# DATA 1030 Final Project Report:

# Predicting Heart Failure Patient Survival

Drew Solomon

Brown University

GitHub: https://github.com/drew-solomon/predicting-heart-failure-survival

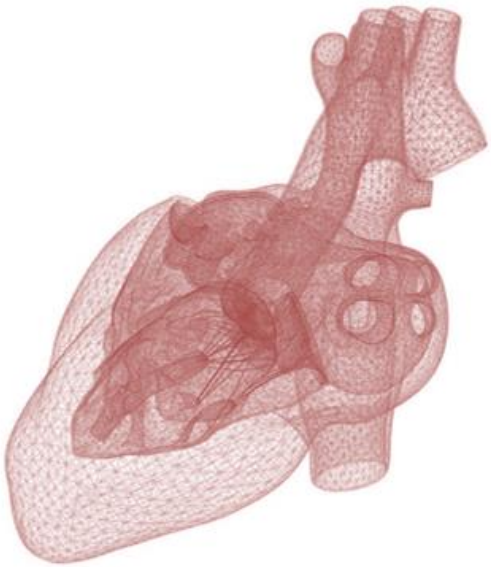

*Image by iStock/Devrimb*

# Table of Contents

# 1. Introduction

*The Problem*

Cardiovascular diseases (CVDs) are the number 1 cause of death globally. In 2016, an estimated 17.9 million people died of CVDs, representing 31% of all global deaths (WHO, 2016). Thus, CVDs represent a major public health burden. In particular, heart failure refers to when the heart cannot pump enough blood to the body. While predicting heart failure is common in medicine, clinicians' predictions of heart failure-related events have limited accuracy (Buchan et al., 2019).

*Target variable*

The target variable is the death event, which measures if the patient died before the end of the follow up period (130 days on average). 203 patients (67.89%) survived and 96 patients (32.11%) died.

*Classification problem*

Since the target variable is a categorical one with two classes (survived or not), the problem here is a binary classification problem. In particular, given the dataset, this classification problem involves predicting heart failure patient survival within a time period of around 130 days, for patients who have already suffered heart.

*Why it matters*

Identifying key features related to patients' survival and building classification models to accurately predict survival may ultimately assist clinicians in extending and improving the lives of people with heart failure.

*Dataset description*

This 2015 dataset contains themedical records of 299 heart failure patients. The dataset has 299 data points (for each individual patient) and 13 features (including the target variable).

The categorical features  are: anaemia, high blood pressure, diabetes, sex, smoking, and death event (if the patient during before the end of the follow-up period).

The continuous features are: age, creatinine phosphokinase, ejection fraction (% of blood leaving the heart at each contraction), platelets, serum creatinine (indicator of kidney dysfunction), serum sodium, and time (days until follow-up). There are no missing values.

This dataset was obtained via UCI.

# 2. EDA

**Selected Figures:**

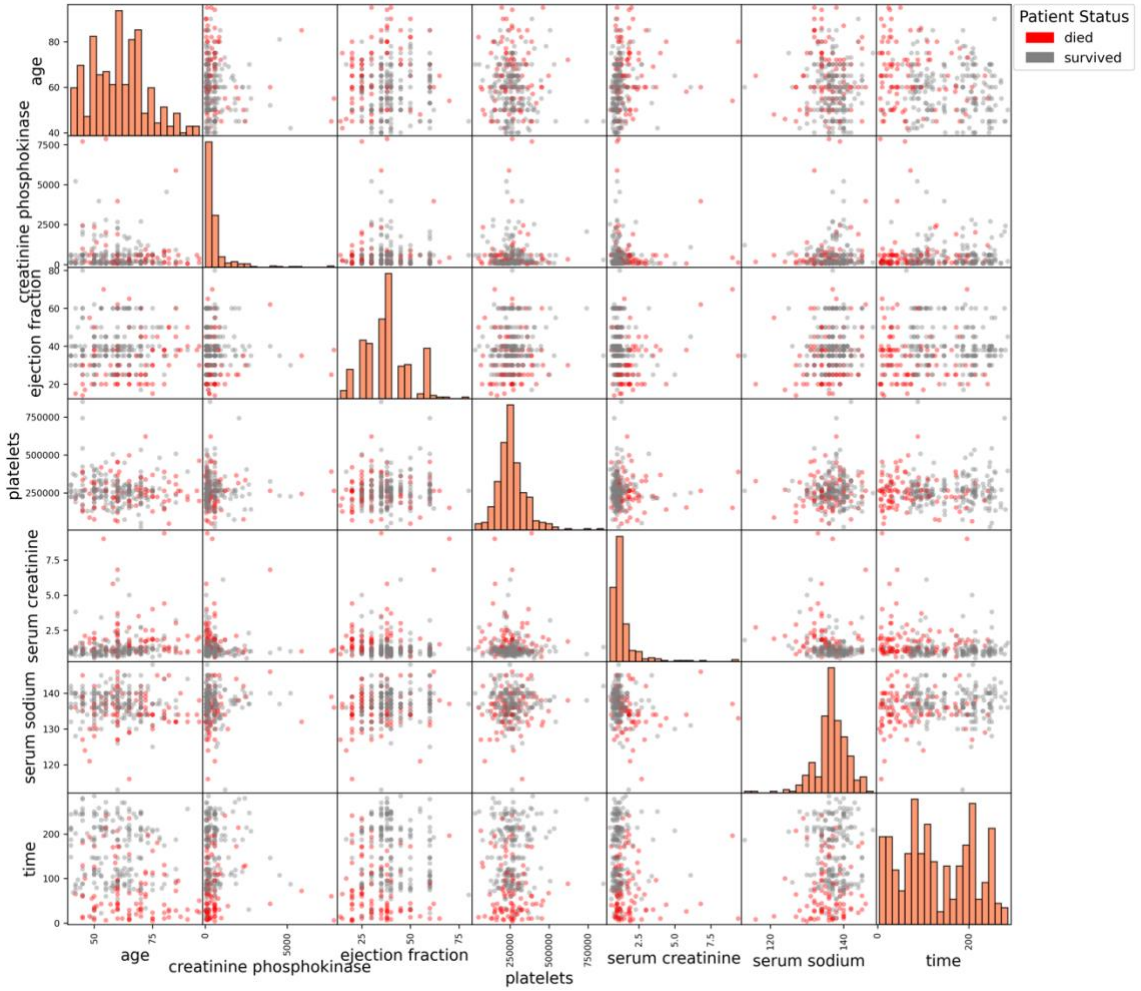


*Figure 1. Scatter matrix of patients' age and clinical information, coded by patients' survival status (red: died, gray: survived).*

The most visible pattern is that patients who died had a shorter time until follow-up, by default.

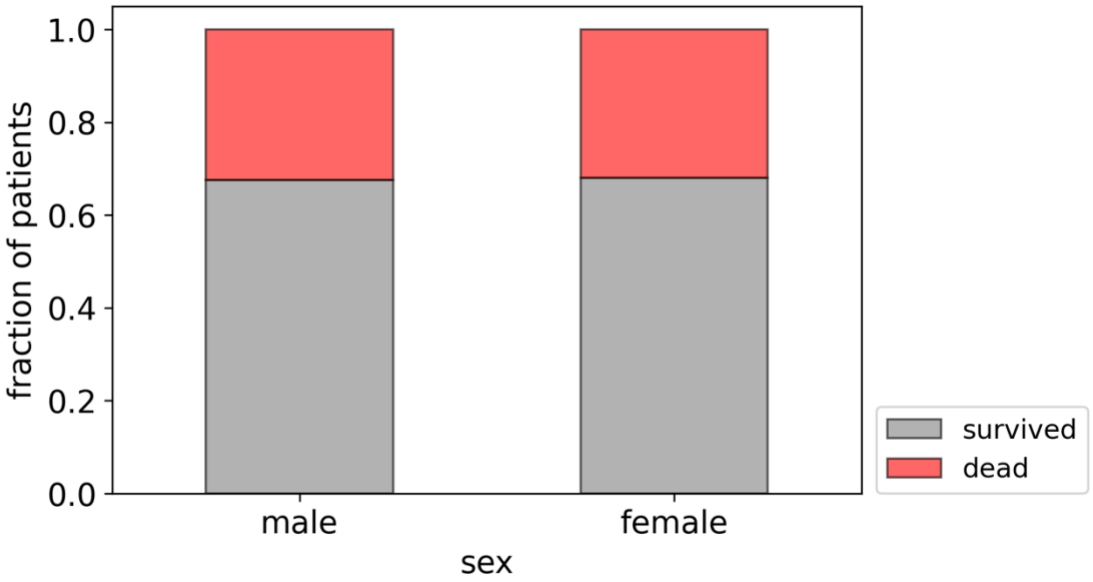


*Figure 2. Stacked bar plot of patients' survival status by sex (normalized).*

The rate of survival between the sexes is remarkably similar, at 68.04% for female patients, and 67.62% for male patients.

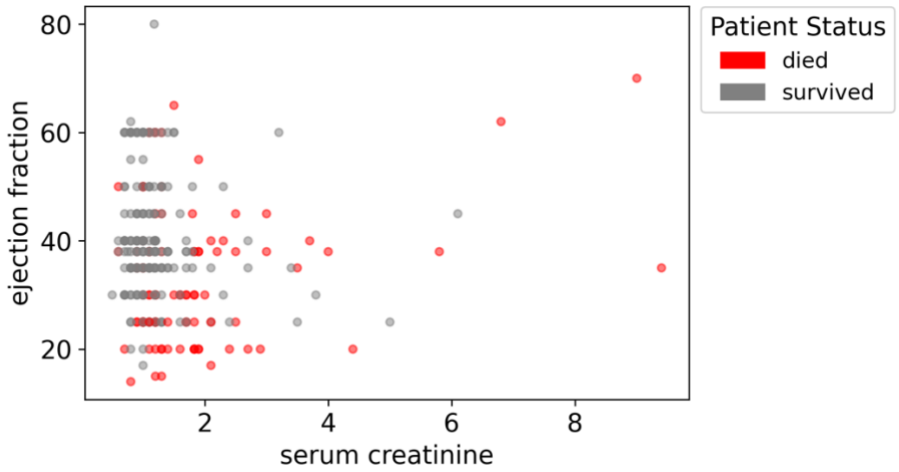


*Figure 3. Scatter plot of serum creatinine vs. ejection fraction, coded by patients' survival status.*

This plot illustrates that patients who did not survived typically had low EF and higher serum creatinine, whereas patients who survived (within the follow-up window) had a higher ejection fraction and lower serum creatinine.

Focusing on EF, the fourth figure illustrates the distribution of patients' ejection fraction measures by survival status.

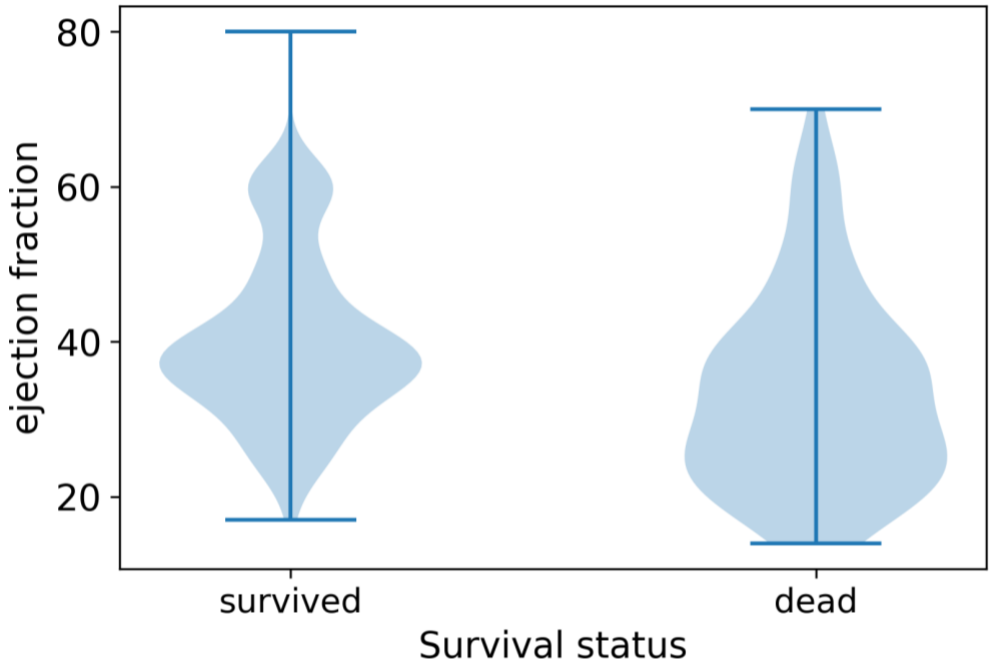


*Figure 4. Violin plot of patients' ejection fraction by survival status.*

There is a clear difference between patients who survived, with a higher EF with a mode around 38%, and patients who did not, with a mode around 23%.

# 3. Methods

### Data preprocessing

### Split proportion

Given the small dataset size, I split the data in 60% training, 20% validation, and 20% test set.

### IID data

The dataset is IID (independent and identically distributed) because each data point represents a unique patient, without repeated visits.

### Split type and justification

Given the IID structure, categorical target variable, small size (*n*=299) of the dataset, and the smaller proportion of positive target cases (death event = 1), I split the dataset using a stratified KFold split. The stratified KFold split - which splits the data into folds that preserve the proportion of samples from each class - results in variation in balance on order of only 1%. Thus, this split ensures that the balance of the target class - patient death or survival - is preserved.

### Splitting the dataset based on the classification goal

Since I am predicting heart failure patients' survival from medical record information, I discarded the 'time' feature, since this information is not available when predicting new patients' survival. By splitting with a stratified KFold split, I avoid the risk of some models being fit largely on data from patients who survived.

### Preprocessors by feature

I standardize all features with the StandardScaler, model estimation and feature ranking.

To prevent leaking statistics, the preprocessor is only fit on the training set, and then used to transform the validation and test sets.

# ML Pipeline

### Evaluation metric

In this dataset, 32.1% of patients did not survive (in the observed window). Therefore, the baseline accuracy score for this data is 67.9%, if all patients are predicted to survive. However, while accuracy measures the fraction of patients correctly classified, in this context it is vital to consider the costs of misclassification. In particular, if a higher risk of mortality guides which patients receive treatment, then incorrectly predicting the survival of a patient who faces higher risks (false negative) may prevent a life-saving treatment. In addition, predicting higher mortality for a patient who is likely to survive (false positive) may result in the misallocation of limited medical resources, which is costly to other high risk patients.

Given the dataset balance and the stakes of misclassification for patients, I chose the Matthew's correlation coefficient (MCC) as my evaluation metric. The MCC is the correlation coefficient between the observed and predicted binary classifications, and returns a value between −1 and +1. The key characteristic of the MCC is that both positive and negative cases must be predicted well for the score to be high.

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

*Figure 5. Matthew's Correlation Coefficient (MCC) formula.*

By taking all 4 confusion matrix classes (TP, TN, FP, FN) into account, the MCC captures the importance of correctly predicting both positive and negative cases within an imbalanced dataset (Chicco & Jurman, 2020).

### Splitting for hyperparameter tuning

Across the ML models, I use a StratifiedKFold with 4 folds to split the data. Using multiple folds to tune hyperparameters is more robust because the best hyperparameters are chosen based on the average score across the folds. By definition, each fold is used once as a validation set, while the others form the training set. Thus, the score for a given hyperparameter combination is the average of the 4 fold scores.

**Feature selection**

Given the small number of features in general (11) and relative to the number of points (299), feature selection is not necessary before training the models, so I save feature selection for after the ML models are trained and tuned.

**Hyperparameter tuning**

For each ML technique, I chose 2-3 relevant hyperparameters to optimize using GridSearchCV. For the given parameter ranges, GridSearchCV trains a model for every parameter combination and evaluates each model's performance on the validation set (by MCC score). GridSearchCV saves the highest scoring hyperparameter combinations as best models, which I then evaluate on the test set to measure how well the models' predict unseen data.

For each ML model, I aimed to find the hyperparameter combination that produces the optimal bias-variance trade-off, observing underfitting and overfitting for neighboring values. Further, to find the combinations that yield consistently high validation scores, I calculated the standard deviation for every possible combination across 4 folds and 6 random states and found the hyperparameters that scored the most standard deviations above the baseline.

Thus, for each ML model, I:

- split and preprocess the data using a stratified KFold with 4 folds
- train and tune the ML model using GridSearchCV, maximizing MCC validation scores and returning best models' hyperparameter combinations and scores
- measure the uncertainty across each fold for all parameter combinations
- calculate the mean MCC test score and standard deviations over all random state iterations (recording split/non-deterministic model uncertainty)
- plot the averaged confusion matrix for the ML model over the random state iterations, to visualize the ML model's classifications

**Models and hyperparameter tuning:**

***Model 1: Logistic Regression (elastic net)***

Hyperparameters tuned:

- C: inverse of regularization strength
    - Range: [1e-4, 1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2]
- l1_ratio: ratio of lasso to ridge regularization (penalizing large coefficients)
    - Range: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.9, 1]

In total, there are 7 x 11 = 77 combinations.

The best hyperparameter combination was: C= 0.1, l1_ratio = 0.2.

These hyperparameters are within the given range, between under and overfitting.

***Model 2: Support Vector Classification (SVC, radial basis function kernel)***

Hyperparameters tuned:

- C: inverse of strength of regularization
    - Range: [1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3]
- Gamma: the SVM's kernel coefficient, determines the spread of influence around each training point
    - Range: [1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2]

In total, there are 6 x 6 = 36 combinations.

The best hyperparameter combination was: C= 10, Gamma = 0.01, each within the under- and overfitting range.

***Model 3: Random Forest***

The hyperparameters were chosen to reduce overfitting by constraining the random forest trees, and introducing randomness in which features are used. The values reflect the small number of points and features in the original dataset.

Hyperparameters tuned:

- max_depth: maximum depth of each tree in the ensemblre
    - Range: [2, 3, 4, 5, 6, 7, 8, 9, 10] (up to number of features)
- max_features: fraction of features considered at each split
    - Range: [0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
- min_samples_split: minimum number of points to split internal node
    - Range: [4, 8, 16, 32, 64]

In total, there were 9 x 6 x 5 = 270 combinations.
The best hyperparameters (in terms of standard deviations above the baseline score) were: max_depth = 2, max_features = 0.5, and min_samples_split = 16.

***ML model 4: XGBoost***

Hyperparameters tuned:

- max_depth: maximum depth of each tree
    - Range: [1, 2, 3, 4, 5]
- reg_alpha: l1 regularization on weights
    - Range: [1e-2, 1e-1, 1e0, 1e1, 1e2]
- reg_lambda: l2 regularization on weights
    - Range: [1e-1, 1e0, 1e1, 1e2, 1e3, 1e4]

In total, there were 5 x 5 x 6 = 150 combinations.
The best hyperparameters were: max_depth = 4, reg_alpha: 0.1, reg_lambda: 10000.

***Model 5: k-NN***

Hyperparameters tuned:

- n_neighbors:
    - Range: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29]
- weights:
    - Range: [ ‘uniform’, distance’]

In total, there were 15 x 2 = 30 combinations.
The best hyperparameters were: n_neighbors = 5, weights = ‘uniform’.
The number of neighbors is between the under- and overfitting range, while uniform weighting performed best on the CV the vast majority of times.

**Measuring uncertainty due to splitting and non-deterministic methods**

I measure the uncertainty in models’ prediction performances due to splitting and non-deterministic methods by cross-validating over 6 random states. Further, to measure uncertainty in tuning, I calculate the standard deviation in validation scores for each hyperparameter combination across 4 folds and 6 random states (i.e. 24

scores for every combination). For prediction, I calculate the standard deviation in test scores over the 6 random states, for each ML model.

# 4. Results

### Baseline Score

The baseline MCC score – when all patients are predicted to survive, by majority class – is 0.

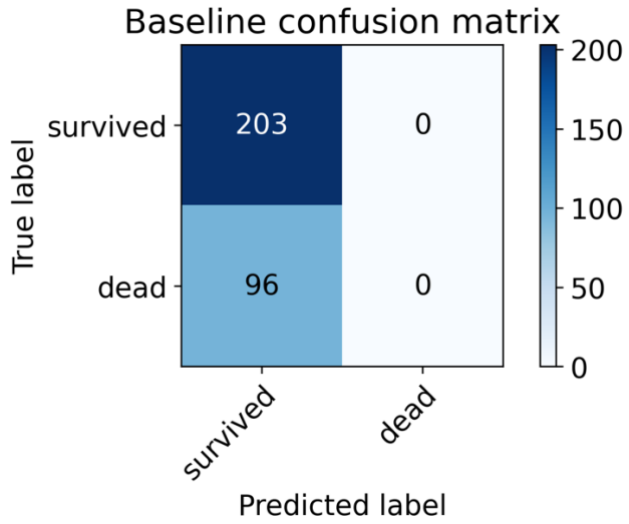


*Figure 6. Baseline confusion matrix, predicted by majority class.*

### ML Models Performance

*Table 1. Summary of each ML test survival prediction results, over 6 random states.*

| Model | Baseline MCC | Mean test MCC | Standard deviation | Standard deviations above baseline |
|---|---|---|---|---|
| Logistic regression | 0 | 0.357 | 0.044 | 8.12 |
| SVC (rbf) | 0 | 0.329 | 0.099 | 3.33 |
| Random forest | 0 | 0.377 | 0.087 | 4.33 |
| XGBoost | 0 | 0.339 | 0.093 | 3.66 |
| k-NN | 0 | 0.137 | 0.103 | 1.33 |

The best classification was achieved by the random forest, while the logistic regression model had the least variance (and relatively good performance).

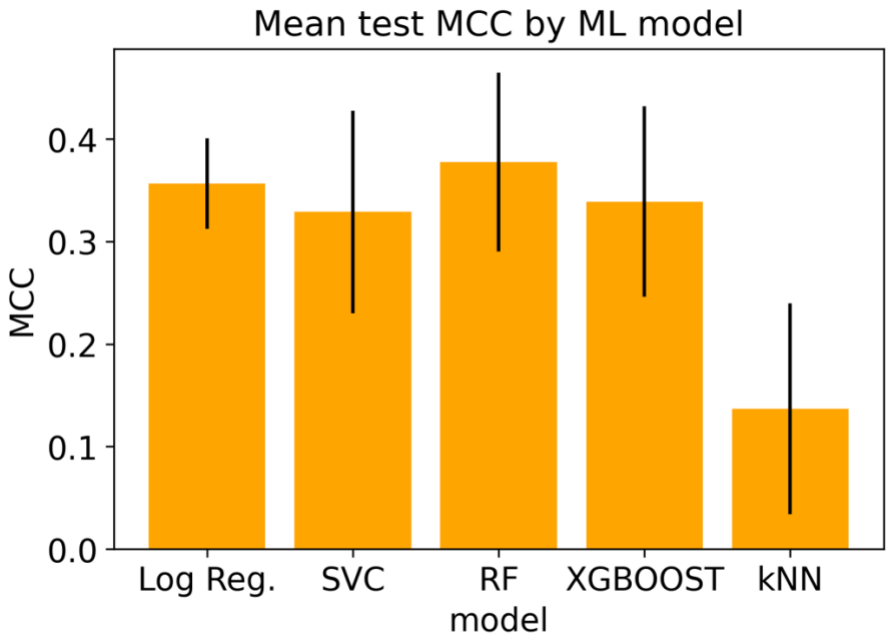


*Figure 7. Bar plot of mean test MCC by ML model, with standard errors.*

To view how each model classified patients exactly, I plot the confusion matrices averaged across the 6 random states for each model (class 0 = 'survived', 1 = 'dead').

***Model 1: Logistic Regression (elastic net)***

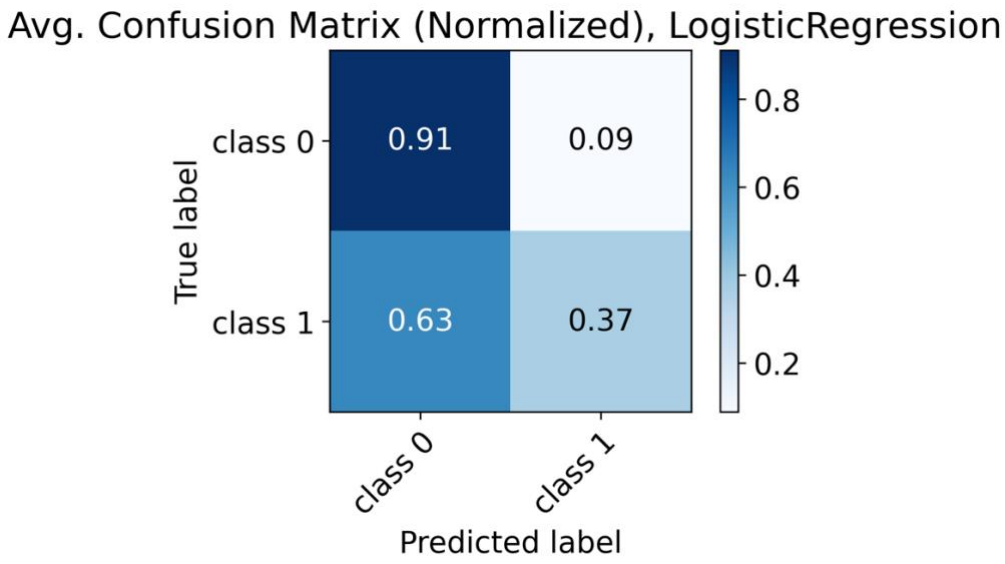


*Figure 8. Average confusion matrix for logistic regression (normalized).*

Thus, 91% of patients who survived were correctly predicted, while only 37% of patients who died were accurately predicted.

***Model 2: Support Vector Classification (SVC)***

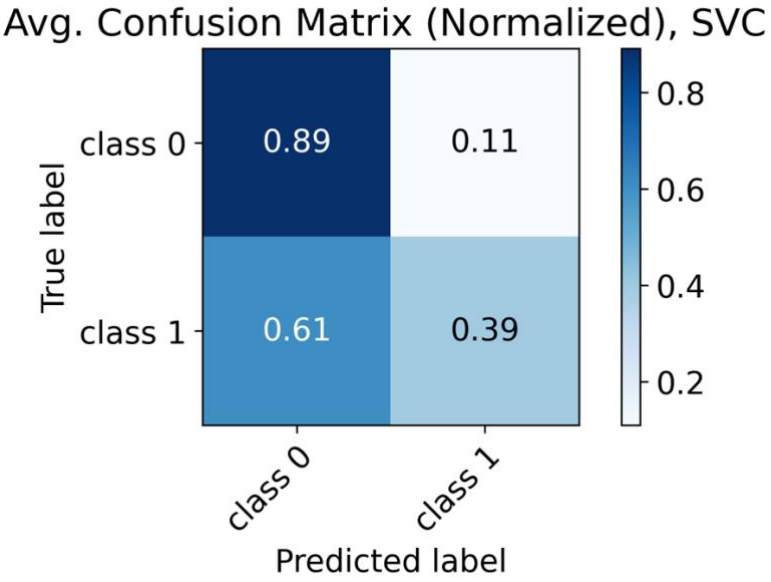


*Figure 9. Average confusion matrix for SVC (normalized).*

Here,  39% of patients who died were accurately predicted.

***Model 3: Random Forest***

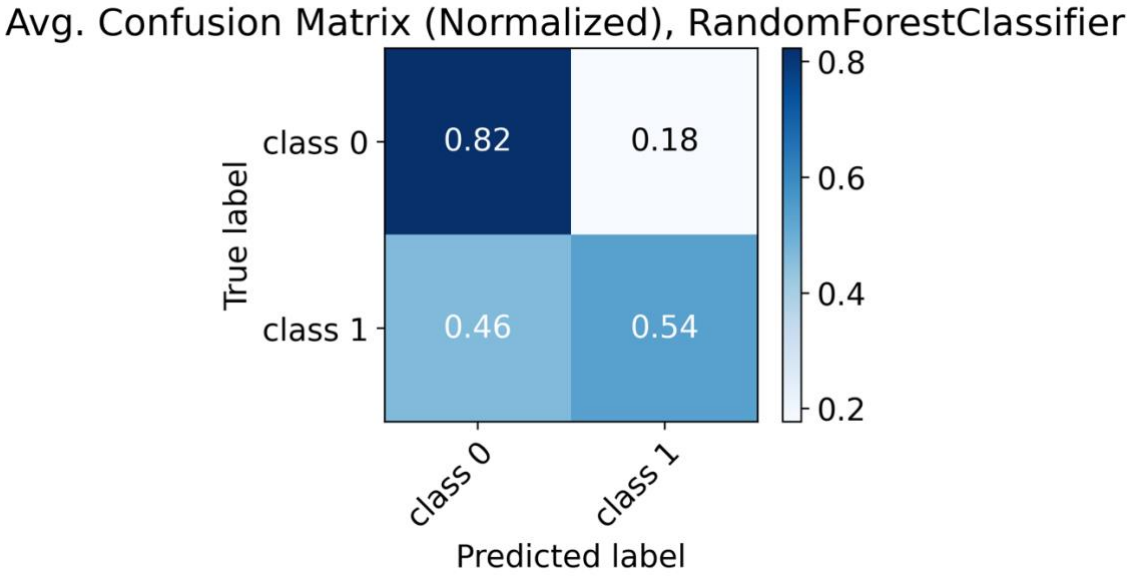


*Figure 10. Average confusion matrix for random forest (normalized).*

Unlike the other models, random forest correctly predicts the majority (54%) of patients who died.

***ML model 4: XGBoost***

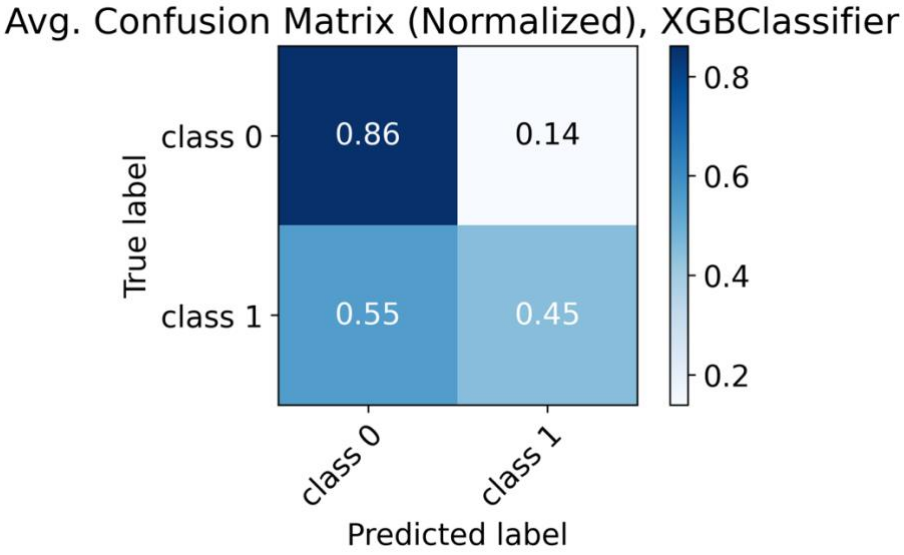


*Figure 11. Average confusion matrix for XGBoost (normalized).*

***Model 5: k-NN***

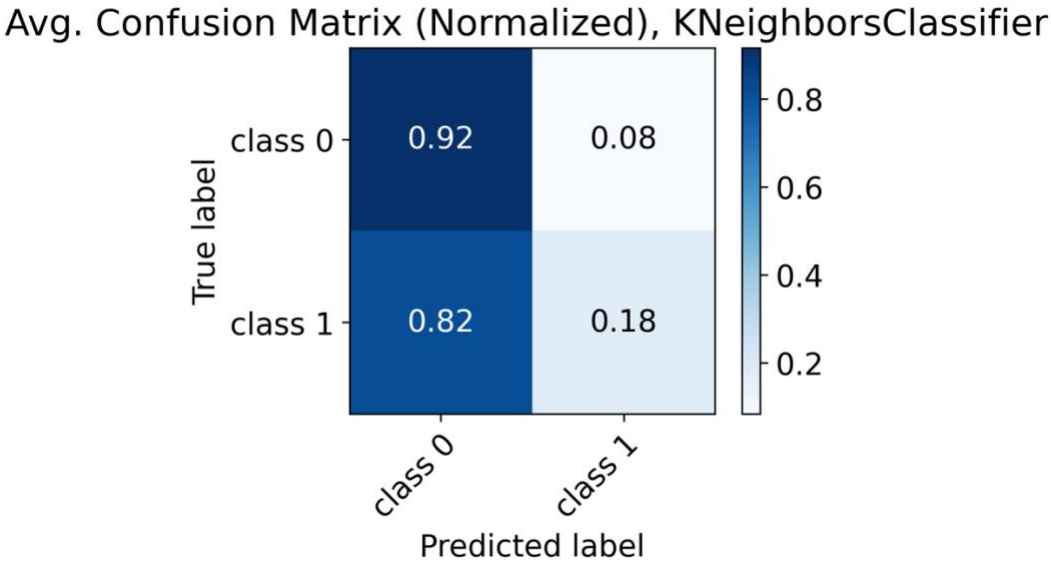


*Figure 12. Average confusion matrix for kNN (normalized).*

The k-NN model did especially poorly, misclassifying 82% of patients who died.

Overall, while the models predicted patients' survival above baseline, the rate of misclassification is still high for patients who died.

**Global feature importances**

To determine which features are most important for predicting survival, I measure global feature importance across the ML models using linear model coefficients, permutation feature importances, Gini impurity decrease, XGBoost metrics, and SHAP values.

***Coefficients of linear model:***

Because all features were standardized during preprocessing (i.e. zero mean and unit variance), the features' coefficients' absolute values reflect feature importance (without original scale).

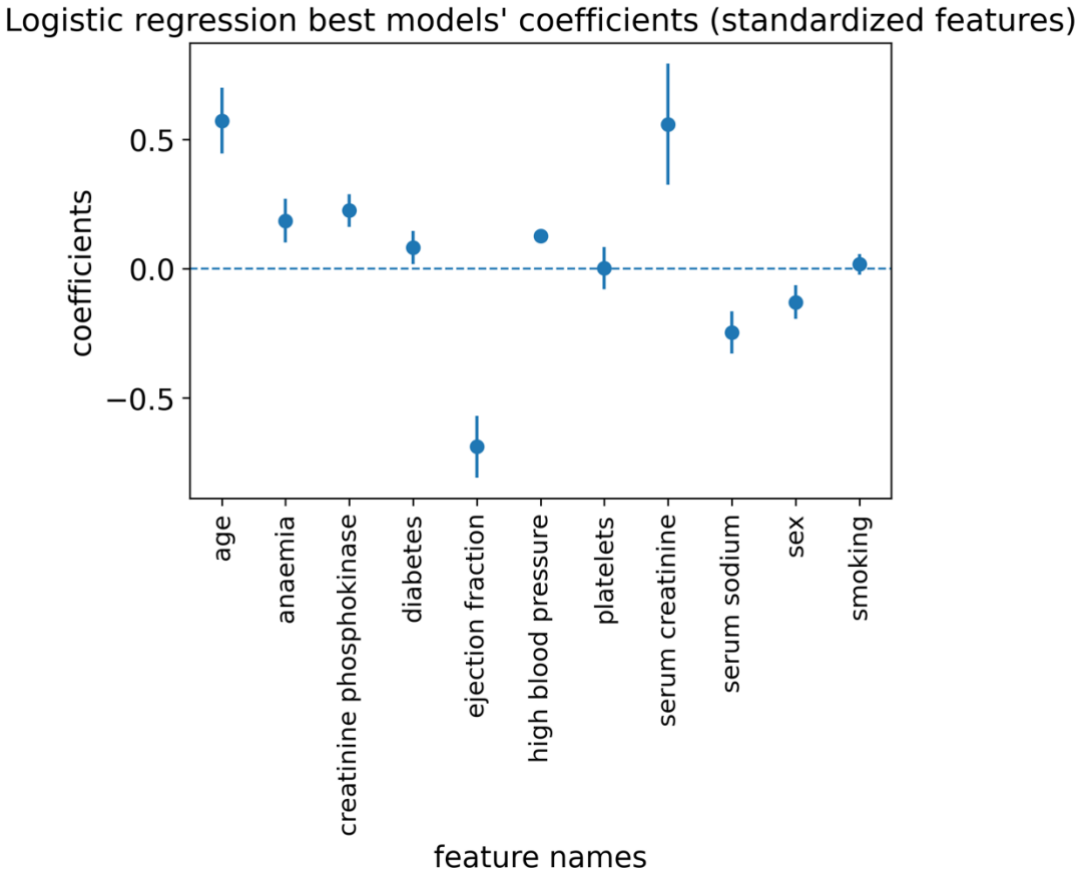


*Figure 13. Logistic regression best models' coefficients, with error (standardized).*

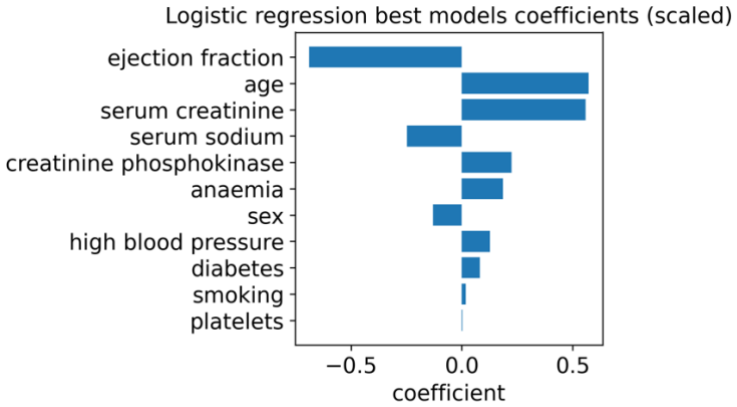


*Figure 14. Logistic regression coefficients, sorted by absolute value.*

Therefore, the top 3 most importance features across the logistic regression best models are: ejection fraction (EF), age, and serum creatinine.

***Permutation feature importance, by model:***

By randomly shuffling one feature at a time and measuring the drop in test score, permutation feature importance indicates which feature are most necessary for good predictions. This method is particularly useful for this data given the lack of highly correlated features (see EDA).

1. Logistic regression:

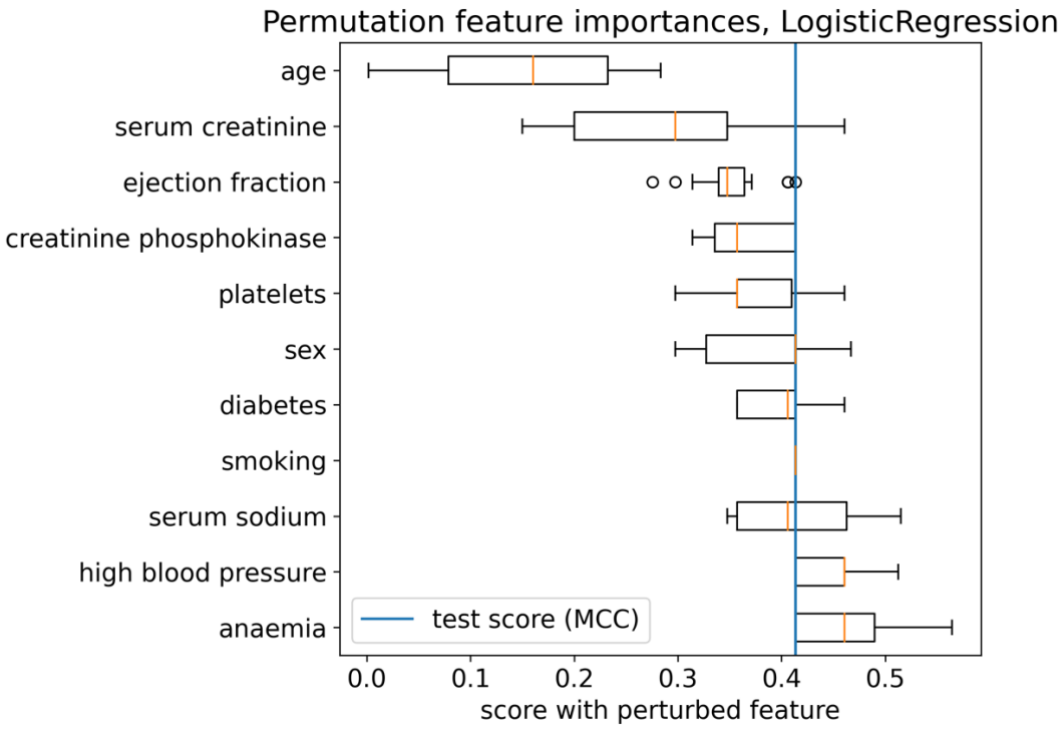


*Figure 15. Permutation feature importances, logistic regression.*

2. SVC:

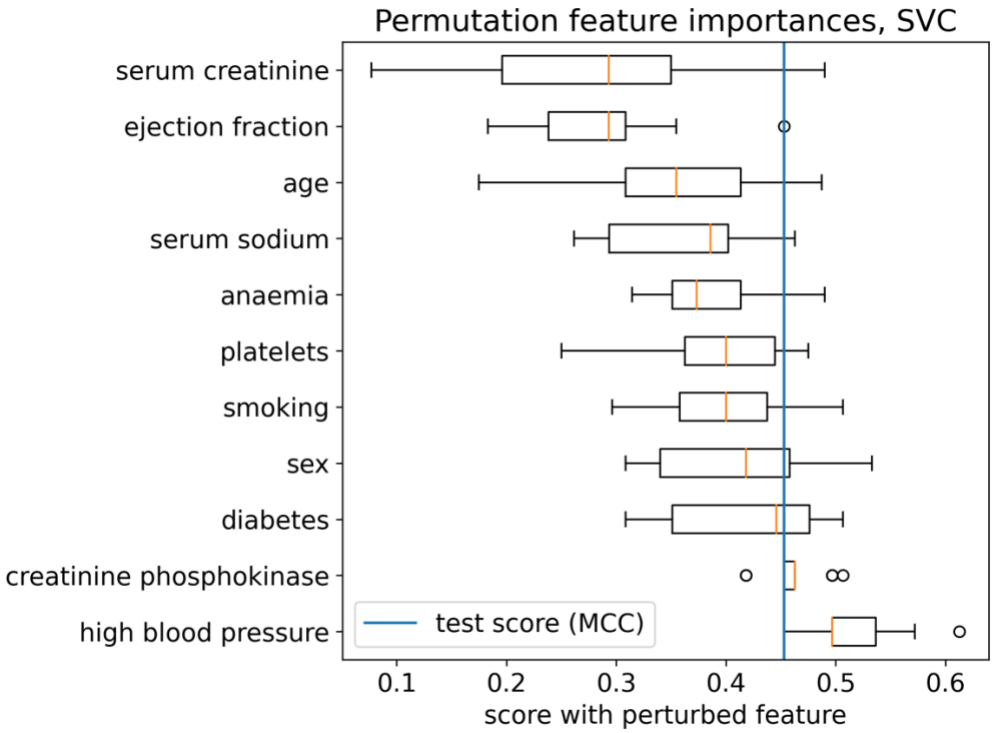


*Figure 16. Permutation feature importances, SVC.*

3. Random Forest:

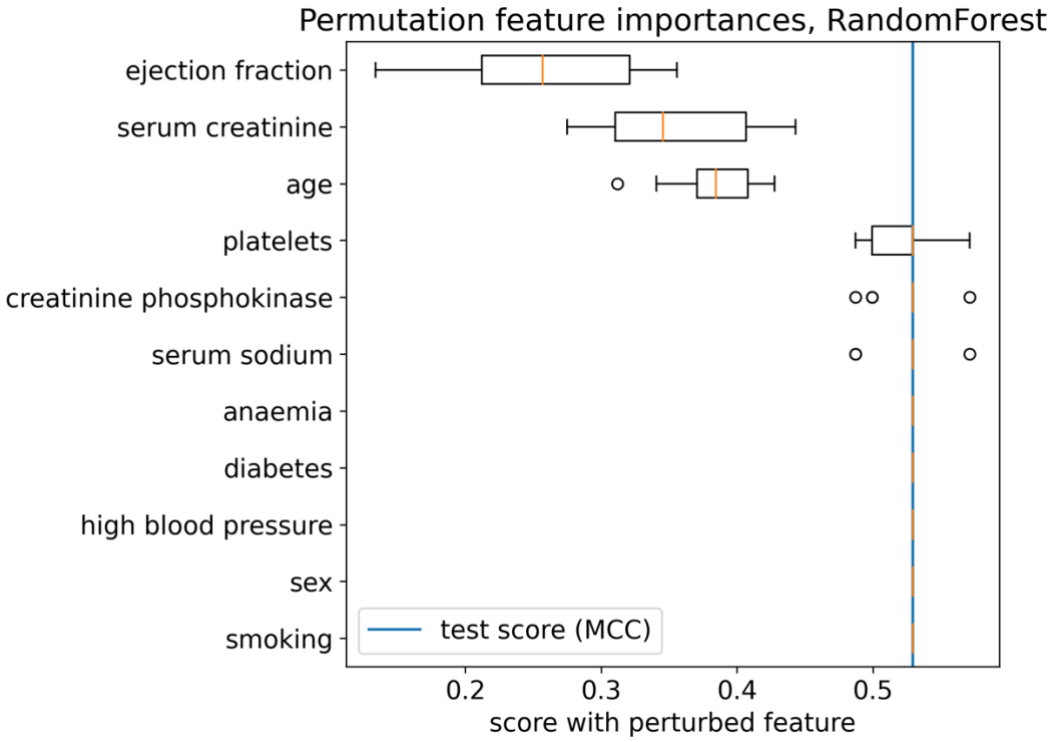


*Figure 17. Permutation feature importances, Random Forest.*

4. XGBoost:

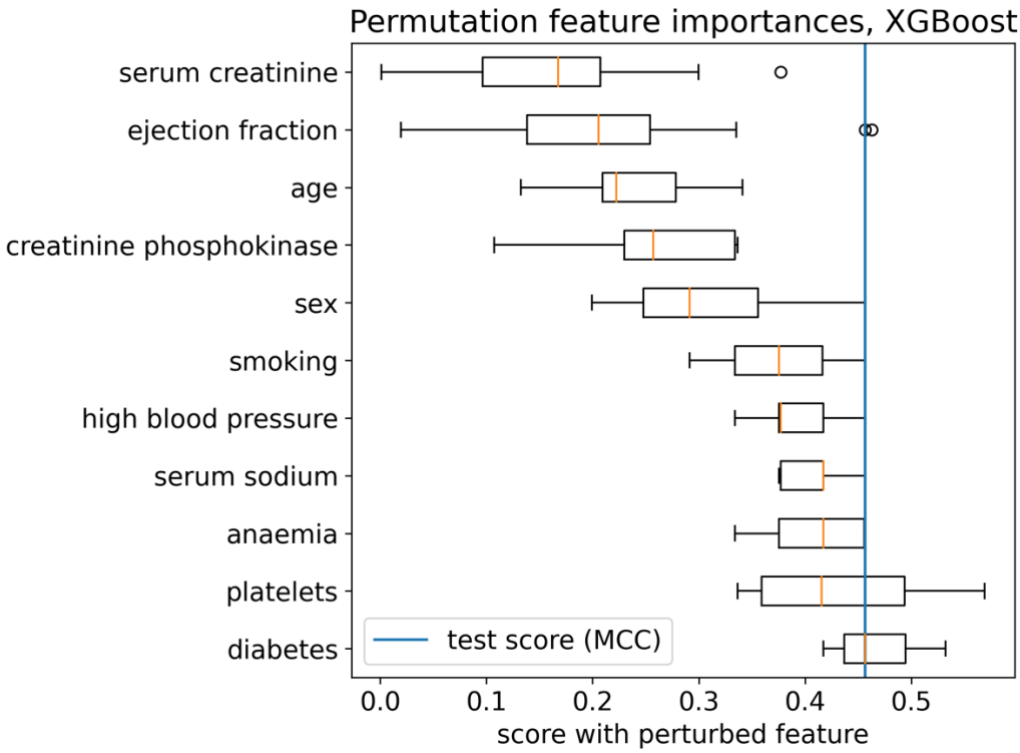


*Figure 18. Permutation feature importances, XGBoost.*

5. K-NN:

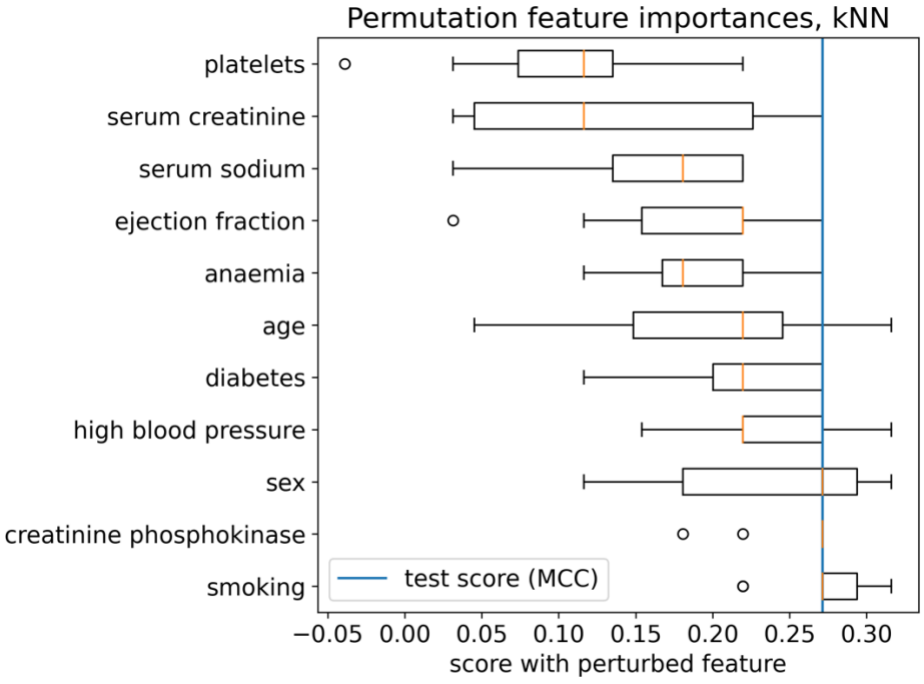


*Figure 19. Permutation feature importances, kNN.*

Aside from KNN (which performs poorly), the permutation feature importances find that age, serum creatinine, and ejection fraction are most important.

***Random Forest – Mean Gini Impurity Decrease***

Gini impurity measures the quality of a decision tree's split by the probability of misclassification in a node. Here, the more the mean Gini impurity decreases when a feature is removed, the more important the feature.

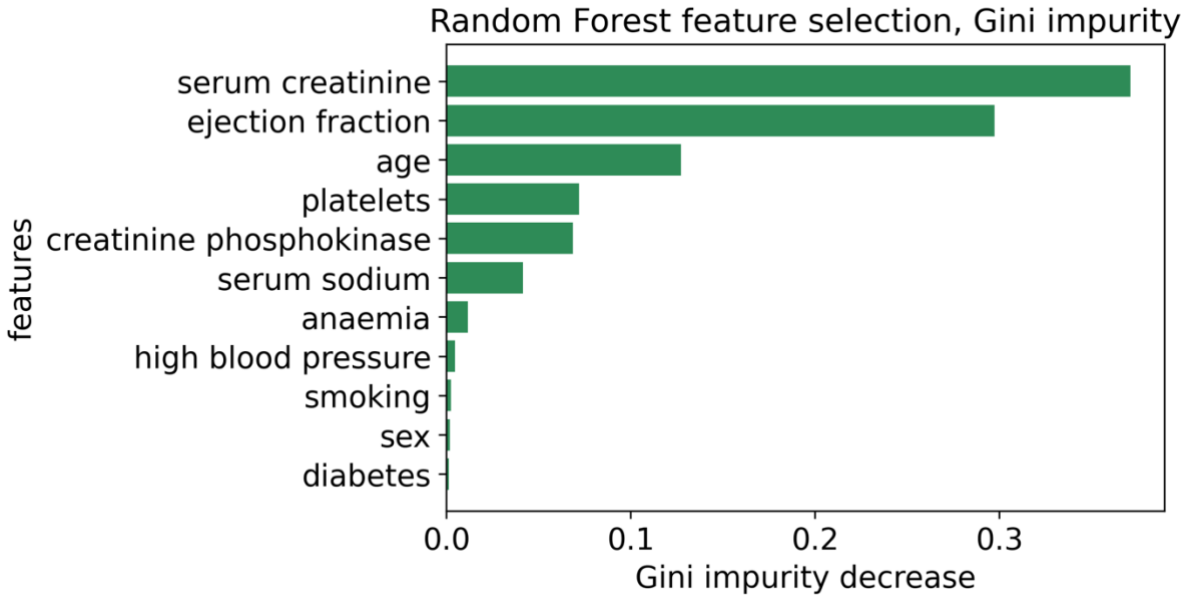


*Figure 20. Random forest feature ranking, Gini impurity decrease.*

***XGBoost metrics***

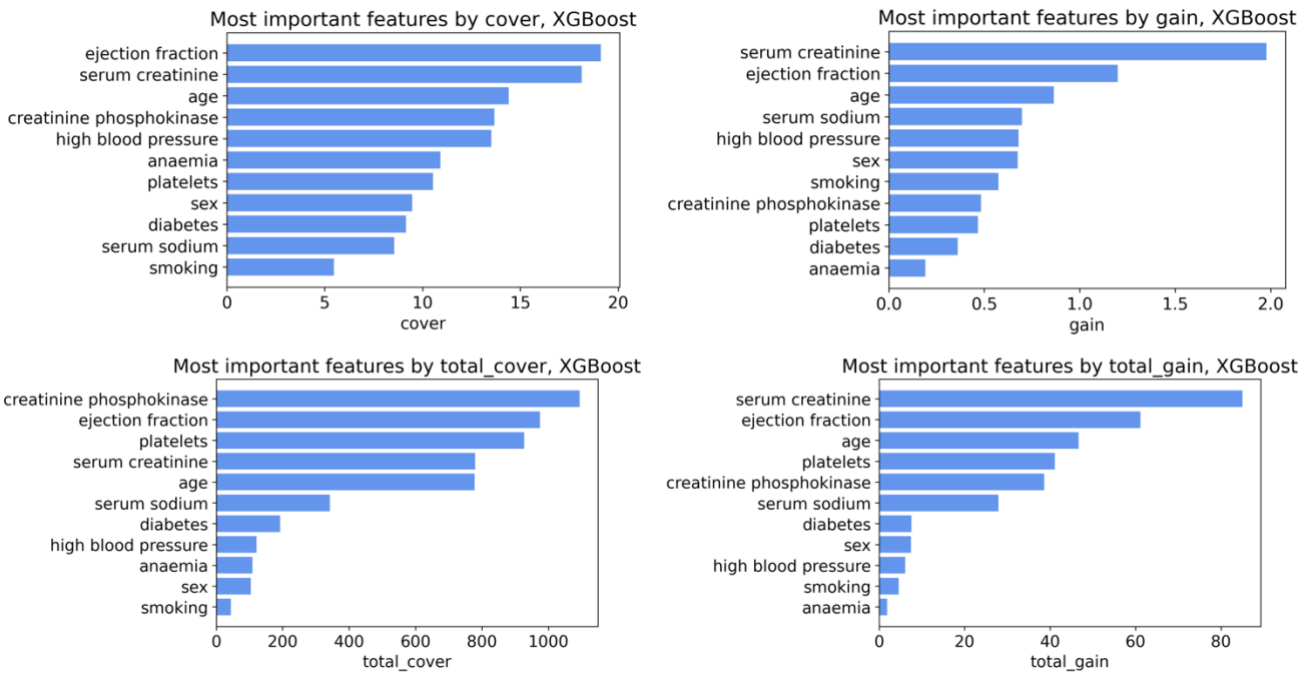


*Figure 21. Feature ranking by XGBoost metrics.*

### *SHAP values*

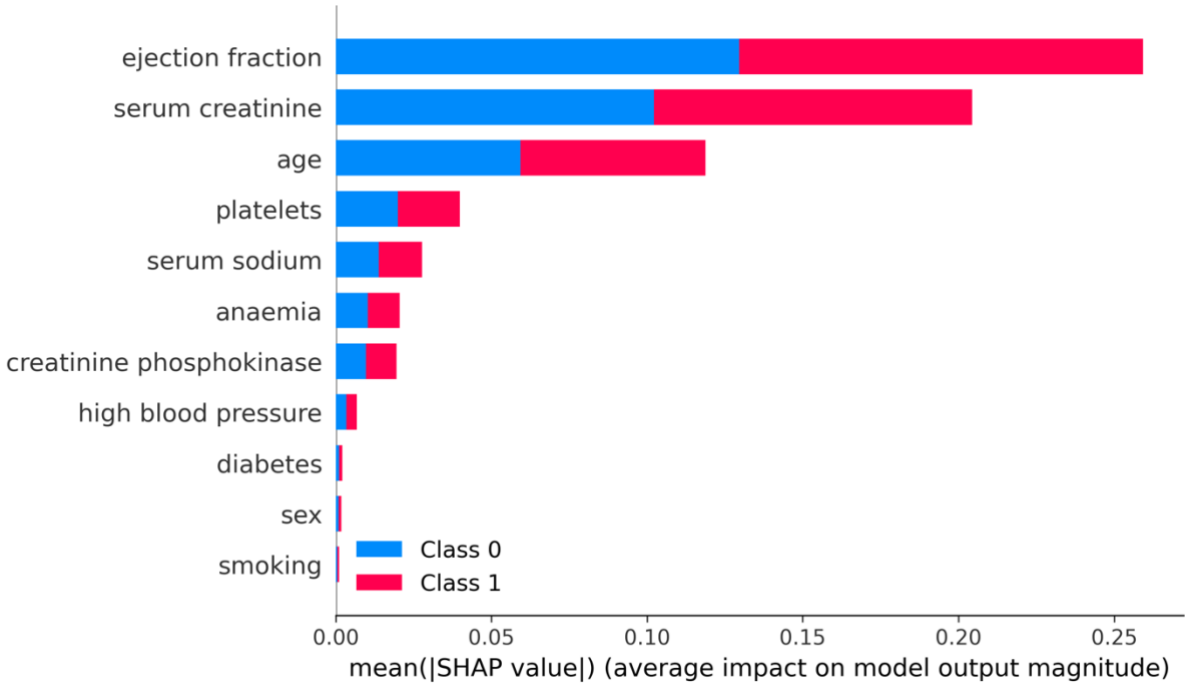


*Figure 22. Mean absolute SHAP values, for RF model.*

Overall, the feature rankings consistently rank age, serum creatinine, and ejection fraction the highest. Thus, these feature are most important for predicting survival. As expected, higher EF decreases the odds of a death event, while higher age and serum creatinine increase the odds of a death event. Interestingly, some of the least important features are smoking and diabetes.

## Local feature importances

SHAP force plot show how much each feature contributes to the final prediction for that specific point. In this context, it indicates what the predicted probability of surviving (class 0) is for that patient, and how the different features pushed this prediction from the baseline.

For example, for patient at index 3, their high ejection fraction and low serum creatinine reduce their risk of dying below the baseline, despite their anaemia:

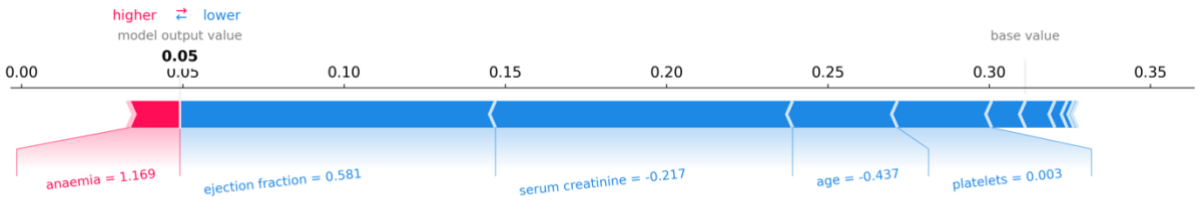


*Figure 23. Force plot of patient, predicting class 1.*

## Interpretation/Discussion

The feature ranking clearly indicate that serum creatinine, ejection fraction, and age are the most importance features in predicting survival.

It is surprising that smoking and diabetes are not important for predicting survival among heart failure patients. However, all patients in this dataset have suffered heart failure, so the additional risk of heart failure from smoking or having diabetes is likely to be smaller than among the general population.

Based on these results, I re-trained the ML models on a dataset containing just the top 3 most predictive features, finding:

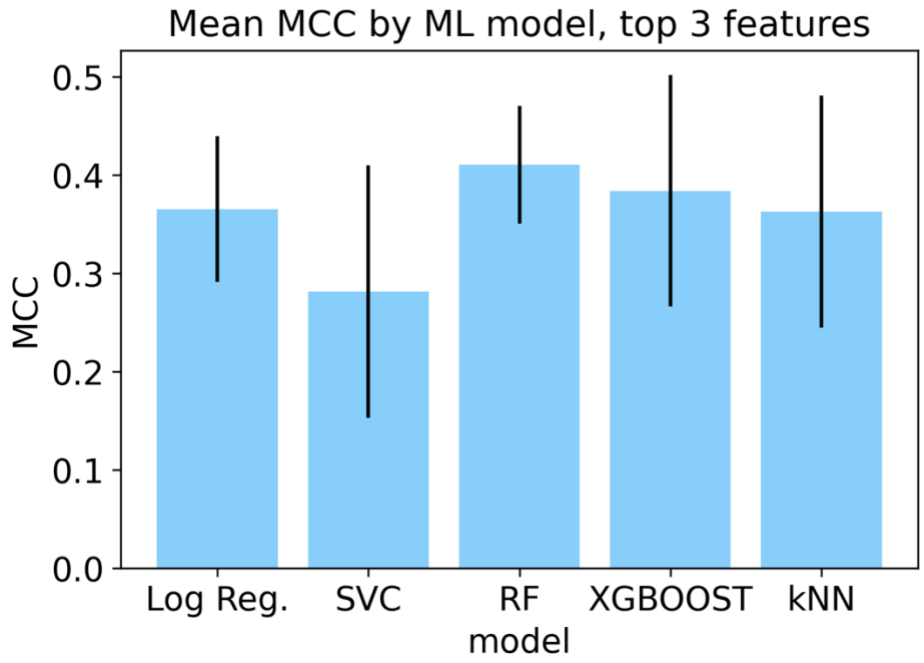


*Figure 24. Mean MCC by ML model, top 3 features.*

Compared to  the full dataset:

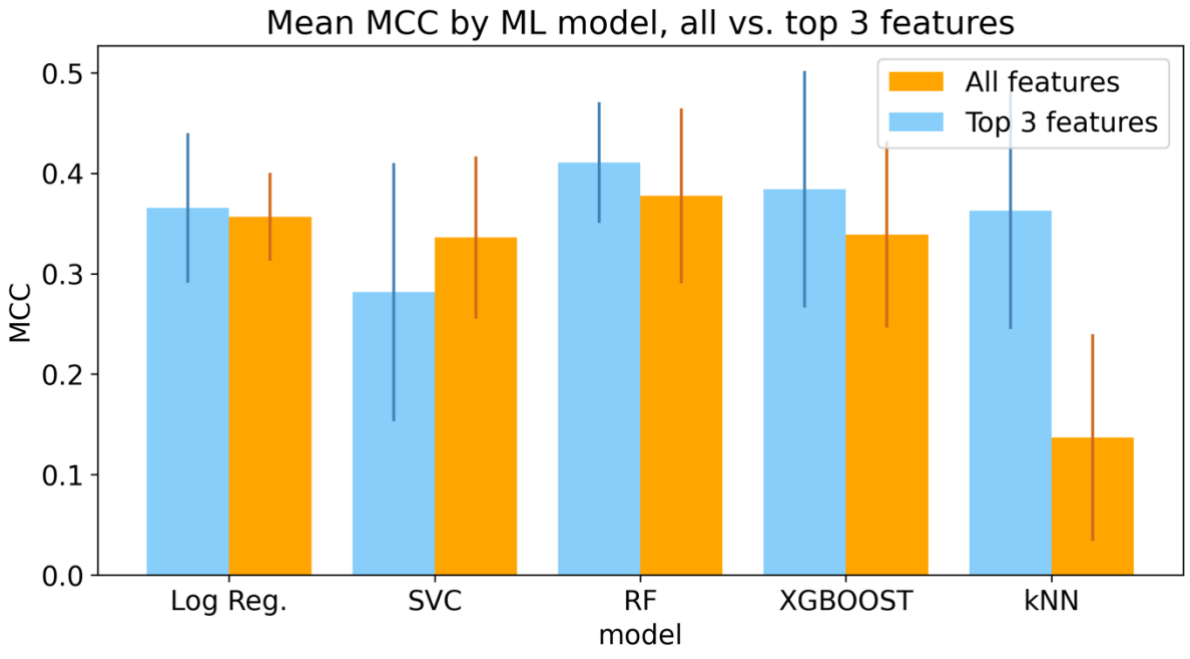


*Figure 25. Comparison of mean MCC by model, all vs. top 3 features.*

Therefore, it appears that using the 3 features along achieves at least as good if not better survival prediction than using the full dataset, especially with a random forest or XGBoost approach, evaluated with the MCC.

# 5. Outlook

### Limitations

The main weakness of this study is the small size of the data set (n=299), as well the oversimplification of whether or not patients survive, instead of how long. In addition, the data is missing other factors related to heart failure such as weight.

### Potential improvements

To improve this study, I would run the ML models on a larger dataset that contains the same predictive features, to verify (or falsify) the results. In addition, I would run the models over a finer grain of hyperparameter combinations, and more random states, to find the best combination and minimize error.

### Further research/techniques

Further research might include using neural networks to predict survival, although they may be harder to interpret. In addition, further research may examine what factors contribute to lower ejection fraction or high serum creatinine, and potentially examine which interventions are most effective in mitigating these risks.

# References

Ahmad, T., Munir, A., Bhatti, S. H., Aftab, M., & Raza, M. A. (2017). Survival analysis of heart failure patients: A case study. PloS one, 12(7), e0181001.

American Heart Association. Classes of Heart Failure. www.heart.org/en/health-topics/heart-failure/what-is-heart-failure/classes-of-heart-failure.

Buchan, T. A., Ross, H. J., McDonald, M., Billia, F., Delgado, D., Posada, J. D., ... & Alba, A. C. (2019). Physician prediction versus model predicted prognosis in ambulatory patients with heart failure. The Journal of Heart and Lung Transplantation, 38(4), S381.

World Health Organization. Cardiovascular Diseases (CVDs). www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).

Chicco, D., & Jurman, G. (2020). Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. BMC medical informatics and decision making, 20(1), 16.

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC genomics, 21(1), 6.

Heart Disease Facts. 8 Sept. 2020, www.cdc.gov/heartdisease/facts.htm.

Raphael, C., Briscoe, C., Davies, J., Whinnett, Z. I., Manisty, C., Sutton, R., ... & Francis, D. P. (2007). Limitations of the New York Heart Association functional classification system and self-reported walking distances in chronic heart failure. Heart, 93(4), 476-482.

Savarese, G., & Lund, L. H. (2017). Global public health burden of heart failure. Cardiac failure review, 3(1), 7.