

Improving Logistic Regression on the Adult Income Dataset

Raul “JR” Saenz

Austin Community College

COSC 3380-003

Professor Katrompas

October 2025

Baseline Results

```
((.venv) ) [jrjaenz@jrjaenz assignment5]$ python income_model.py
=== Logistic Regression Results ===
Accuracy: 0.8456
Precision: 0.7362
Recall: 0.5874
F1 Score: 0.6534
ROC-AUC: 0.9017
```

```
=== Confusion Matrix ===
[[6331  472]
 [ 925 1317]]

=== Classification Report ===
```

	precision	recall	f1-score	support
0	0.87	0.93	0.90	6803
1	0.74	0.59	0.65	2242
accuracy			0.85	9045
macro avg	0.80	0.76	0.78	9045
weighted avg	0.84	0.85	0.84	9045

Experiment Log

ID	Change	Files	Accuracy	Precision	Recall	F1	ROC-AUC	Notes
E1	CLASSWEIGHT="balanced"	income_config.py	0.8086	0.5787	0.8381	0.6846	0.9017	Balanced weights penalize mistakes on minority class, reducing FN substantially. ROC-AUC~unchanged.
E2a	SCALER = "minmax01"	income_config.py	0.8432	0.7302	0.5830	0.6483	0.9006	Small declines across metrics; ROC-AUC ~ unchanged. MinMax not beneficial for this LR setup.
E2b	SCALER = "minmax01" CLASSWEIGHT='balanced'	income_config.py	0.8052	0.5732	0.8385	0.6809	0.9005	Compared to baseline you get much higher recall and lower precision.
E3	Added code to income_model.py for threshold tuning CLASSWEIGHT="balanced"	income_config.py income_model.py	0.8086	0.5787	0.8381	0.6846	0.9017	Added threshold to sweep best F1 at t=0.575; chose t based on precision-recall trade-off.
E4	Added code to income_model.py for threshold tuning CLASSWEIGHT=None	income_config.py income_model.py	0.8456	0.7362	0.5874	0.6534	0.9017	No change from baseline
E5a	COUNTRYTHRESHOLD=50	income_config.py	0.8452	0.7347	0.5879	0.6531	0.9019	No meaningful change

ID	Change	Files	Accuracy	Precision	Recall	F1	ROC-AUC	Notes
E5b	COUNTRYTHRESHOLD=150	income_config.py	0.8453	0.7356	0.5870	0.6529	0.9016	No meaningful change
E6	CORRELATIONTHRESHOLD = 0.95	income_config.py	0.8456	0.7362	0.5874	0.6534	0.9017	No meaningful change
E7	CORRELATIONTHRESHOLD = 0.99	income_config.py	0.8456	0.7362	0.5874	0.6534	0.9017	No meaningful change

Findings and Discussions

No meaningful changes to report, but with small changes I did improve the operating point (F1) somewhat.

E1: Changing CLASSWEIGHT='balanced' caused Recall to jump to approximately ~0.84, precision to drop to ~0.58, and F1 to rise to ~0.685.

E2a: Changing SCALER='minmax01' (weights=None) caused small declines across metrics; ROC-AUC stayed ~the same. MinMax didn't help LR here.

E2b: SCALER='minmax01' + CLASSWEIGHT='balanced' kept recall high (~0.84) but dropped precision (~0.57), so F1 dipped to ~0.681. Standard scaling works better for this setup.

E3: Added threshold tuning (swept t from 0.20–0.80). Best F1 ≈ 0.691 at $t \approx 0.575$ ($P \approx 0.62$, $R \approx 0.78$). I kept the table's headline metrics at the default $t=0.50$ for apples-to-apples comparison.

E4: Reverting to CLASSWEIGHT=None (with the tuning code present) returned metrics to baseline—no change, as expected.

E5a/b: Tweaking COUNTRYTHRESHOLD (tried 50 and 150) made no meaningful change. Grouping rare countries didn't move the model.

E6: CORRELATIONTHRESHOLD=0.95 removed nothing (no pairs that high), so metrics were unchanged.

E7: CORRELATIONTHRESHOLD=0.99 likewise removed nothing—again no change. (Note: lowering this too far starts dropping race dummies due to one-hot anti-correlation, which I don't think is desirable here.)

Code Choices for the Heart Data

I recreated the same four-file structure for Cleveland Heart Disease:

- **heart_config.py** – one place to control data, pre-processing, and model knobs:

TESTSIZE=0.2, RANDOMSTATE=42, SCALER="standard", CLASSWEIGHT=None,
MAXITER=1000, CORRELATIONTHRESHOLD=0.85, WINSORIZE=False.

Feature typing:

- **Categorical (one-hot):** cp, restecg, slope, thal (these are coded as numbers but are **categories**).
- **Binary categoricals (keep 0/1):** sex, fbs, exang.
- **Numeric:** age, trestbps, chol, thalach, oldpeak, ca (coerced from object to numeric because of “?” in source).
- **heart_datafunctions.py** – small, single-purpose helpers (e.g., safe one-hot with drop_first=True, standard/minmax scalers, correlation/heatmap utilities). No prints inside functions.
- **heart_dataprocess.py** – script-style pipeline:
 - Load heart_disease_cleveland.csv.
 - Replace “?” with NaN, **drop rows with missing data** (6 rows → 297 remain).
 - **Binarize target:** any non-zero becomes 1.
 - One-hot encode multi-class categoricals; keep binary columns as is.
 - **Scale numeric features** using StandardScaler (mean 0, var 1).

- Compute correlation matrix/heatmap; **no features dropped** at $|r| > 0.85$.
- Split to train_... / test_... CSVs (80/20 split).
- **heart_model.py** – mirrors the income model: read train/test, split X/y, fit LogisticRegression (liblinear, max_iter=1000, class_weight=None), evaluate (accuracy, precision, recall, F1, ROC–AUC), confusion matrix, classification report, and a simple coefficient table.

Differences vs. income code.

- Heart has **numeric-looking categoricals** (cp, restecg, slope, thal) that must be treated as categories (OHE). Income already had clear string categoricals.
- No “country grouping” logic was needed for heart (no long-tail text categories).
- Heart is **small (n≈300)** and fairly balanced after binarization; I avoided heavy feature pruning and left WINSORIZE=False.
- Otherwise, I kept the architecture the same for clarity and reproducibility.

Experiments on the Heart Data

Baseline (standard scaler, no class weights):

Accuracy **0.867**, Precision **0.885**, Recall **0.821**, F1 **0.852**, ROC–AUC **0.946**.

Confusion matrix: [[29, 3], [5, 23]] on the 60-row test set.

What I tried / considered.

- **Outliers:** I added a config flag (WINSORIZE) but left it **False**. With LR + standardization and a small dataset, capping “extremes” risks removing real signal.

- **Correlation pruning:** at **0.85** nothing was dropped (consistent with the small, curated UCI dataset).
- **Class weights / threshold tuning:** I kept class weights at **None**; the set isn't strongly imbalanced after binarization, and baseline metrics were already strong. I would reuse the income threshold sweep here if the use-case prioritized recall/precision differently, but I didn't change the decision threshold for this assignment to avoid over-fitting the small test split.

Did results improve over time? I focused on getting the pre-processing right and validating a clean baseline. Given the size of the data, the model is already performing well.

Would Graphing Help?

Yes. I generated and kept:

- A **correlation heat-map** for both datasets (audit for redundancy).
- The **confusion matrix** printouts (quick error shape).

Optional but useful in a report appendix: **ROC** and **Precision–Recall** curves. For this assignment, the textual metrics plus the heat-map work; curves are nice-to-have, not must-have.

What I Learned

- **Proving “no improvement” is still real work.** I documented parameter sweeps and showed why most changes didn't help, which builds trust in both the pipeline and the results.
- **Data work drives model quality.** Correctly identifying **numeric-looking categoricals** and one-hot encoding them was critical on the heart set.
- **Centralized configs** make iteration safe and fast. I changed one thing at a time and kept the code stable.

- **Metrics \neq one number.** ROC–AUC barely moved while precision/recall traded off; choosing an operating threshold can be more impactful than swapping scalars.
- **Correlation and OHE caveat.** One-hot columns can be (anti)correlated; blindly dropping by correlation can remove meaningful dummies.
- **Small data demands restraint.** With ~300 rows, aggressive pruning or winsorization can hurt; a clean, well-typed baseline with LR performed well.

Overall, my view shifted toward **experiment discipline and transparency**. I didn't just chase a bigger number; I built a reproducible process, logged decisions, and explained why certain knobs didn't change outcomes.