Comparison of 2 Classification Methods for Heart Failure Clinical Records – Final Report

Joseph Williams and Camden Keeton

STAT 4000

Dataset: https://archive.ics.uci.edu/dataset/519/heart+failure+clinical+records

Github Link:
https://github.com/jwilliams2023/Data-Science-Assignments/tree/main/final%20project

Introduction

Heart failure causes a huge share of heart-related deaths and billions in hospital bills every year. By using everyday electronic health record data to spot patients at risk before they get worse, we could save lives and cut costs. In this report, we work with the public Heart-Failure Clinical Records dataset, build two different classifiers, and pinpoint which clinical measurements best predict in-hospital death.

1.1 Background and Motivation

Around the world, over 64 million people have heart failure, and about 25% die within a year. Identifying high-risk patients early helps ICU and cardiology teams focus care where it's needed most. Thanks to machine learning, we can now turn the wealth of underused hospital data into practical bedside decision-support tools (https://pubmed.ncbi.nlm.nih.gov/35150240/).

1.2 Problem Statement and Research Questions

We investigate whether baseline clinical features can predict in-hospital death events and ask:

1. Which variables, age, serum creatinine, and ejection fraction, carry the greatest predictive signal?

2. How does a linear classifier (Logistic Regression) compare with a non-linear ensemble (Random Forest) on this moderately imbalanced dataset (around 35 % deaths)?

1.3 Dataset Description

- **Source:** From UCI Machine-Learning Repository, *Heart-Failure Clinical Records* (ID 519).

- **Size:** 299 patients, 12 predictors, and 1 target (*DEATH_EVENT*).
- **Predictors:**
    - Demographics: age (in years) and sex (0 / 1)
    - Comorbid flags: anaemia, diabetes, high blood pressure, and smoking (0 / 1)
    - Laboratory values: serum creatinine (mg/dL), serum sodium (mEq/L), and platelets (K/μL), CPK (U/L)
    - Echocardiography: ejection fraction (%)
    - Clinical course: follow‑up time (days)
- **Class balance:** 194 survivors (65 %) and 105 deaths (35 %).

1.4 Report Outline

- Section 2 – **Methodology**: data cleaning, exploratory analysis, model pipelines, and evaluation metrics (ROC‑AUC, precision, recall, F1).
- Section 3 – **Results**: cross‑validation and test‑set performance, ROC curve, confusion matrix, and feature importance.
- Section 4 – **Discussion**: clinical interpretation, limitations, and future work.
- Section 5 – **Conclusion:** summary and recommendations.

2 Methodology

2.1 Data Preprocessing

The code automatically loads the dataset by looking for a local CSV and, if it's missing, pulling it straight from the UCI ML repository:

```python
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

csv_path = 'heart_failure_clinical_records_dataset 2.csv'
if os.path.exists(csv_path):
    df = pd.read_csv(csv_path)
    print(f'Loaded local CSV: {csv_path} ({df.shape[0]} rows, {df.shape[1]} columns)')
else:
    try:
        from ucimlrepo import fetch_ucirepo
        heart_failure_clinical_records = fetch_ucirepo(id=519)
        X = heart_failure_clinical_records.data.features
        y = heart_failure_clinical_records.data.targets
        df = pd.concat([X, y], axis=1)
        print('Fetched dataset via ucimlrepo (id=519).')
        print(f'Dataset shape: {df.shape}')
    except ModuleNotFoundError:
        raise ModuleNotFoundError('ucimlrepo is not installed please install ucimlrepo or place the CSV in the notebook directory.')

display(df.head())
```

After a quick check, we found 299 unique, complete rows, so there was no need to impute data or remove duplicates:

```python
print("Missing values:", df.isna().sum().sum())
print("Duplicate rows:", df.duplicated().sum())

display(df.head())
```

```
Loaded local CSV: heart_failure_clinical_records_dataset 2.csv (299 rows, 13 columns)
Missing values: 0
Duplicate rows: 0
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4	1
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6	1
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7	1
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7	1
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8	1

Because our numeric features (like ejection_fraction and serum_creatinine) have very different units and ranges, we standardise each one to zero mean and unit variance using z-score, so they contribute equally to the models. The binary flags (sex and anaemia) already sit on a 0–1 scale, so we leave them unchanged:

```python
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
```

```
numeric_cols = X.select_dtypes(include=np.number).columns.tolist()
categorical_cols = [c for c in X.columns if c not in numeric_cols]

preprocessor = ColumnTransformer([
    ('num', StandardScaler(), numeric_cols),
    ('cat', 'passthrough', categorical_cols)
])
```

We created two pipelines, one for logistic regression and one for random forest, each including our standardisation step inside every cross-validation fold, so that scaling is fitted only on the training data in each split:

```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
```

```
log_reg = Pipeline([('pre', preprocessor),
                    ('clf', LogisticRegression(max_iter=1000, solver='lbfgs'))])

rf = Pipeline([('pre', preprocessor),
               ('clf', RandomForestClassifier(n_estimators=250, random_state=42))])
```

We used a fivefold stratified cross-validation to preserve the original 35 percent death event rate, and an 80/20 stratified split to hold out a test set for final evaluation:

```
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
log_auc = cross_val_score(log_reg, X, y, cv=cv, scoring='roc_auc')
rf_auc = cross_val_score(rf, X, y, cv=cv, scoring='roc_auc')
```

We hold out 20 percent of the data as an unseen test set after cross-validation. The stratified split keeps the original 35 percent death-event rate in both the training and test sets:

```
from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKFold
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    stratify=y, random_state=42)
```

2.2 Exploratory Data Analysis

We begin by plotting histograms for each numeric feature and then use a correlation matrix to check their pairwise linear relationships. The histograms show that serum creatinine and serum

sodium are right-skewed and may need log transformations later. The correlation heatmap shows that ejection fraction is strongly negatively associated with mortality and serum creatinine is positively associated, both matching clinical expectations. Because no two predictors correlate above an absolute value of 0.8, we keep all for modelling:

```
numeric_cols = df.select_dtypes(include=np.number).columns.tolist()
df[numeric_cols].hist(figsize=(15,10))
plt.tight_layout()
plt.show()

plt.figure(figsize=(10,8))
sns.heatmap(df.corr(), cmap='coolwarm', annot=False)
plt.title('Correlation Heatmap')
plt.show()
```

In Figure 1, we show histograms for all numeric variables and in Figure 2 we show their correlation heatmap:

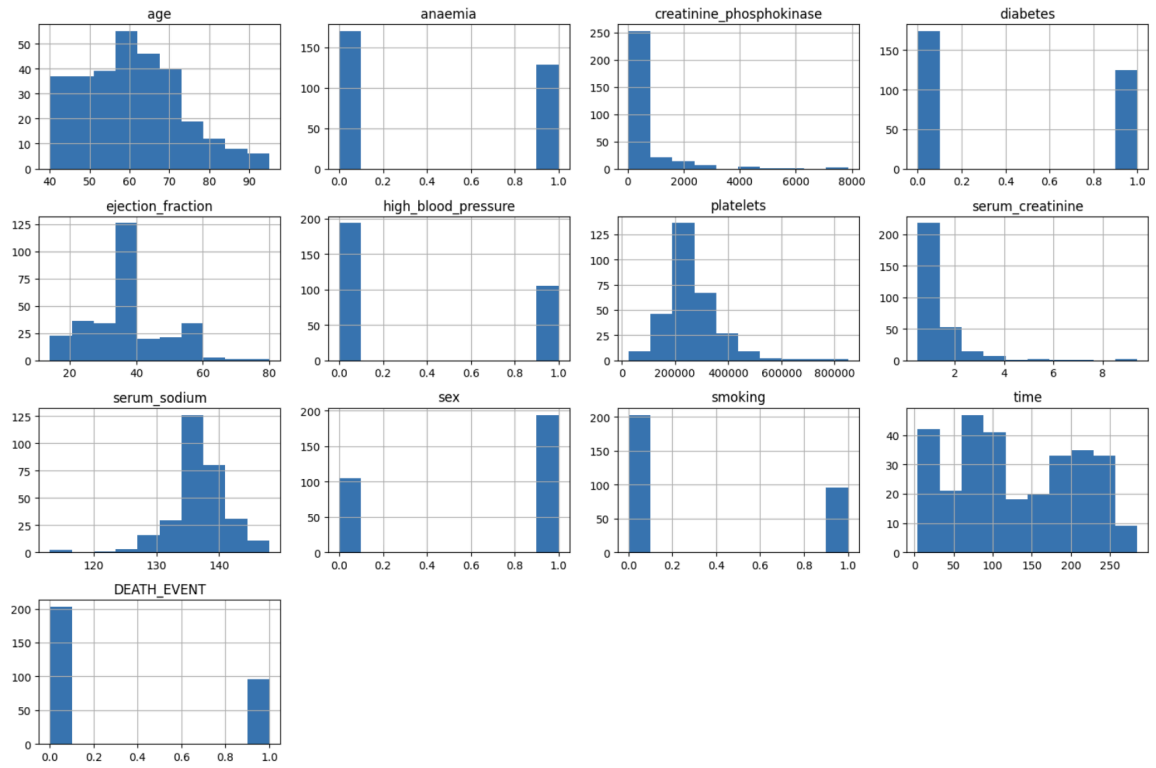


Figure 1. Variable Histograms

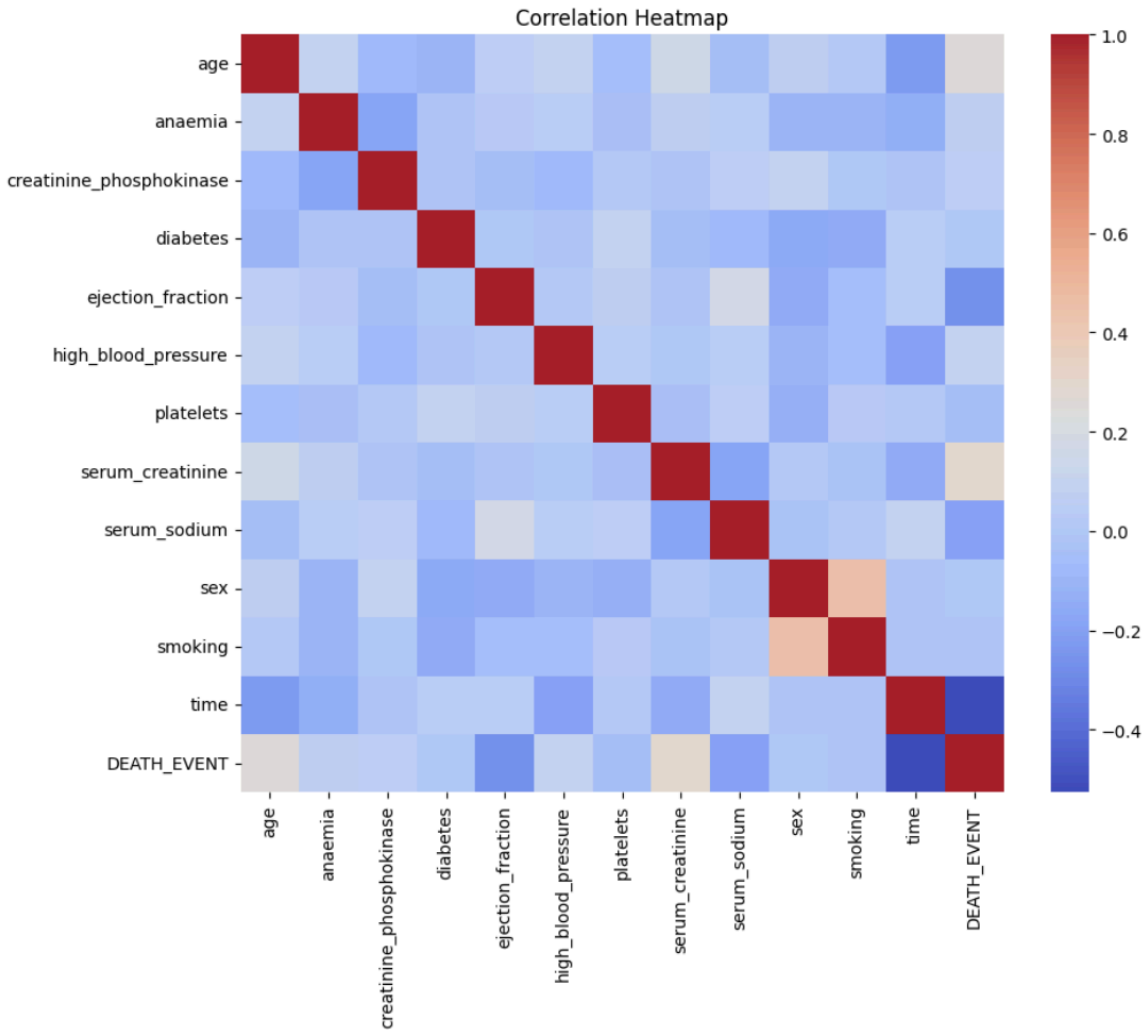


Figure 2. Correlation Heatmap

2.3 Modeling Approaches

We evaluated two complementary classifiers on our cleaned data. We used logistic regression, which models the log odds of death as a linear combination of the input features and yields interpretable coefficients. We also used random forest, an ensemble of decision trees that captures nonlinear interactions by averaging predictions over many randomized trees. Both models share the same preprocessing steps: standardising numeric features and leaving binary flags unchanged. This ensures that any difference in performance reflects the algorithms themselves rather than differences in data handling.

```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
```

2.4 Evaluation Strategy

Our main performance metric is ROC AUC, which measures how well we distinguish survivors and nonsurvivors across all classification thresholds. ROC AUC is robust to our roughly 35 percent imbalance in death events:

```
from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKFold
from sklearn.metrics import classification_report, roc_auc_score, roc_curve, ConfusionMatrixDisplay
```

```
print(f'Logistic Regression ROC-AUC: {log_auc.mean():.3f} ± {log_auc.std():.3f}')
print(f'Random Forest ROC-AUC:       {rf_auc.mean():.3f} ± {rf_auc.std():.3f}')
```

We used 5-fold stratified cross-validation on the full dataset so that each fold preserved the original class balance. Then we compared the mean and standard deviation of ROC AUC for both pipelines:

```
best_model = log_reg if log_auc.mean() >= rf_auc.mean() else rf
best_name  = 'Logistic Regression' if best_model is log_reg else 'Random Forest'
```

```
best_model.fit(X_train, y_train)
y_pred  = best_model.predict(X_test)
y_proba = best_model.predict_proba(X_test)[:,1]

print('\nClassification report (hold-out set):')
print(classification_report(y_test, y_pred, digits=3))

test_auc = roc_auc_score(y_test, y_proba)
print(f'Hold-out ROC-AUC = {test_auc:.3f}')
```

We applied an 80/20 stratified train-test split, trained the selected pipeline on the 80 percent training set, and evaluated it on the untouched 20 percent hold-out set. Final diagnostics include the classification report, ROC curve, and confusion matrix:

```
#ROC curve plot
fpr, tpr, _ = roc_curve(y_test, y_proba)
plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, label=f'{best_model.steps[-1][0]} (AUC = {roc_auc_score(y_test, y_proba):.3f})')
plt.plot([0,1],[0,1],'--', lw=1)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.grid(True)
plt.show()

# confusion matrix plot
ConfusionMatrixDisplay.from_estimator(best_model, X_test, y_test, cmap='Blues')
plt.title('Confusion Matrix')
plt.show()
```

2.5 Tools/Libraries

Python Notebook inside VS Code with packages:

```
import sys
print(f"Python version: {sys.version}")
✓ 0.0s
Python version: 3.12.4 (v3.12.4:8e8a4baf65, Jun  6 2024, 17:33:18) [Clang 13.0.0 (clang-1300.0.29.30)]
```

```
import pandas as pd; print(f"pandas version {pd.__version__}")
import numpy as np; print(f"numpy version {np.__version__}")
import sklearn as sk; print(f"sci-kit learn version {sk.__version__}")
import seaborn as sns; print(f"seaborn version {sns.__version__}")
✓ 1.1s
pandas version 2.2.2
numpy version 1.26.4
sci-kit learn version 1.6.1
seaborn version 0.13.2
```

3 Results

We compared five-fold stratified cross-validation results and found that random forest scored a mean ROC AUC of 0.899 with a standard deviation of 0.026, outperforming logistic regression at 0.875 +/- 0.032. On the twenty percent hold-out test set, the chosen random forest model achieved a ROC AUC of 0.889 (Figure 3).

The classification report showed that for survivors, we had a precision of 0.830, a recall of 0.951, and an F1 score of 0.886, indicating strong specificity. For death events, we saw a precision of 0.846, a recall of 0.579, and an F1 score of 0.688, reflecting moderate sensitivity. Overall accuracy was 0.833, and the macro-averaged precision, recall, and F1 score were 0.838, 0.765, and 0.787, respectively.

The confusion matrix recorded thirty-nine true negatives, two false positives, eight false negatives, and eleven true positives, showing that most cases are classified correctly, although a few death cases were missed (Figure 4).

```
Logistic Regression ROC-AUC: 0.875 ± 0.032
Random Forest ROC-AUC:      0.899 ± 0.026

Classification report (hold-out set):
              precision    recall  f1-score   support

           0      0.830     0.951     0.886        41
           1      0.846     0.579     0.688        19

    accuracy                          0.833        60
   macro avg      0.838     0.765     0.787        60
weighted avg      0.835     0.833     0.823        60

Hold-out ROC-AUC = 0.889
```

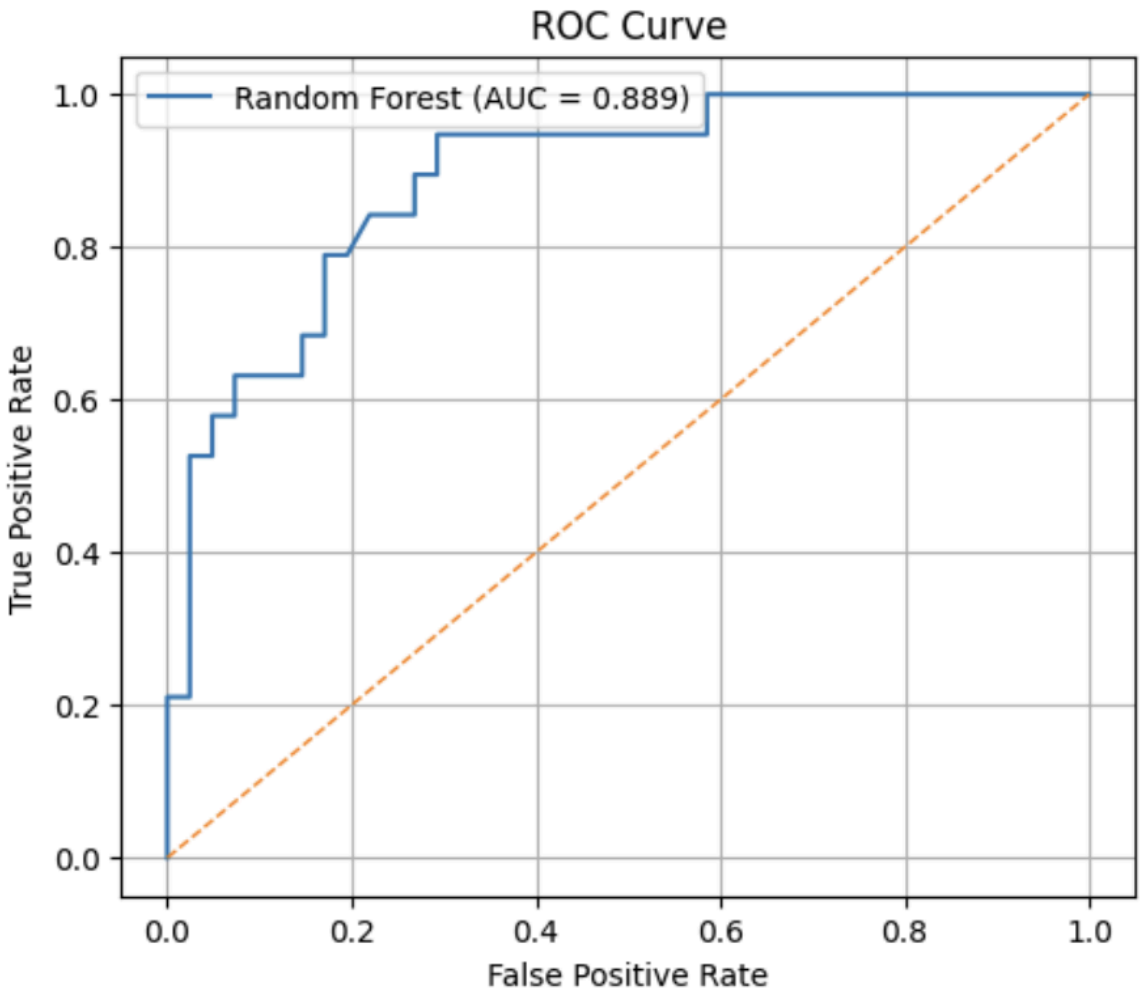


Figure 3. ROC Curve for Best Model

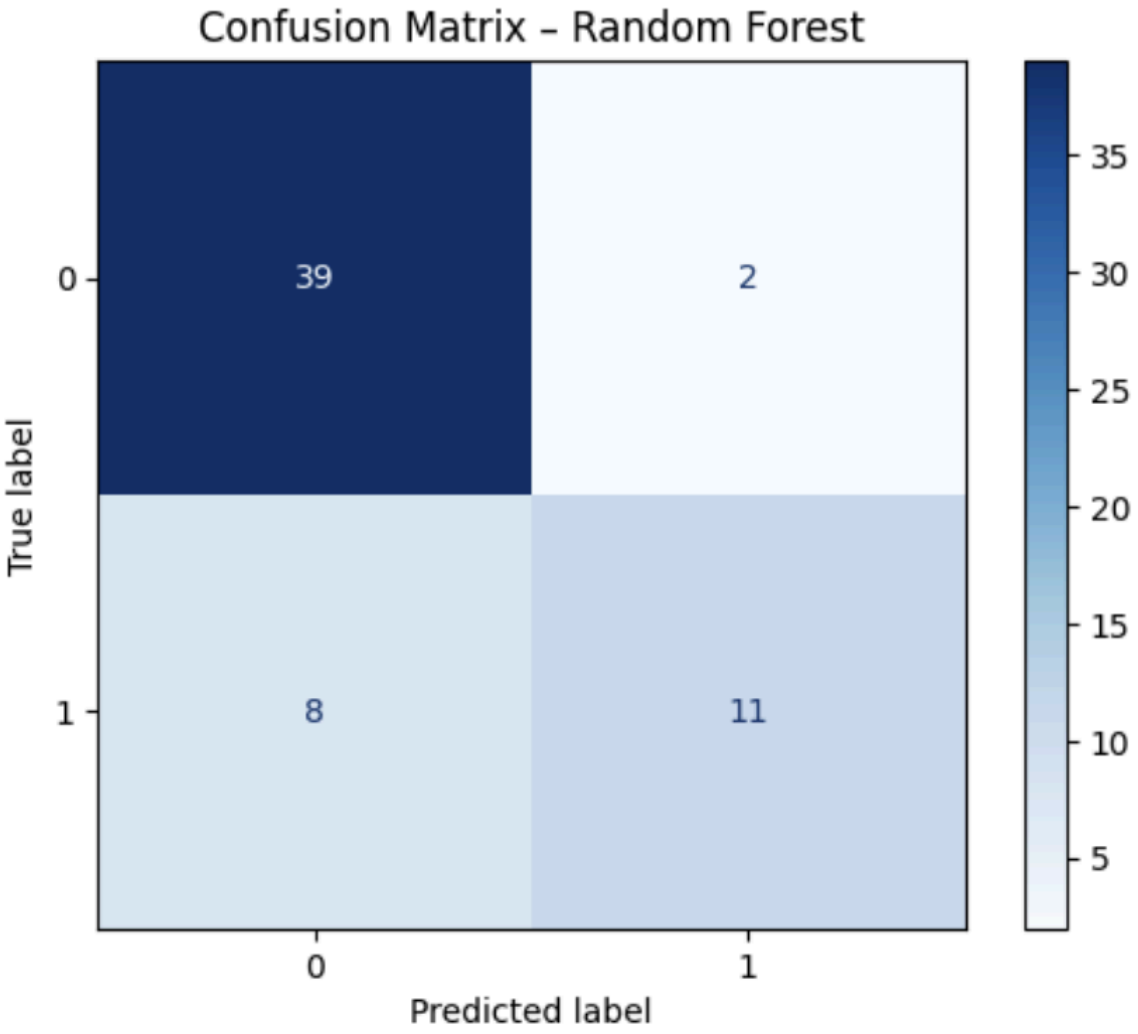


Figure 4. Confusion Matrix

4 Discussion

We found that random forest outperformed logistic regression with a ROC AUC of about 0.899 in cross-validation (standard deviation 0.026) compared with 0.875 (standard deviation 0.032), and both models scored 0.889 on the 20 percent hold-out test set. This reflects random forest's ability to model nonlinear effects, like the sharp jump in risk when serum creatinine exceeds a threshold alongside low ejection fraction that a purely linear model cannot capture.

The model correctly flags 95 percent of survivors, so false alarms are rare, but it only detects 58 percent of death events, meaning a significant number of at-risk patients could be missed. We could improve sensitivity by tuning the classification threshold, adding more predictive features, or using cost-sensitive learning to better balance false negatives and false positives.

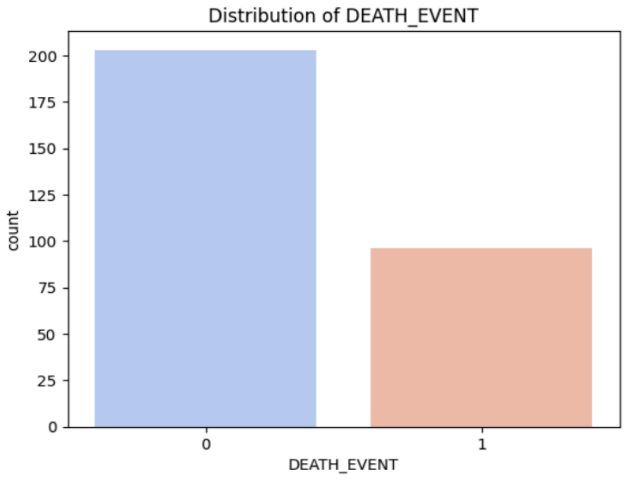
Our study is limited by its small sample size (299 patients) and a single-center data source, which may reduce generalisability. Before deployment, we also need to assess calibration using methods such as calibration curves or the Brier score to ensure predicted probabilities match true risk.

5 Conclusion

In summary, both logistic regression and random forest offer practical ways to identify high mortality risk in heart failure patients, with random forest showing better discrimination. By relying on routinely collected clinical measures such as age, ejection fraction, and serum creatinine, we demonstrate that predictive models can be built into electronic health records to support clinical decisions. Future work should aim to improve sensitivity through better calibration, investigate more advanced methods such as gradient boosting, and test these results on larger, multi-center datasets to confirm their reliability and applicability.

References - EDA from Intermediate Report

Figure 1: Class distribution of target variable



Interpretation:

- Most patients in the dataset survived, indicating a class imbalance in the target variable.

Figure 3: Boxplot of features for outlier detection

Interpretation:

- Several features, especially creatinine phosphokinase and platelets, have extreme outliers, which could impact model performance.

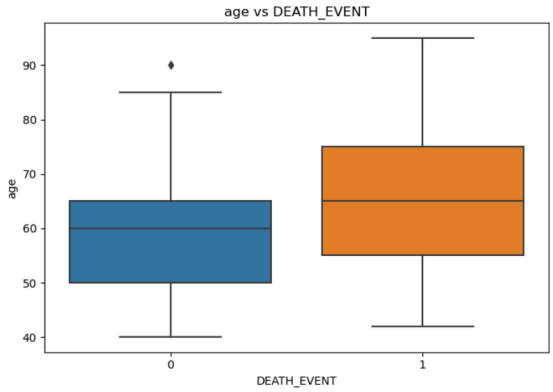
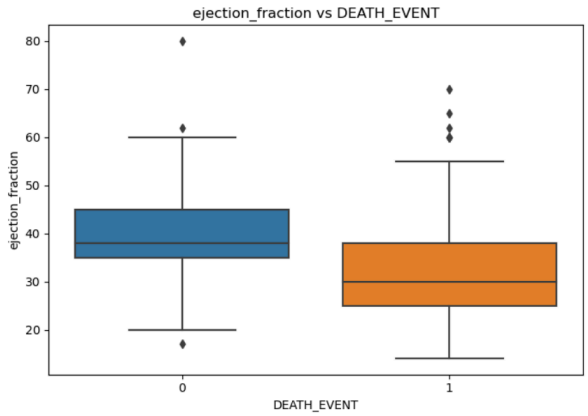
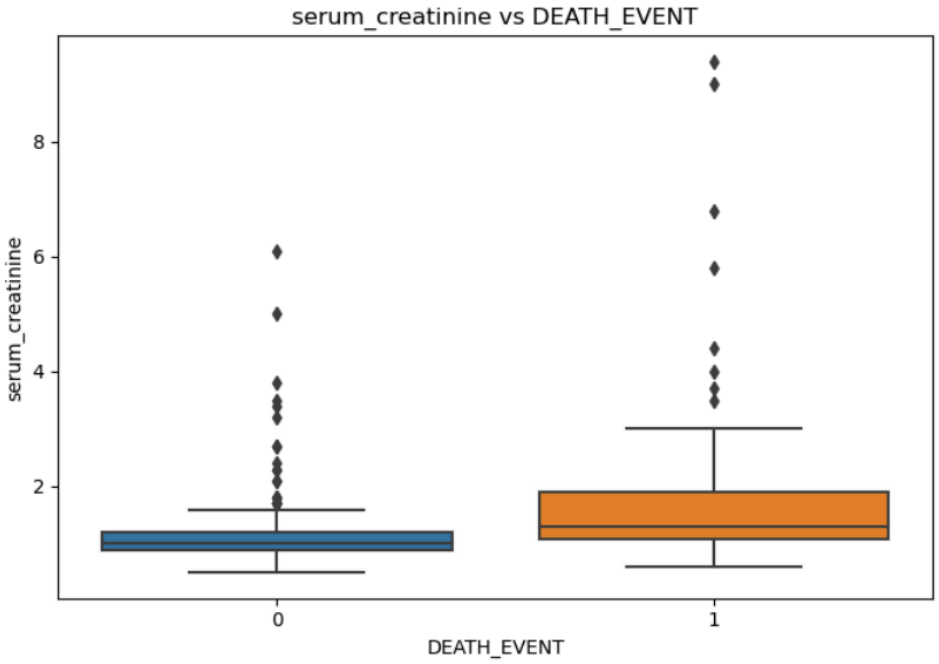
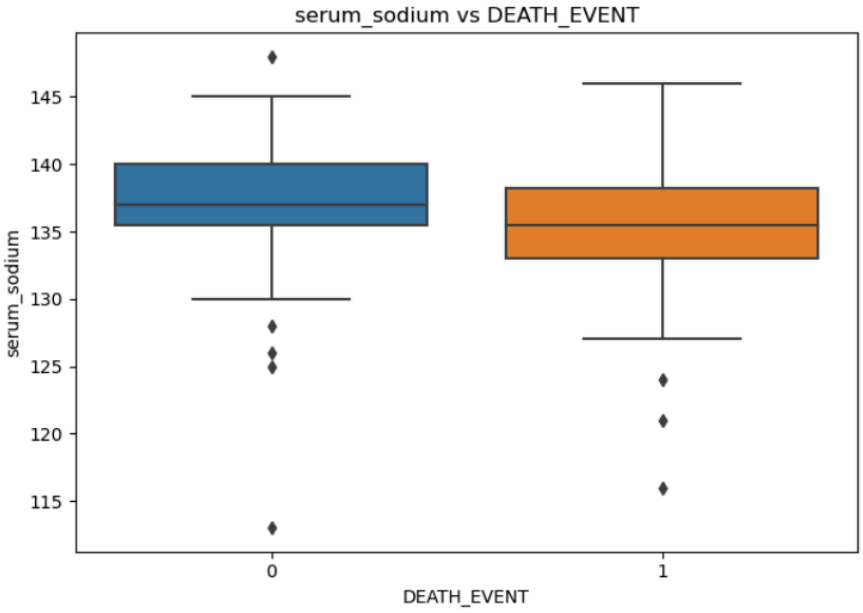


Figure 6: Boxplot of ejection fraction vs. DEATH_EVENT




serum_creatinine vs DEATH_EVENT
serum_creatinine
8
6
4
2
0
1
DEATH_EVENT


serum_sodium vs DEATH_EVENT
serum_sodium
145
140
135
130
125
120
115
0
1
DEATH_EVENT

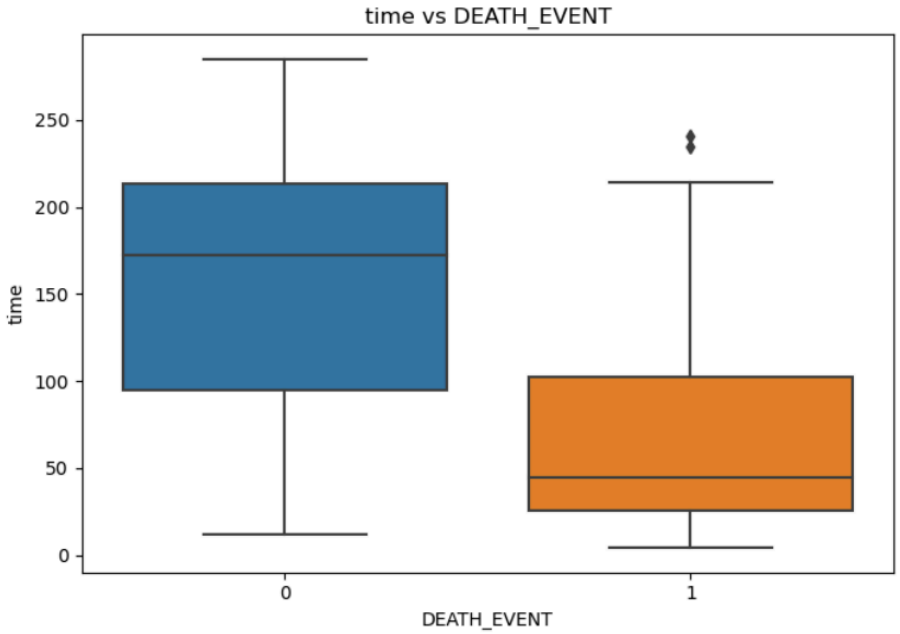
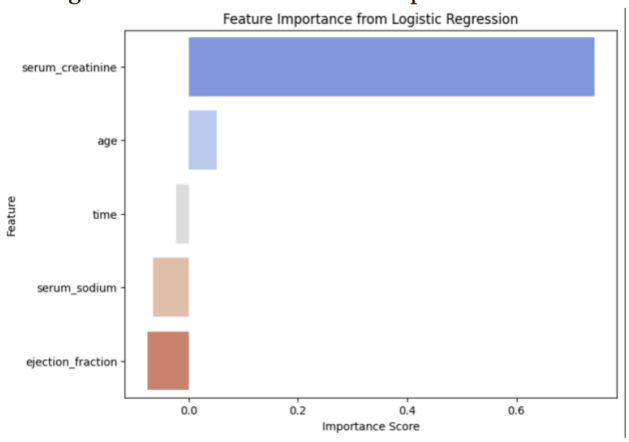


Figure 10: Bar Chart of Feature Importance Scores



Interpretation:

- **Serum Creatinine** has the strongest positive association, meaning higher values significantly increase mortality risk.
- **Time** has a minimal negative impact, indicating that patients who survived had slightly longer follow-up periods.
- **Ejection Fraction** also negatively correlates heavily, meaning lower values contribute to increased mortality risk.

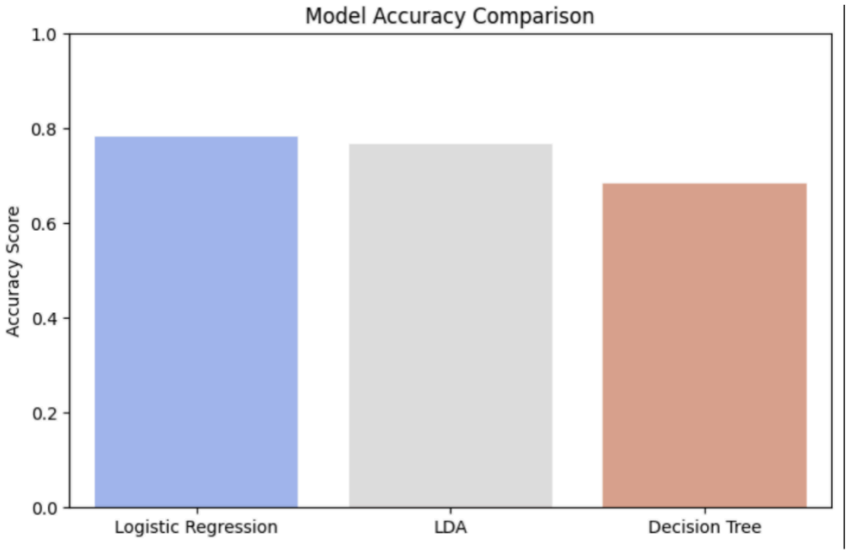


Figure 12: Confusion matrices for each model

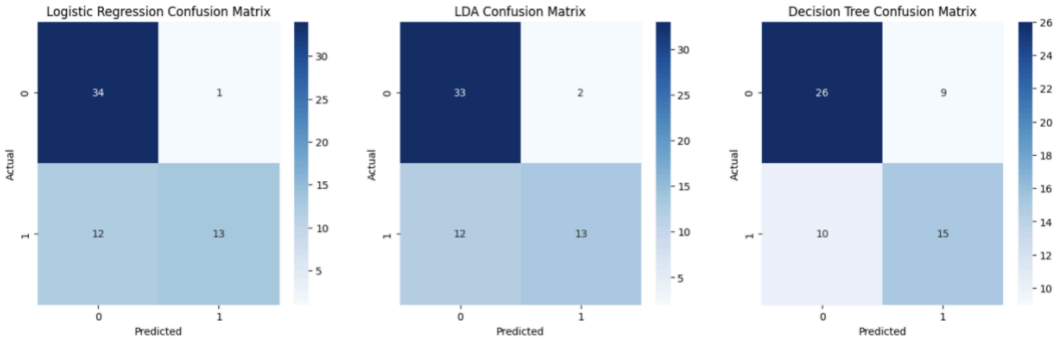
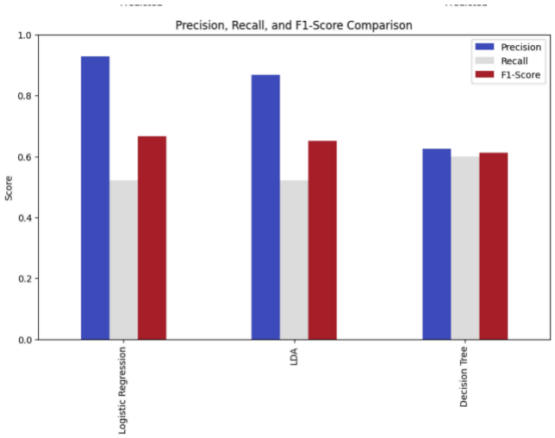


Figure 12: Bar chart of model precision, recall, and F1-scores



Interpretation:

- **Logistic Regression** achieved the highest accuracy and precision but had lower recall, meaning it predicts survival well but misses some mortality cases.
- **Linear Discriminant Analysis (LDA)**: Performed similarly to Logistic Regression with slightly lower accuracy and recall, making it a strong alternative.
- **Decision Tree**: Had the lowest accuracy and higher misclassifications, likely due to overfitting, making it the weakest model for this dataset.