

Supplementary Material S2

Reproducibility Details: Preprocessing, Hyperparameter Grids, and SHAP Configuration

Manuscript: Explainable ML for Rehabilitation Outcome Prediction | sci-4282552

1. Global Reproducibility Settings

All stochastic components used a global random seed of 42, including train/test splitting, K-Means clustering initialization, model training, cross-validation shuffling, and bootstrap resampling.

Component	Setting	Value
Global random seed	All stochastic components	42
Train/test split	random_state	42
K-Means clustering	random_state, n_init	42, 20
Cross-validation	random_state (StratifiedKFold)	42
Bootstrap resampling	np.random.seed	42
XGBoost	random_state	42
LightGBM	random_state	42
Gradient Boosting	random_state	42
Random Forest	random_state	42
MLP	random_state	42
Logistic Regression	random_state	42

2. Preprocessing Pipeline

The preprocessing pipeline was applied in the following sequence:

Step	Operation	Details
1	Data loading	Cleaned_Dataset.csv (n=3,419, 82 features)
2	Leakage removal	All discharge BI columns removed (COD_26, COD_26_1 to COD_26_11)
3	Target engineering	normalized_gain = (BI_discharge - BI_admission) / (100 - BI_admission), clipped [0,1]
4	Binary outcome	outcome_binary = 1 if normalized_gain >= median (0.743), else 0
5	Stratified split	80/20 train/test, stratified by outcome_binary, random_state=42
6	Feature scaling	StandardScaler applied to Age only (other features already encoded/binary)
7	Cluster scaling	Separate StandardScaler fitted on full training set for K-Means only
8	K-Means clustering	K=2, n_init=20, random_state=42, fitted on training set only

Step	Operation	Details
9	Cluster assignment	Training: fit_predict; Test: predict (no re-fitting)
10	Missing values	fillna(0) applied to all numeric features
11	Final feature set	71 features + 1 cluster membership binary variable = 72 total

3. Hyperparameter Search Grids

3.1 Search Strategy

Hyperparameter tuning was applied to XGBoost (All Patients) and Gradient Boosting (Neurological subgroup) using RandomizedSearchCV with `n_iter=40` and 10-fold stratified cross-validation (`scoring='roc_auc'`, `random_state=42`). All other models used default scikit-learn/XGBoost/LightGBM parameters as listed in Table S3.3.

3.2 Search Grids (Tuned Models)

XGBoost — All Patients

Hyperparameter	Search Space	Best Value
<code>n_estimators</code>	[200, 300, 500]	200
<code>max_depth</code>	[4, 5, 6, 7]	4
<code>learning_rate</code>	[0.01, 0.05, 0.1]	0.05
<code>subsample</code>	[0.7, 0.8, 0.9]	0.8
<code>colsample_bytree</code>	[0.7, 0.8, 0.9]	0.8
<code>min_child_weight</code>	[1, 3, 5]	5
<code>gamma</code>	[0, 0.1, 0.2]	0.2
<code>eval_metric</code>	Fixed	logloss
Best CV AUC	—	0.8302

Gradient Boosting — Neurological Subgroup

Hyperparameter	Search Space	Best Value
<code>n_estimators</code>	[200, 300, 500]	200
<code>max_depth</code>	[3, 4, 5, 6]	4
<code>learning_rate</code>	[0.01, 0.05, 0.1]	0.05
<code>subsample</code>	[0.7, 0.8, 0.9]	0.8
<code>min_samples_leaf</code>	[3, 5, 10]	10
<code>max_features</code>	['sqrt', 'log2', 0.7]	log2
Best CV AUC	—	0.8251

3.3 Default Parameters (Non-Tuned Models)

Model	Key Parameters
Logistic Regression	C=0.1, max_iter=1000, solver='lbfgs', penalty='l2'
Random Forest	n_estimators=200, max_depth=12, min_samples_leaf=5, n_jobs=-1
LightGBM	n_estimators=300, max_depth=6, learning_rate=0.05, subsample=0.8
MLP	hidden_layer_sizes=(128, 64, 32), activation='relu', max_iter=500, learning_rate_init=0.001, early_stopping=True
Voting Ensemble	estimators: XGBoost+LightGBM+GradientBoosting+RandomForest, voting='soft'
Stacking Ensemble	estimators: XGBoost+LightGBM+GradientBoosting+MLP, final_estimator=LogisticRegression(C=1.0), cv=5

4. SHAP Configuration

Setting	Value / Description
SHAP version	0.44.0
Explainer type	shap.TreeExplainer (model-specific, exact Shapley values for tree ensembles)
Primary model	XGBoost (best overall AUC) — used for all four levels of analysis
Secondary model	Gradient Boosting — used for cross-model SHAP agreement analysis
Input data	Held-out test set only (n=684) — training data not used for SHAP computation
SHAP output	Per-sample SHAP values matrix (684 × 72); aggregated via mean SHAP
Global explanation	Beeswarm plot + bar chart of mean absolute SHAP values
Local explanation	Waterfall plots for representative individual patients
Group analysis	SHAP computed separately for Orthopedic (n=369) and Neurological (n=315) subsets
Cluster analysis	SHAP computed separately for Profile 0 (n=404) and Profile 1 (n=280) subsets
Cross-model agreement	Spearman rank correlation between XGBoost and Gradient Boosting SHAP rankings

5. Software and Package Versions

Package	Version	Use
Python	3.12	Runtime environment
scikit-learn	1.8.0	ML models, preprocessing, CV
XGBoost	2.0.3	Primary classification model
LightGBM	4.1.0	Classification model
SHAP	0.44.0	Explainability analysis

Package	Version	Use
NumPy	1.26.0	Numerical computation
pandas	2.1.3	Data manipulation
matplotlib	3.8.0	Visualization
seaborn	0.13.2	Statistical visualization
SciPy	1.11.0	Statistical tests

Note: All analyses were conducted on Kaggle Notebooks (GPU P100 environment).