VYTAUTAS MAGNUS UNIVERSITY

FACULTY OF INFORMATICS

DEPARTMENT OF APPLIED INFORMATICS

Deep Learning Systems (INF4039)

Individual assignment.

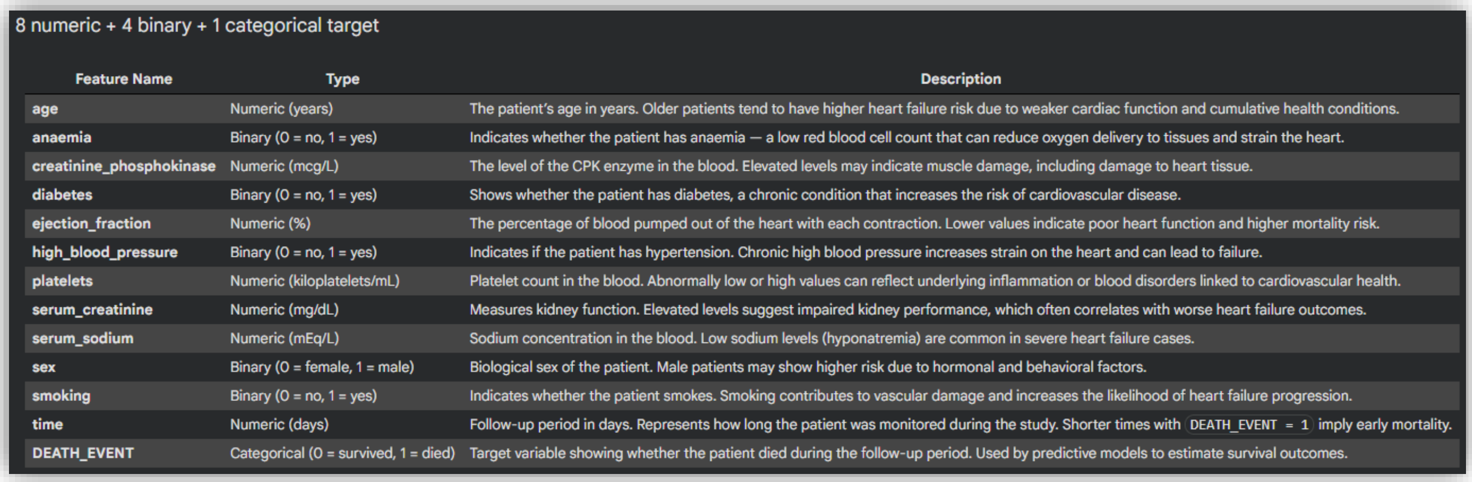
Machine Learning analysis on a chosen dataset.

Student: Rostyslav Malynovskyi **IF2200086**

Professor: Vytautas Kučinskas

Kaunas 2025

Dataset Description: Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone.



8 numeric + 4 binary + 1 categorical target

Feature Name	Type	Description
age	Numeric (years)	The patient's age in years. Older patients tend to have higher heart failure risk due to weaker cardiac function and cumulative health conditions.
anaemia	Binary (0 = no, 1 = yes)	Indicates whether the patient has anaemia — a low red blood cell count that can reduce oxygen delivery to tissues and strain the heart.
creatinine_phosphokinase	Numeric (mcg/L)	The level of the CPK enzyme in the blood. Elevated levels may indicate muscle damage, including damage to heart tissue.
diabetes	Binary (0 = no, 1 = yes)	Shows whether the patient has diabetes, a chronic condition that increases the risk of cardiovascular disease.
ejection_fraction	Numeric (%)	The percentage of blood pumped out of the heart with each contraction. Lower values indicate poor heart function and higher mortality risk.
high_blood_pressure	Binary (0 = no, 1 = yes)	Indicates if the patient has hypertension. Chronic high blood pressure increases strain on the heart and can lead to failure.
platelets	Numeric (kiloplatelets/mL)	Platelet count in the blood. Abnormally low or high values can reflect underlying inflammation or blood disorders linked to cardiovascular health.
serum_creatinine	Numeric (mg/dL)	Measures kidney function. Elevated levels suggest impaired kidney performance, which often correlates with worse heart failure outcomes.
serum_sodium	Numeric (mEq/L)	Sodium concentration in the blood. Low sodium levels (hyponatremia) are common in severe heart failure cases.
sex	Binary (0 = female, 1 = male)	Biological sex of the patient. Male patients may show higher risk due to hormonal and behavioral factors.
smoking	Binary (0 = no, 1 = yes)	Indicates whether the patient smokes. Smoking contributes to vascular damage and increases the likelihood of heart failure progression.
time	Numeric (days)	Follow-up period in days. Represents how long the patient was monitored during the study. Shorter times with `DEATH_EVENT = 1` imply early mortality.
DEATH_EVENT	Categorical (0 = survived, 1 = died)	Target variable showing whether the patient died during the follow-up period. Used by predictive models to estimate survival outcomes.

DATASET SUMMARY:

- Total samples: 299 patients
- Features: 12 clinical features + 1 categorical target (DEATH_EVENT)
- Feature types: 8 numeric (e.g., age, ejection_fraction) + 4 binary (e.g., diabetes, anaemia) + 1 categorical target
- Target variable: DEATH_EVENT (0 = survived, 1 = died)
- Classes: 2 (Survived, Died)
- Samples per class: ~231 survived (0), ~68 died (1)
- Missing values: In AGE column
- Outliers: Minimal — natural variability in clinical measurements
- Data quality: Almost clean, small data cleaning required

Main Goal: to analyze and predict mortality in heart failure patients using clinical parameters. The project aims to develop, train, and compare both traditional machine learning and neural network models to identify the most effective approach for predicting survival outcomes and understanding key health risk factors.

Data preparation and exploration

- Load and inspect the heart failure dataset, checking missing values, duplicates, and data types.
- Visualize key features and their relationship with DEATH_EVENT; compute

summary statistics and correlations to highlight the strongest risk factors (time, serum creatinine, ejection fraction, age).

Model development

- Build an interpretable baseline with a tuned Decision Tree classifier.
- Develop and compare neural models for tabular data: a Multilayer Perceptron (with scaling and regularization) and an Autoencoder-based classifier on compressed features.
- Use a common stratified train–test split to ensure fair comparison between all models.

Evaluation and analysis

- Evaluate models using Accuracy, Precision, Recall, F1-score, and ROC-AUC, with special focus on Recall and True Positives for the "died" class.
- Study the effect of key hyperparameters (tree depth, min samples, MLP regularization and architecture) on overfitting and death-case detection.
- Compare models and feature effects to identify the most informative clinical risk factors and the most reliable model for detecting high-risk patients.

Clinical risk hypothesis

Patients with low ejection fraction, high serum creatinine, and shorter follow-up time have a markedly higher probability of death, reflecting combined heart and kidney dysfunction plus early mortality.

Demographic hypothesis

Older patients, particularly those with anaemia or hypertension, show increased mortality risk, although age alone is a weaker predictor than core clinical markers like ejection fraction, creatinine, and time.

Model performance hypothesis

A well-tuned Decision Tree is expected to match or outperform more complex ensemble and neural models on this small tabular dataset, especially in recall for the "died" class, while MLP and Autoencoder models may suffer from overfitting and class imbalance.

Data Preparation

1. Check for missing values. df.isnull().sum() => as a result, we have 16 empty cells in age feature.
2. Drop these values. df = df.dropna()
3. Check for duplicates.
4. Detect and remove outliers.

```
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns

for col in numeric_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]

print("Shape after outlier removal:", df.shape)
```

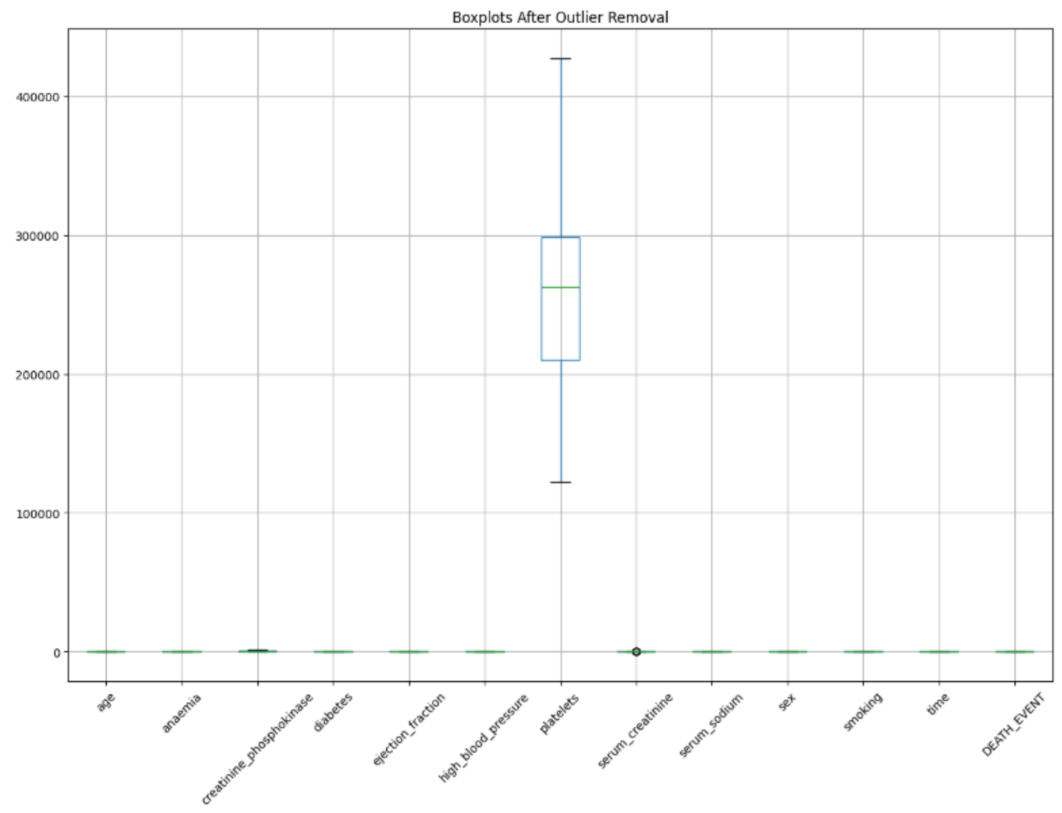
```
Shape after outlier removal: (212, 13)
```

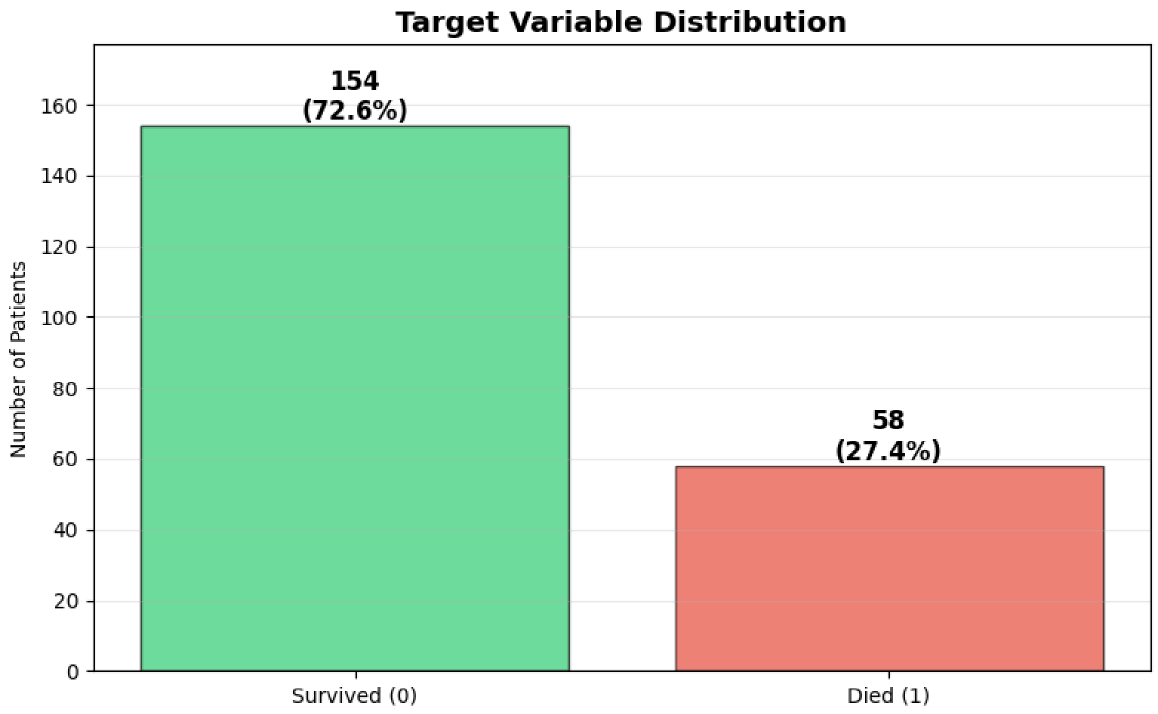
- For each numeric feature (like age, creatinine, etc.),
- Find the middle range of values.
- Remove rows where the value is too far from the middle.
- Repeat for all features.

This helped to make sure that my data doesn't have extreme values (outliers) that might mess up analysis.

Outlier removal worked.

- No extreme values remain far outside whiskers
- Platelets still has a wide distribution
- No missing values visible in plot
- Many biomedical features still show natural clinical variability

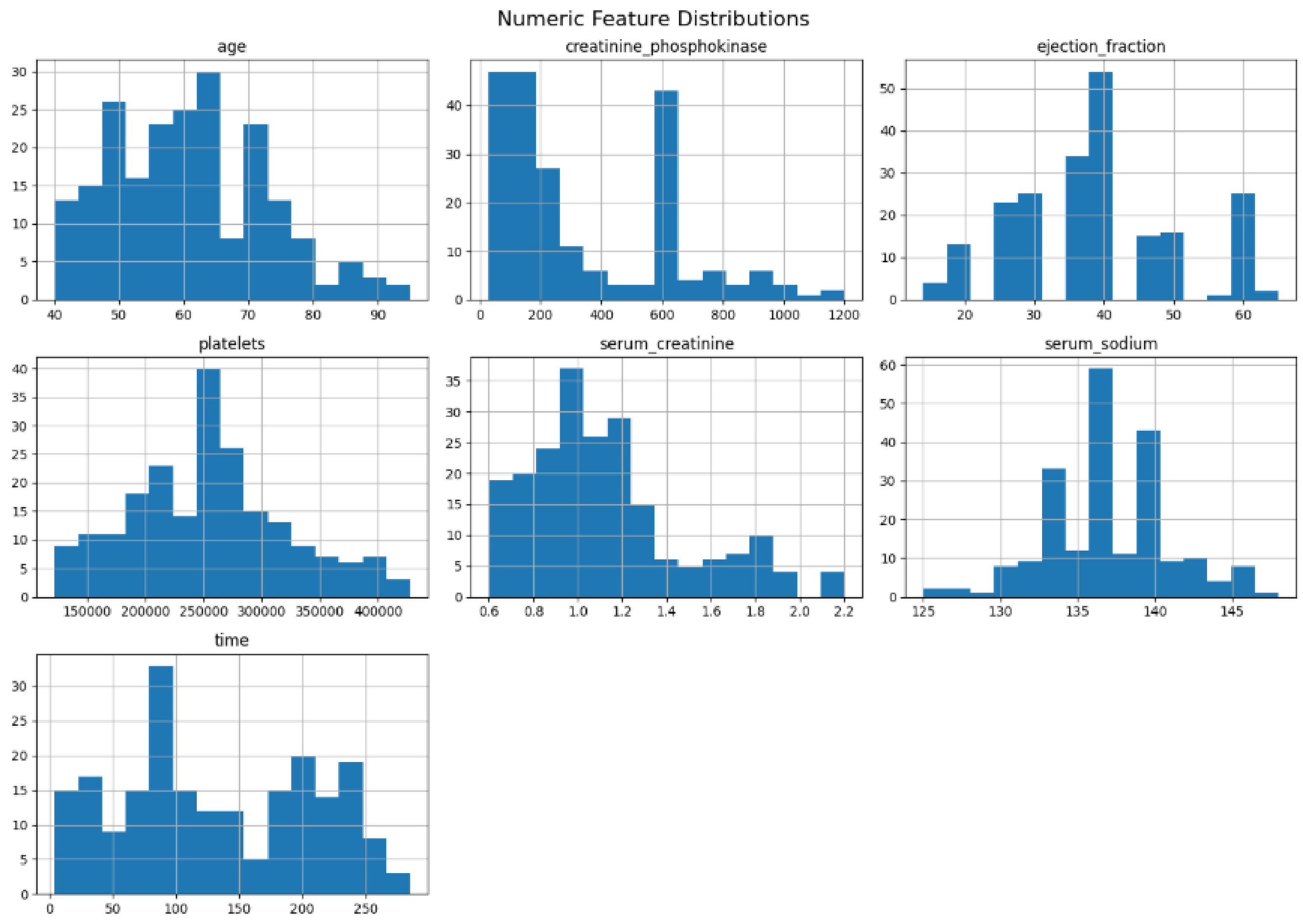




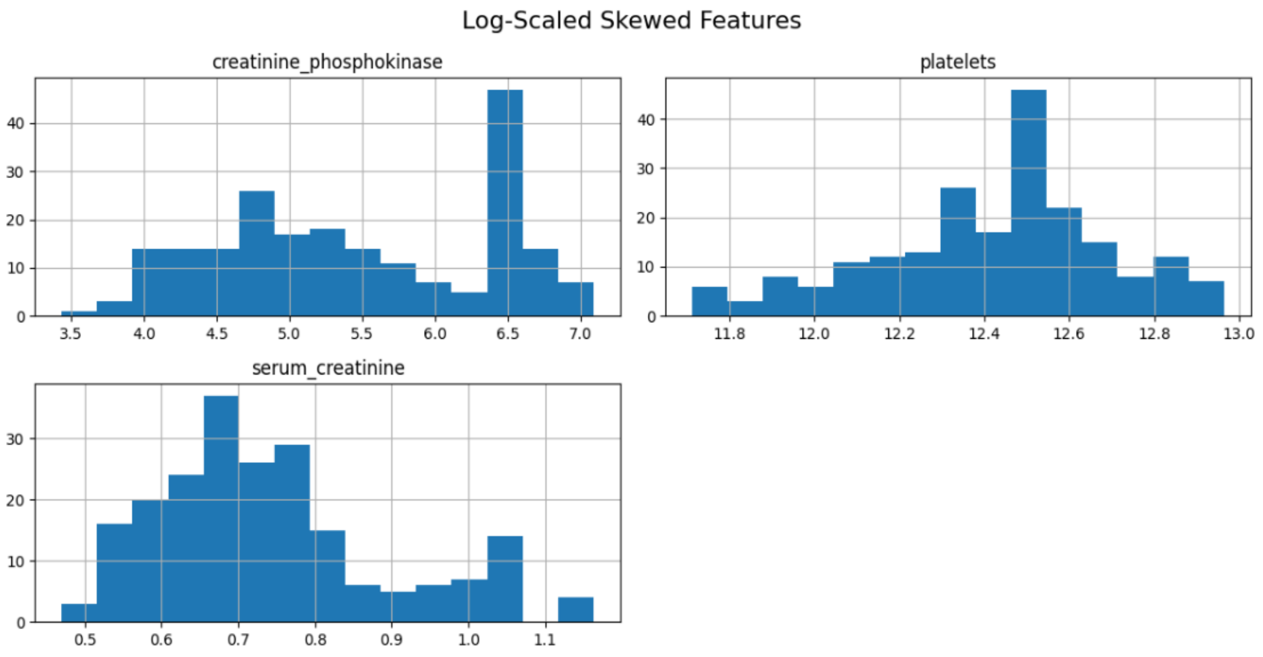
Class Imbalance Ratio: 2.66:1

Minority class (Died): 27.4%

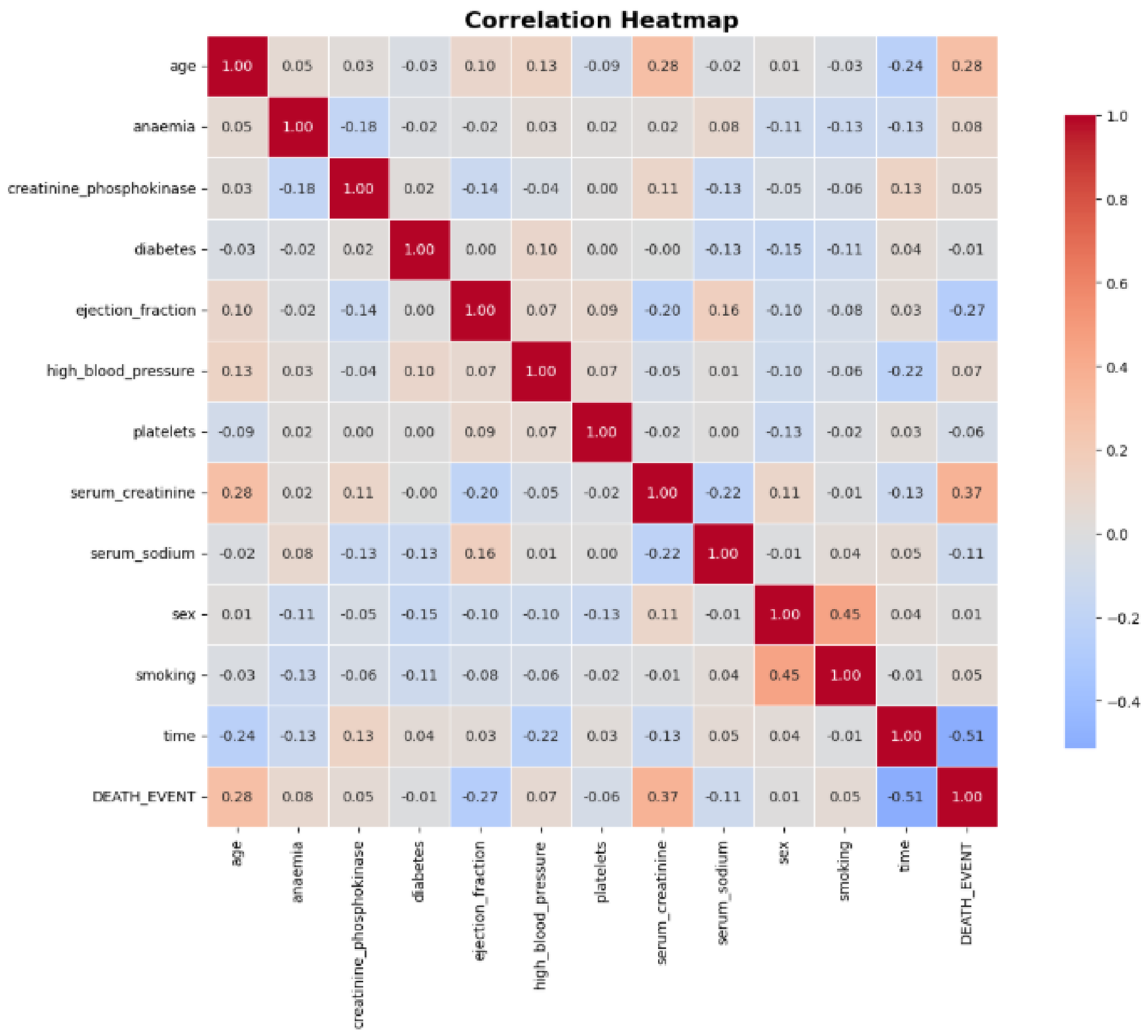
Numeric feature distributions



After log transformation



Log transformation was applied to the three most heavily skewed features: creatinine phosphokinase, platelets, and serum creatinine, because their original values had long tails and extreme outliers that could distort model predictions. After transforming, these features' distributions became much more balanced and normal-like, which helps machine learning models like logistic regression and neural nets identify patterns more effectively. This adjustment ensures that no single set of extreme values overwhelms the analysis and allows all important features to contribute fairly on a comparable scale.



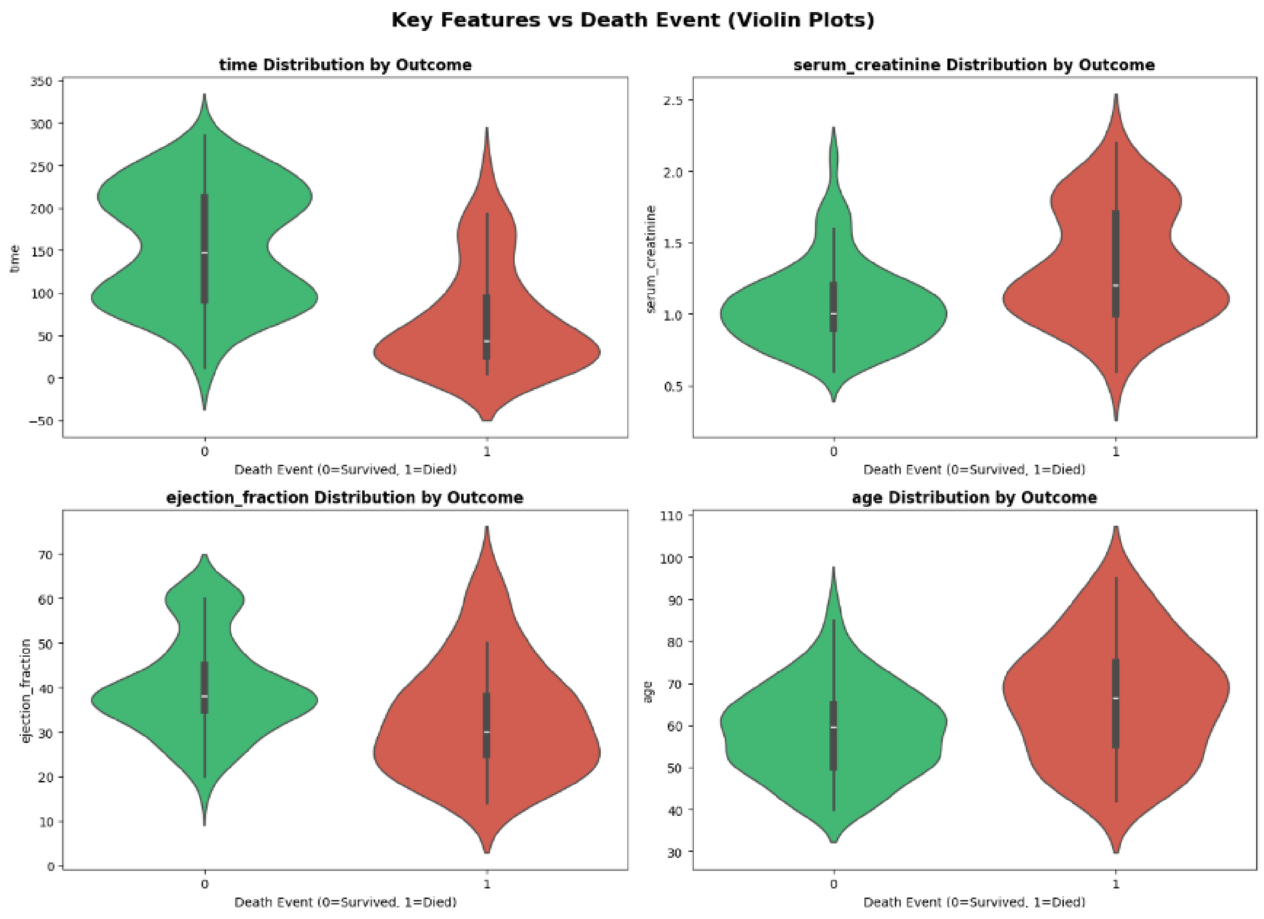
	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
age	1.00	0.05	0.03	-0.03	0.10	0.13	-0.09	0.28	-0.02	0.01	-0.03	-0.24	0.28
anaemia	0.05	1.00	-0.18	-0.02	-0.02	0.03	0.02	0.02	0.08	-0.11	-0.13	-0.13	0.08
creatinine_phosphokinase	0.03	-0.18	1.00	0.02	-0.14	-0.04	0.00	0.11	-0.13	-0.05	-0.06	0.13	0.05
diabetes	-0.03	-0.02	0.02	1.00	0.00	0.10	0.00	-0.00	-0.13	-0.15	-0.11	0.04	-0.01
ejection_fraction	0.10	-0.02	-0.14	0.00	1.00	0.07	0.09	-0.20	0.16	-0.10	-0.08	0.03	-0.27
high_blood_pressure	0.13	0.03	-0.04	0.10	0.07	1.00	0.07	-0.05	0.01	-0.10	-0.06	-0.22	0.07
platelets	-0.09	0.02	0.00	0.00	0.09	0.07	1.00	-0.02	0.00	-0.13	-0.02	0.03	-0.06
serum_creatinine	0.28	0.02	0.11	-0.00	-0.20	-0.05	-0.02	1.00	-0.22	0.11	-0.01	-0.13	0.37
serum_sodium	-0.02	0.08	-0.13	-0.13	0.16	0.01	0.00	-0.22	1.00	-0.01	0.04	0.05	-0.11
sex	0.01	-0.11	-0.05	-0.15	-0.10	-0.10	-0.13	0.11	-0.01	1.00	0.45	0.04	0.01
smoking	-0.03	-0.13	-0.06	-0.11	-0.08	-0.06	-0.02	-0.01	0.04	0.45	1.00	-0.01	0.05
time	-0.24	-0.13	0.13	0.04	0.03	-0.22	0.03	-0.13	0.05	0.04	-0.01	1.00	-0.51
DEATH_EVENT	0.28	0.08	0.05	-0.01	-0.27	0.07	-0.06	0.37	-0.11	0.01	0.05	-0.51	1.00

Strong predictors (will be important for the model):

- time (-0.51) - the longer the observation period, the lower the probability of death
- serum_creatinine (0.37) - the higher, the higher the risk
- age (0.28) - the older you are, the higher the risk
- ejection_fraction (-0.27) - the higher (better heart function), the lower the risk

Weak predictors (almost no impact):

- diabetes, sex, smoking, anemia - very weak association with the target variable



Violin plot is helpful because, in one picture, you can see not only the average and range but also whether the data is clumped together, has more than one peak, or is spread out differently in different groups, making it easier to compare patterns.

Time

Conclusion: The most important indicator! The longer a patient is observed, the greater the chance of survival.

Serum Creatinine

Conclusion: High creatinine = bad sign (kidney problems are associated with mortality).

Ejection Fraction

Conclusion: Low ejection fraction = high risk of death (the heart is pumping blood poorly).

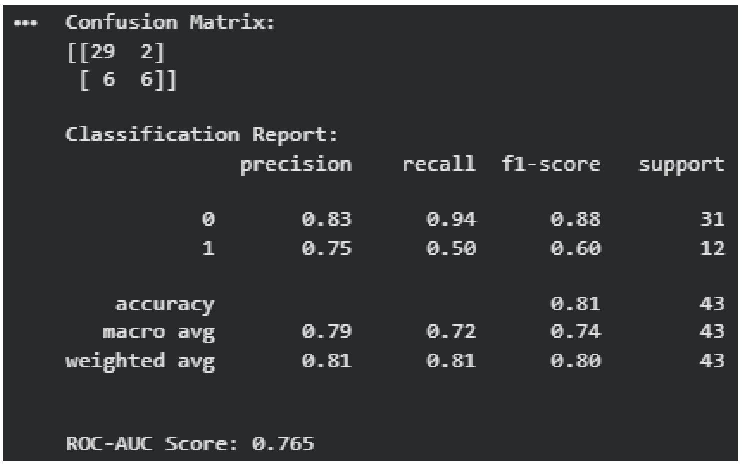
Age

Conclusion: Older age = higher risk, but the difference is not as strong as with other features.

MACHINE LEARNING IMPLEMENTATION

1. ***Decision Tree with max_depth = 5***

Metric	Explanation	Outcome
Confusion matrix	Shows prediction counts for each class; your model correctly identified 29 survivors and 6 deaths, but missed 6 deaths (false negatives) and gave 2 false alarms for death.	See matrix
Precision	Measures how many predicted deaths were correct; "died" class precision is 0.75, meaning 75% of death predictions matched true deaths.	0.75 ("died")
Recall	Measures how many true deaths were caught; recall for "died" is 0.50, so the model found half of the actual deaths.	0.50 ("died")
F1-score	The F1-score is a single number that measures how well the model balances catching actual deaths (recall) and not raising false alarms (precision). For deaths, F1-score is 0.60, which means model is moderately effective at correctly finding true deaths without making too many mistakes	0.60 ("died")
Support	This is just a count of how many actual examples there are for each result. In a test set, there were 31 people who survived and 12 who died.	31 / 12
Accuracy	Accuracy tells how many predictions the model got right out of all the cases. In my results, 81% of predictions were correct, but this number can be misleading if one group (like survivors) is much bigger.	0.81
Macro avg	This averages the model's performance on deaths and survivors as if both were equally important, even if the numbers in each group are different. My macro F1 average is 0.74, so overall performance is decent on both groups.	0.74
Weighted avg	The average takes into account how many survivors and deaths there are (so more weight goes to the bigger group). My weighted F1 is 0.80, reflecting strong prediction on the larger survivor group.	0.80
ROC-AUC score	This measures how well the model separates people who died from those who survived (1 means perfect, 0.5 is as good as guessing). In my results, score of 0.765 means the model is fairly good at telling the two apart.	0.765

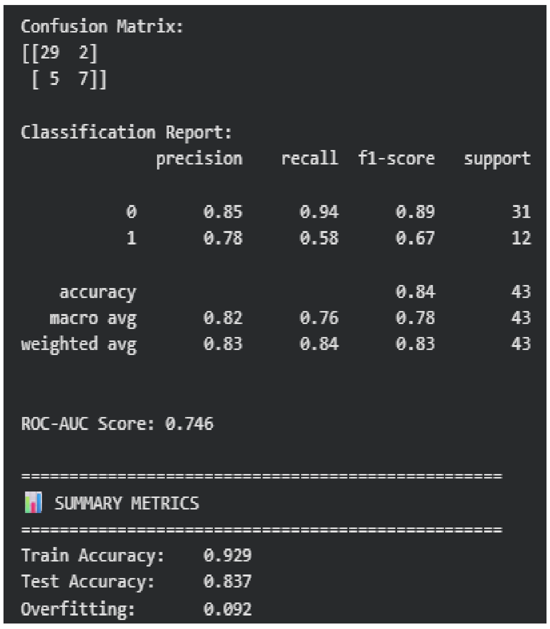


```
••• Confusion Matrix:
    [[29  2]
     [ 6  6]]

    Classification Report:
                  precision    recall  f1-score   support

               0       0.83      0.94      0.88        31
               1       0.75      0.50      0.60        12

        accuracy                           0.81        43
       macro avg       0.79      0.72      0.74        43
    weighted avg       0.81      0.81      0.80        43


    ROC-AUC Score: 0.765
```

2. ***<u>Decision Tree with max_depth = 8</u>***

- ❖ max_depth increased to 8 (was 5)
- ❖ min_samples_split set to 5 (was 10)
- ❖ min_samples_leaf set to 3 (was 5)

Metric	Result
Precision (died)	0.78 (improved)
Recall (died)	0.58 (improved)
F1-score (died)	0.67 (improved)
Support (died)	12
Accuracy	0.84 (improved)
Macro avg	0.78 (improved)
Weighted avg	0.83 (improved)
ROC-AUC score (died)	0.746
Train accuracy	0.93
Test accuracy	0.84
Overfitting	0.09



```
Confusion Matrix:
[[29  2]
 [ 5  7]]

Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.94      0.89        31
           1       0.78      0.58      0.67        12

    accuracy                           0.84        43
   macro avg       0.82      0.76      0.78        43
weighted avg       0.83      0.84      0.83        43


ROC-AUC Score: 0.746

==================================================
📊 SUMMARY METRICS
==================================================
Train Accuracy:    0.929
Test Accuracy:     0.837
Overfitting:       0.092
```

Correct predictions: 36/43 (83.7% accuracy)

- Survived → Survived: 29
- Died → Died: 7

Errors: 7/43

- Survived → Died: 2 (False Positive)
- Died → Survived: 5 (False Negative - still there is a problem)

Test Accuracy - This is the percentage of correct predictions your model makes on the test dataset—the data it never saw during training. High test accuracy means your model is good at generalizing and works well for new, unseen cases, which is what matters most in real-world use. *Train accuracy (0.93) is a bit higher than test accuracy (0.84), so model learned well but did not simply memorize the data.*

Train Accuracy - This is the percentage of correct predictions on the training dataset—the data your model learned from. It shows how well your model memorized or fit patterns in the training data; very high train accuracy is usually expected.

Overfitting - is when your model performs much better on the training data than on new, unseen test data. It means the model may have learned details specific to the training set instead of general rules; the difference between train and test accuracy helps you see if overfitting is happening. T*he overfitting gap (0.09) is small, suggesting that model generalizes well and isn't just remembering the training set.*

Conclusion

With the deeper decision tree, all the important metrics have improved compared to the previous model. The F1-score for deaths increased, and the model now finds more real deaths, though it still misses some (5 cases). Overfitting is low, meaning the tree generalizes fairly well, and overall reliability—especially for predicting death—has gotten better.

3. ***Random Forest***

Results are the same as for Decision Tree with depth 5.

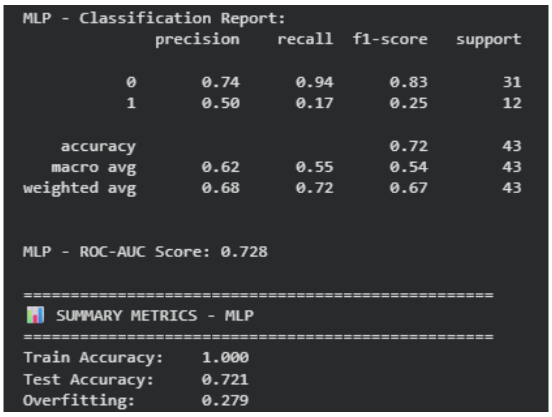
Model	**True Positive**	**False Negative**	**Recall (died)**
Decision Tree (depth=5)	6	6	50%
Decision Tree (depth=8)	7	5	58.3%
Random Forest	6	6	50%

Decision Tree with depth=8 turned out to be better than Random Forest. Why?

1. Random Forest is too conservative with these parameters.
2. Small dataset (212 samples) - Random Forest doesn't always help.
3. A single deep tree sometimes works better on small data.

4. MULTILAYER PERCEPTRON (MLP)

Metric	Result
Precision (died)	0.50
Recall (died)	0.17
F1-score (died)	0.25
Support (died)	12
Accuracy	0.72
Macro avg	0.56
Weighted avg	0.62
ROC-AUC score (died)	0.73
Train accuracy	1
Test accuracy	0.72
Overfitting	0.28



```
MLP - Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.94      0.83        31
           1       0.50      0.17      0.25        12

    accuracy                           0.72        43
   macro avg       0.62      0.55      0.54        43
weighted avg       0.68      0.72      0.67        43


MLP - ROC-AUC Score: 0.728

==================================================
📊 SUMMARY METRICS - MLP
==================================================
Train Accuracy:    1.000
Test Accuracy:     0.721
Overfitting:       0.279
```

Conclusions

- The MLP model has very low recall for deaths (0.17), meaning it misses most high-risk patients.
- **Train accuracy is perfect (1.00), but test accuracy drops sharply (0.72), which**

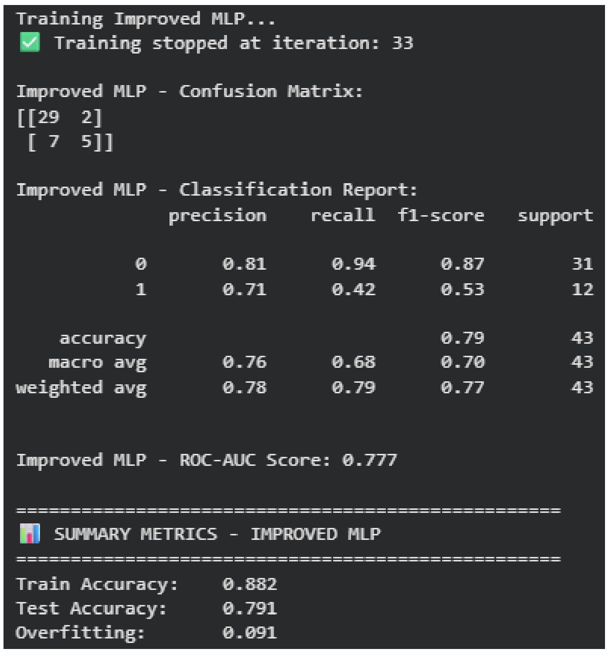
means the model is overfitting - it memorizes training data but struggles with new cases.

- F1-score for deaths is only 0.25, much lower than previous models.
- Only 2 actual deaths are detected (Died → Died), and 10 deaths are missed (Died → Survived), making it unreliable for finding high-risk patients in your dataset.
- Overfitting is high (0.28), so the model's predictions on new data are not trustworthy.

5. ***IMPROVED MLP with Regularization***

Parameter	Value	Description
hidden_layer_sizes	(64, 32)	The neural network has two middle layers that do the "thinking", with 64 and 32 small processing units ("neurons") in each layer.
activation	relu	This adds flexibility, letting the network spot complex patterns, not just straight lines.
solver	adam	This is the method the model uses to learn and adjust itself step by step.
alpha	0.01	This helps prevent the model from memorizing the training data too closely, so it works better on new data.
batch_size	16	The network looks at this many cases at once before it updates itself (here, 16 at a time).
learning_rate	adaptive	The "speed" at which the model learns; it slows down if the model isn't getting better, so it won't miss important details.
learning_rate_init	0.001	The "springboard" speed for learning when training begins.
max_iter	500	The maximum times the network is allowed to look through the data before stopping.
early_stopping	True	Stops training if no improvement on validation set
validation_fraction	0.2	20% of training data is used for validation
n_iter_no_change	20	Stop early if no improvement after 20 epochs
random_state	42	Makes results reproducible by fixing the random seed
verbose	False	Turns off progress output during training

Metric	Result
Precision (died)	0.71
Recall (died)	0.42
F1-score (died)	0.53
Support (died)	12
Accuracy	0.79
Macro avg	0.68
Weighted avg	0.77
ROC-AUC score (died)	0.78
Train accuracy	0.88
Test accuracy	0.79
Overfitting	0.09



```
Training Improved MLP...
✅ Training stopped at iteration: 33

Improved MLP - Confusion Matrix:
[[29  2]
 [ 7  5]]

Improved MLP - Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.94      0.87        31
           1       0.71      0.42      0.53        12

    accuracy                           0.79        43
   macro avg       0.76      0.68      0.70        43
weighted avg       0.78      0.79      0.77        43


Improved MLP - ROC-AUC Score: 0.777

==================================================
📊 SUMMARY METRICS - IMPROVED MLP
==================================================
Train Accuracy:    0.882
Test Accuracy:     0.791
Overfitting:       0.091
```

Conclusion

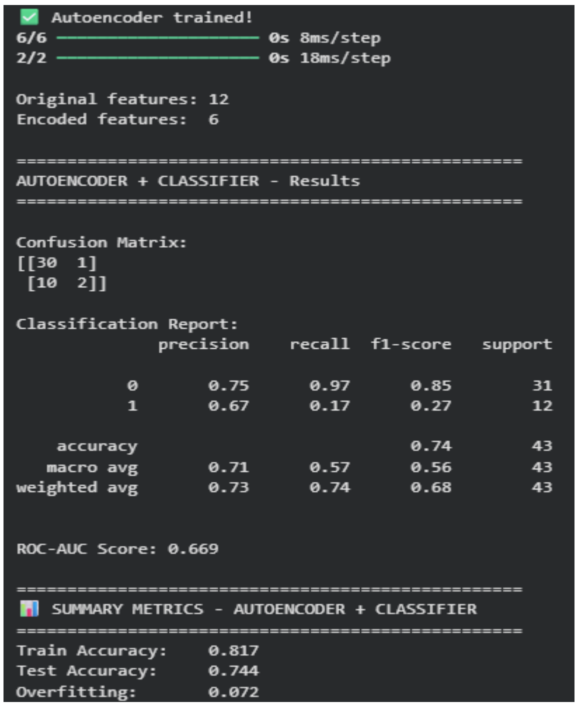
The Improved MLP shows better generalization compared to the original MLP, increasing test accuracy from 0.721 to 0.791 while significantly reducing overfitting. It also greatly improved precision for predicting deaths, meaning when it predicts a patient will die, it is correct most of the time. However, the recall for deaths remains relatively low, so it still misses many actual death cases. Overall, the model is now more balanced and performs closer to the Decision Tree, though the Decision Tree still detects more true death cases. The Improved MLP also achieves a strong ROC-AUC score, showing good ability to rank patients by risk. In summary, the Improved MLP is more reliable than the previous version, but the Decision Tree remains the best model for correctly identifying patients who are likely to die.

6. <u>Autoencoder</u>

Parameter	Value	Description
input_dim	X_train_scaled.shape	Number of features going into the autoencoder
encoding_dim	6	Size of compressed hidden layer (“bottleneck”); outputs 6 features
activation	relu (encoding), sigmoid (decoding)	“relu” helps learn patterns; “sigmoid” outputs values between 0 and 1

optimizer	adam	Learning method that adjusts weights efficiently
loss	mse	Model tries to minimize mean squared error (difference)
epochs	50	Model scans through data up to 50 times during training
batch_size	16	Number of samples processed at once before updating weights
validation_split	0.1	Reserves 10% of training data to check for overfitting during training
shuffle	True	Randomly mixes data before each training cycle

Metric	Result
Precision (died)	0.67
Recall (died)	0.17
F1-score (died)	0.27
Support (died)	12
Accuracy	0.74
Macro avg	0.56
Weighted avg	0.68
ROC-AUC score (died)	0.67
Train Accuracy	0.82
Test Accuracy	0.74
Overfitting	0.08



```
✅ Autoencoder trained!
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step

Original features: 12
Encoded features:  6

============================================================
AUTOENCODER + CLASSIFIER - Results
============================================================

Confusion Matrix:
[[30  1]
 [10  2]]

Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.97      0.85        31
           1       0.67      0.17      0.27        12

    accuracy                           0.74        43
   macro avg       0.71      0.57      0.56        43
weighted avg       0.73      0.74      0.68        43


ROC-AUC Score: 0.669

============================================================
📊 SUMMARY METRICS - AUTOENCODER + CLASSIFIER
============================================================
Train Accuracy:    0.817
Test Accuracy:     0.744
Overfitting:       0.072
```

Conclusion

Compared to both the basic MLP and the MLP with regularization, the Autoencoder + Classifier approach compresses feature information but does **not improve recall or F1-score for "died" cases**—these metrics (recall = 0.17, F1 = 0.27) are still very low, meaning most deaths are missed. Test accuracy and AUC are similar or slightly lower than MLP models, and overall reliability for predicting deaths remains poor. Feature compression did not solve the main problem, and regularized MLPs still outperform autoencoder-based models for critical death risk detection in this dataset.

7. ***FINAL COMPARISON***

Model	Train Accuracy	Test Accuracy	Overfitting	Precision (Died)	Recall (Died)	F1 Score (Died)	ROC-AUC	True Positives	False Negatives
Decision Tree (depth=5)	0.90	0.81	0.09	0.75	0.50	0.60	0.77	6	6
Decision Tree (depth=8)	0.93	0.84	0.09	0.78	0.58	0.67	0.75	7	5
Random Forest	0.99	0.81	0.18	0.75	0.50	0.60	0.80	6	6
MLP	1.00	0.72	0.28	0.50	0.17	0.25	0.73	2	10
MLP (regularization)	0.88	0.79	0.09	0.71	0.42	0.53	0.78	5	7
Autoencoder	0.82	0.74	0.08	0.67	0.17	0.27	0.67	2	10

Conclusions

This table shows how well each model predicts heart failure death cases. The most important columns - **Recall (Died)** and **True Positives** - show how many real death cases the model correctly finds. The **Decision Tree (depth=8)** performed best, identifying 7 true deaths and missing only 5, making it the most clinically reliable for identifying high-risk patients. The **Random Forest** has a strong ROC-AUC, meaning it separates classes well, but its recall is still too low for medical safety. The **MLP** and **Autoencoder** models suffer from overfitting (high train but low test accuracy) and fail to detect most deaths, making them much less useful for risk prediction. In summary, **Decision Tree (depth=8)** is the most effective model here - it finds the most high-risk cases with low overfitting, while more complex neural models miss too many deaths to be practical.