FYS-STK 3155 project 3: A Study of Heart Failure

Theodor Midtbø Alstad

15th December 2020

Abstract

This report analyzes a dataset on heart failure published by user Larxel on Kaggle, using a custom Logistic Regressor, a custom Neural Network, a Random Forest from scikit-learn, and a custom Ridge regressor. It finds that scikit-learns Random Forest has the highest rate of accuracy, and that the Logistic Regressor has the highest rate of positives detected.

1 Introduction

Cardiovascular diseases are the leading causes of death in the world[1], and understanding their causes can help efforts in combating death.

This report discusses a dataset regarding heart failure[2], building on theory from previous projects[3][4]. This dataset contains 12 features, one of which is a response variable, and 299 datapoints.

In this project, I analyze the aforementioned dataset using 4 different methods, with the intent of finding the best algorithm for this dataset: ScikitLearns Random Forest, Logistic Regression, Neural Network, and Ridge linear regression.

Structurally, this report:

- 1: Builds on theory previously described in an earlier project[4].
- 2: Describing the dataset used.
- 3: Referencing other literature.
- 4: Describing what algorithms were used and what distinguishes them.
- 5: Providing the results found.
- 6: Discussing the results and finding conclusions.

2 Theory

I am using algorithms developed in a previous project[4], where the theory of Ridge, basic SGD, momentum, minibatching, varying step length, and l_2 regularization parameters is described, so I will refer to that, but will extend the theory on Neural Network and Logistic Regression.

2.1 Confusion matrix

When using a binary classification algorithm, you could either get a correct answer or an incorrect answer. This could be further detailed by describing in what way the classification is correct or incorrect, by describing whether the algorithm is correct or incorrect and what option it chose. You can describe this detail using a confusion matrix:

Table 1: An example confusion matrix

	Positive	Negative
Pred. Positive	True Positive	False Positive
Pred. Negative	False Negative	True Negative

2.2 Logistic Regression

Building on the theory described in my previous report[4], this section goes into more detail on the actual algorithm of my Logistic Regressor.

2.2.1 Prediction

A Logistic Regressor takes a set of datapoints, $\mathbf{X}$, and uses a set of weights, θ, in a matrix multiplication $\mathbf{X} \times \theta = \mathbf{z}$ with the shape $(1 \times p) \times (p \times k) = (k \times 1)$. Since this is a matrix multiplication $\mathbf{X}$ can be expanded to contain multiple sets of datapoints, $\mathbf{X} \in \mathbb{R}^{n \times p}$ to make multiple predictions at once without changing the weights. The resulting z is finally put through an output function, here Softmax, giving a final prediction.:

$$\text{softMax}(z) = \frac{e^z}{\sum_{i=0}^{k-1} (z_i)}$$

2.2.2 Correction/Backpropagation

The first time a Logistic Regressor is used θ is somehow initialized to a set of values, here randomly uniformly distributed between 0 and 1. The first prediction a Logistic Regressor makes is likely to be very wrong due to the random weights, and this is where the training process starts. When the regressor makes a prediction, it is compared to set of values that is known to be the correct prediction for the given datapoints, $\bar{z}$, and an error is generated by using the appropriate cost function, here simply being the difference between the two: $C(z) = z - \bar{z}$. This error is then applied to the weights through matrix multiplication, $\theta \leftarrow \theta - \mathbf{X}^T \times C(z)$. This prediction-and-correction cycle is repeated for a number of sets of data, repeatedly training the regressor, becoming more and more accurate.

2.3 Neural Network

A Neural Network functions like an extension to the Logistic Regressor, where there are layers, called hidden layers, put between the input and the output

layers. The network makes predictions, much like the Logistic Regressor, by multiplying the input, $\mathbf{X}$ by a set of weights and putting the result through an activation function (sigmoid, ReLU, etc.), except that for the network, this result is multiplied by another set of weights and put through the activation function again, repeating for each hidden layer (creating what is known as activation values, a_{ln} for a single node or a_l for an entire layer, the output of a node in a layer that is to be put through the activation function, where $a_0 = \mathbf{X}$) until an output is generated; for a network with $l = 3$ hidden layers, it would function as $((((\mathbf{X} \times \theta_0) \times \theta_1) \times \theta_2) \times \theta_f) = \mathbf{X} \times \prod_{i=0}^{2}(\theta_i) \times \theta_f = z$. Similarly to the Logistic Regressor, an error-value is generated, $\epsilon = z - \bar{z}$, and is backpropagated through the network in a similar, if more convoluted, way; $\theta_i \leftarrow \theta_i - (\epsilon \times \theta_f f'(a_l)) \times (\theta_{l-1} f'(a_{l-1})) \times (\theta_{l-2} f'(a_{l-2}))... \times (\theta_i f'(a_i)) \times f(a_i)$, where f() is the activation function.

The fact that the same error is multiplied throughout the network can lead to issues where the gradient, how much the weights change, vanishes or explodes as it is calculated further down the network, but the details if this occurrence is not within the scope of this report.

Each node can use a different activation function, and each layer could have a different number of nodes, but in the case of the network used in this report, each layer has the same number of nodes, and all nodes use the same activation function (sigmoid, which was found through a grid search to give optimal results) except for the final ones (using the softMax function).

2.4 Random Forest

2.4.1 Decision Trees

Since the structure of a Random Forest relies heavily on the structure of a decision tree, it is only natural to describe it first.

A decision tree tries to predict data by separating it into several binary or discretized decisions, splitting into a sub-tree for each possible decision, repeating this process until it comes to a "leaf", where the training algorithm has decided on a prediction. A relevant example could be the algorithm asking whether the patient smokes, going to the sub-tree for "no" if the patient doesn't smoke, then asking whether the ejection_fraction is above or below 0.4, going to the relevant sub-tree, finally asking whether the patient has diabetes, and reaching a decision on DEATH_EVENT from this. Of note is that any continuous feature has to be discretized to fit into a decisions tree, usually by defining ranges and creating sub-trees based on what range the relevant feature is in (as was shown in the relevant example).

A decision tree is very simple to implement, use, and interpret, is good at predicting, but is very prone to overfitting[5].

2.4.2 Reducing Overfitting

In order to reduce overfitting in decision trees, they are organized into structures called Random Forests, which contains multiple decision trees, but each decision tree is constructed from a bootstrapped[3] version of the dataset, and estimating a prediction based on the prediction in each component decision tree.

2.5 Other literature

Sakhiya, Nayan[6] graphically describes each individual feature in terms of its correlation to DEATH_EVENT and decides to model the data using the features "time", "ejection_fraction", and "serum_creatinine" and manages to get an accuracy of up to 93.33% with a Gradient Booster Classifier and 90.00% with a Random Forest Classifier and Logistic Regressor. They do, however, inadvisably use the time feature and use a random state in their train-test data split that, according to a commenter named BloodCoder16, would lead to an accuracy of 85% with other random states. Whilst this leads to doubt about the useability of the predictions themselves, the individual features are well analyzed.

3 Dataset

The dataset used[2] deals with heart failure as a binary classification problem. There are 12 input data and 1 response datum. It was published by the user Larxel on Kaggle June 2020. 6 of the features are discrete and 6 are continuous. The data is scaled by dividing each feature by the average value of that feature.

3.1 Features

Table 2: Input features

Feature	Description	(datatype) [Unit]
Age	Age of the patient	(float) [years]
Anaemia	Whether the patient is anaemic	(boolean) [true/false]
Creatinine Phosphokinase	Level of the CPK enzyme in the patients blood	(float) [mcg/L]
Diabetes	Whether the patient was diabetic	(boolean) [true/false]
Ejection Fraction	Percentage of blood in heart being pumped out at each contraction	(percentage) [%]
High Blood Pressure	Whether the patient has hypertension	(boolean) [true/false]
Platelets	Level of platelets in the blood	(float) [kiloplatelets/mL]
Serum Creatinine	Level of serum creatinine in the blood	(float) [mg/dL]
Serum Sodium	Level of serum sodium in the blood	(float) [mEq/L]
Sex	Whether the patient is male of female	(binary) [male/female]
Smoking	Whether the patient smokes	(boolean) [true/false]
Time	The duration of the study on the subject	(integer) [days]

The time feature is the amount of time the study went on for and is ignored here because it is only gained at the end of the study and can thus not only only be known ahead of time and is not useable as a predictor, but also has a strong correlation to DEATH_EVENT, and will artificially inflate accuracy.

The sex feature is given as binary, but will be treated as a boolean (using the already binary 0 and 1 as false and true, respectively) for ease of implementation.

Table 3: Response features

Feature	Description	(datatype) [Unit]
DEATH_EVENT	Whether the subject died during the study	(boolean) [true/false]

A negative response variable indicates that the study terminated without the subject dying. It is assumed that this means that the subject was written off as healthy and does not pose a risk of dying of heart disease.

4 Methods

4 algorithms were tested on this dataset, 3 of which were custom-made (Random Forest was imported from sklearn[7], included to include professionally constructed algorithms) and 3 of which is traditionally used for classification (Ridge is typically used for regression on continuous response variables, included to include non-traditional methods). They were all evaluated using a grid search to find optimal parameters

5 Results

5.1 Features

Features represented using tables and histograms to show their individual correlation to DEATH_EVENT. The tables include an extra row showing survival rate for each possibility. The histograms have been normalized for a total area of 1 for ease of comparison. A higher DEATH_EVENT False divided by DEATH_event True, i.e. a higher survival rate, is better.

Table 4: Table of correlation between boolean anaemia and DEATH_EVENT

	anaemia True	anaemia False
DEATH_EVENT True	46.00	50.00
DEATH_EVENT False	83.00	120.00
DEATH_EVENT False divided by DEATH_EVENT True	1.80	2.40

Table 5: Table of correlation between boolean diabetes and DEATH_EVENT

	diabetes True	diabetes False
DEATH_EVENT True	40.00	56.00
DEATH_EVENT False	85.00	118.00
DEATH_EVENT False divided by DEATH_EVENT True	2.12	2.11

Table 6: Table of correlation between boolean high_blood_pressure and DEATH_EVENT

	high_blood_pressure True	high_blood_pressure False
DEATH_EVENT True	39.00	57.00
DEATH_EVENT False	66.00	137.00
DEATH_EVENT False divided by DEATH_EVENT True	1.69	2.40

Table 7: Table of correlation between boolean sex and DEATH_EVENT

	sex True	sex False
DEATH_EVENT True	62.00	34.00
DEATH_EVENT False	132.00	71.00
DEATH_EVENT False divided by DEATH_EVENT True	2.13	2.09

Table 8: Table of correlation between boolean smoking and DEATH_EVENT

	smoking True	smoking False
DEATH_EVENT True	30.00	66.00
DEATH_EVENT False	66.00	137.00
DEATH_EVENT False divided by DEATH_EVENT True	2.20	2.08

Table 9: Table of correlation between boolean DEATH_EVENT and DEATH_EVENT. This is included for completeness

	DEATH_EVENT True	DEATH_EVENT False
DEATH_EVENT True	96.00	0.00
DEATH_EVENT False	0.00	203.00

From the tables, the two features that stand out as having a larger individual impact are high blood pressure and anaemia with a decreased survival rate of 29.58% and 25.00%, respectively, followed by smoking and sex, increasing survival rate by 5.77% and 1.91%. respectively.

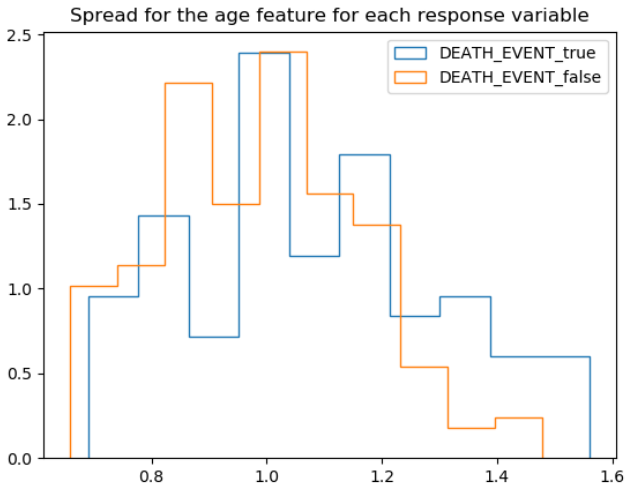

Figure 1: Histogram presenting the age feature for DEATH_EVENT being True and False

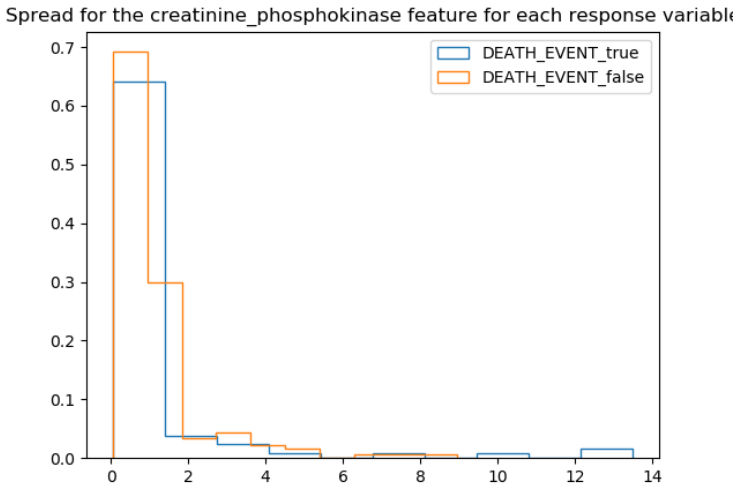

Figure 2: Histogram presenting the creatinine_phosphokinase feature for DEATH_EVENT being True and False

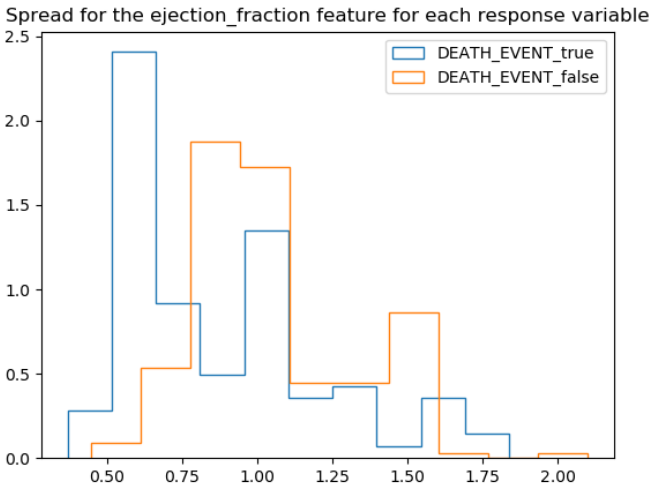

Figure 3: Histogram presenting the ejection_fraction feature for DEATH_EVENT being True and False

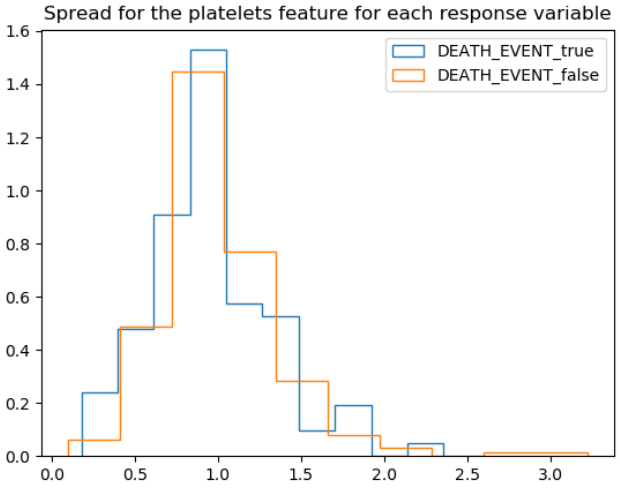

Figure 4: Histogram presenting the platelets feature for DEATH_EVENT being True and False

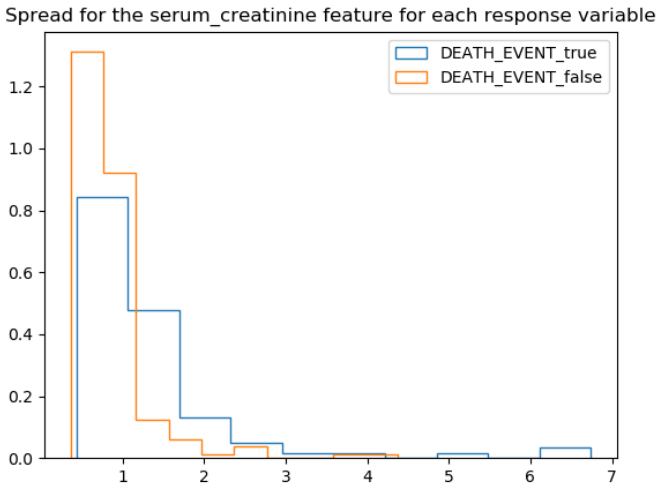

Figure 5: Histogram presenting the serum_creatinine feature for DEATH_EVENT being True and False

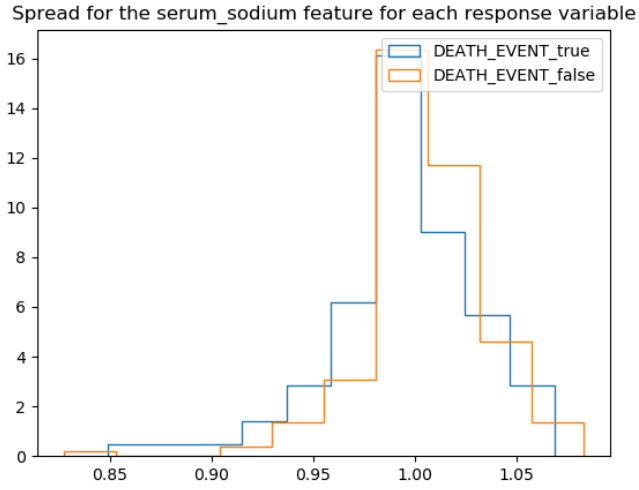

Figure 6: Histogram presenting the serum_sodium feature for DEATH_EVENT being True and False

Given time constraints, the histogram correlations will not be numerically analyzed, but I will refer to N. Sakhiya[6] choosing ejection_fraction and serum_creatinine as strong predictors.

5.2 Prediction power

All confusion matrices generated as an average over 1000 different train/test splits and trainings. The confusion matrices are generated from the testing samples and thus contain $ceil(299/4) = 75$ total samples.

Table 10: Confusion table for Neural Network as an average over 1000 runs, with an overall accuracy score of 70.07%

Neural Network	Positive	Negative
Pred. Positive	8.13	6.45
Pred. Negative	15.99	44.42
Rate of Detection	0.34	0.87

Table 11: Confusion table for Random Forest as an average over 1000 runs, with an overall accuracy score of 73.50%

Random Forest	Positive	Negative
Pred. Positive	9.96	5.79
Pred. Negative	14.08	45.17
Rate of Detection	0.42	0.89

Table 12: Confusion table for Ridge as an average over 1000 runs, with an overall accuracy score of 73.14%

Ridge	Positive	Negative
Pred. Positive	9.50	5.43
Pred. Negative	14.71	45.35
Rate of Detection	0.40	0.89

Table 13: Confusion table for Logistic Classifier as an average over 1000 runs, with an overall accuracy score of 73.42%

Logistic Classifier	Positive	Negative
Pred. Positive	10.71	6.57
Pred. Negative	13.37	44.36
Rate of Detection	0.45	0.87

6 Discussion

Looking at the predictive power of each algorithm, All but the Neural Network perform equally well an overall classifier. Surprisingly enough, I implemented Ridge as an attempt to demonstrate that it is an algorithm traditionally used to regress on continuous values and did not expect it to perform on par with other methods. As an overall predictor, one could pick any one of Random Forest,

Ridge, and Logistic Classifier, but given the nature of the dataset, in that it deals with heart failure, weight should be put on minimizing false negatives or maximizing rate of positive detection, which the Logistic Classifier seems to do the best, at 45% detection rate compared to the closest Random Forest, performing at a 42% detection rate.

7 Conclusion

In this report I have analyzed a dataset on heart failure, looked into the predictive power of individual features, looked at overall predictive power and predictive power for certain modes of the response variable DEATH_EVENT. I found that scikit-learns Random Forest has the most overall accurate predictions, but not significantly more accurate than Ridge and the Logistic Classifier, and I found that the Logistic Classifier has the highest rate of positive detections (i.e. the lowest rate of false negatives compared to true positives).

7.1 Considerations for improvements

This report includes training on all features, even though several have been demonstrated to have much higher predictive power and feature selection, like seen done by N. Sakhiya[6], could be beneficial. Unfortunately, there has not been enough time to make use of the analysis on the individual impacts from each feature.

Given the nature of this dataset, and how false positives are preferable over false negatives, the regressions could have benefited from changed threshold in prediction, so that not just the most likely outcome was predicted, but rather positives had a bias so that the false positives detection could with benefit be artificially inflated.

References

[1] "The top 10 causes of death." World Health Organization, https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death, Dec. 2020. Accessed: 2020-12-12.

[2] Larxel, "Heart Failure Prediction." Kaggle, https://kaggle.com/andrewmvd/heart-failure-clinical-data, June 2020. Accessed: 2020-11-28.

[3] T. M. Alstad, *FYS-STK 3155 project 1.* Oct 2020.

[4] T. M. Alstad, *FYS-STK 3155 project 2.* Nov 2020.

[5] P. Flom, "Why do decision trees have a tendency to overfit to the training set? - quora." Quora, https://www.quora.com/Why-do-decision-trees-have-a-tendency-to-overfit-to-the-training-set?share=1, Oct 2019. Accessed: 2020-12-14.

[6] N. Sakhiya, "Heart fail:analysis and quick-prediction." Kaggle, https://kaggle.com/nayansakhiya/heart-fail-analysis-and-quick-prediction. Accessed: 2020-12-13.

[7] "scikit-learn." https://scikit-learn.org/stable/index.html. Accessed: 2020-12-12.