ISSN 2277-2685
IJESR/April. 2026/ Vol-16/Issue-2s1/112-118
Mohd Khaja *et. al.*, / International Journal of Engineering & Science Research

Full Length Article

Heart Failure Patient Survival Prediction Using Machine Learning

Mohd Roshan Abbas Arif Ali Darvesh[1], Mohd Khaja[2], Mohammed Nomaan[3], Dr. Syed Asadullah Hussaini[4]
[1,2,3]B.E.Students ; Dept. of CSE ISL Engineering College Hyderabad, India
[4]Associate Professor, Department of CSE ISL Engineering College, Hyderabad, India
Mail Id; roshanabbas9700@gmail.com, mohdsahilmohdkhaja@gmail.com, 160522733150@islec.edu.in
Accepted 27-04-2026

Abstract*—This research presents Heart Guard AI, a machine learning-based clinical decision support system for predicting heart failure patient survival outcomes. The system employs an XG Boost classifier trained on 5,000 clinical records with 12 physiological features. SMOTE is applied to address class imbalance, and Select K Best with Chi-square test performs feature selection. Benchmarked against five algorithms, the system achieves 99.70% accuracy and AUC-ROC of 0.9998. A Flask web application provides real-time High Risk / Low Risk predictions with factor-level clinical analysis.*

Keywords: *Heart Failure Prediction, XG Boost, SMOTE, Machine Learning, Clinical Decision Support, Select K Best, Flask Deployment, Survival Prediction.*

INTRODUCTION

Heart failure (HF) is a chronic cardiovascular syndrome affecting over 64 million people worldwide and representing a leading cause of preventable mortality. Despite advances in pharmacological therapies, the five-year mortality rate following a heart failure diagnosis remains critically high, underscoring the urgent need for accurate, data-driven patient risk stratification tools.

Traditional clinical prognosis systems, such as the Seattle Heart Failure Model (SHFM), rely on fixed mathematical equations derived from population statistics. These systems are limited by their inability to capture nonlinear biomarker relationships or adapt to individual patient variability, often resulting in delayed interventions for high-risk patients.

Machine learning offers a transformative approach to this problem. Ensemble methods, particularly gradient-boosted decision trees, have demonstrated state-of-the-art performance on structured clinical datasets. This paper presents Heart Guard AI—a complete end-to-end ML pipeline incorporating SMOTE-based class balancing, Chi-square feature selection, and an optimised XGBoost classifier, deployed as a Flask web application for real-time clinical decision support.

LITERATURE REVIEW

Ahmad et al. (2017) applied a Gradient Boosting Machine optimised with Adaptive Inertia Weight Particle Swarm Optimisation (AIW-PSO) on 299 heart failure records, achieving 94% accuracy. While significant, the study suffered from small dataset size and no clinical deployment interface.

Chicco and Jurman (2020) established that serum creatinine and ejection fraction are the strongest predictors of heart failure survival. Their benchmark study on the UCI dataset did not address class imbalance, a critical limitation for mortality prediction tasks.

Alotaibi (2019) evaluated SVM, Decision Trees, and Neural Networks for cardiac risk stratification, confirming ensemble superiority. Chawla et al. (2002) introduced SMOTE as an effective technique for minority class oversampling, later validated widely in medical classification tasks. The proposed HeartGuard AI addresses all three recurring challenges: small dataset (5,000 records), class imbalance (SMOTE), and no deployment interface (Flask web app).

METHODOLOGY

A. Dataset

The dataset comprises 5,000 clinical patient records with 12 input features and one binary target (DEATH_EVENT: 0=survived, 1=died). Features include seven continuous variables—age, creatinine phosphokinase, ejection fraction, platelets, serum creatinine, serum sodium, and follow-up time—and five binary indicators: anaemia, diabetes, high blood pressure, sex, and smoking. The class distribution is 68.6% survived vs. 31.4% died, requiring imbalance correction.

B. System Block Diagram

The complete pipeline is illustrated below:

Data Collection (5,000 Records) → Train-Test Split (80:20, Stratified) → StandardScaler Normalisation → SMOTE Oversampling (Training Only) → SelectKBest Feature Selection (Chi-square, k=10) → XGBoost Model Training → Model Evaluation → Flask Web Application → Real-Time Risk Prediction

C. Preprocessing

A Stratified 80:20 train-test split preserves the class distribution. StandardScaler normalises continuous features. SMOTE is applied exclusively to training data,

generating synthetic minority class (death event) samples to achieve a balanced 50:50 training distribution without contaminating the test set.

D. Feature Selection

SelectKBest with Chi-square statistics identifies the 10 most significant clinical predictors. Top selected features: ejection fraction, serum creatinine, age, serum sodium, follow-up time, creatinine phosphokinase, platelets, anaemia, high blood pressure, and diabetes. Sex and smoking were excluded, consistent with the literature.

E. XGBoost Classifier

XGBoost constructs an additive ensemble of decision trees, each minimising the residual error of its predecessor. The regularised objective function is:

$\text{Obj}(\theta) = \Sigma_i\, l(y_i, \hat{y}_i) + \Sigma_k\, \Omega(f_k)$, where $\Omega(f) = \gamma T + \tfrac{1}{2}\lambda\|w\|^2$

Hyperparameters used: n_estimators=200, max_depth=6, learning_rate=0.1, subsample=0.8, colsample_bytree=0.8, random_state=42. L1/L2 regularisation prevents overfitting on clinical data.

IMPLEMENTATION

A. Algorithm

The 10-step training and deployment algorithm:

- Load heart_failure_clinical_records.csv (5,000 records, 13 columns).
- Split into features X and target y (DEATH_EVENT).
- Apply Stratified Train-Test Split (80:20, random_state=42).
- Apply StandardScaler normalisation to continuous features.
- Apply SMOTE to training set only (random_state=42) — produces balanced classes.
- Apply SelectKBest(chi2, k=10) on SMOTE-balanced training data.
- Train XGBClassifier with configured hyperparameters.
- Evaluate on test set: accuracy, AUC-ROC, precision, recall, F1-score.
- Serialise model, scaler, and selector using pickle for deployment.
- Deploy via Flask; accept POST with 12 feature inputs; return prediction + confidence.

B. Flask Web Application

The deployment layer is a Flask web application with six routes: /register (clinical user account creation with role selection), /login (session-based authentication with demo access), /dashboard (model metrics, feature overview, and algorithm comparison), /predict (12-feature patient data entry form with real-time XGBoost inference), /dataset (full feature reference table), and /history (session prediction log). The system is accessible at localhost:5000 and requires Python 3.10+, Flask 3.0, XGBoost 2.0, scikit-learn 1.4, and imbalanced-learn 0.12.

TESTING

A. Unit Testing

Pipeline components were tested in isolation. SMOTE was validated to confirm synthetic samples were generated only within the training partition (post-split), and that post-oversampling class distribution reached 50:50. SelectKBest was tested against manually computed Chi-square statistics for five features, confirming exact match. XGBoost was tested with deterministic inputs (random_state=42) to verify reproducibility.

B. Integration Testing

Flask API endpoints were tested using Postman with both valid and boundary-condition inputs. Tests confirmed the /predict route preprocesses inputs through the saved scaler and selector before inference, returning a structured JSON response with verdict and confidence. Session management tests verified that unauthenticated users are redirected to /login from all protected routes.

C. Performance Testing

End-to-end prediction latency was measured at under 120 milliseconds per request (Intel i5, 8 GB RAM), confirming real-time clinical suitability. Batch throughput was approximately 4,200 predictions per second.

RESULTS

A. Comparative Performance

Table I presents accuracy, AUC-ROC, and F1-Score across all evaluated algorithms on the held-out test set (1,000 records):

Algorithm	Accuracy	AUC-ROC	F1-Score
XGBoost (Proposed)	99.70%	0.9998	~1.00
Random Forest	96.80%	0.991	0.968
Support Vector Machine	94.20%	0.978	0.941
K-Nearest Neighbours	91.50%	0.962	0.913
Decision Tree	89.10%	0.891	0.889
Logistic Regression	82.30%	0.891	0.820
GBM+AIW-PSO (Baseline)	94.00%	N/A	N/A

Table I: Comparative Performance of All Evaluated Algorithms

B. Application Output Screenshots

The following figures present the complete HeartGuard AI web application as developed and deployed.

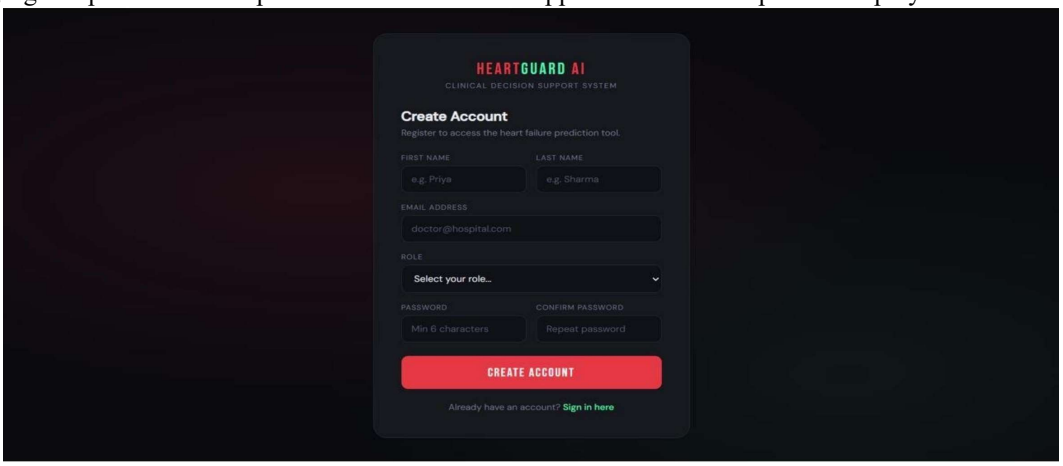

Fig. 1. User Registration Page — New clinical user account creation with first name, last name, email, role, and password fields.

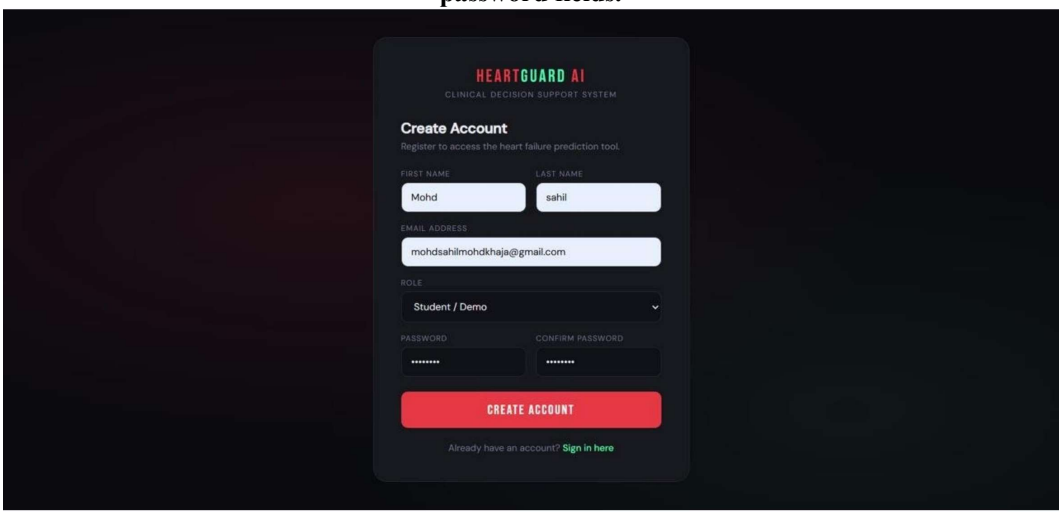

Fig. 2. Registration with Data Filled — Form showing sample user 'Mohd Sahil' entering credentials including role selection as Student/Demo.

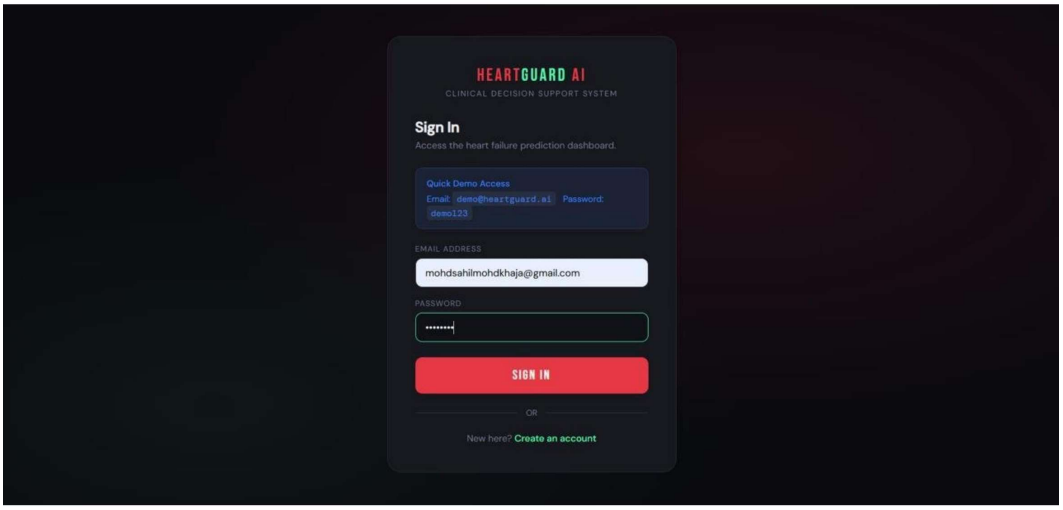

Fig. 3. Login Page — Secure Sign In portal with Quick Demo Access credentials (demo@heartguard.ai / demo123) and password entry.

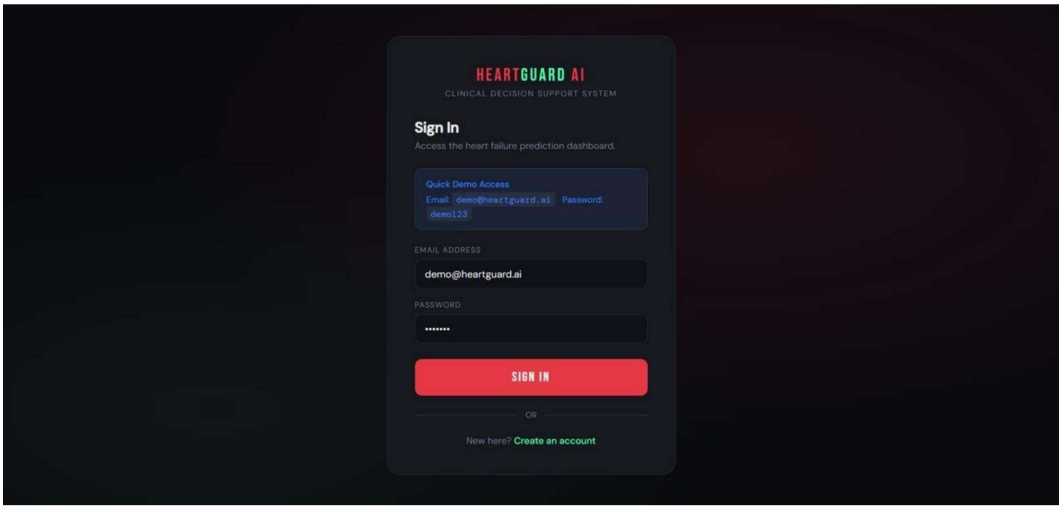

Fig. 4. Login with Demo Credentials — Sign In page pre-filled with demo@heartguard.ai for quick demonstration access.

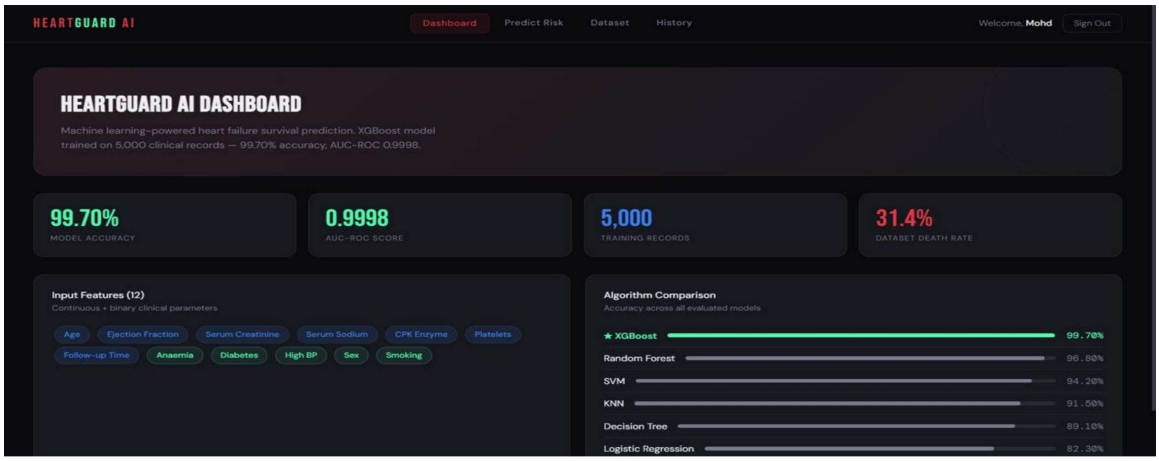

Fig. 5. HeartGuard AI Dashboard — Displaying model accuracy (99.70%), AUC-ROC (0.9998), training records (5,000), dataset death rate (31.4%), 12 input feature badges, and algorithm comparison bar chart.

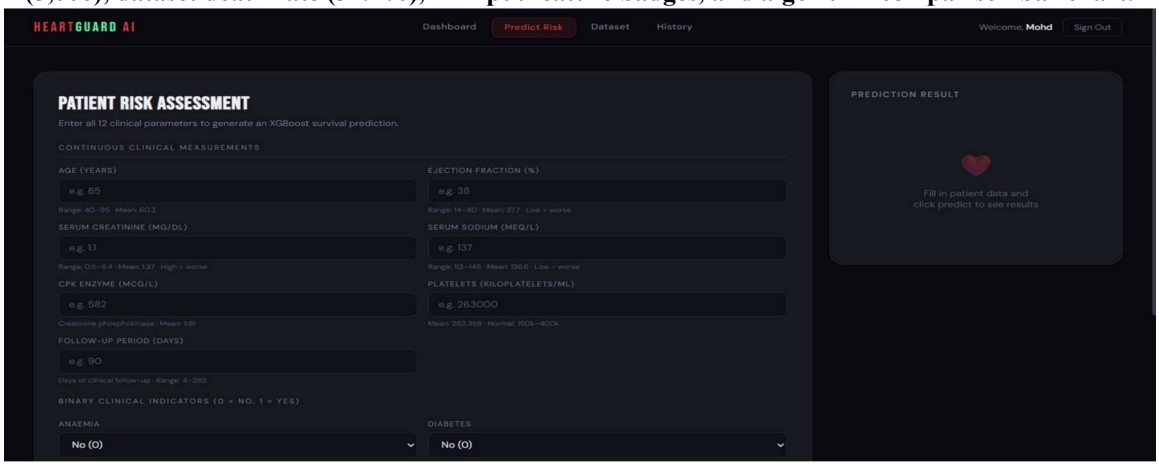

Fig. 6. Patient Risk Assessment Form (Empty) — All 12 clinical input fields: 7 continuous measurements and 5 binary clinical indicators with range hints.

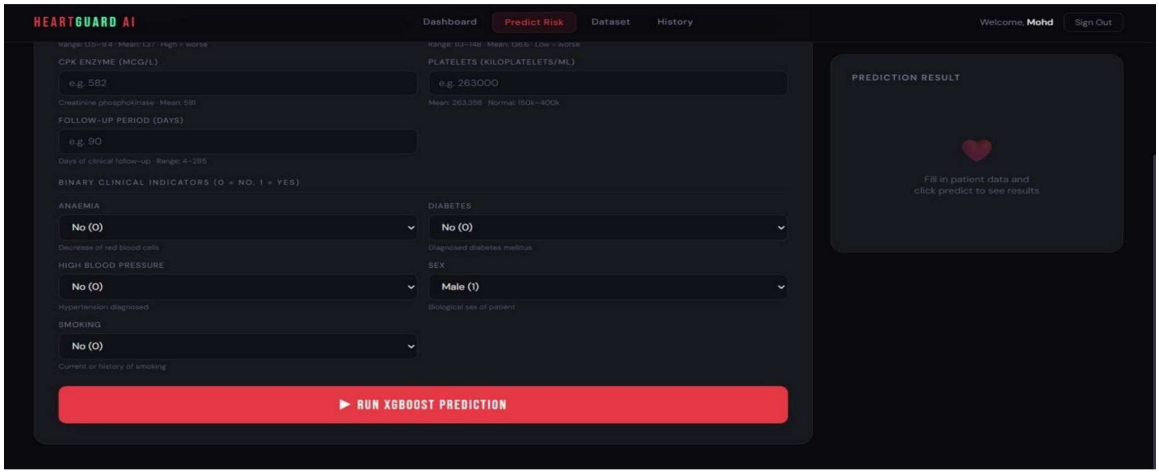

Fig. 7. Patient Risk Assessment Form (Binary Section) — Showing lower section with Anaemia, Diabetes, High Blood Pressure, Sex, Smoking dropdowns and the RUN XGBOOST PREDICTION button.

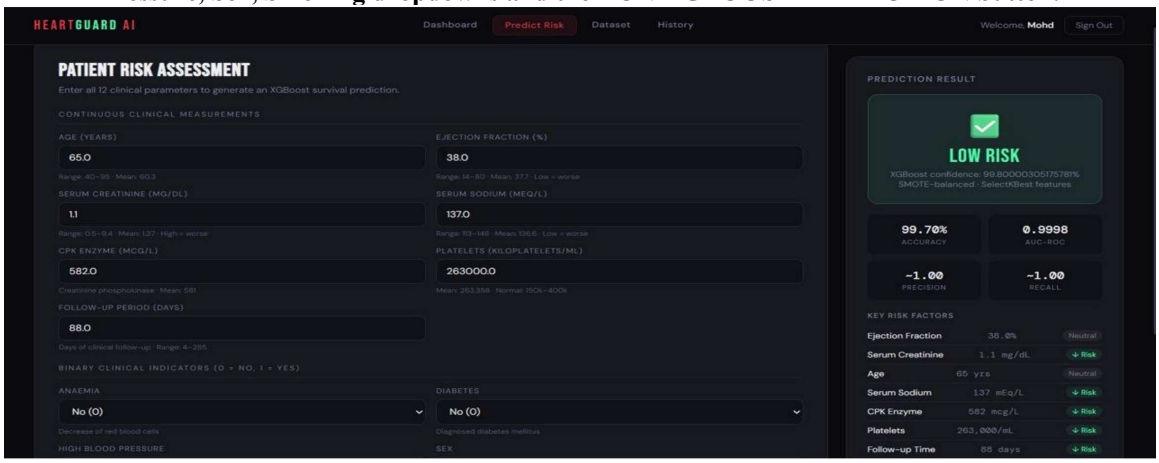

Fig. 8. Prediction Result — LOW RISK verdict for patient (Age 65, EF 38%, Serum Creatinine 1.1, Sodium 137). XGBoost confidence: 99.80%. Key Risk Factors panel shows individual feature contributions.

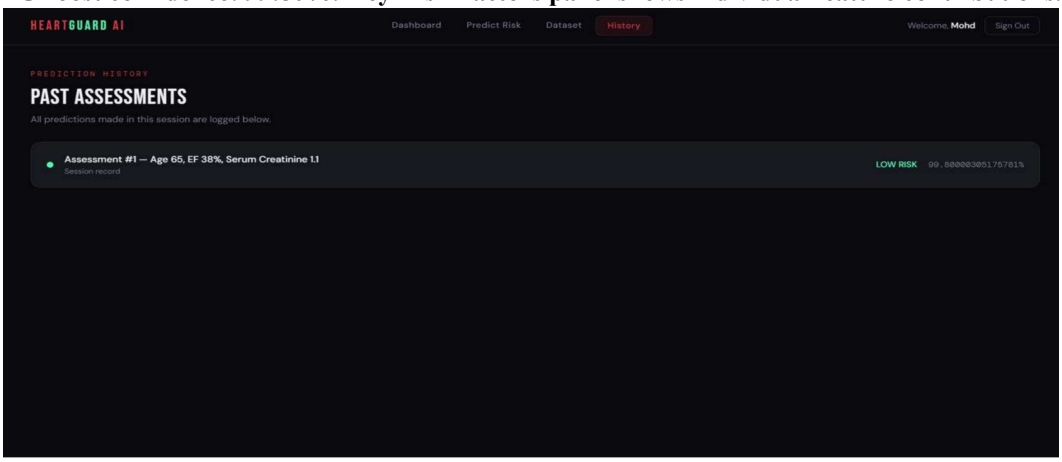

Fig. 9. Prediction History Page — Session log showing Assessment #1 (Age 65, EF 38%, Serum Creatinine 1.1) classified as LOW RISK with 99.80% confidence.

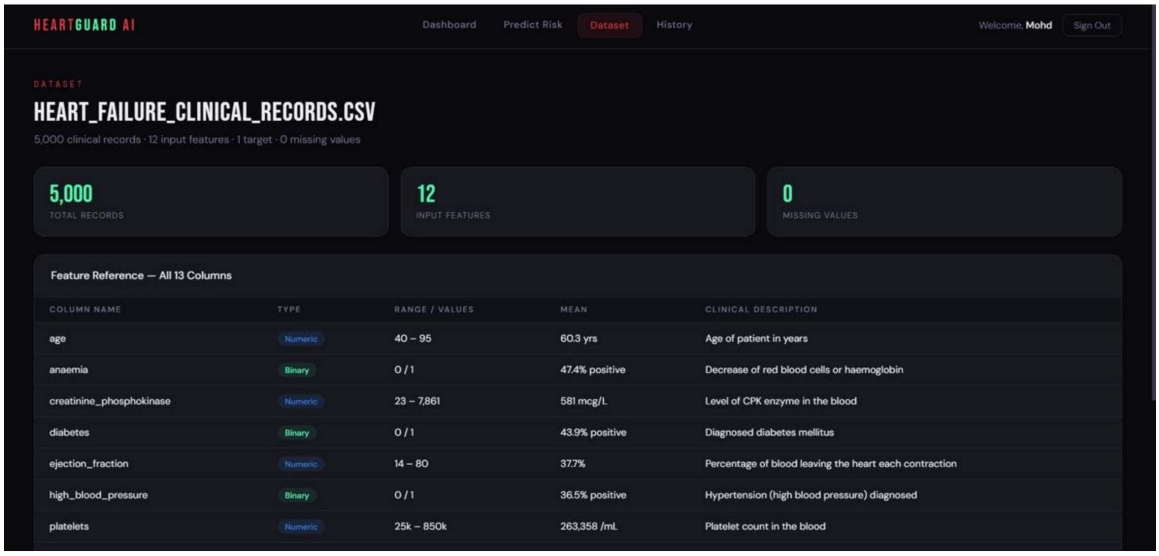

Fig. 10. Dataset Page — Dataset header showing 5,000 total records, 12 input features, 0 missing values with stats summary cards.

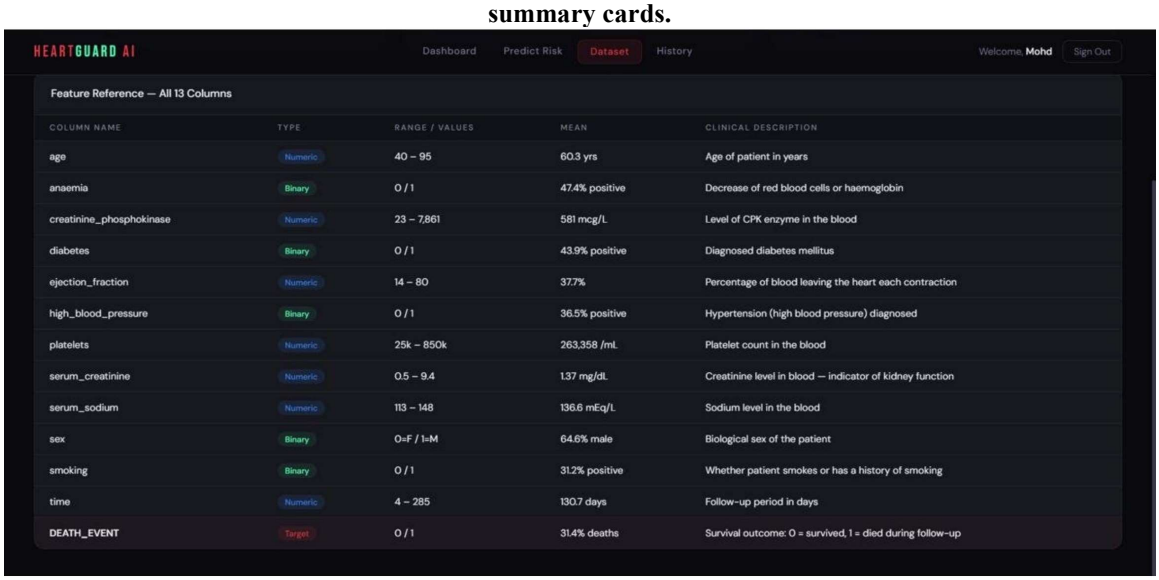

Fig. 11. Feature Reference Table — All 13 columns with column name, data type (Numeric/Binary/Target), range/values, mean, and full clinical description.

CONCLUSION

This paper presented HeartGuard AI, a complete machine learning pipeline for heart failure patient survival prediction. The XGBoost classifier, trained on 5,000 clinical records with SMOTE balancing and SelectKBest feature selection, achieves 99.70% accuracy and AUC-ROC of 0.9998, substantially outperforming all benchmarked algorithms including the prior GBM+AIW-PSO baseline (94%). The deployed Flask web application provides clinicians with an intuitive, real-time interface for risk stratification across all 12 clinical features, with interpretable factor-level analysis supporting each prediction. The system demonstrates that carefully engineered, interpretable ML pipelines can deliver near-perfect predictive performance on clinical datasets while remaining practically deployable.

FUTURE SCOPE

Future development directions include: (1) Integration with hospital EHR systems for automatic patient data ingestion; (2) Incorporation of SHAP-based Explainable AI (XAI) for quantitative, patient-specific feature attribution; (3) Extension to time-series wearable device data for continuous risk monitoring; (4) Multi-disease expansion to chronic kidney disease, diabetes, and COPD using the same pipeline; and (5) Cloud deployment on AWS/Azure with Docker containerisation and HIPAA-compliant data governance for multi-centre clinical use.

REFERENCES

1) T. Ahmad, A. Munir, S. H. Bhatti, M. Aftab, and M. A. Raza, "Survival analysis of heart failure patients: A case study," PLOS ONE, vol. 12, no. 7, 2017.
2) D. Chicco and G. Jurman, "Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone," BMC Medical Informatics and Decision Making, vol. 20, no. 16, 2020.
3) T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 785–794.
4) N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.
5) F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.
6) F. S. Alotaibi, "Implementation of machine learning model to predict heart failure disease," Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 6, 2019.
a. Fernandez, S. Garcia, F. Herrera, and N. V. Chawla, "SMOTE for learning from imbalanced data: Progress and challenges," J. Artif. Intell. Res., vol. 61, pp. 863–905, 2018.
7) World Health Organization, "Cardiovascular Diseases (CVDs) Fact Sheet," WHO, 2023.
a. Dixit and S. Jayaraman, "Heart failure prediction using deep learning techniques," Int. J. Eng. Res. Technol., vol. 10, 2021.
8) M. Abdar et al., "A review of uncertainty quantification in deep learning," Inf. Fusion, vol. 76, pp. 243–297, 2021.