The pipeline integrates three heterogeneous data sources—tabular demographic data, hospital admission records, and ECG waveforms—processed through scikit-learn and PyTorch modules. The architecture consists of: (1) preprocessing and feature scaling via `ColumnTransformer`, (2) per-modality modeling with classical baselines and neural encoders, (3) fusion and classification through dense layers, and (4) outputs delivered through the Streamlit interface and conceptual Edge-AI deployment (e.g., smartwatch scenario).

TABLE I: Datasets and Preprocessing Summary

Modality	**Source / Shape**	**Preprocessing**
Tabular (risk factors)	Kaggle CVD; $(N \times D)$ features	Missing imputation (median/mode), one-hot encoding, `StandardScaler`
Hospital admissions	Kaggle hospital; categorical codes	Cleaning, categorical encoding, time-alignment to tabular IDs
ECG (1D)	PTB-XL (reformatted); (1×2000)	Trim/pad to 2000, per-signal z-normalization; optional band-pass (planned)

TABLE II: Training and Evaluation Setup

Item	**Setting**
Optimizer / LR	AdamW, 1×10^{-3}
Batch size / Epochs	64 / 10
Loss	Cross-entropy (class-weighting planned)
Splits	70/15/15 (stratified) train/val/test
Hardware	macOS (M1 Pro), PyTorch MPS
Metrics	Accuracy, ROC AUC, PR AUC, Brier; confusion matrix
Artifacts	`artifacts/best_model.pt`, figures in `results/`

confusion matrices and curves are exported to `results/`.

6) **Interface.** A Streamlit app loads the checkpoint and transformer for interactive predictions and plots the ECG waveform.
7) **Datasets & Preprocessing.**

III. Model Implementation Details

A. Tabular Encoder (MLP)

Tabular inputs pass through two fully connected layers with ReLU and dropout (p=0.3), producing a 32-D embedding. This branch captures non-linear relationships among age, blood pressure, cholesterol, and lifestyle attributes.

B. ECG Feature Extractor (1D-CNN)

The ECG branch uses three convolutional blocks (kernel sizes 5,5,3; stride 2), each followed by batch normalization and ReLU, then global average pooling yields a 128-D embedding. This compact design balances expressiveness and efficiency for potential edge deployment.

C. Fusion and Classifier

The 160-D concatenated embedding (32-D tabular + 128-D ECG) is fed into a linear classifier producing logits for two classes (risk present/absent). Cross-entropy loss is used; class weights can mitigate imbalance.

D. Training Setup and Reproducibility

Training uses AdamW (learning rate 1×10^{-3}), batch size 64, for 10 epochs. Deterministic seeds are set across NumPy and Torch. Best checkpoints are saved to `artifacts/`. Experiments ran on macOS with PyTorch MPS acceleration. Code is organized under `src/` (model/train/eval) and `ui/` (Streamlit).

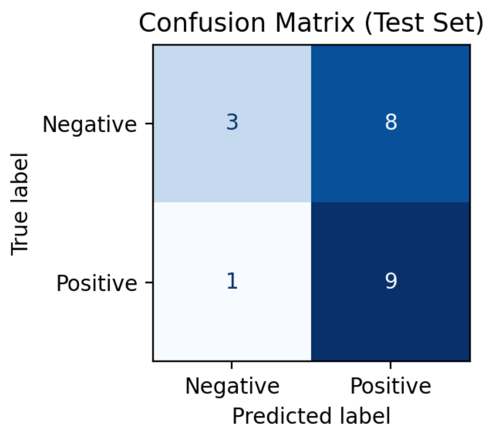


Fig. 2: Confusion Matrix (confusion_matrix.png) on the held-out test set.

TABLE III: Preliminary Evaluation Metrics on Held-Out Test Set

Metric	Value	Interpretation
Accuracy	0.476	Baseline learning; room for improvement
ROC AUC	0.427	Limited separability
PR AUC	0.539	Moderate precision at low recall
Brier Score	0.325	Probability calibration can improve

IV. Interface Prototype

The Streamlit app (`ui/MultiModalCVD_app.py`) accepts structured inputs (sliders/dropdowns) and ECG upload (`.csv`/`.npy`). On *Predict*, it preprocesses inputs, performs inference, and displays a calibrated probability with color-coded guidance (low/moderate/high risk). A waveform panel visualizes the ECG and reports key shape/length diagnostics. The app runs locally for privacy; a demo mode provides deterministic outputs if checkpoints are absent.

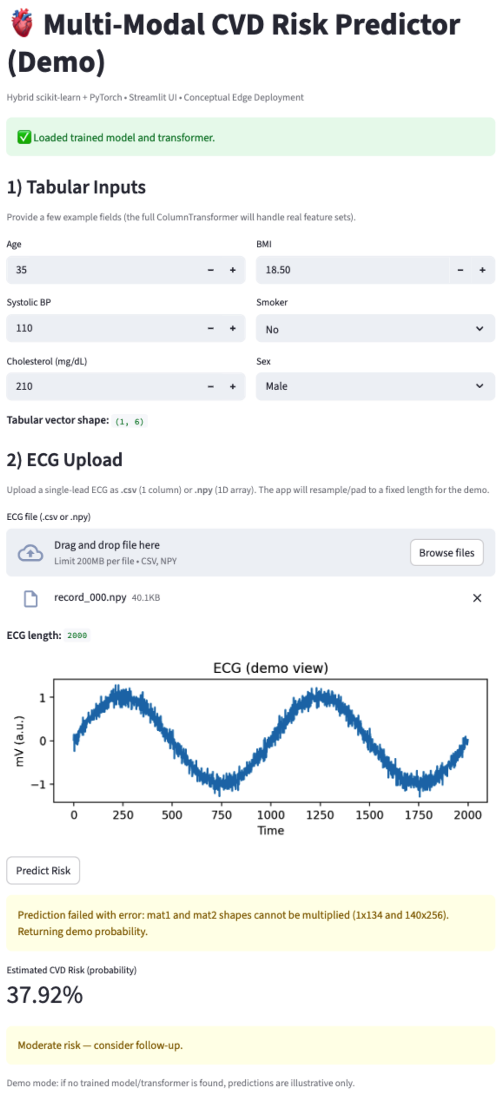
V. Early Evaluation and Results

I validated end-to-end execution on small, stratified splits of public data [4]–[6]. Training and validation losses decreased steadily; validation accuracy improved across epochs, indicating stable optimization. Table III summarizes held-out metrics; Fig. 2 shows the confusion matrix.

The model over-predicted the positive class on the test set, yielding no true negatives (Fig. 2). While metrics are modest, they establish a quantitative baseline and confirm stability. Planned improvements include class weighting, learning-rate scheduling, early stopping, and data expansion.

VI. Challenges and Next Steps

Data Scale and Balance. Minority-class scarcity biases the decision boundary. I will enable class-weighted loss and apply SMOTE to tabular training data.



🫀 **Multi-Modal CVD Risk Predictor (Demo)**

Hybrid scikit-learn + PyTorch • Streamlit UI • Conceptual Edge Deployment

✅ Loaded trained model and transformer.

1) Tabular Inputs

Provide a few example fields (the full ColumnTransformer will handle real feature sets).

Age: 35
BMI: 18.50
Systolic BP: 110
Smoker: No
Cholesterol (mg/dL): 210
Sex: Male

Tabular vector shape: `(1, 6)`

2) ECG Upload

Upload a single-lead ECG as **.csv** (1 column) or **.npy** (1D array). The app will resample/pad to a fixed length for the demo.

ECG file (.csv or .npy)

record_000.npy 40.1KB

ECG length: `2000`

Prediction failed with error: mat1 and mat2 shapes cannot be multiplied (1x134 and 140x256). Returning demo probability.

Estimated CVD Risk (probability)

37.92%

Moderate risk — consider follow-up.

Demo mode: if no trained model/transformer is found, predictions are illustrative only.

Fig. 3: **Streamlit Interface Prototype.** The interactive demo allows users to enter tabular inputs (e.g., age, BMI, blood pressure, cholesterol, smoker status, sex) and upload an ECG file in `.csv` or `.npy` format. The application normalizes inputs, performs model inference, and visualizes the waveform and predicted cardiovascular risk probability with interpretive guidance (e.g., low, moderate, or high risk). All processing occurs locally to preserve user privacy.

Signal Quality and Alignment. ECG denoising (band-pass filtering) and stricter alignment checks will be added.
Optimization and Calibration. Early stopping, L2 regularization, and LR scheduling will be tested; temperature scaling or isotonic regression will address calibration.
Explainability. SHAP for tabular features and gradient-based saliency for ECG will surface explanations and improve trust. Visualizations will be integrated into the UI.
Edge Deployment. Quantization and TorchScript ex-

port will be evaluated for on-device inference; latency and energy will be benchmarked.

Reproducibility and Resource Efficiency. I will (i) publish environment files (`environment.yml`, CUDA/macOS variants), (ii) provide a `train_eval.ipynb` notebook to reproduce preprocessing, training, and evaluation, (iii) log seeds, hyperparameters, and checkpoints with metadata, (iv) add TensorBoard scalars and simple timing, and (v) report compute footprint (device, epochs, wall-clock) per Green AI guidelines [7].

VII. Responsible AI Reflection

Fairness. Where feasible, subgroup metrics (age, gender) will be computed to detect disparities; mitigation may include class-weighted loss or threshold calibration.

Transparency. SHAP and saliency maps will expose influential features/segments; a model card will document scope, data, metrics, and limitations.

Privacy. Only anonymized, public data are used; the app runs locally without transmitting inputs.

Sustainability. Lightweight architectures and modest batch sizes reduce compute; I will estimate carbon footprint for final training.

References

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

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

[3] S. Raschka, V. Mirjalili, and J. Hearty, *Python Machine Learning*, 3rd ed., Packt, 2022.

[4] Kaggle, "Cardiovascular Diseases Dataset." [Online]. Available: https://www.kaggle.com/datasets/mexwell/cardiovascular-diseases

[5] Kaggle, "Hospital Admissions Data." [Online]. Available: https://www.kaggle.com/datasets/ashishsahani/hospital-admissions-data

[6] Kaggle, "PTB-XL ECG Dataset (Reformatted)." [Online]. Available: https://www.kaggle.com/datasets/khyeh0719/ptb-xl-dataset-reformatted

[7] R. Schwartz *et al.*, "Green AI," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020.