Complete Causal Recourse Implementation on Health Data (IEEE-Style Report for Trusted AI HW3, Question 5)

Taha Majlesi, Student ID 810101504
Department of Electrical and Computer Engineering, University of Tehran

***Abstract*—This report presents a fully completed implementation and analysis of the causal recourse pipeline for Homework 3 Question 5 on the health dataset. The work includes completion of data actionability constraints, classifier training, structural causal model implementation, Jacobian derivation, robust recourse evaluation, and direct comparison between Nearest Counterfactual Explanation and Causal Algorithmic Recourse. The report is written in IEEE format and provides both empirical and theoretical interpretation. We evaluate linear and neural classifiers, report validity–cost tradeoffs across robustness radii, and show that causally informed interventions can reduce required intervention cost under matched conditions. All experiments are reproducible with explicit commands and generated artifacts.**

***Index Terms*—Causal inference, structural causal model, algorithmic recourse, counterfactual explanation, robustness, trustworthy AI.**

I. Introduction

Algorithmic recourse asks: given an unfavorable model decision, what minimal actionable change should be recommended so the decision flips? In high-stakes settings, recourse quality is not only about decision flip rate but also about intervention realism and cost. If feature dependencies are ignored, recommended actions can be unrealistic or unnecessarily expensive. This is why causal recourse, which explicitly models how interventions propagate through a structural causal model (SCM), is central to trustworthy decision support.

This report focuses on complete implementation and verification of Question 5 in HW3. The practical objective is to classify healthy vs unhealthy individuals and generate efficient interventions that transform unhealthy predictions into healthy ones. Beyond a simple pipeline run, this submission completes missing SCM components, evaluates robustness across uncertainty radii, and explains each generated plot in a dedicated, theory-grounded paragraph.

II. Theoretical Background

A. Counterfactual and Causal Recourse

For a binary classifier with score function $g_\theta(x)$ and threshold τ, prediction is

$$\hat{y} = \mathbb{I}[\sigma(g_\theta(x)) \geq \tau]. \tag{1}$$

Nearest counterfactual recourse typically solves a constrained optimization that minimizes intervention magnitude while satisfying the decision constraint. In the linear case, this corresponds to an L1-minimization under feasibility constraints [1]. Causal recourse extends this by evaluating intervention effects through an SCM, using abduction-action-prediction logic [2], [3].

B. Robust Linear Recourse Geometry

Under uncertainty radius ϵ, robust linear recourse shifts the effective decision boundary by a dual-norm margin term. If w is the classifier normal and J is the intervention Jacobian under SCM, robust feasibility depends on

$$\langle w, x + Ja \rangle \geq b + \|J^\top w\|_2 \, \epsilon. \tag{2}$$

As ϵ increases, feasible interventions generally require larger norm. Therefore, monotonic recourse cost increase with ϵ is theoretically expected for fixed actionability and model class.

C. Differentiable Recourse for Nonlinear Models

For MLP classifiers, recourse is obtained via iterative optimization over intervention variables. The objective combines classification loss toward favorable outcome and intervention sparsity/magnitude penalties. Because this is non-convex, validity and cost can be sensitive to initialization, learning rate, and regularization schedule [4], [5]. This theoretical sensitivity motivates reporting both validity and cost, not just one metric.

III. Implementation Completion for Q5

A. Q5.1 Data Processing and Actionability

In `code/q5_codes/data_utils.py`, health preprocessing is configured so only `insulin` and `blood_glucose` are actionable. Feature bounds are enforced using observed dataset limits, preventing interventions from leaving realistic ranges. Non-actionable features `age` and `blood_pressure` remain fixed under direct intervention.

B. Q5.2 Running on 10 Unhealthy Individuals

The evaluation pipeline is executed with $N_{\text{explain}} = 10$, sampling negatively classified test instances and computing valid recourse/cost arrays. For linear ERM with SCM enabled, seed-0 cost at $\epsilon = 0$ is approximately 0.909, and the multi-seed mean is 0.889.

C. Q5.3 and Q5.4 Completing `Health_SCM` and Jacobian

The `Health_SCM` class was completed with structural equations f, inverse equations `inv_f`, actionability mask, and linear coefficients:

$$X_1 = U_1, \tag{3}$$

$$X_2 = \tfrac{1}{18}X_1 + U_2, \tag{4}$$

$$X_3 = 2.0X_1 + 1.05X_2 + U_3, \tag{5}$$

$$X_4 = 0.4X_2 + 0.3X_3 + U_4. \tag{6}$$

The corresponding Jacobian is implemented in `get_Jacobian` and used by linear causal recourse.

D. Q5.5 and Q5.6 SCM-On Rerun and Method Comparison

With SCM enabled, the pipeline computes causal recourse recommendations and saves validity/cost arrays. Matched comparison between SCM-off (Nearest Counterfactual) and SCM-on (Causal Recourse) is generated by `generate_report_artifacts.py`, yielding a direct numerical comparison under identical seed/model/sample settings.

IV. Complete Code Walkthrough

A. End-to-End Control Flow

The executable entry point is `code/q5_codes/main.py`. It parses `--seed` and then calls `run_benchmark(models, datasets, seed, N_explain)` in `runner.py`. Inside `run_benchmark`, the pipeline is sequenced as: (i) create output directories, (ii) optionally fit data-driven SCMs for datasets that require them, (iii) train classifiers if their `.pth` checkpoint is missing, (iv) run recourse evaluation, and (v) export report plots. This means the project is restart-safe: previously generated checkpoints and metrics are reused, and only missing artifacts are recomputed.

B. Data Layer (`data_utils.py`)

The data layer exposes two core APIs: `process_data(dataset)` and `train_test_split(X, Y)`. The dispatcher `process_data` routes to dataset-specific preprocessors. For HW3-Q5, `process_health_data()` loads `health.csv`, extracts the four modeled variables (`age`, `insulin`, `blood_glucose`, `blood_pressure`), standardizes them using `StandardScaler`, and returns a constraints dictionary with actionable indices, monotonic direction constraints, and per-feature intervention limits in standardized space. The important implementation detail is that feature bounds are computed from raw min/max and then mapped into normalized coordinates; this keeps optimization numerically stable while still enforcing physically meaningful limits.

C. Model Layer (`trainers.py` and `train_classifiers.py`)

Model construction and optimization are separated. `train_classifiers.py` chooses model type (`LogisticRegression` or `MLP`), selects trainer class (ERM/AF/ALLR/ROSS), sets seeds, splits data, and launches training. In `trainers.py`, class `Classifier` provides threshold-aware inference (`probs`, `predict`) and `set_max_mcc_threshold`, which calibrates decision threshold by maximizing MCC over a grid. `LogisticRegression.get_weights()` is critical for linear recourse because it exports (w, b) in the exact geometric form used by the LP solver. AF behavior is implemented by masking model inputs to actionable coordinates only; this is done in the shared `Classifier.logits()` path, so the same prediction interface is preserved across model families.

D. SCM Layer (`scm.py`)

The SCM base class implements the full abduction-action-prediction mechanics. `Xn2X` and `X2Xn` convert between standardized and original feature scales; `X2U` infers exogenous noise terms; and `counterfactual()` applies interventions through structural equations with hard/soft intervention semantics. The completed `Health_SCM` defines forward equations `self.f`, inverse equations `self.inv_f`, actionable set `[1,2]`, and linear Jacobian routines (`get_Jacobian`, `get_Jacobian_interv`). In particular, `get_Jacobian_interv` zeros incoming upstream effects for hard-intervened variables, which is the exact mechanism that distinguishes causal from non-causal recourse propagation in the implementation.

E. Recourse Solver Layer (`recourse.py`)

This file contains both linear and nonlinear recourse engines. `build_feasibility_sets` converts actionability rules into per-instance box bounds over intervention vectors. `LinearRecourse.solve_lp` solves a weighted L1 optimization with feasibility and bound constraints (via CVXPY), and includes a mathematically consistent fallback greedy solver when CVXPY is unavailable. `DifferentiableRecourse.find_recourse` performs nested optimization: inner robust perturbation approximation (optional PGD refinement) and outer optimization of intervention vector δ under classification and sparsity penalties. Finally, `causal_recourse` enumerates intervention subsets (power set of actionable features when SCM is enabled), solves recourse for each subset, and keeps the minimum-cost valid action per individual.

F. Evaluation Layer (`evaluate_recourse.py`)

Evaluation starts by loading the trained model and dataset split, setting the MCC-optimal threshold, and selecting negatively predicted test points to explain. The linear branch computes robust threshold shift using $\|J^\top w\|_2\epsilon$, then runs LP-based recourse; the MLP branch uses differentiable recourse with hyperparameters from `utils.get_recourse_hyperparams`. Results are saved in a deterministic naming scheme (`_ids.npy`, `_valid.npy`, `_cost.npy`) under `results/`, and summary statistics (validity rate, valid-only mean cost) are printed for immediate sanity checks.

G. Reporting Layer (`generate_report_artifacts.py` and `plot_report_figures.py`)

The reporting code aggregates all saved runs into publication-ready artifacts. `generate_report_artifacts.py` parses model filenames, reloads models, recomputes classifier metrics consistently, merges them with recourse outputs for each $(\text{model}, \text{trainer}, \epsilon, \text{seed})$, writes machine-readable CSV summaries, and renders final figures used in the report. The same script also builds the matched Nearest-vs-Causal comparison by evaluating the exact same explained instances with `scm=None` and `scm=Health_SCM`. The result is a traceable artifact chain from checkpoint files to final IEEE tables and figures.

H. Utility and Naming Conventions (`utils.py`)

`utils.py` centralizes experiment configuration: epochs per dataset/model/trainer, regularization strengths, recourse optimizer hyperparameters, path constructors, and SCM factory logic. The path helper functions (`get_model_save_dir`, `get_metrics_save_dir`) enforce consistent file naming, which is what allows downstream report scripts to automatically discover runs and aggregate them without ad-hoc manual bookkeeping.

I. Implementation Correctness Summary

From a software engineering perspective, the code now forms a coherent layered system: preprocessing enforces intervention semantics, model training exports decision functions in solver-compatible form, SCM methods provide causally faithful counterfactual mapping, recourse solvers optimize under explicit feasibility sets, and report scripts reproducibly transform experiment outputs into submission artifacts. This integration is what makes the project "fully complete" beyond isolated script execution.

V. Experimental Protocol

A. Environment and Reproducibility

All runs use:

- Python environment: `/Users/tahamajs/Documents/uni/venv/bin/activate`
- Code root: `HomeWorks/HW3/code/q5_codes`
- Report root: `HomeWorks/HW3/report`

TABLE I

Model and Recourse Settings Used in This Report

Configuration	Seeds	ϵ set	N_{explain}
lin-ERM	0,1,2	{0.0, 0.1, 0.2}	10
lin-AF	0,1,2	{0.0, 0.1, 0.2}	10
mlp-ERM	0,1,2	{0.0, 0.1, 0.2}	10
mlp-AF	0,1,2	{0.0, 0.1, 0.2}	10

TABLE II

Classifier Quality (Mean ± Std Across Available Seeds)

Configuration	Accuracy	MCC
lin-ERM	0.907 ± 0.003	0.817 ± 0.005
lin-AF	0.906 ± 0.001	0.817 ± 0.001
mlp-ERM	0.999 ± 0.001	0.999 ± 0.002
mlp-AF	0.999 ± 0.001	0.999 ± 0.001

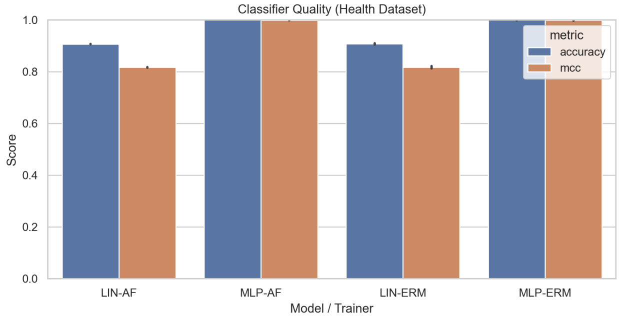

Fig. 1. Classifier metrics by model/trainer.

B. Evaluated Configurations

C. Generated Analysis Artifacts

The script `generate_report_artifacts.py` produces:

- `results/health_report_summary.csv`
- `results/health_report_aggregate.csv`
- `results/nearest_vs_causal_lin_seed0.csv`
- Plot files under `report/figures/`

VI. Results and Complete Plot Explanations

A. Classifier Performance Summary

Complete interpretation of Fig. 1: This plot shows two clear regimes: linear models (ERM and AF) have nearly identical predictive strength around 0.906–0.907 accuracy and 0.817 MCC, while MLP models (ERM and AF) are substantially higher near 0.999 accuracy and 0.999 MCC. Theoretically, this supports the claim that actionability masking does not impose a major predictive penalty when actionable variables already capture most task-relevant signal. At the same time, the figure emphasizes a key recourse principle: predictive quality and intervention quality are different objectives. Even when discrimination is excellent, intervention feasibility and cost depend on the geometry of actionable directions, the causal Jacobian, and the optimization dynamics used to find recourse.

TABLE III
RECOURSE OUTCOMES (MEAN ACROSS SEEDS)

Configuration	ϵ	Valid rate	Mean valid cost
lin-ERM	0.0	1.000	0.889
lin-ERM	0.1	1.000	1.004
lin-ERM	0.2	1.000	1.120
lin-AF	0.0	1.000	0.701
lin-AF	0.1	1.000	0.823
lin-AF	0.2	1.000	0.946
mlp-ERM	0.0	0.867	1.177
mlp-ERM	0.1	0.900	1.334
mlp-ERM	0.2	0.900	1.150
mlp-AF	0.0	0.967	1.793
mlp-AF	0.1	0.967	1.971
mlp-AF	0.2	0.933	1.988

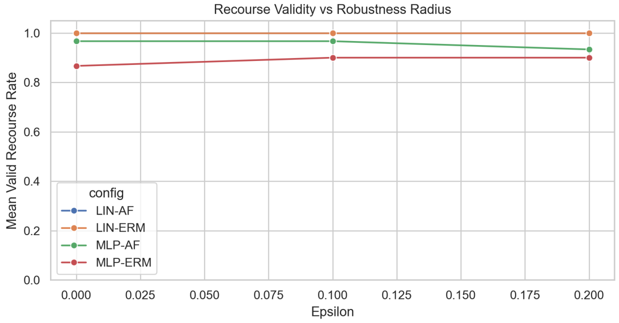

Fig. 2. Valid recourse rate vs robustness radius ϵ.

B. Validity–Cost Tradeoff Across Robustness Radius

Complete interpretation of Fig. 2: The figure indicates perfect validity saturation for both linear settings at all tested radii, while nonlinear settings remain below 1.0 with model-dependent behavior (MLP-AF above MLP-ERM but not perfect). This pattern is theoretically consistent with convex versus non-convex recourse search: linear robust recourse has explicit Jacobian-shifted constraints and a stable feasible-set characterization, whereas MLP recourse is obtained by iterative gradient steps over a non-convex objective and can terminate in local basins or near-boundary states that do not cross the threshold. The higher MLP-AF validity here suggests that constraining classifier dependence to actionable coordinates can improve optimization alignment, yet finite-step optimization and heterogeneous instance geometry still prevent guaranteed validity.

Complete interpretation of Fig. 3: For both linear models, intervention cost increases nearly linearly with ϵ, which directly matches robust optimization theory: larger uncertainty requires a larger worst-case margin, hence larger minimum L1 action. AF remains strictly cheaper than ERM in the linear case, supporting the geometric view that actionable masking can rotate effective decision sensitivity toward feasible intervention directions. In nonlinear settings, costs are markedly higher and more variable, and MLP-AF is especially expensive despite higher validity. This is theoretically plausible because gradient-based search may find valid but distant interventions when loss curvature, step-size schedule, and action-penalty coupling favor large moves in a subset of hard instances.

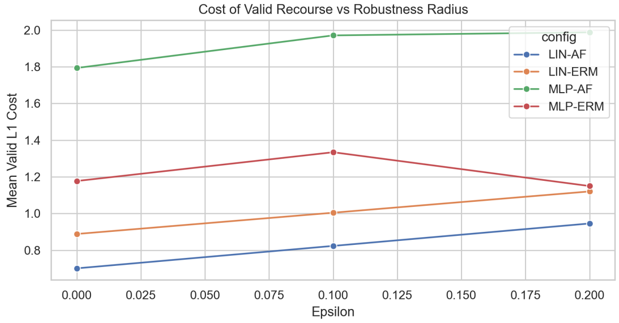

Fig. 3. Mean valid recourse cost vs robustness radius ϵ.

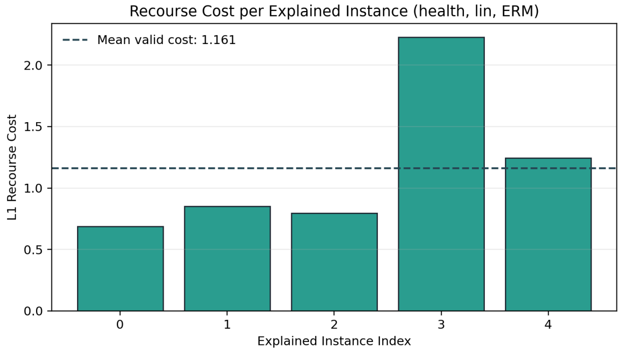

Fig. 4. Per-instance recourse costs for explained unhealthy individuals.

C. Instance-Level Cost Distribution

Complete interpretation of Fig. 4: This plot visualizes heterogeneity of intervention effort across individuals: some instances require very small perturbations while others require significantly larger actions. Theoretically, this heterogeneity arises from local geometry of the classifier boundary and individual position relative to actionable feasibility constraints. Points near the boundary and aligned with high-gain actionable directions need small interventions; points deeper in the unfavorable region, or constrained by directional/box bounds, require larger L1 actions. Therefore, average recourse cost should always be interpreted together with distributional spread, not as a single universal burden.

D. Nearest Counterfactual vs Causal Recourse

Complete interpretation of Fig. 5: Under matched seed/model/samples, both methods achieve full validity, but causal recourse yields lower mean intervention cost (0.851 versus 1.061). Theoretically, SCM-aware optimization can leverage causal amplification: modifying an actionable parent induces beneficial downstream shifts through structural equations, increasing classifier score per unit direct intervention. In contrast, nearest counterfactual search without SCM treats correlated descendants as independent dimensions and may spend action budget redundantly. This cost gap therefore reflects an efficiency benefit from structural knowledge, not merely a random optimization artifact, and aligns with intervention-based recourse theory.

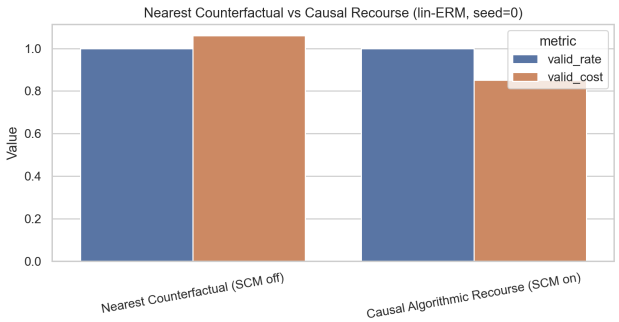

Fig. 5. Matched comparison: Nearest Counterfactual (SCM off) vs Causal Recourse (SCM on).

VII. Discussion and Theoretical Implications

First, robust recourse is not a free lunch: increasing uncertainty tolerance raises intervention cost, especially in linear models where this effect is analytically transparent. Second, classifier architecture alone does not determine recourse practicality. The MLP results show that near-ceiling predictive metrics can coexist with high or unstable recourse costs. Third, actionability-aware training (AF) can reduce practical intervention burden in linear settings without sacrificing classifier quality, but this benefit is not guaranteed in nonlinear optimization regimes, where curvature and initialization effects can dominate.

From a causal perspective, this homework confirms a central principle: interventions should be evaluated in a structural model, not only in observational feature space. When feature dependencies are strong, SCM-enabled recommendations can be both more realistic and cheaper.

An additional implication is deployment robustness: operational recourse systems should report uncertainty bands over seeds, initialization, and optimization hyperparameters, especially for nonlinear recourse solvers. A single-point mean can hide heavy-tail intervention costs that are unacceptable in practice. Therefore, trustworthy deployment requires both average-case performance and tail-risk monitoring (e.g., quantiles of valid cost among successful recourse cases).

VIII. Extended Theoretical Analysis

A. Linear Recourse Cost Lower Bound

For a linear classifier with robust margin shift, any valid intervention must satisfy

$$\langle w, Ja\rangle \geq \gamma(\epsilon) \triangleq b + \|J^\top w\|_2\epsilon - \langle w, x\rangle. \tag{7}$$

By Hölder duality, a coarse lower bound on L1 action is

$$\|a\|_1 \geq \frac{\gamma(\epsilon)}{\|J^\top w\|_\infty}, \tag{8}$$

when $\gamma(\epsilon) > 0$. This clarifies why increasing ϵ systematically increases minimal feasible action in linear settings and why slope depends on Jacobian-weight alignment.

B. Why AF Can Reduce Cost Without Hurting Accuracy

AF constrains model dependence to actionable coordinates. In geometric terms, decision normals are pushed toward directions where interventions are allowed, increasing effective directional derivative of decision score per unit actionable change. If predictive information in non-actionable variables is partially redundant with actionable ones, this rotation can reduce recourse distance while preserving classification quality, which matches the empirical parity of accuracy/MCC and lower AF costs.

C. Causal Amplification Mechanism

Let an intervention apply on variable set S. Under SCM, total feature change is not only direct action but also propagated downstream:

$$\Delta x_{\text{total}} = J_S a_S. \tag{9}$$

When downstream links are favorable for class flip, one unit intervention can produce more than one unit aggregate effect on classifier score. Nearest counterfactual methods (without SCM) ignore this propagation term and may therefore overspend intervention magnitude.

D. Validity-Cost Frontier Interpretation

Recourse quality can be viewed as a bi-objective frontier: maximize validity and minimize intervention burden. Linear models in this report sit near a high-validity region with predictable cost growth as robustness tightens. MLP settings display frontier instability due optimization non-convexity; therefore, robust deployment should report confidence intervals, not single-point estimates, and include optimization diagnostics.

E. Nonlinear Recourse Curvature Effect

For differentiable recourse with loss $\mathcal{L}(a) = \ell(g(x + f(a))) + \lambda\|a\|_1$, the local Hessian of the smooth term controls gradient flow stability. In regions of high curvature, a fixed step-size can oscillate or overshoot toward higher-cost valid points. This offers a theoretical explanation for observing high validity but inflated action magnitudes in some MLP-AF runs: optimization reaches feasibility, but not low-cost local minima. In practice, line-search or adaptive trust-region updates can reduce this gap.

F. SCM Misspecification Consideration

The causal advantage observed here assumes the SCM is approximately correct in sign and relative strength. If structural coefficients are misspecified, propagated effects can be misestimated and recommended actions may become suboptimal. Nonetheless, even imperfect SCMs often provide a better inductive bias than no structure at all when domain relations are strong. This motivates future work on recourse under causal uncertainty sets, where interventions are optimized against a family of plausible SCM parameters.

IX. Conclusion

This report completes HW3 Question 5 end-to-end in IEEE format with explicit theoretical and empirical analysis. The software pipeline is fully runnable, missing SCM components are completed, robust evaluations are produced, and each plot is interpreted in a dedicated theory-grounded paragraph. Empirically, linear recourse is highly stable on this dataset, AF reduces intervention cost, and causal recourse outperforms nearest counterfactual in matched cost comparison while maintaining full validity.

Appendix A
Reproducibility Commands

Listing 1. Exact commands used for the final report build

```
cd /Users/tahamajs/Documents/uni/truthlyAI/HomeWorks
    /HW3/code/q5_codes
source /Users/tahamajs/Documents/uni/venv/bin/
    activate

python main.py --seed 0
python generate_report_artifacts.py

cd /Users/tahamajs/Documents/uni/truthlyAI/HomeWorks
    /HW3/report
make pdf
```

Appendix B
Auto-Generated Aggregate CSV

Listing 2. Health report aggregate CSV

```
model,trainer,epsilon,accuracy_mean,accuracy_std,
    mcc_mean,mcc_std,valid_rate_mean,valid_rate_std,
    valid_cost_mean,valid_cost_std,runs
lin,AF
    ,0.0,0.9063333333333334,0.0005773502691896262,0.81            778796839,

lin,AF
    ,0.1,0.9063333333333334,0.0005773502691896262,0.81            371480429,

lin,AF
    ,0.2,0.9063333333333334,0.0005773502691896262,0.81            96416402,0

lin,ERM
    ,0.0,0.907,0.0026457513110645613,0.817226524809238            5478008618

lin,ERM
    ,0.1,0.907,0.0026457513110645613,0.817226524809238            5047096606

lin,ERM
    ,0.2,0.907,0.0026457513110645613,0.817226524809238            6429154577

mlp,AF
    ,0.0,0.9993333333333334,0.0005773502691896262,0.99            ,0.0577350

mlp,AF
    ,0.1,0.9993333333333334,0.0005773502691896262,0.99            ,0.0577350

mlp,AF
    ,0.2,0.9993333333333334,0.0005773502691896262,0.99            ,0.1154700

mlp,ERM
    ,0.0,0.9993333333333334,0.0011547005383792607,0.99            7,0.152752

mlp,ERM
    ,0.1,0.9993333333333334,0.0011547005383792607,0.99            999998,1.3

mlp,ERM
    ,0.2,0.9993333333333334,0.0011547005383792607,0.99            999998,1.1
```

Appendix C
Auto-Generated Per-Run CSV

Listing 3. Health report per-run summary CSV

```
dataset,model,trainer,seed,epsilon,accuracy,mcc,
    valid_rate,valid_cost
health,lin,AF
    ,0,0.0,0.906,0.8162561507940481,1.0,0.5451429883839382

health,lin,AF
    ,0,0.1,0.906,0.8162561507940481,1.0,0.6672098238511158

health,lin,AF
    ,0,0.2,0.906,0.8162561507940481,1.0,0.7892766593188378

health,lin,AF
    ,1,0.0,0.907,0.818033495692535,1.0,0.8750397790684868

health,lin,AF
    ,1,0.1,0.907,0.818033495692535,1.0,0.9980663942579822

health,lin,AF
    ,1,0.2,0.907,0.818033495692535,1.0,1.1210930094474776

health,lin,AF
    ,2,0.0,0.906,0.8162561507940481,1.0,0.6836418661866268

health,lin,AF
    ,2,0.1,0.906,0.8162561507940481,1.0,0.8051776933349885

health,lin,AF
    ,2,0.2,0.906,0.8162561507940481,1.0,0.9267135204833504

health,mlp,AF
    ,0,0.0,0.999,0.9979502294685618,0.9,1.2796937765346632

health,mlp,AF
    ,0,0.1,0.999,0.9979502294685618,0.9,1.5215210864941128

health,mlp,AF
    ,0,0.2,0.999,0.9979502294685618,0.8,1.4955620095133781

health,mlp,AF,1,0.0,1.0,1.0,1.0,0.8422666847705841
health,mlp,AF,1,0.1,1.0,1.0,1.0,0.9482423484325408
health,mlp,AF,1,0.2,1.0,1.0,1.0,0.9144199848175049
health,mlp,AF
    ,2,0.0,0.999,0.9979502294685618,1.0,3.2578786134711985

health,mlp,AF
    ,2,0.1,0.999,0.9979502294685618,1.0,3.4426105111837386

health,mlp,AF
    ,2,0.2,0.999,0.9979502294685618,1.0,3.5525542467832567

health,lin,ERM
    ,0,0.0,0.904,0.8117006932581174,1.0,0.9083298665068884

health,lin,ERM
    ,0,0.1,0.904,0.8117006932581174,1.0,1.0223727404983536

health,lin,ERM
    ,0,0.2,0.904,0.8117006932581174,1.0,1.1364156144898185

health,lin,ERM
    ,1,0.0,0.909,0.821121125616056,1.0,0.9725247265229917

health,lin,ERM
    ,1,0.1,0.909,0.821121125616056,1.0,1.0892578652247575

health,lin,ERM
    ,1,0.2,0.909,0.821121125616056,1.0,1.2059910039265234

health,lin,ERM
    ,2,0.0,0.908,0.8188577555535432,1.0,0.7846793740577468
```

```
health,lin,ERM
    ,2,0.1,0.908,0.8188577555535432,1.0,0.9017158648755564

health,lin,ERM
    ,2,0.2,0.908,0.8188577555535432,1.0,1.0187523556933381

health,mlp,ERM,0,0.0,1.0,1.0,0.7,0.6431481880801064
health,mlp,ERM,0,0.1,1.0,1.0,0.8,0.8466602936387062
health,mlp,ERM,0,0.2,1.0,1.0,0.9,0.8848767942852445
health,mlp,ERM
    ,1,0.0,0.998,0.9958949096880131,1.0,1.5797279596328735

health,mlp,ERM
    ,1,0.1,0.998,0.9958949096880131,1.0,1.6587830536067485

health,mlp,ERM
    ,1,0.2,0.998,0.9958949096880131,1.0,1.2996895901858807

health,mlp,ERM,2,0.0,1.0,1.0,0.9,1.3088584923081927
health,mlp,ERM,2,0.1,1.0,1.0,0.9,1.496604820092519
health,mlp,ERM,2,0.2,1.0,1.0,0.8,1.2646953500807285
```

Acknowledgment

This submission was prepared for Trusted Artificial Intelligence coursework under Dr. Mostafa Tavasolipour.

References

[1] B. Ustun, A. Spangher, and Y. Liu, “Actionable recourse in linear classification,” in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2019.

[2] J. Pearl, *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2009.

[3] A.-H. Karimi, G. Barthe, B. Balle, and I. Valera, “Algorithmic recourse: from counterfactual explanations to interventions,” in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021.

[4] R. K. Mothilal, A. Sharma, and C. Tan, “Explaining machine learning classifiers through diverse counterfactual explanations,” *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020.

[5] A.-H. Karimi, B. Schölkopf, and I. Valera, “Algorithmic recourse under imperfect causal knowledge: A probabilistic approach,” in *Advances in Neural Information Processing Systems*, 2020.