LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CU6051NI Artificial Intelligence

75% Individual Coursework

Submission: Final Submission

Academic Semester: Autumn Semester 2025

Credit: 15 credit semester long module

Student Name: Prashanna Sthapit

London Met ID: 23050386

College ID: NP01CP4A230194

Assignment Due Date: 21/01/2026.

Assignment Submission Date: 21/01/2026

Submitted To: Mr. Binod Bhattarai

GitHub Link	*https://github.com/prashannasthapit/AI_CW2_23050386*

I confirm that I understand my coursework needs to be submitted online via MST Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Table of Contents

1. Title of the Project 1
2. Introduction 1
3. Background 2
   - 3.1. Related Works 2
   - 3.2. Problem Statement 3
   - 3.3. Research Questions 3
   - 3.4. Objectives 4
4. Solution 4
   - 4.1. Dataset Description 4
   - 4.2. Algorithms Used 5
   - 4.3. Pseudocode 6
     - 4.3.1. Logistic Regression 6
     - 4.3.2. Random Forest 8
     - 4.3.3. XGBoost 10
   - 4.4. Diagrammatical representations of the solution 13
     - 4.4.1. Overall 13
     - 4.4.2. Logistic Regression 14
     - 4.4.3. Random Forest 15
     - 4.4.4. XGBoost 16
     - 4.4.5. Model Comparison Process 17
   - 4.5. Explanation of the development process 18
     - 4.5.1. Methodology 18
     - 4.5.2. Evaluation Metrics 18
     - 4.5.3. Project Timeline 19
     - 4.5.4. Tools and Technologies 19
   - 4.6. Achieved results 20
5. References 28

Table Of Figures

Figure 1: Overall Flowchart 13
Figure 2: Logistic Regression Flowchart 14
Figure 3: Random Forest Flowchart 15
Figure 4: XGBoost Flowchart 16
Figure 5: Model Comparison Diagram 17
Figure 6: Logistic Regression Classification Report 20
Figure 7: Random Forest Classification Report 20
Figure 8: XGBoost Classification Report 21
Figure 9: Model Performance Comparison 21
Figure 10: Model Performance Comparison Bar Chart 22
Figure 11: ROC Curves Comparison 22
Figure 12: Train vs Test Accuracy 23
Figure 13: Train vs Test AUC-ROC 23
Figure 14: Logistic Regression Confusion Matrix 24
Figure 15: Random Forest Confusion Matrix 24
Figure 16: XGBoost Confusion Matrix 25
Figure 17: Top features Random Forest 25
Figure 18: Top features Random Forest 26
Figure 19: Top features Logistic Regression 26
Figure 20: 5-Fold Cross-Validation AUC-ROC Distribution 27
Figure 21: Final Model Comparison Summary 27

Table Of Tables

Table 1: Table of Algorithms 5

1. Introduction

1.1. Title of the Project

Predicting Emergency Readmission of Discharged Heart Failure Patients: A Comparative Analysis of XGBoost, Random Forest, and Logistic Regression

1.2. Explanation of the Problem Domain

Congestive heart failure is a chronic and progressive cardiovascular condition often causing with frequent hospitalizations and high readmission rates, with a quarter of patients being readmitted to the hospital within 30 days after discharge (Health Recovery Solutions, 2024).

Early emergency readmissions often reflect inadequate discharge planning, unresolved clinical issues, or insufficient post-discharge monitoring. Reducing avoidable readmissions is a major priority for healthcare systems, as it improves patient outcomes while lowering costs.

This project focuses on using machine learning to predict heart failure patients likely to be readmitted. By analysing relevant clinical features related to heart failure, the models aim to support hospital staff in identifying patients who might benefit from enhanced follow up care, interventions, or delayed discharge, ultimately improving patient safety and quality of care.

1.3. Explanation of the AI Concepts Used

This project applies supervised machine learning techniques to a healthcare prediction task. They are trained with historical clinical data where outcome (*emergency readmission or not*) is already known. Three algorithms are used and compared:

- Logistic Regression: It is a statistical classification model commonly used in medical research for binary outcomes, provides interpretability and insight into feature importance.

- Random Forest: It is a ensemble machine learning method. It combines many decision trees to improve the prediction accuracy and reduce overfitting.
- XGBoost: It is a powerful gradient boosting algorithm, builds trees sequentially to optimize predictive performance and handle complex nonlinear relationships.

These models learn patterns from heart failure–specific clinical indicators to estimate the probability of emergency department readmission after discharge.

2. Background

Heart failure affects millions globally, and preventing avoidable readmissions is a key performance metric for hospitals. Traditional methods rely on simple clinical rules that often miss complex interactions between other diseases for instance, diabetes, renal failure and cardiac markers like, BNP, LVEF.

An ML based solution is needed to effectively model these non-linear relationships and improve the accuracy of predicting which patients will require emergency care post-discharge.

2.1. Related Works

a. (Bobak J. Mortazavi, 2016) compared machine learning algorithms for predicting readmission of heart failure patients and found that Random Forest outperformed logistic regression by capturing complex interactions between patient vitals and lab results.

b. (Sara Bersche Golas M, 2018) used XGBoost algorithm to predict 30 day readmissions in heart failure patients, achieving a higher area under the curve (AUC) compared to traditional clinical risk scores, demonstrating the value of gradient boosting in handling missing medical data.

c. (Fuhai Li, 2021) developed and validated an XGBoost based prediction model for in hospital deaths among ICU admitted patients suffering from

heart failure using the MIMIC-III database, showing superior discrimination and calibration compared with the GWTG-HF risk score and demonstrating the effectiveness of machine learning for risk in critically ill heart failure patients.

d. (Samir E AbdelRahman, 2014) analyzed electronic health records for HF patients and concluded that feature selection is critical. Variables like Brain Natriuretic Peptide (BNP) and creatinine levels were identified as top predictors by Random Forest algorithms.

e. (Cida Luo, 2022) demonstrated that ensemble techniques like XGBoost could effectively handle class imbalance, a common issue where the number of readmitted patients is significantly smaller than those not readmitted.

2.2. Problem Statement

High rates of emergency readmission for heart failure patients suggest gaps in discharge protocols and risk assessment. The problem this study addresses is the difficulty in accurately identifying which discharged patients are likely to return to the emergency department, using a comparative analysis of ML algorithms to find the most robust predictor.

2.3. Research Questions

a. Which algorithm (XGBoost, Random Forest, or Logistic Regression) provides the best sensitivity and accuracy for predicting emergency readmissions?
b. How do specific heart failure indicators (e.g., LVEF, NYHA class, Pro BNP) correlate with the risk of emergency return?
c. Can machine learning models effectively handle the class imbalance inherent in readmission data?

2.4. Objectives

a. To train and validate three distinct models: Logistic Regression, Random Forest, and XGBoost.
b. To evaluate model performance using relevant metrics (Recall, F1-Score, ROC-AUC) to ensure high-risk patients are not missed.

3. Solution

3.1. Dataset Description

a. Dataset source: https://physionet.org/content/heart-failure-zigong/1.3/
b. Type: Binary Classification
c. Instances: 2008 patient records
d. Features: Clinical variables including NYHA classification, LVEF, Brain Natriuretic Peptide, and comorbidities
e. Target Variable: *return.to.emergency.department.within.6.months*
f. Missing Values: No

3.2. Algorithms Used

Table 1: Table of Algorithms

	Logistic Regression	**Random Forest**	**XGBoost**
Type	Linear	Ensemble (Bagging)	Ensemble (Boosting)
Learning	Direct optimization	Parallel trees	Sequential trees
Scaling Required	Yes	No	No
Handles Non-linearity	No (linear boundary)	Yes	Yes
Regularization	L1/L2 penalty	Tree pruning, max_depth	L1/L2 + tree pruning
Interpretability	High (coefficients)	Medium (feature importance)	Medium (feature importance)
Class Imbalance	class_weight	class_weight	scale_pos_weight

3.3. Pseudocode

3.3.1. Logistic Regression

```
ALGORITHM: Logistic Regression for Binary Classification

INPUT: X_train (scaled features), y_train (binary labels)
OUTPUT: Predicted probabilities and class labels

FUNCTION sigmoid(z):
    RETURN 1 / (1 + exp(-z))

FUNCTION train_logistic_regression(X, y, learning_rate, max_iterations):
    // Initialize weights and bias
    weights ← vector of zeros (size = number of features)
    bias ← 0

    FOR iteration = 1 TO max_iterations:
        // Forward pass: compute predictions
        z ← X · weights + bias          // Linear combination
        predictions ← sigmoid(z)        // Apply sigmoid activation

        // Compute gradients (with class weights for imbalanced data)
        error ← predictions - y
        weighted_error ← error × class_weights

        gradient_weights ← (X^T · weighted_error) / n_samples
        gradient_bias ← mean(weighted_error)

        // Update parameters
        weights ← weights - learning_rate × gradient_weights
        bias ← bias - learning_rate × gradient_bias

        // Check for convergence
```

```
        IF loss_change < tolerance:
            BREAK

    RETURN weights, bias

FUNCTION predict(X, weights, bias, threshold=0.5):
    z ← X · weights + bias
    probabilities ← sigmoid(z)
    predictions ← IF probabilities ≥ threshold THEN 1 ELSE 0
    RETURN predictions, probabilities
```

3.3.2. Random Forest

```
ALGORITHM: Random Forest Classifier

INPUT: X_train (features), y_train (labels), n_trees, max_depth
OUTPUT: Ensemble predictions

FUNCTION build_decision_tree(X, y, depth, max_depth):
    // Base cases
    IF depth ≥ max_depth OR all y values are same OR n_samples < min_split:
        RETURN LeafNode(majority_class(y))

    // Find best split (using random subset of features)
    feature_subset ← randomly_select(sqrt(n_features))
    best_feature, best_threshold ← None, None
    best_gini ← infinity

    FOR each feature IN feature_subset:
        FOR each threshold IN possible_splits(X[feature]):
            left_mask ← X[feature] ≤ threshold
            right_mask ← X[feature] > threshold

            gini ← weighted_gini(y[left_mask], y[right_mask])

            IF gini < best_gini:
                best_gini ← gini
                best_feature ← feature
                best_threshold ← threshold

    // Split and recurse
    left_child ← build_decision_tree(X[left], y[left], depth+1, max_depth)
    right_child ← build_decision_tree(X[right], y[right], depth+1, max_depth)

    RETURN DecisionNode(best_feature, best_threshold, left_child, right_child)
```

```
FUNCTION train_random_forest(X, y, n_trees):
    forest ← empty list

    FOR i = 1 TO n_trees:
        // Bootstrap sampling (random sample with replacement)
        X_sample, y_sample ← bootstrap_sample(X, y)

        // Build tree on bootstrap sample
        tree ← build_decision_tree(X_sample, y_sample, depth=0, max_depth)
        forest.append(tree)

    RETURN forest

FUNCTION predict(X, forest):
    all_predictions ← empty matrix

    FOR each tree IN forest:
        tree_predictions ← tree.predict(X)
        all_predictions.append(tree_predictions)

    // Majority voting for classification
    final_predictions ← mode(all_predictions, axis=trees)

    // Probability = proportion of trees voting for class 1
    probabilities ← mean(all_predictions, axis=trees)

    RETURN final_predictions, probabilities
```

3.3.3. XGBoost

```
ALGORITHM: XGBoost Classifier

INPUT: X_train (features), y_train (labels), n_rounds, learning_rate
OUTPUT: Ensemble of weak learners (boosted trees)

FUNCTION compute_gradient_hessian(y_true, y_pred):
    // For binary classification with log loss
    probabilities ← sigmoid(y_pred)
    gradient ← probabilities - y_true          // First derivative
    hessian ← probabilities × (1 - probabilities)  // Second derivative
    RETURN gradient, hessian

FUNCTION find_best_split(X, gradient, hessian, lambda, gamma):
    best_gain ← 0
    best_feature, best_threshold ← None, None

    // Sum of gradients and hessians for current node
    G ← sum(gradient)
    H ← sum(hessian)

    FOR each feature IN X.columns:
        // Sort by feature values
        sorted_indices ← argsort(X[feature])
        G_left, H_left ← 0, 0

        FOR each split_point IN sorted_indices:
            G_left ← G_left + gradient[split_point]
            H_left ← H_left + hessian[split_point]
            G_right ← G - G_left
            H_right ← H - H_left

            // XGBoost gain formula with regularization
```

```
        gain ← 0.5 × [
            (G_left²)/(H_left + lambda) +
            (G_right²)/(H_right + lambda) -
            (G²)/(H + lambda)
        ] - gamma

        IF gain > best_gain:
            best_gain ← gain
            best_feature ← feature
            best_threshold ← X[feature][split_point]

    RETURN best_feature, best_threshold, best_gain

FUNCTION build_xgb_tree(X, gradient, hessian, depth, max_depth):
    IF depth ≥ max_depth OR n_samples < min_samples:
        // Leaf weight formula
        weight ← -sum(gradient) / (sum(hessian) + lambda)
        RETURN LeafNode(weight)

    feature, threshold, gain ← find_best_split(X, gradient, hessian)

    IF gain ≤ 0:
        weight ← -sum(gradient) / (sum(hessian) + lambda)
        RETURN LeafNode(weight)

    // Split and recurse
    left_mask ← X[feature] ≤ threshold
    left_child ← build_xgb_tree(X[left], gradient[left], hessian[left], depth+1)
    right_child ← build_xgb_tree(X[right], gradient[right], hessian[right], depth+1)

    RETURN DecisionNode(feature, threshold, left_child, right_child)

FUNCTION train_xgboost(X, y, n_rounds, learning_rate):
    // Initialize predictions (adjusted for class imbalance)
```

```
    base_score ← log(sum(y==1) / sum(y==0))  // log-odds of positive class
    F ← array of base_score (size = n_samples)

    trees ← empty list

    FOR round = 1 TO n_rounds:
        // Compute gradients and hessians
        gradient, hessian ← compute_gradient_hessian(y, F)

        // Apply scale_pos_weight for class imbalance
        gradient[y==1] ← gradient[y==1] × scale_pos_weight
        hessian[y==1] ← hessian[y==1] × scale_pos_weight

        // Build tree to fit negative gradients
        tree ← build_xgb_tree(X, gradient, hessian, depth=0, max_depth)

        // Update predictions with shrinkage (learning rate)
        F ← F + learning_rate × tree.predict(X)

        trees.append(tree)

    RETURN trees, base_score

FUNCTION predict(X, trees, base_score, learning_rate):
    F ← array of base_score

    FOR each tree IN trees:
        F ← F + learning_rate × tree.predict(X)

    probabilities ← sigmoid(F)
    predictions ← IF probabilities ≥ 0.5 THEN 1 ELSE 0

    RETURN predictions, probabilities
```

3.4. Diagrammatical representations of the solution

3.4.1. Overall

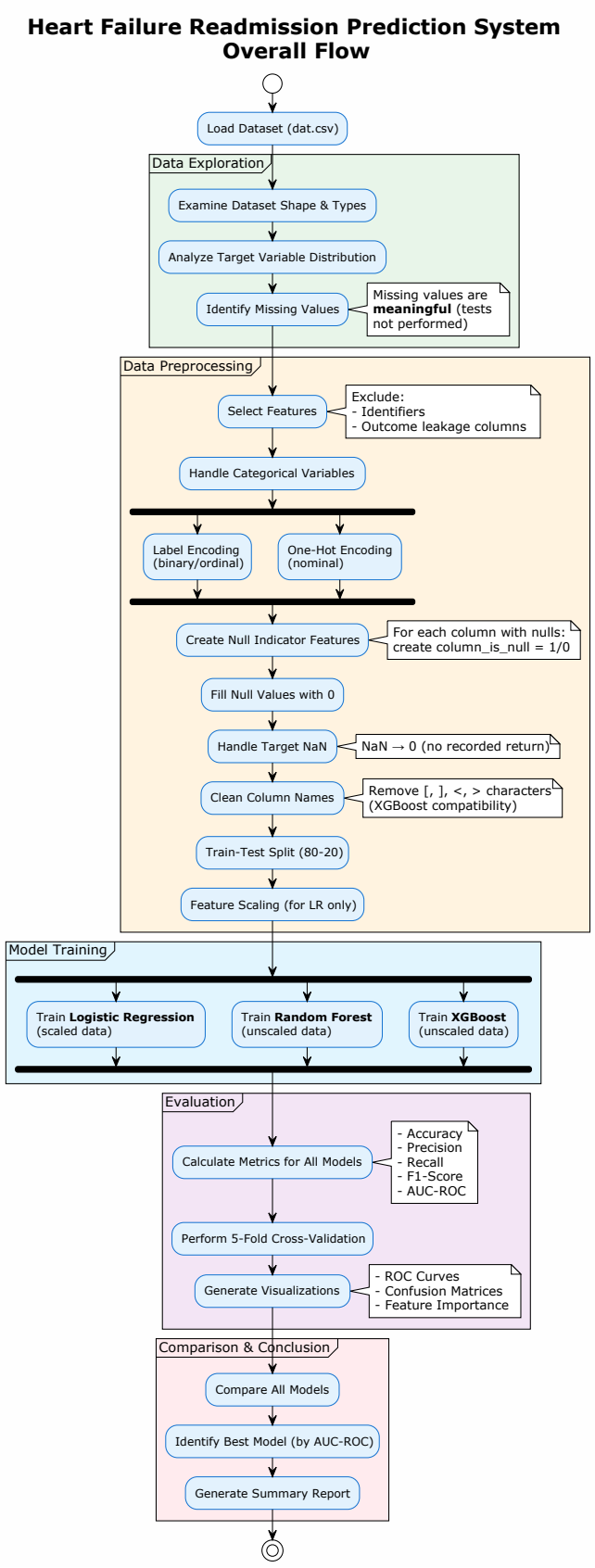

Figure 1: Overall Flowchart

3.4.2. Logistic Regression

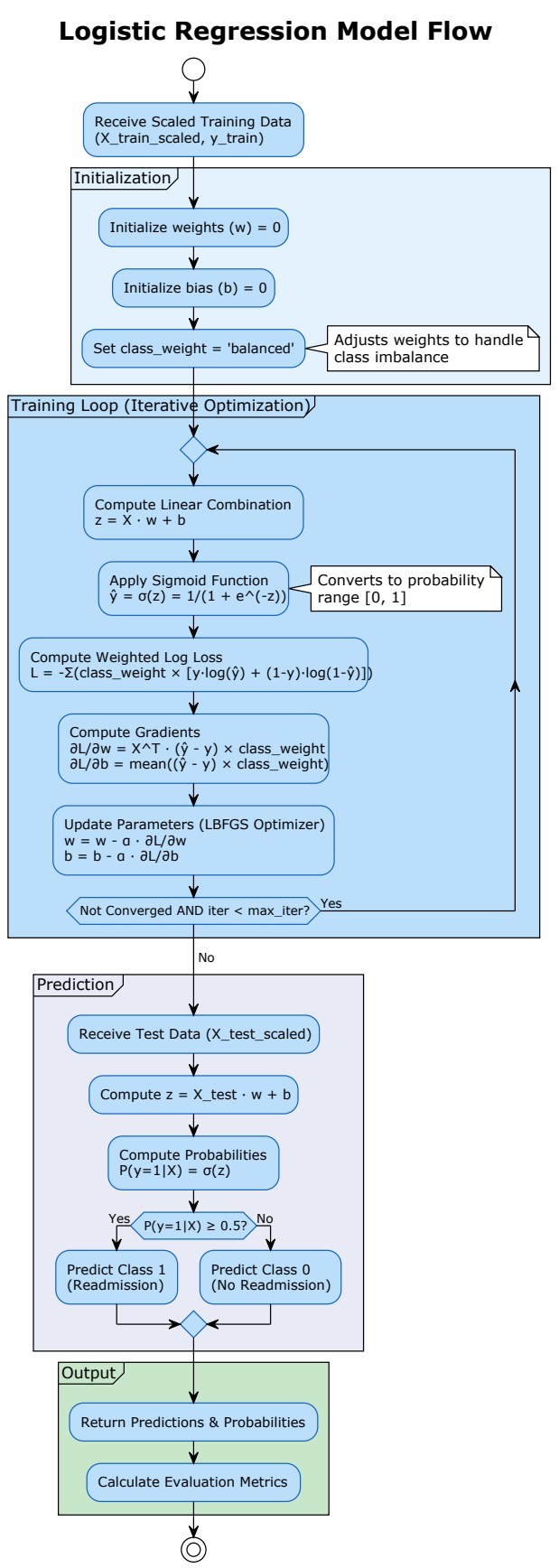

Figure 2: Logistic Regression Flowchart

3.4.3. Random Forest

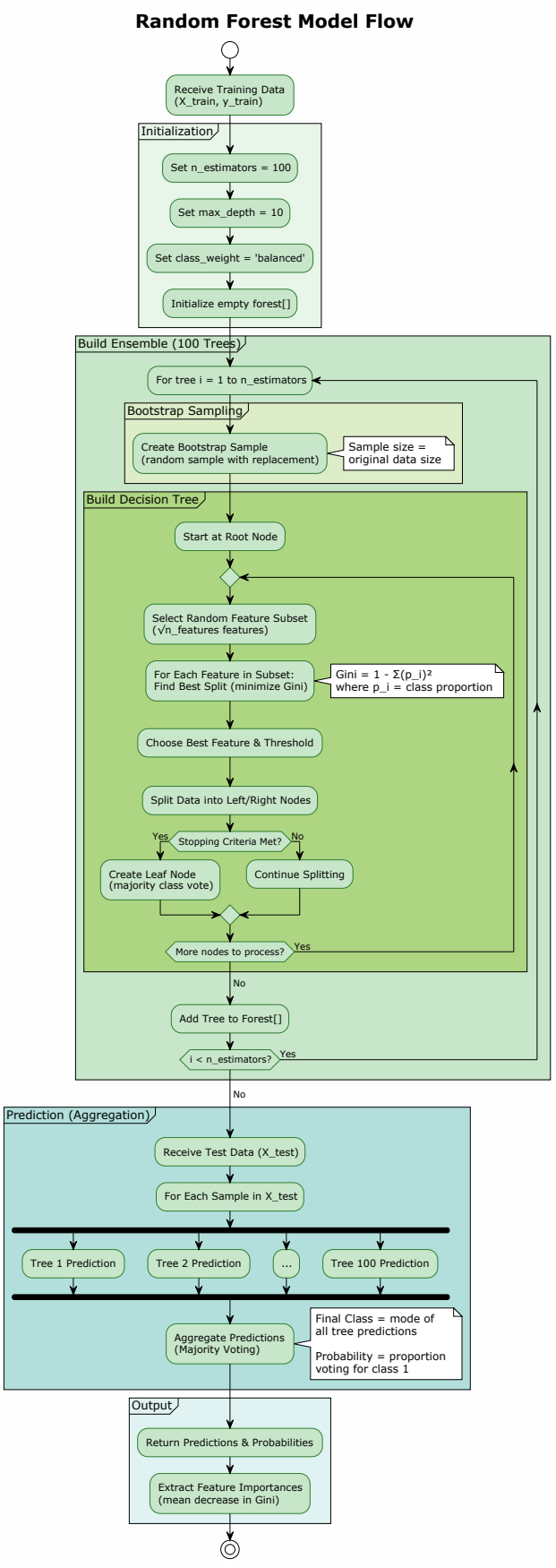

Figure 3: Random Forest Flowchart

3.4.4. XGBoost

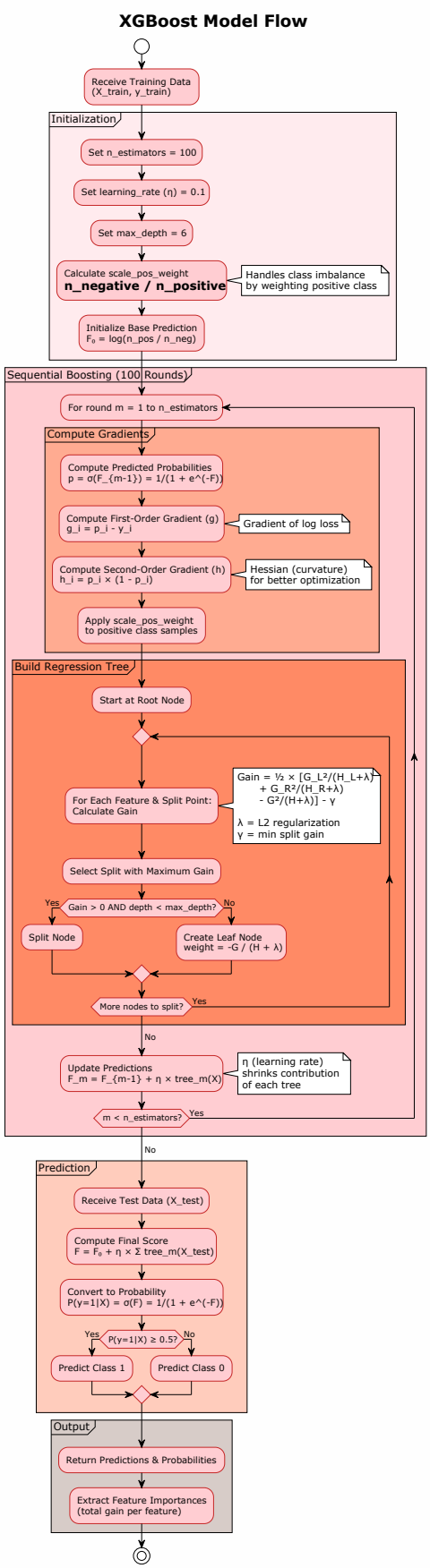

Figure 4: XGBoost Flowchart

3.4.5. Model Comparison Process

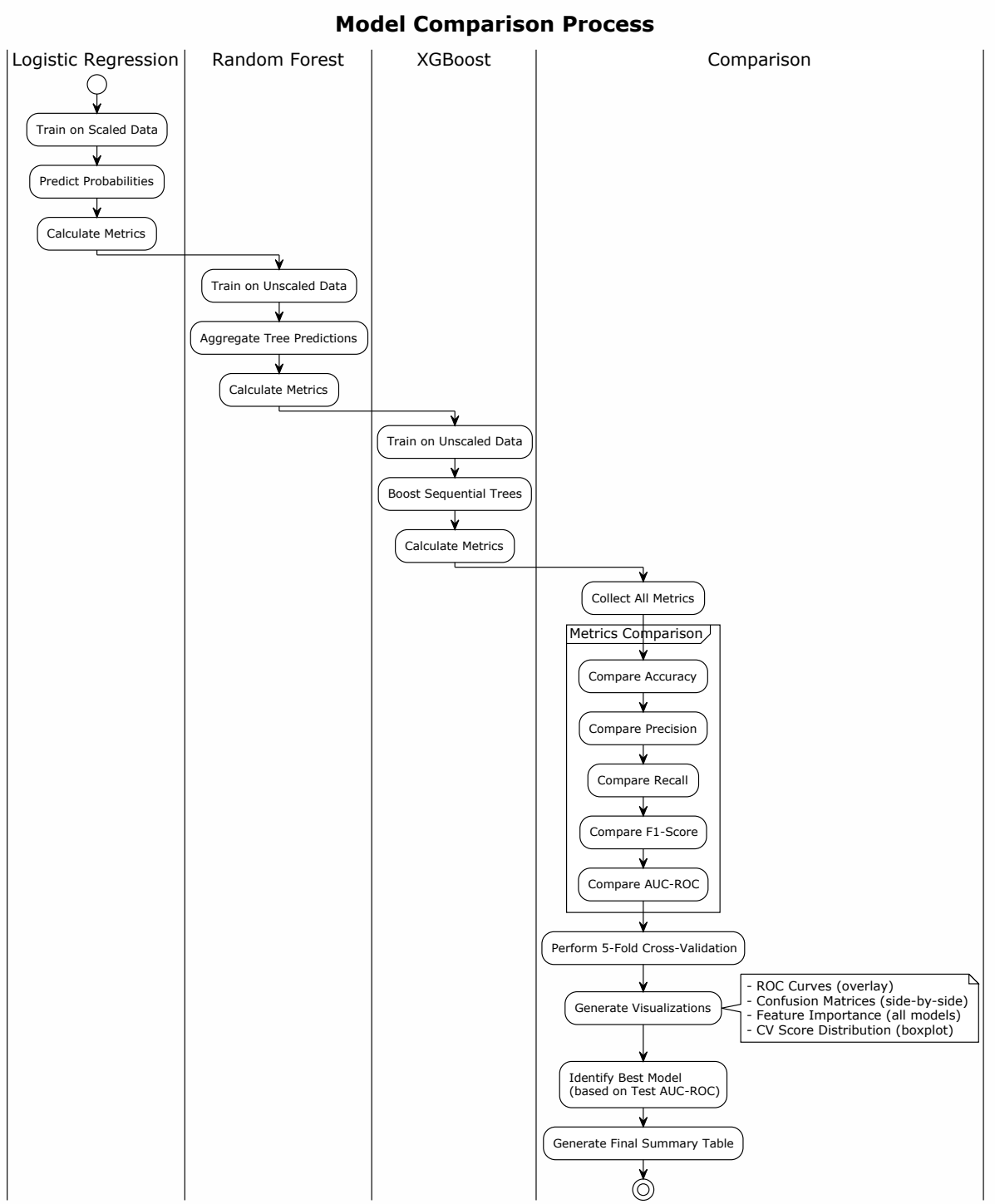

Figure 5: Model Comparison Diagram

3.5. Explanation of the development process

3.5.1. Methodology

a. Data Ingestion: Load the heart failure dataset subset.
b. Preprocessing:
    i. Impute missing values (e.g., mean imputation for LVEF, mode for NYHA class).
    ii. Encode categorical features (e.g., Outcome, Gender) using One-Hot Encoding.
    iii. Scale numerical features (e.g., BNP, Age) for Logistic Regression.
c. EDA: Visualize relationships between LVEF, NYHA class, and Readmission.
d. Split: Divide into training (80%) and testing (20%) sets.
e. Modelling: Train Logistic Regression, Random Forest, and XGBoost with hyperparameter tuning (e.g., tree depth, learning rate).
f. Evaluation: Compare using ROC-AUC and Recall (Sensitivity).

3.5.2. Evaluation Metrics

a. Recall (Sensitivity): The most critical metric, as missing a patient likely to be readmitted (False Negative) carries high medical risk.
b. ROC-AUC: To evaluate the model's discrimination capability across different thresholds.
c. Accuracy: General performance measure.
d. F1-Score: To balance precision and recall given the likely class imbalance.

3.5.3. Project Timeline

Week 1: Dataset understanding and cleaning (focusing on HF specific columns).

Week 2: Data Analysis and Cleaning.

Week 3: Train models.

Week 4: Final evaluation, result compilation, and report generation.

3.5.4. Tools and Technologies

Python: Programming language.

Pandas: To be used in data manipulation.

Scikit learn: To use for Logistic Regression and Random Forest implementation.

Jupyter Notebook: For interactive coding and visualization.

3.6. Achieved results

```
============================================================
LOGISTIC REGRESSION
============================================================

Classification Report (Test Set):
--------------------------------------
                precision    recall  f1-score   support

No Readmission       0.75      0.65      0.70       247
   Readmission       0.54      0.66      0.59       155

      accuracy                           0.65       402
     macro avg       0.65      0.65      0.65       402
  weighted avg       0.67      0.65      0.66       402


5-Fold Cross-Validation AUC-ROC: 0.6304 (+/- 0.0553)
```

Figure 6: Logistic Regression Classification Report

```
============================================================
RANDOM FOREST
============================================================

Classification Report (Test Set):
--------------------------------------
                precision    recall  f1-score   support

No Readmission       0.69      0.83      0.76       247
   Readmission       0.61      0.41      0.49       155

      accuracy                           0.67       402
     macro avg       0.65      0.62      0.62       402
  weighted avg       0.66      0.67      0.66       402


5-Fold Cross-Validation AUC-ROC: 0.6391 (+/- 0.0327)
```

Figure 7: Random Forest Classification Report

```
============================================================
XGBOOST
============================================================

Classification Report (Test Set):
----------------------------------------
                precision    recall  f1-score   support

No Readmission       0.70      0.79      0.74       247
   Readmission       0.57      0.45      0.50       155

      accuracy                           0.66       402
     macro avg       0.64      0.62      0.62       402
  weighted avg       0.65      0.66      0.65       402


5-Fold Cross-Validation AUC-ROC: 0.6299 (+/- 0.0315)
```

Figure 8: XGBoost Classification Report

```
================================================================================
MODEL PERFORMANCE COMPARISON
================================================================================

Test Set Performance Metrics:
------------------------------------------------------------
                    Accuracy  Precision  Recall  F1-Score  AUC-ROC
Model
Logistic Regression    0.6542     0.5426  0.6581    0.5948   0.6829
Random Forest          0.6716     0.6095  0.4129    0.4923   0.6729
XGBoost                0.6592     0.5750  0.4452    0.5018   0.6681


Best Model for Each Metric:
--------------------------------------
Accuracy: Random Forest (0.6716)
Precision: Random Forest (0.6095)
Recall: Logistic Regression (0.6581)
F1-Score: Logistic Regression (0.5948)
AUC-ROC: Logistic Regression (0.6829)
```

Figure 9: Model Performance Comparison

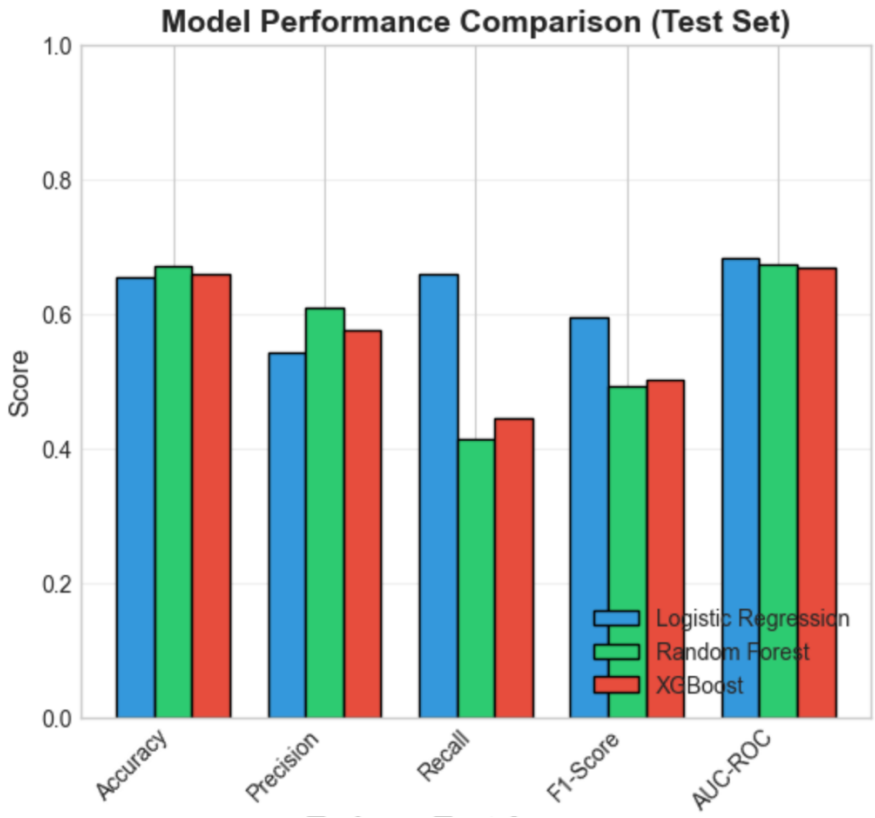

Figure 10: Model Performance Comparison Bar Chart

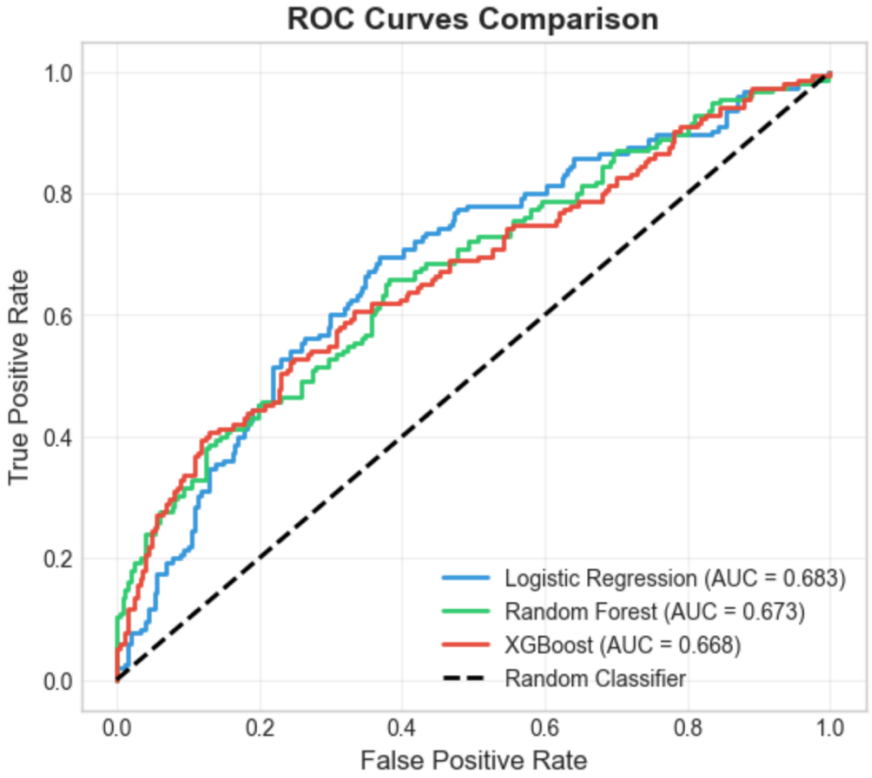

Figure 11: ROC Curves Comparison

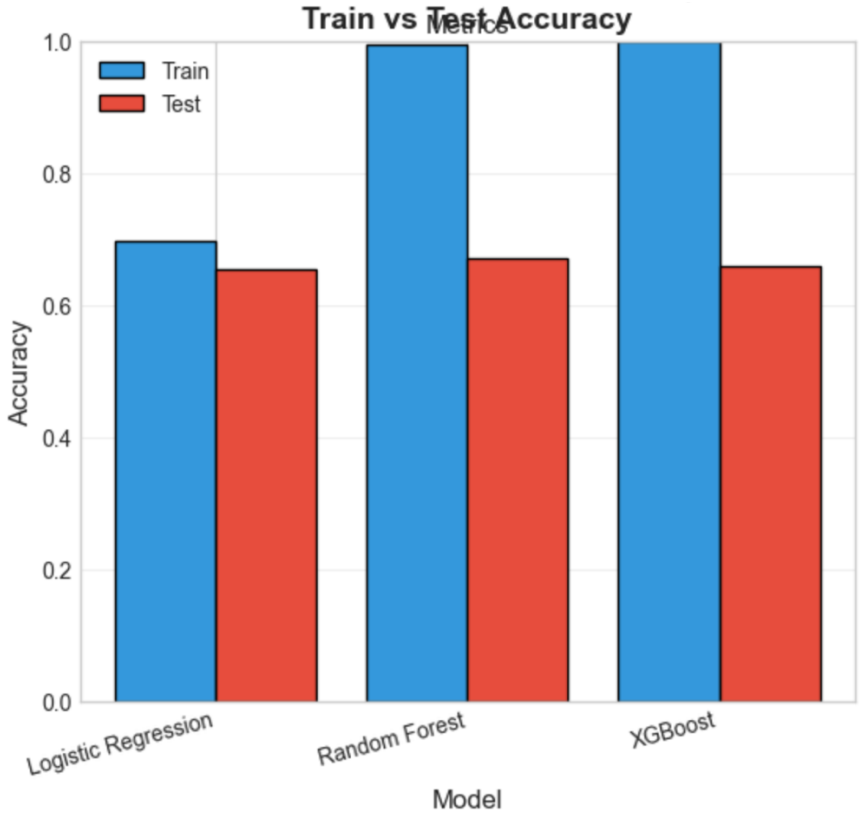

Figure 12: Train vs Test Accuracy

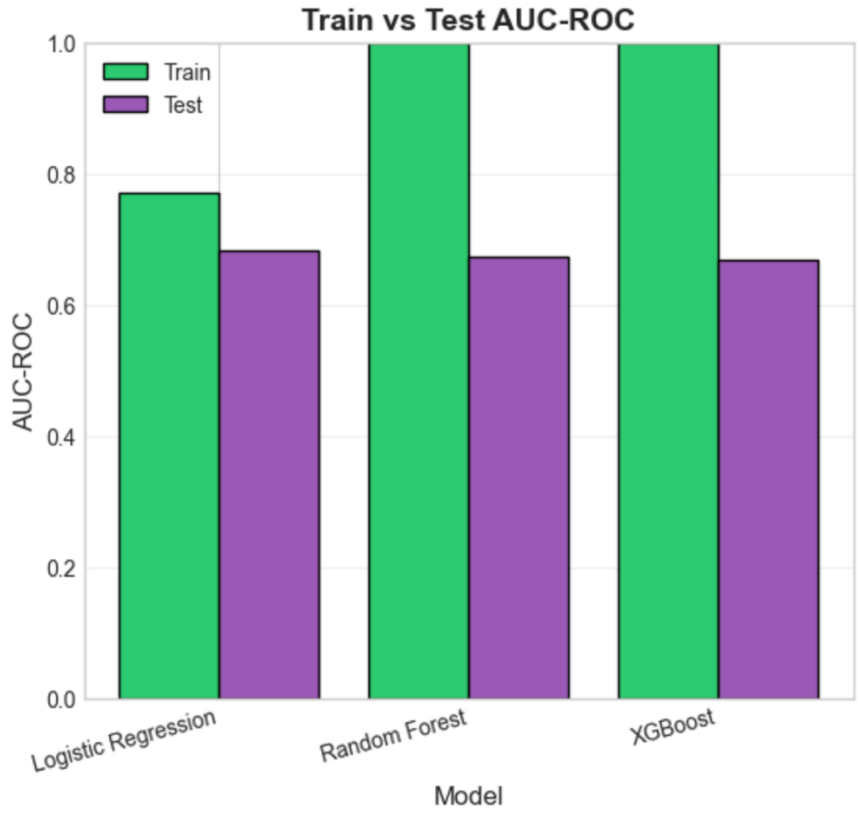

Figure 13: Train vs Test AUC-ROC

Figure 14: Logistic Regression Confusion Matrix

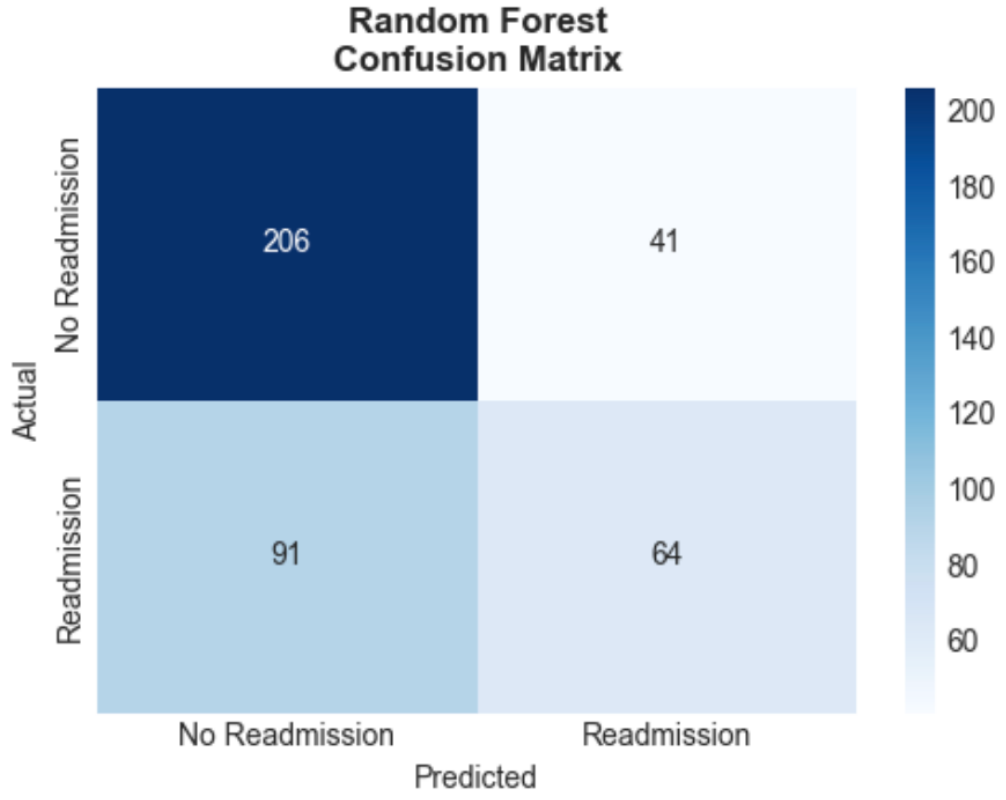

Figure 15: Random Forest Confusion Matrix

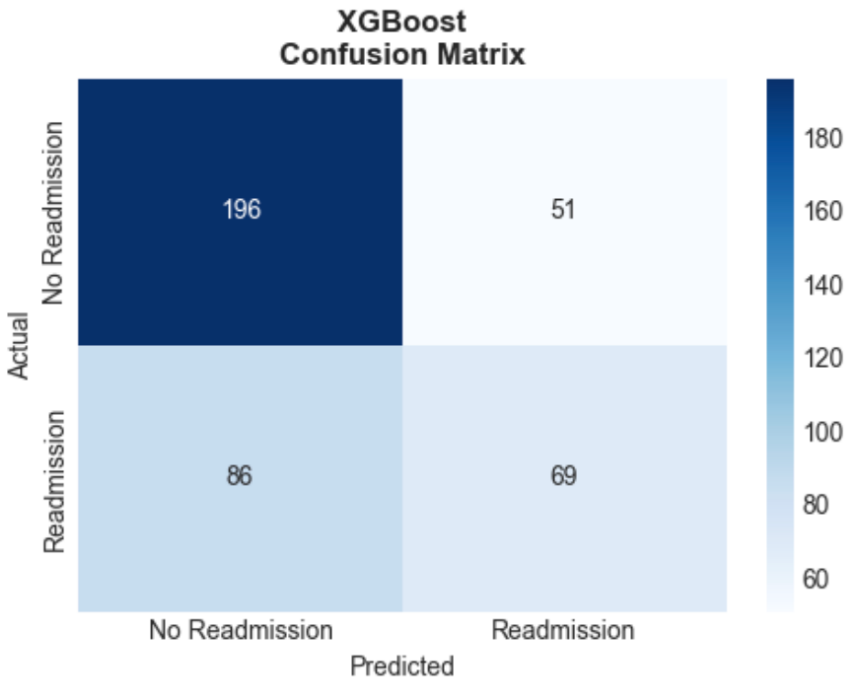

Figure 16: XGBoost Confusion Matrix

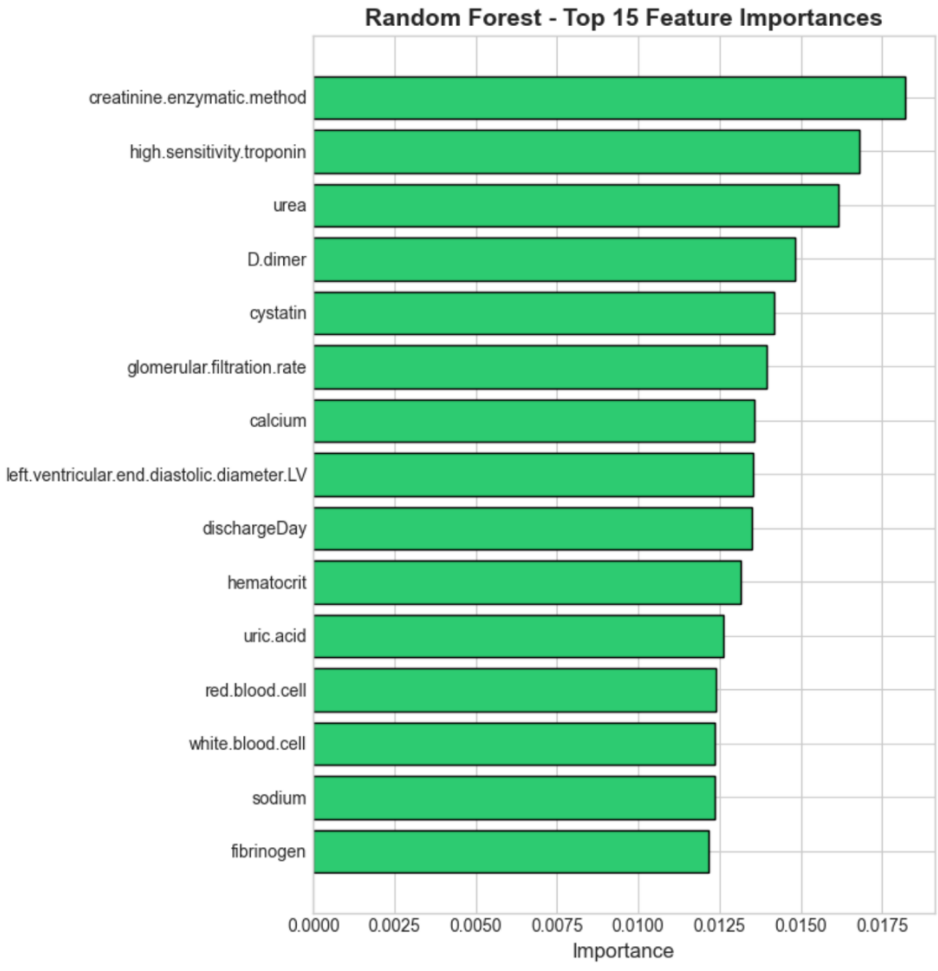

Figure 17: Top features Random Forest

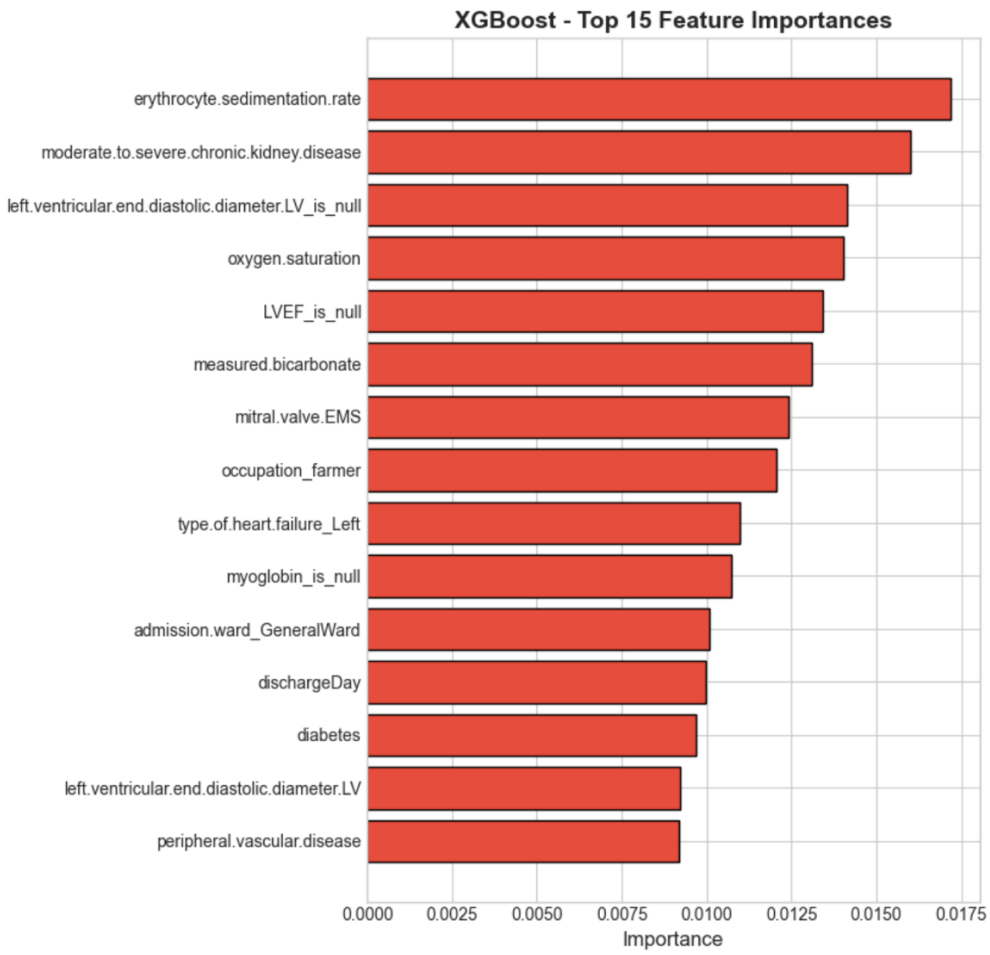

Figure 18: Top features Random Forest

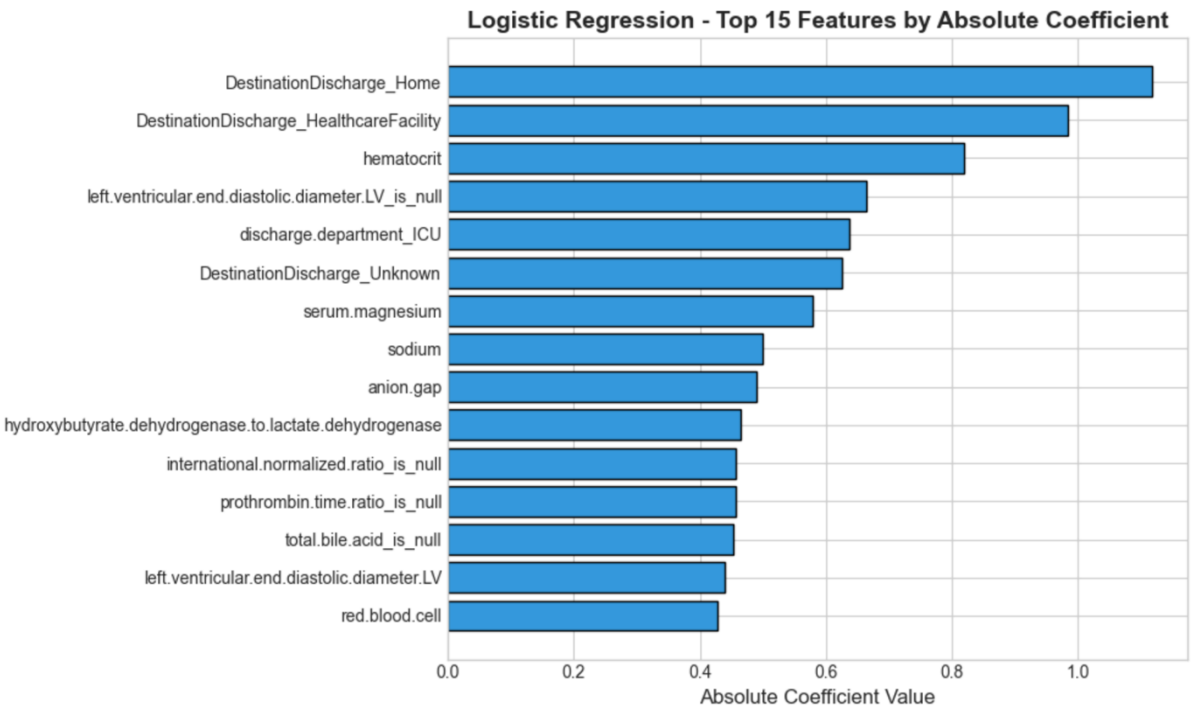

Figure 19: Top features Logistic Regression

```
============================================================
CROSS-VALIDATION COMPARISON (5-Fold Stratified)
============================================================

Cross-Validation Results:
                Model  Mean AUC-ROC  Std AUC-ROC  Min AUC-ROC  Max AUC-ROC
Logistic Regression      0.618250     0.015782     0.606886     0.649542
      Random Forest      0.635926     0.034254     0.600581     0.699771
            XGBoost      0.630183     0.028887     0.593090     0.674431
```

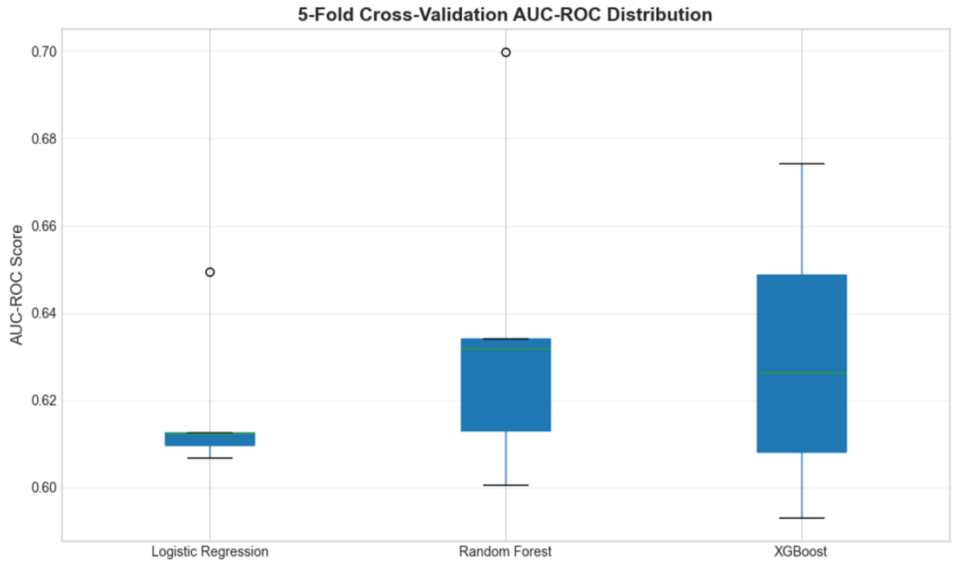

Figure 20: 5-Fold Cross-Validation AUC-ROC Distribution

```
================================================================================
FINAL MODEL COMPARISON SUMMARY
================================================================================

         Metric Logistic Regression Random Forest XGBoost
  Test Accuracy              0.6542        0.6716  0.6592
 Test Precision              0.5426        0.6095  0.5750
    Test Recall              0.6581        0.4129  0.4452
  Test F1-Score              0.5948        0.4923  0.5018
   Test AUC-ROC              0.6829        0.6729  0.6681
CV Mean AUC-ROC              0.6182        0.6359  0.6302
 CV Std AUC-ROC              0.0158        0.0343  0.0289

================================================================================
BEST PERFORMING MODEL (based on Test AUC-ROC): Logistic Regression
AUC-ROC Score: 0.6829
================================================================================
```

Figure 21: Final Model Comparison Summary

4. Conclusion

4.1. Analysis of the work done

This study evaluated the effectiveness of three supervised machine learning models, Logistic Regression, Random Forest, and XGBoost, for predicting emergency readmission of heart failure patients following hospital discharge. The models were trained and assessed using standard classification metrics, with particular emphasis on recall and ROC-AUC due to the clinical importance of correctly identifying high-risk patients.

The findings indicate that **Logistic Regression achieved the best overall performance** on this dataset. This result is consistent with the characteristics of the problem domain and the data used. The dataset size was moderate, and the clinical features exhibited largely linear and stable relationships with readmission risk. Under such conditions, Logistic Regression benefits from regularisation and feature scaling, enabling strong generalisation and reliable performance. Although ensemble methods are often expected to outperform linear models, this study demonstrates that increased model complexity does not necessarily lead to better results.

Random Forest produced competitive outcomes but did not consistently exceed the performance of Logistic Regression. **XGBoost underperformed relative to expectations**, which may be attributed to its sensitivity to dataset size, feature distribution, and hyperparameter configuration. These findings highlight the importance of aligning model choice with data characteristics rather than assuming superiority of advanced algorithms.

Although ensemble models are often expected to outperform linear models, Logistic Regression achieved the best performance in this study, likely due to the relatively small dataset size and the predominantly linear relationships between clinical features and readmission risk.

4.2. How the application/solution addresses real world problems

The developed models demonstrate how machine learning can support hospitals in identifying heart failure patients who are at higher risk of emergency readmission. By using commonly available clinical indicators, the solution can assist healthcare staff in improving discharge planning, prioritising follow-up care, and allocating medical resources more effectively.

The strong performance of Logistic Regression is particularly valuable in a clinical setting, as the model is both accurate and easy to interpret. This makes it suitable for real-world use, where transparency and trust are essential for medical decision-making.

4.3. Further work

- Improve XGBoost performance through deeper hyperparameter tuning and feature engineering.
- Test the models on larger and more diverse datasets to improve generalisation.
- Extend the prediction task to include patient mortality and long-term outcomes.
- Integrate the model into hospital information systems for real-time risk assessment.
- Apply explainable AI techniques to further support clinical trust and adoption.

5. References

Bobak J. Mortazavi, P. N. S. D. M. E. M. B. M. P. K. D. M. M. A. M. M. S.-X. L. P. S. N. N. P. a. H. M. K. M. S., 2016. Analysis of Machine Learning Techniques for Heart Failure Readmissions. *AHA|ASA Journals,* 9(6), p. 12.

Cida Luo, Y. Z. Z. R. L. G. C. Z. W., 2022. A machine learning-based risk stratification tool for in-hospital mortality of intensive care unit patients with heart failure. *Springer Nature Link,* 20(136), p. 9.

Fuhai Li, H. X. Z. M. F. J. Z. ,. Z. L., 2021. Prediction model of in-hospital mortality in intensive care unit patients with heart failure: machine learning-based, retrospective analysis of the MIMIC-III database. *BMJ Journals,* 1(1), p. 17.

Health Recovery Solutions, 2024. *Preventing Heart Failure Readmissions: 4 Key Strategies.* [Online]
Available at: https://www.healthrecoverysolutions.com/blog/preventing-heart-failure-readmissions-4-key-strategies
[Accessed 14 December 2025].

Samir E AbdelRahman, M. Z. B. E. B. K. K., 2014. A three-step approach for the derivation and validation of high-performing predictive models using an operational dataset: congestive heart failure readmission case study. *Springer Nature Link,* 14(41), p. 19.

Sara Bersche Golas M, T. S. S. A. H. O. J. S. T. N. T. H. G. K. J. F. S. K. J. K. K. J., 2018. A machine learning model to predict the risk of 30-day readmissions in patients with heart failure: a retrospective analysis of electronic medical records data. *Springer Nature Link,* 18(44), p. 19.