Heart Failure Modeling via Random Forest Analysis

Rahul Khanna

505943969

University of California, Los Angeles

AOS C111: Introduction to Machine Learning for Physical Sciences

1. Introduction

Heart failure is a serious condition in which the heart fails to pump sufficient blood for the body, often resulting from an overly weak or stiff heart muscle. The underlying cause can vary from coronary artery disease to high blood pressure, but the results can often be fatal, with heart failure being the largest contributor to "background cardiovascular diseases", claimed to be responsible for over 17 million annual deaths globally. It is important to note that heart failure is not a disease itself, but rather a condition that is often the end state of various heart-related disorders. Heart failure is also somewhat unpredictable: a 2019 meta-analysis estimated a 5-year survival rate of about 57%, and a 10-year survival rate of about 35%. This can make it difficult to predict in advance which patients are more susceptible to death following a heart failure event, which in turn affects treatment. This is particularly an issue among older adults, a group that is more significantly affected by heart failures than their younger counterparts. The high mortality rate among older patients has ripple effects on healthcare systems and infrastructure, as treatment is difficult, expensive, and quickly variable depending on the severity of the condition for each patient.

Creating a model to predict whether patients will die or not during a specific time is important, because it can provide needed insight into the treatments and options given. It can also help deduce the most important features that affect fatality following heart condition, which can help with preventative treatments that aim to reduce the mortality risk of heart failure. While there exist numerous factors that should be taken into account when diagnosing a patient with heart failure, this report will focus specifically on just a few, including age, anemia, CPK enzyme level, diabetes, heart blood ejection, serum sodium levels, sex, and platelet levels.

2. Data and Preprocessing Analysis

This report utilizes the "Heart Failure Clinical Records" dataset sourced in 2020 from the UCI Machine Learning repository. It contains data for 299 patients, with 12 features:

- Age (measured in years)
- Anaemia (whether or not patient experienced a decrease in red blood cells)
- Creatinine Phosphokinase (amount of CPK enzyme present in blood, measured in mcg/L)
- Diabetes (whether or not the patient has diabetes)
- Heart blood ejection (amount of blood leaving heart per contraction, measured in %)
- Blood Pressure (whether or not patient had high blood pressure)
- Platelets (amount of platelets present in patient blood, measured in kiloplatelets/mL)
- Serum Creatinine (amount of serum creatinine in blood, measured in mg/dL)
- Serum Sodium (amount of serum sodium in blood, measured in mEq/L)
- Sex (whether patient is biological man or woman)
- Smoking (whether or not patient smokes)
- Follow-Up time (measured in days)

The response variable for the model will be Death Event (whether or not the patient died during the follow-up period). The purpose of this report is to create a Random Forest model that predicts binary value “Death Event”, given some subset of the 12 features. Due to the binary nature of the response variable, this can be interpreted as a classification problem.

The dataset was provided with no missing values, so there was no need to do initial replacement or removal of invalid values. I decided to select my subset of features (to use for the model) by examining the correlations between each of the features, and the target value “Death Event”. While the relationship between “death_event” and the set of features is likely non-linear, correlation value can still provide a strong baseline in determining which features had a greater effect on death_event than others. Doing so led to the following results:

```
correlations = X_original.corrwith(y['death_event'])
print(correlations.sort_values(ascending=False))
features_to_keep = correlations[abs(correlations) > 0.1].index
X = X_original[features_to_keep]
```

```
serum_creatinine            0.294278
age                         0.253729
high_blood_pressure         0.079351
anaemia                     0.066270
creatinine_phosphokinase    0.062728
diabetes                   -0.001943
sex                        -0.004316
smoking                    -0.012623
platelets                  -0.049139
serum_sodium               -0.195204
ejection_fraction          -0.268603
time                       -0.526964
```

Figure 1: Table of features and their correlation with target variable

Here, correlation ranges from -1 to 1, with 1 indicating a strong positive correlation and -1 indicating a strong negative correlation. An initial examination of these values suggests that time has the “strongest” correlation with Death Event at -0.526, suggesting that as time increases, patients are less likely to have died during the observation period. Indeed, further analysis showed that the average follow-up period for patients that died was 70.885, compared to the average follow-up time period for patients that survived being 158.33. This could point to survival being increasingly likely over longer time frames, perhaps due to medical intervention or higher mortality risk early after heart failure. On the other hand, this could also be a result of observation bias or survival bias as there may have been increased studies done on long-term survivors, skewing results.

For the purposes of this model, I decided to focus on features whose absolute correlation values were at least equal to 0.1. This threshold was found to improve my random forest accuracy through experimentation, and resulted in the examined features being age, ejection_fraction, serum_creatinine, serum_sodium, and time.

I decided to examine these features using the .describe() functionality, to get a better idea of their distributions. This led to the following:

	age	ejection_fraction	serum_creatinine	serum_sodium	time
count	299.000000	299.000000	299.00000	299.000000	299.000000
mean	60.833893	38.083612	1.39388	136.625418	130.260870
std	11.894809	11.834841	1.03451	4.412477	77.614208
min	40.000000	14.000000	0.50000	113.000000	4.000000
25%	51.000000	30.000000	0.90000	134.000000	73.000000
50%	60.000000	38.000000	1.10000	137.000000	115.000000
75%	70.000000	45.000000	1.40000	140.000000	203.000000
max	95.000000	80.000000	9.40000	148.000000	285.000000

Figure 2: Table of features and their descriptive statistics

Since each of these were continuous values, I decided to further examine their distributions when accounting for death_event. In other words, I separated the dataset into two distinct sets, one where patients survived during their follow-up period, and one where they died, then compared the results for each feature. This can help provide more initial insight into which features may be most important in predicting death_event. This was visualized using box plots.

2a. Age

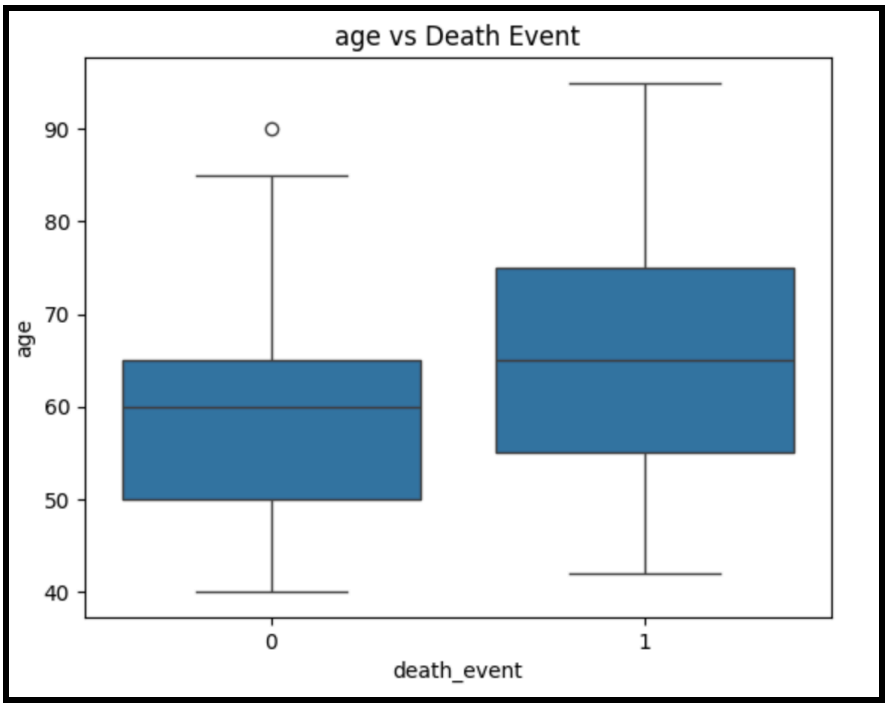

Figure 3: Box plots showing age distribution of surviving and dying patients

This boxplot suggests that those patients who died tended to be older than those that survived. The median, first quartile, and third quartile ages are each higher for patients that died compared to those who survived. There were some outliers (notably, the patient aged close to 90 that survived), but the trend suggests that an age at or above 70 greatly increases mortality rate. There is also a significant amount of overlap in the 55-65 range, which suggests other factors may dominate if age is within a certain threshold.

2b. Ejection Fraction

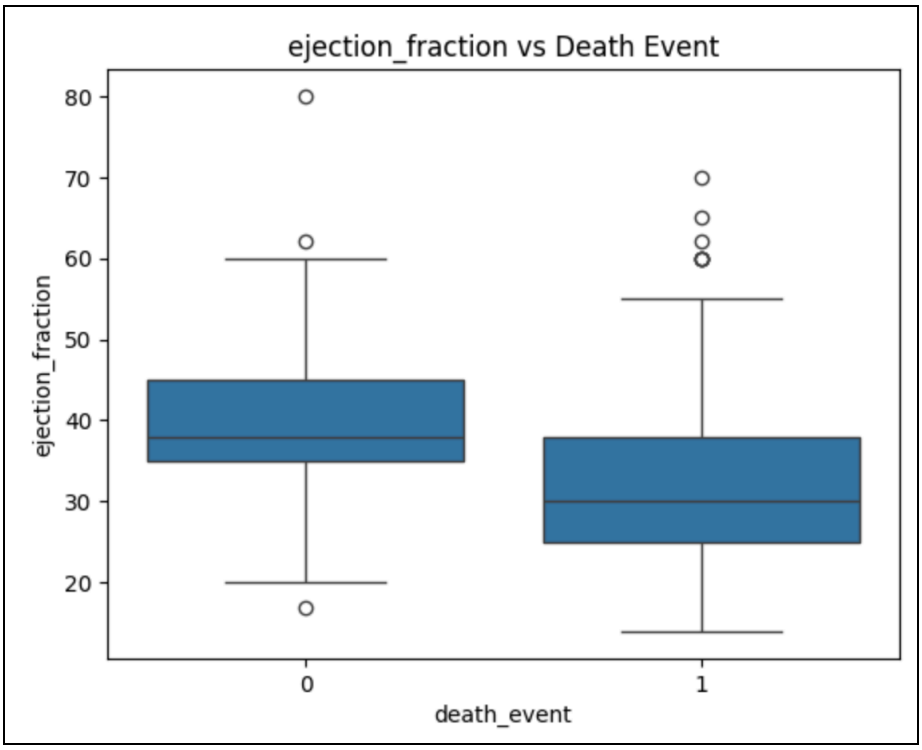

Figure 4: Box plots showing ejection fraction distribution of surviving and dying patients

This boxplot suggests a large discrepancy in the ejection fraction amount for patients who died compared to those who survived. Patients who survived tended to have an ejection fraction amount of about 35%, compared to those who died at about 30%. The interquartile range (25%-50%) of patients that survived had overall higher ejection fraction amounts than those who died, with 35%-45% compared to 25-40%. This implies that being able to pump more blood out of the heart leads to higher chances of survival, which is logical as being able to pump more blood could be correlated with a stronger heart.

2c. Serum Creatinine

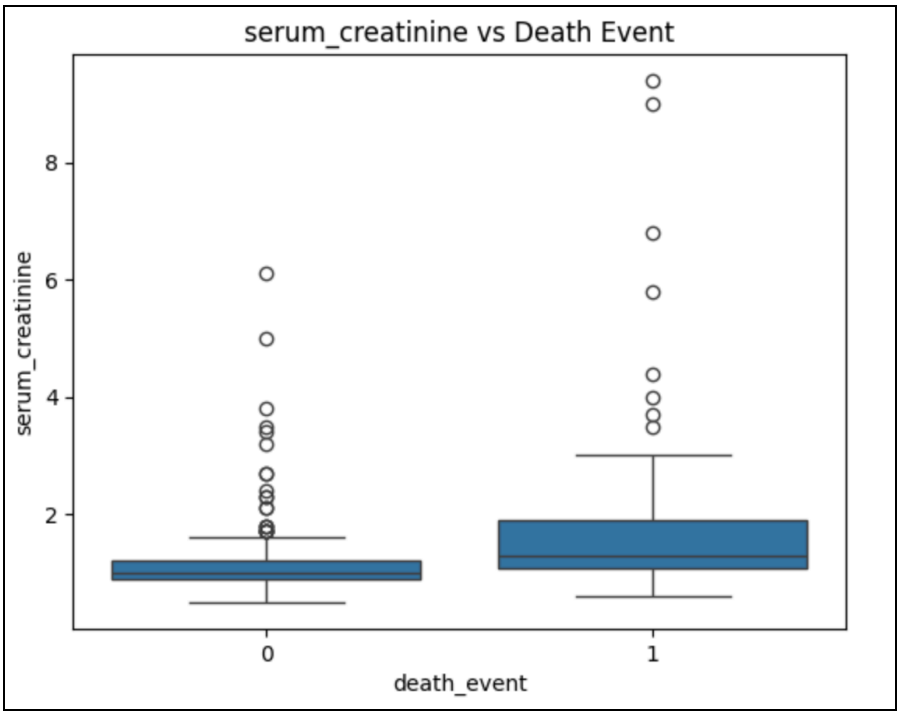

Figure 5: Box plots showing serum creatinine distribution of surviving and dying patients

This boxplot is less meaningful compared to the others due to the large number of outliers indicated by the white circles. The middle 50% of patients that survived tended to have less serum creatinine levels in their blood compared to those that died (about 1 mg/dL compared to about 1.75 mg/dL), but the range for patients that died was much larger, ranging from 0.5 mg/dL to 9.4 mg/dL. However, both distributions seem to have many outliers making it difficult to draw many conclusions from this data alone.

2d. Serum Sodium

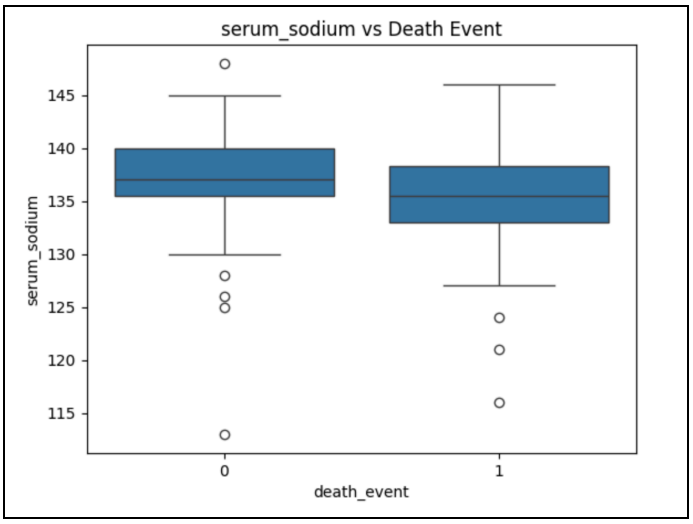

Figure 6: Box plots showing serum sodium distribution of surviving and dying patients

This boxplot suggests that patients who survived tended to have more serum sodium in their blood compared to those who died, with a median of about 137.5 mEq/L compared to about 135 mEq/L. The middle range was also larger, but interestingly, the maximum values were around the same at about 145 mEq/L (excluding the outlier for the death_event=0). Overall, this seems to point towards the opposite trend compared to the serum creatinine.

2e. Death Event

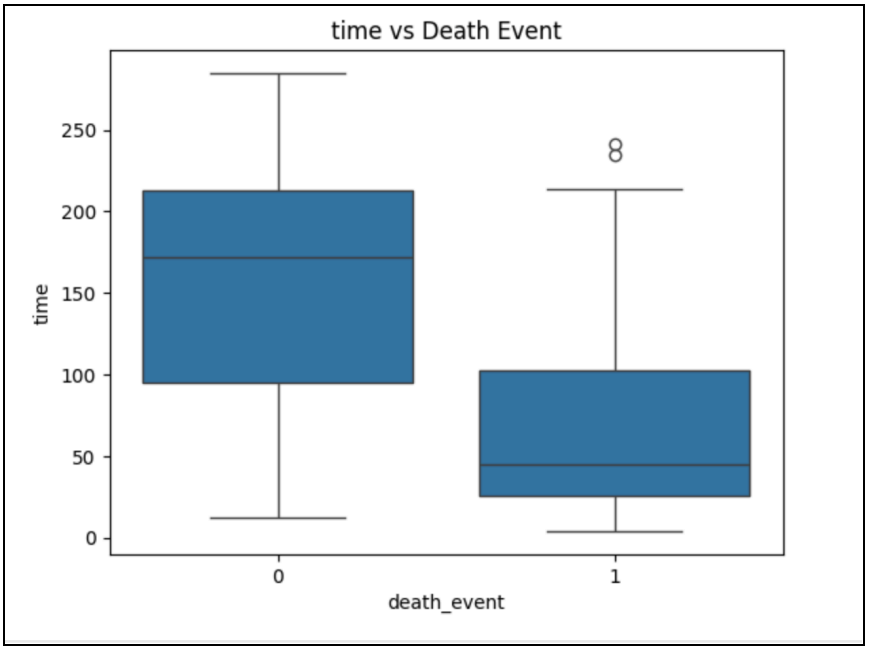

Figure 7: Box plots showing follow-up time distribution of surviving and dying patients

This boxplot suggests a similar trend to that discussed above; the patients who survived tended to have longer follow-up periods compared to those that died, with median value at about 175 days compared to 50 days. This is the strongest discrepancy visualized, with a relatively firm threshold at about 100 days, suggesting that follow-up periods over 100 days tend to be correlated with survival, whereas periods under 100 days tend to be correlated with death. However, this is clearly not a firm rule as there are outliers and the overall distribution still has overlap.

3. Modeling

Since one seeks to utilize the above features to classify patients as either "likely to die within time period" or "not likely to die within time period", a random forest is an appropriate model to use due to its ability to high dimensionality resulting from multiple variables. A random forest is an ensemble learning method that combines multiple decision trees (each created during training), and aggregates their results to provide voting-based predictions. Decision Trees are useful in this context, because they can combine multiple variables (as is present here) to reach a classification, which can help show non-linear trends compared to methods like linear or logistic regression. Random forests are well-equipped to handle non-linear, high-dimensional data, and have the additional benefit of easily quantifying feature importance via impurity reduction. This can result in a clear visualization of which features were most important in predicting mortality.

The primary model was created by importing RandomForestClassifier from sklearn.ensemble; I started by performing a 80-20 train-test split on the original data, then created a classifier with 100 estimators (100 decision trees) and a specified random state. I then fit the model to the training data and extracted the predictions, which could then be used when analyzing the results. The overall code looks as follows:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn import tree
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


X_train, X_test, y_train, y_test = train_test_split(X, y['death_event'], test_size=0.2, random_state=0)
rf = RandomForestClassifier(n_estimators=100, random_state=0)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```

Below is 1 of the 100 decision trees created by the random forest, specifically the tree indexed at 0. This can provide an idea of the general structure that most trees likely resembled, but with varying thresholds and values.

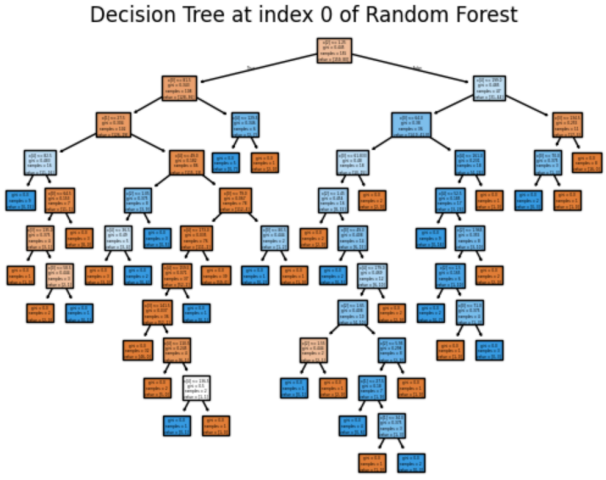

Figure 8: One of the decision trees created by random forest to predict death_event

4. Results + Analysis

Using the accuracy_score function, we could determine that the Random forest had an accuracy of about 0.8666 corresponding to 86.67%. This indicates that the Random Forest was able to correctly predict the death_event for 86.67% of the test-set, which comes out to 52 out of the 60 test data points. This is overall a relatively high accuracy score, and suggests that the random forest model was a strong choice in terms of predicting death_event from the subset of features. Interestingly, repeating the above method but with all features led to an accuracy of 85%, suggesting that our choice to filter the features on correlation was helpful. This is likely because using fewer features helps keep the dimensionality smaller, which can prevent overfitting on the training set, which in turn generally increases accuracy on the test set.

We can further examine the test results by looking at the confusion matrix:

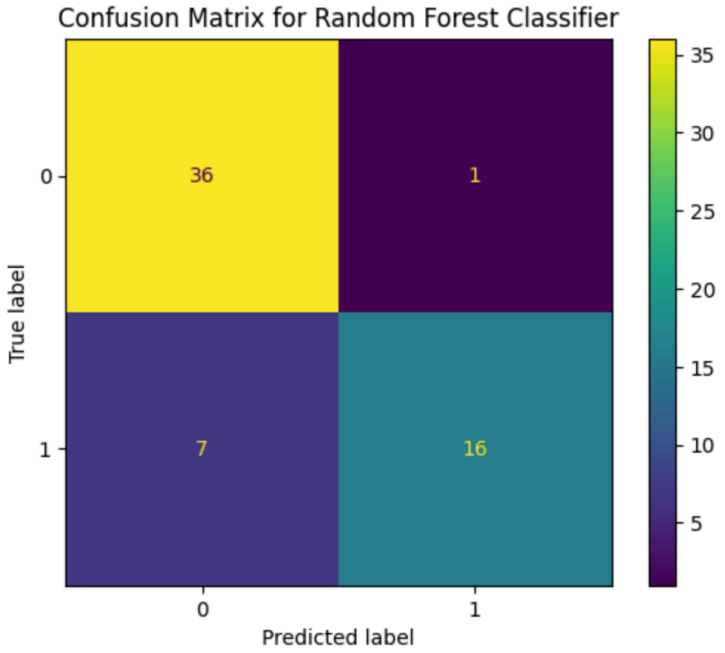

Figure 9: Confusion matrix for random forest classifier predictions and actual values

Our above confusion matrix indicates that the random forest produced 36 true negatives, 16 true positives, 1 false positive, and 7 false negatives. Depending on one's priorities, this can have mixed implications - the large number of true negatives and true positives suggests that the model does a good job of correctly assessing most patients' mortality given the other features. The fact that there was only 1 false positive indicates that the model will rarely predict that a patient will die, if they will actually live within the follow-up time period.

On the other hand, there was a non-trivial number of false negatives, suggesting that the model may predict survival when the patient will in fact die. This could be considered a weakness of our model, as we'd like to minimize false negatives that give patients an incorrect understanding of their situation following heart failure. Arguably, false negatives are also more harmful than false positives, as false negatives may result in weaker/less effective treatment being given under the pretense that the patient is not in danger, whereas false positives would just lead to increased caution without as much risk. These statistics are also reflected by the following:

```
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
```

```
Precision: 0.9411764705882353
Recall: 0.6956521739130435
F1 Score: 0.8
```

Here, precision indicates proportion of correctly predicted positive instances out of all instances predicted as positive, equivalent to minimizing false positives. Our high precision is in agreement with our earlier conclusion that our model has relatively few false positives.

Unfortunately, the recall value of 0.69 is relatively low - this indicates that of all actual positive instances (e.g patients who died), our model was only able to predict about 0.69 of their deaths. Ideally, the random forest would have a higher recall value even if it leads to lower precision; this could possibly be accomplished by using a different subset of features or adjusting the hyperparameters like number of estimators. The F1 score is a combination of recall and precision, its value of 0.8 is relatively average and suggests that the model is good, but not perfect, at minimizing false predictions. Further analysis can be done by examining the resulting ROC curve.

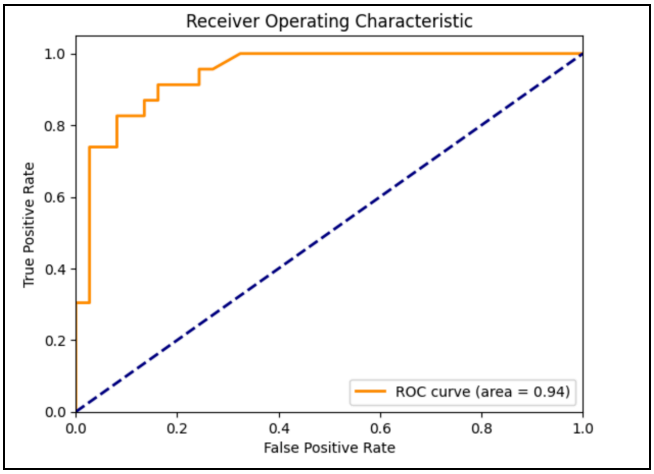

Figure 10: ROC curve for random forest

This ROC curve shows that our model can achieve a relatively high TPR (true positive rate) while minimizing FPR (false positive rate), such as by having a TPR of about 0.8 and FPR of about 0.1. Further improvements to TPR are possible, but would result in a much higher FPR; for instance, increasing the TPR to about 0.92 would require an FPR closer to 0.35. In this case, we'd like our model to have a high TPR rate since we'd like to predict as many deaths as

possible (to increase/improve treatment), so we may prefer to have a higher TPR even if it leads to an increased FPR rate. This can be done by adjusting the thresholds/parameters of the forest.

Finally, one can use the resulting Random Forest model to extract the relative feature importance scores for each of the selected features. This is done by examining, across all 100 random decision trees created, which features were most important in pruning, as quantified by GINI impurity. The resulting graph is shown below:

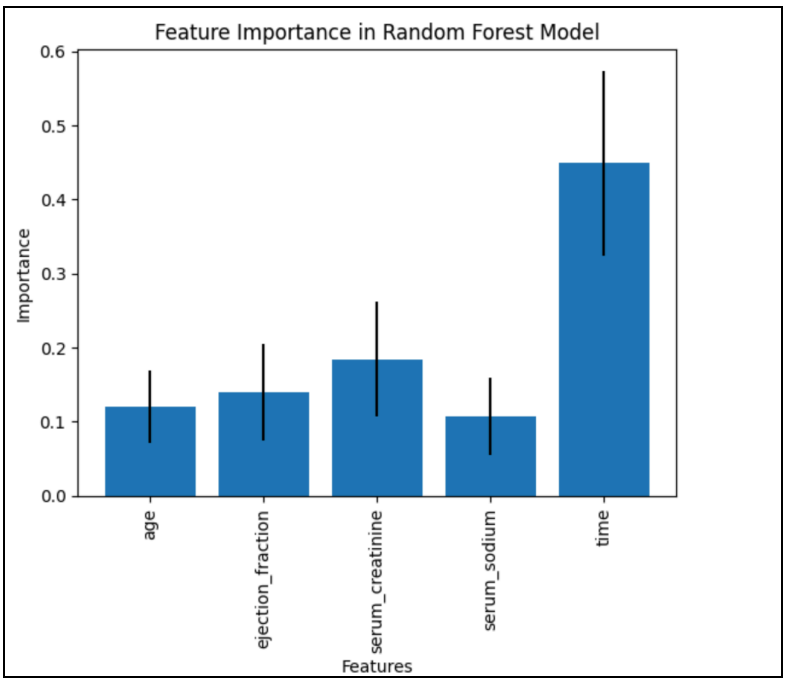

Figure 11: Bar chart showing the relative importance level of each feature in model

From the bar graph, it is clear that time was the single most important feature across the random forest. This is unsurprising, as our initial examination of the correlation and box plots led to a similar conclusion. This is also not very productive from a medical standpoint, since the observation period realistically has no effect on a person's mortality; rather, this could point to hidden trends like early mortality or survival bias, that are difficult to work off. More interestingly, the second-most important factor ended up being serum creatinine levels, followed closely by ejection fraction and then age. This suggests that serum creatinine levels could be pivotal in predicting and treating post-heart-failure conditions. Treatments that result in a favorable change in serum creatinine, or more blood pumped from the heart, could possibly result in higher chances of survival for patients. Finally, the importance of age suggests that older patients should be treated with increased caution when it comes to heart failure, as they're at a higher risk of dying afterwards.

5. Conclusion

Overall, the random forest model was a strong predictor for classifying patients as either dying or surviving a specified follow-up time period. Its strengths included a relatively high overall accuracy as well as an exceptionally low false positive rate, while its main weakness was its non-trivial false negative rate, which could be dangerous if interpreted incorrectly. The most important features that appeared to affect death_event were follow-up period time, serum creatinine levels, ejection fraction amount, and age. Serum sodium levels were also moderately important.

One possible way to improve the model is to take into account probabilities of survival/death as opposed to merely looking at binary outputs, since this is more accurate to the real world and can be less misleading if being practically used. Tweaking the hyperparameters (such as changing the number of estimators) could also result in increased accuracy or lower false negative rate. The ROC curve also suggests that adjusting the threshold could lead to a higher TPR, which may be desirable for a model of this nature.

One drawback to note of the data is the relatively small number of samples; the original dataset only had data for 299 patients, which likely isn't enough to create a representative sample of the larger population suffering from heart failure. Re-training the model with a larger dataset could lead to more accurate results when applied externally, but this is difficult to achieve without grabbing a larger dataset.

Another important result of using a small dataset, is that it's highly possible the random forest ended up overfitting on this small sample of patients. This is especially problematic if the patients all come from a similar background, because the results of the model may be skewed towards that specific population. One way to possibly reduce overfit is to manually limit the depths of the random decision trees; the single tree presented above had multiple layers as a result of the many features, which makes it more likely to overfit. Having a smaller depth could prevent overfit, though it would come at the cost of decreased accuracy on the training set.

In summary, this model should be taken in context when applied to new patients. The small sample size and possible overfitting render it relatively misleading if used on an arbitrary patient, and its relatively significant false negative rate can have worrying implications if used irresponsibly. More valuable could be the other takeaways from the mode, including feature importance; if doctors have access to the factors that most strongly predict death after heart failure, its possible that they can take measures to minimize those deaths, whether those are

measures before or after heart failure occurs. In this way, the results of the model can be important for those looking to address the systemic issue of heart failure fatality, and more analysis should be done to look further into other causes. For instance, more experimentation could be done on the other features of the dataset that were originally excluded, to decide what effect, if any, they have on fatality rates.

References

Heart Failure Clinical Records [Dataset]. (2020). UCI Machine Learning Repository. https://doi.org/10.24432/C5Z89R.

Johnson, Jon. “Congestive Heart Failure Life Expectancy: Prognosis and Stages.” *Medical News Today*, MediLexicon International, www.medicalnewstoday.com/articles/321538. Accessed 6 Dec. 2024.