Heart Failure Risk Prediction Project Report

University of Luxembourg

Submitted by: Faisal Sediqi, Mohamed Lemouissi & Miguel Soares Barbosa

Course: Data Science

Table of Contents

1. Introduction

Heart failure is one of the major causes of death worldwide. In this project, we worked with a clinical dataset containing health-related measurements from 299 patients who had experienced heart failure. Our objective was to understand which clinical features are most predictive of patient survival and to build models that can accurately predict the risk of death.

This project simulates a real-world healthcare task where early detection of patient risk can lead to better treatment planning and potentially save lives.

2. Dataset Overview

The dataset contains 13 medical features and a target column "DEATH_EVENT", which tells us whether a patient died during follow-up. Important features include:

- Age

- Ejection fraction (heart function)

- Serum creatinine (kidney function)

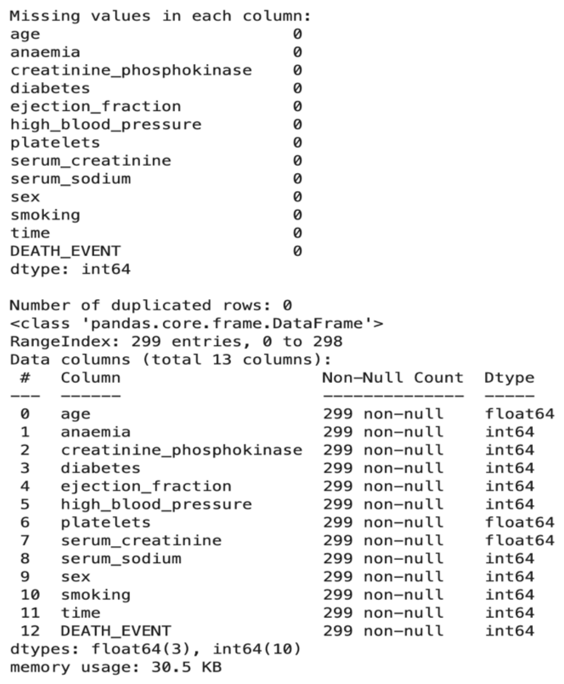
- High blood pressure, anaemia, and diabetes

- Time of follow-up

Before we start visualizing or building models, it's important to check if the data is clean. We need to find missing values, duplicated rows, and check the data types.

2. Data Preparation and Cleaning

Before analysis, we verified the dataset quality:



```
Missing values in each column:
age                         0
anaemia                     0
creatinine_phosphokinase    0
diabetes                    0
ejection_fraction           0
high_blood_pressure         0
platelets                   0
serum_creatinine            0
serum_sodium                0
sex                         0
smoking                     0
time                        0
DEATH_EVENT                 0
dtype: int64

Number of duplicated rows: 0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age                       299 non-null    float64
 1   anaemia                   299 non-null    int64
 2   creatinine_phosphokinase  299 non-null    int64
 3   diabetes                  299 non-null    int64
 4   ejection_fraction         299 non-null    int64
 5   high_blood_pressure       299 non-null    int64
 6   platelets                 299 non-null    float64
 7   serum_creatinine          299 non-null    float64
 8   serum_sodium              299 non-null    int64
 9   sex                       299 non-null    int64
 10  smoking                   299 non-null    int64
 11  time                      299 non-null    int64
 12  DEATH_EVENT               299 non-null    int64
dtypes: float64(3), int64(10)
memory usage: 30.5 KB
```

Since there were no missing values or duplicate rows, we didn’t need to do any cleaning. However, it’s always important to verify these before analysis.

Improvement Suggestion: If this were a larger or messier dataset, additional steps like outlier removal, imputation, or scaling might be needed.

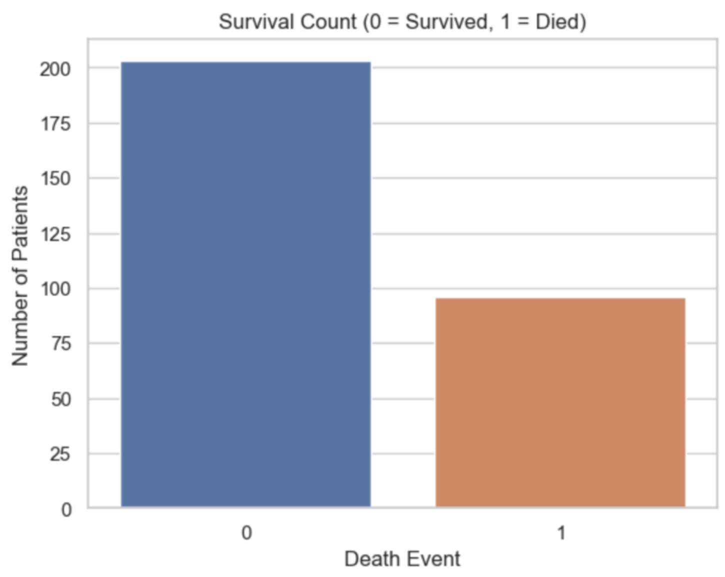
3. Exploratory Data Analysis (EDA):

We visualized the dataset to understand key patterns:

3.1 Survival Class Distribution:

Before jumping into the details of the features, we wanted to see how many patients in the dataset actually survived versus how many passed away. This helps us understand what kind of problem we're working with — is it balanced or are we trying to predict something that happens rarely?

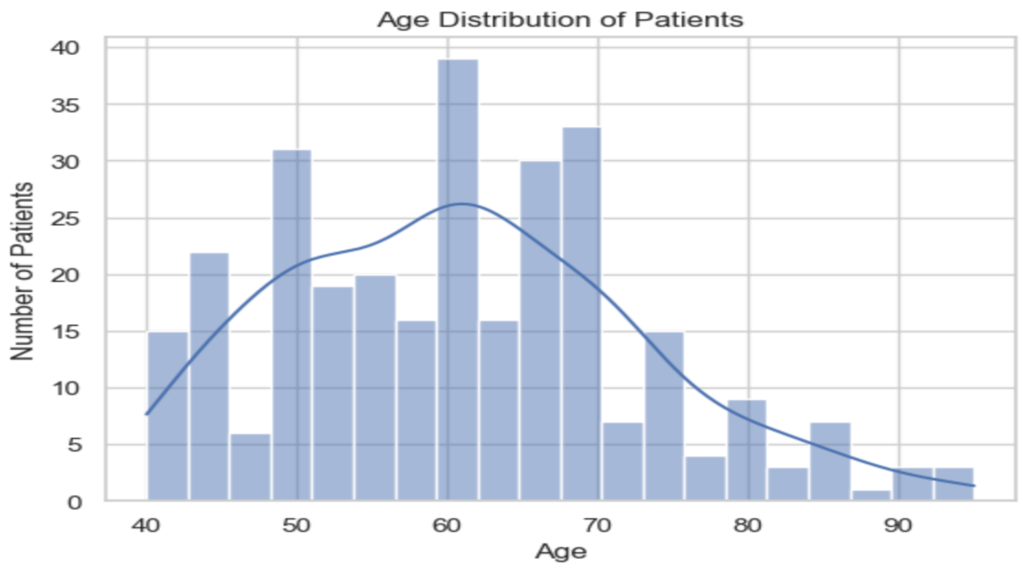
We created a simple bar chart showing the number of patients who survived (0) and those who died (1):



From the chart, we can see that the majority of patients survived, while a smaller portion died. This tells us that the dataset is **slightly imbalanced**, but not severely — which means we can continue without needing to apply special balancing techniques (like oversampling) at this stage.

It’s important to check if our target classes are balanced. If one class dominates, a model could give high accuracy by just predicting that class — even if it's not actually learning. In our case, the imbalance is mild and manageable.

3.2 Age Distribution of Patients: To get a better sense of the age spread in our dataset, we plotted a histogram of all patient ages.

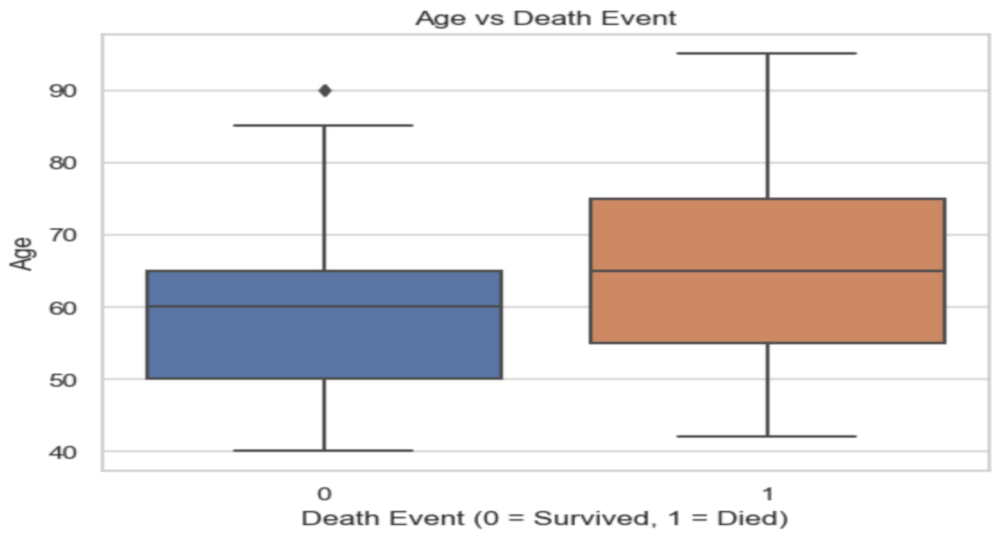


The chart shows that **most patients are between 50 and 70 years old**, which makes sense since heart failure is more common in older adults. There are only a few patients under 45 or over 85.

The line on the chart (the KDE curve) gives a smooth view of how the ages are distributed overall.

If the dataset had more young patients, we could explore whether younger patients survive at higher rates. But in our case, the data is mostly focused on middle-aged to older adults, which fits the real-world context of heart disease.

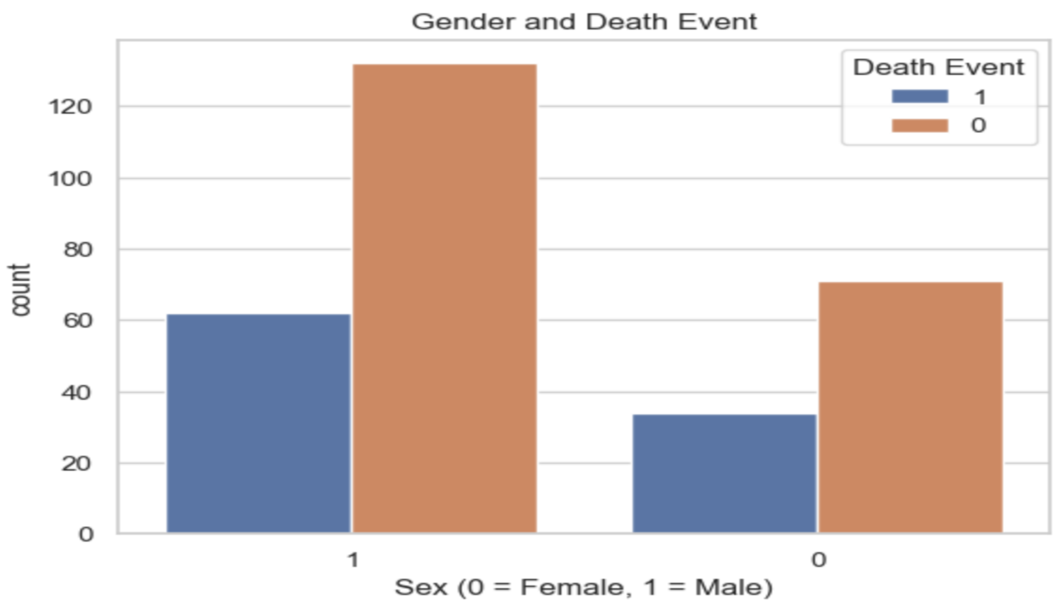
3.3 Relationship Between Age and Survival: To explore how age is related to survival, we used a boxplot to compare the age distribution of patients who **survived (0)** and those who **died (1).**



From the plot, it's clear that **patients who died tend to be older** than those who survived. The median age of those who passed away is higher, and the age range is slightly wider too.

This suggests that **age could be an important predictor** in our model. In medical terms, it's also logical — older patients are generally at higher risk in heart failure cases.

3.4 Gender and Survival Outcome**:** Next, we looked at whether gender (male or female) had any noticeable relationship with survival.



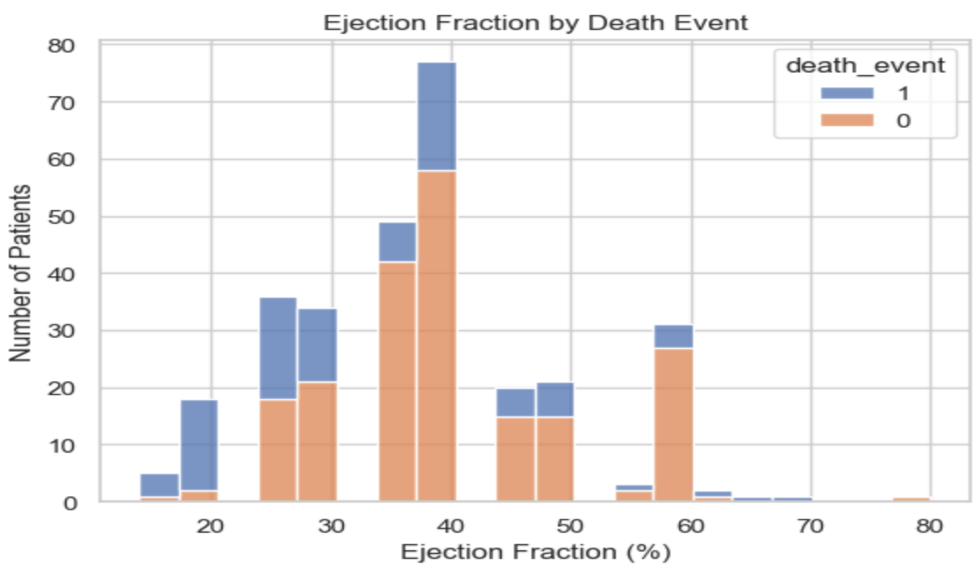
In the bar chart above, we compared the number of male (1) and female (0) patients, split by whether they survived (0) or died (1).

The result? **There isn’t a major difference** between genders when it comes to death rate in this dataset. While there are more male patients overall, the proportion of those who died compared to those who survived is fairly similar for both males and females.

We often assume gender might influence health outcomes, but in this case, it doesn’t seem to be a strong predictor. That’s useful to know — it means gender might not add much value to our model on its own.

If we had a larger dataset or more detailed gender-related features (like hormone levels, lifestyle data, or age-gender interaction), we might uncover more subtle patterns.

3.5 Ejection Fraction and Survival: We used a stacked histogram to explore how ejection fraction (a measure of heart function) relates to survival. Ejection fraction is the percentage of blood the heart pumps out with each beat — and it's a critical value in heart failure diagnosis.

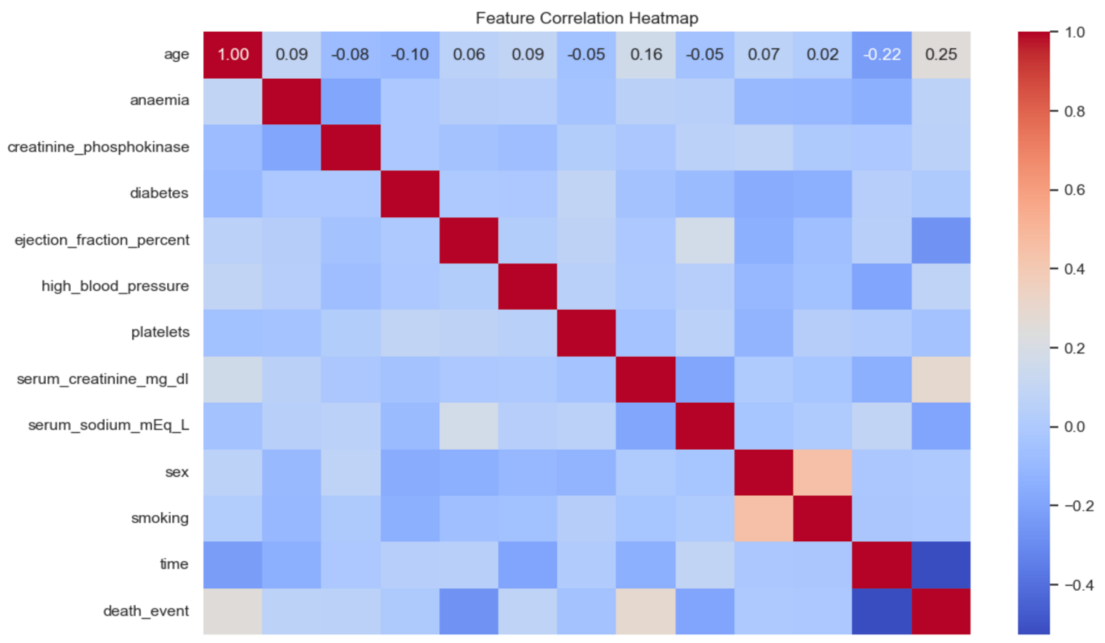


In the chart, the patients who died (1) are mostly concentrated in the lower ejection fraction range — especially below 40%. Meanwhile, patients who survived (0) tend to have higher values, often above 40–45%.

This trend matches real medical knowledge: a low ejection fraction means the heart isn't pumping blood efficiently, which increases the risk of death. It confirms that ejection fraction is a strong predictor in this dataset.

- Patients with **low heart function are at higher risk**
- This variable will likely be **very important for prediction** in our machine learning model.

4. Feature Correlation Heatmap: To get a quick overview of how all the features relate to each other — especially to the target death_event — we used a correlation heatmap.



This chart shows us how strongly (or weakly) features are linearly related, using values from -1 to 1. Positive values mean that as one feature increases, the other tends to increase too. Negative values mean the opposite.

- Death is positively correlated with age and serum_creatinine
  → This suggests that older patients and those with kidney issues (high creatinine) are more at risk.
- Death is negatively correlated with ejection_fraction and serum_sodium
  → These are signs of stronger heart and body function, so it makes sense they're linked to survival.
- Other features like smoking, anaemia, or sex don't seem to have strong correlations. That doesn't mean they're useless — just that they don't have a strong direct linear relationship with death in this dataset.

5. Preparing the Data for Modeling: Before training any models, we need to prepare our dataset. This includes two main tasks:

1. Separating the features (X) from the target variable (death_event)
2. Splitting the data into a training set and a testing set

We used an 80/20 split — meaning 80% of the data will be used to train the model, and 20% will be used to test how well the model performs on unseen data. This is a common practice that helps ensure the model can generalize to new cases.

We also made sure that the death_event column is stored as an integer, since some models require the target to be in numeric format.

Without splitting the data, we wouldn't be able to tell whether the model is actually learning patterns — or just memorizing the training examples. Testing on separate data helps us evaluate model accuracy in a fair and realistic way.

Now that the data is prepared, we'll train and compare different machine learning models to find the one that best predicts patient survival.

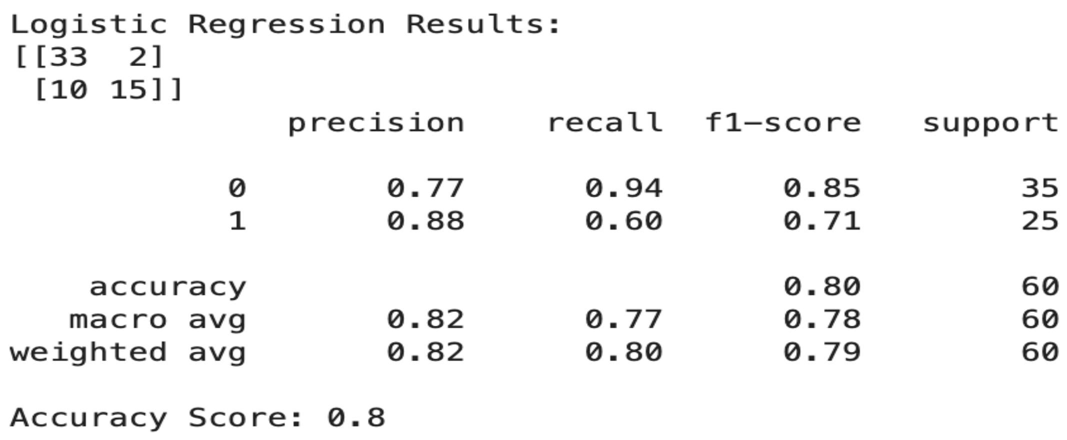
We'll start with:

- Logistic Regression
- Random Forest Classifier
- K-Nearest Neighbors (KNN)

6. Logistic Regression Model: We started by training a Logistic Regression model, which is a simple and interpretable classification algorithm. It's often used as a baseline in machine learning projects.

After fitting the model on the training data, we used it to predict the outcomes for the test set. Then we evaluated the results using:

- Confusion Matrix
- Precision, Recall, F1-score
- Overall Accuracy



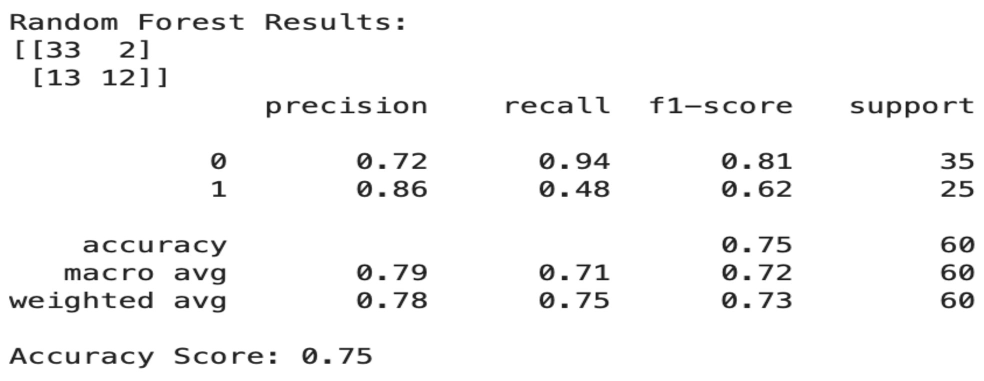
```
Logistic Regression Results:
[[33  2]
 [10 15]]
              precision    recall  f1-score   support

           0       0.77      0.94      0.85        35
           1       0.88      0.60      0.71        25

    accuracy                           0.80        60
   macro avg       0.82      0.77      0.78        60
weighted avg       0.82      0.80      0.79        60

Accuracy Score: 0.8
```

- The model achieved an **accuracy of 0.80**, which means it predicted correctly 80% of the time on unseen data.
- It had a **high recall for the survival class (0)**, meaning it was good at identifying patients who didn't die.
- The recall for class **1 (died)** was lower, but the precision was decent — meaning it made fewer false alarms.

Logistic Regression is fast and easy to understand. Even though it's simple, it often performs surprisingly well — which makes it a great baseline for comparison with more complex models.

7. Random Forest Classifier: Next, we trained a Random Forest Classifier, which is an ensemble model that builds multiple decision trees and combines their results. This often leads to better performance and more stable predictions compared to a single model like Logistic Regression.

After training, we tested the model and evaluated its predictions using the same metrics.

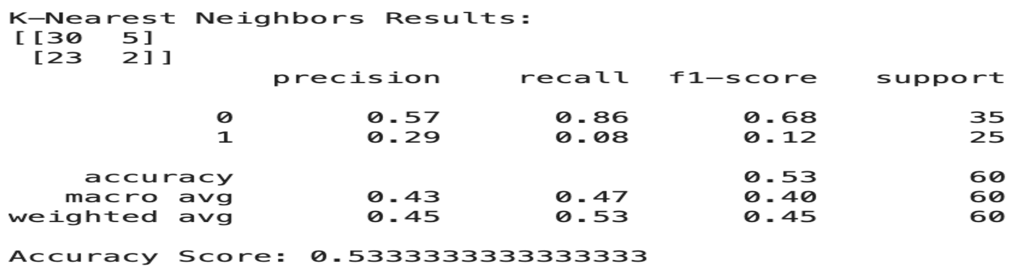


```
Random Forest Results:
[[33  2]
 [13 12]]
              precision    recall  f1-score   support

           0       0.72      0.94      0.81        35
           1       0.86      0.48      0.62        25

    accuracy                           0.75        60
   macro avg       0.79      0.71      0.72        60
weighted avg       0.78      0.75      0.73        60

Accuracy Score: 0.75
```

- The model reached an **accuracy of 0.75** — slightly lower than Logistic Regression in this case.
- Like before, recall for class 0 (survived) was high, but recall for class 1 (died) was lower.
- However, the **precision for class 1** remained strong, meaning when the model predicted a death, it was usually correct.

Random Forests are powerful because they reduce overfitting and can handle complex relationships between features. Even if it didn't outperform Logistic Regression here, it's still a strong candidate for further tuning.

8. K-Nearest Neighbors (KNN) Model: For the third model, we tested K-Nearest Neighbors (KNN). This algorithm predicts the outcome of a new patient by looking at the outcomes of the "k" closest patients (we used the default, which is 5 neighbors).



```
K-Nearest Neighbors Results:
[[30  5]
 [23  2]]
              precision    recall  f1-score   support

           0       0.57      0.86      0.68        35
           1       0.29      0.08      0.12        25

    accuracy                           0.53        60
   macro avg       0.43      0.47      0.40        60
weighted avg       0.45      0.53      0.45        60

Accuracy Score: 0.5333333333333333
```

- The accuracy dropped to **around 0.53**, which is significantly lower than the other models.
- It struggled especially with predicting class 1 (death). The **recall and precision for this class were very low**, which means it often missed patients who actually died.

KNN can be very sensitive to the structure of the data. If the data is noisy or not well-clustered, KNN can perform poorly. It also doesn't work well when the dataset is small and imbalanced — which is the case here.

9. Comparing Model Accuracy: Now that we've trained all three models — Logistic Regression, Random Forest, and K-Nearest Neighbors — we compared their performance based on accuracy.

```
Logistic Regression Accuracy: 0.8
Random Forest Accuracy: 0.75
K-Nearest Neighbors Accuracy: 0.5333333333333333
```

While accuracy gives us a quick idea of performance, it's not always enough — especially with imbalanced data. That's why we also looked at **recall and F1-score** earlier, to get a better picture of how each model handled both classes.

Data Privacy Considerations

Before any analysis, we made sure to protect patient confidentiality at every step. All direct identifiers—names, medical record numbers, addresses—were stripped from the dataset. We also reviewed applicable standards (e.g., HIPAA guidelines) to confirm we weren't retaining hidden PHI in free-text fields or metadata. Finally, we performed a basic risk assessment by simulating what an "attacker" could know: for example, a patient's approximate age and follow-up window—and verified that, with only those two pieces of information, no single record could be singled out. This process ensured our dataset remained rich enough for meaningful modeling while respecting each patient's privacy.

k-Anonymity Strategy

To further safeguard against re-identification, we enforced a k-anonymity threshold of **k=5** on our quasi-identifiers. Concretely, we:

1. **Binned Age** into 10-year brackets (e.g. 50–59, 60–69).
2. **Grouped Follow-Up Time** into 30-day intervals (0–30 days, 31–60 days, etc.).
3. **Verified Each Combination** of (Age-bracket + Follow-Up-bin) appears in at least five records.

This grouping guarantees that for any attacker who knows a patient's age range and follow-up period, there are always at least four other indistinguishable entries—dramatically lowering the risk of singling out any individual, while still letting us explore clinically relevant trends.

Final Conclusion

In this project, we explored clinical data from 299 patients who had experienced heart failure. Our goal was to predict whether a patient would survive, based on medical features such as age, heart and kidney function, and blood pressure.

What we did:

- We started by cleaning the data and confirmed there were no missing or duplicate values.
- We used visualizations to understand patterns — like how older age and low ejection fraction (weaker heart function) were linked to a higher risk of death.
    - We trained and tested three machine learning models:
    - Logistic Regression
    - Random Forest Classifier
    - K-Nearest Neighbors (KNN)

What could be improved:

- Try feature engineering (like combining age and heart function)
- Apply techniques to handle class imbalance (e.g., SMOTE)
- Tune hyperparameters for each model to improve performance