Universitat Politècnica de Catalunya

Machine Learning

Survival Prediction of Heart Failure Patients

Author:

Benjamí Parellada

Fall Term 2021/2022

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Contents

1 Introduction

Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide [2, 6]. The rising epidemic of CVDs is primarily attributed to population ageing and increasing rates of obesity, diabetes, and other risk factors. Heart failure is a common event caused by CVDs, however a smart understanding of which characteristics are predictors of defunction could help these patients survive.

The objective of this project is to predict, using various machine learning methods, the survival of patients who suffered heart failure based on a number of different medical features. To achieve this goal, we have at our disposal a dataset from UCI [4] from heart failure patients which was donated by Davide Chicco after being used in his paper [2]. A second objective is to check our results against Chicco's and see if they are comparable.

The dataset contains 299 unique registers each with 13 distinct medical features. All 299 patients suffered *left ventricular systolic dysfunction* (heart failure), after this event, we are trying to predict if the patient will die/survive in the follow-up period. The event is summarized as a binary feature which will be the target feature we are trying to predict. To be able to predict the defunction, we have 12 other medical attributes available, some numeric (age, levels of blood serum bio-markers, etc.) and some binary (does the patient have diabetes, is the patient a smoker, etc.).

Using these other features, we train and fit various machine learning classifiers in order to predict if the patient will die in the follow-up period. The classifiers that seem to work the best are, in order: Quadratic Discriminant Analysis, Multi-Layer Perceptrons, and Random Forests. We obtain accuracies around 75%, comparable to the results presented in Chicco's work [2]. The main features that help in predicting the survival are Ejection Fraction and Serum Creatinine, the same as they found in the mentioned literature.

Our results conclude that having high values of Serum Creatinine is a strong indicator that the patient will not survive. On the contrary, having high values of Ejection Fraction seem a strong indicator that the patient will survive.

The remainder of the project is organized as follows. In section 2 a brief background of the previous work done with this dataset is presented. Section 3 presents a more qualitative exploration of the data: explanation of the features, their relationship to the target feature, and pre-processing done. As well as applying some clustering and dimensionality reduction techniques to see if we can understand our data better. Afterwards, in section 4, we present the models we have used, why we have used them, and the results of these models. Moreover, we try and explain how some of these models try to classify. Finally,

in section 5 we present the conclusions to the project, as well as some things we could have done better or differently.

2 Previous Work

There are 2 main bodies of work done on this dataset:

- The first study, Tanvir Ahmad et al. [1], was focused on survival analysis of heart failure patients who were admitted to the Institute of Cardiology and Allied hospital Faisalabad-Pakistan during April-December (2015). The study uses classical statistical models such as Cox regression to model mortality considering various of the available features; as well as Kaplan Meier plots to study the pattern of survival, and Martingale residuals were used to assess the functional form of variables.

  The most relevant risk factors for mortality among heart failure patients were: Age, Renal Dysfunction, Blood Pressure, Ejection Fraction and Anemia.

- The second study, Davide Chicco [2], was focused on predicting patients' survival from their data and finding the most relevant features among those included in their medical records. The study uses several machine learning classifiers to both predict the patients survival, and rank the features corresponding to the most important risk factors. The classifiers include one linear statistical method (Logistic Regression), three tree-based methods (Random Forests, One Rule, Decision Tree), one Neural Network (Multi-Layer Perceptron), two Support Vector Machines (linear, and with Gaussian radial kernel), one instance-based learning model (K-Nearest Neighbors), one probabilistic classifier (Naïve Bayes), and an ensemble boosting method (Gradient Boosting).

  The most relevant risk factors for mortality among heart failure patients were: Serum Creatinine and Ejection Fraction.

Seeing the methods used in the second study, we will compare our results with theirs.

3 Data Exploration

In this following sections, we analyze and describe transformations done on the dataset previous to modeling. We also explore the dataset with some dimensionality reduction algorithms and clustering profiling.

This dataset [2, 4] consists of 299 records, each one with 13 features. It has 6 binary variables, including the target feature, and 7 numeric variables. We present a more in depth description of all the features and their summary statistics:

Categorical variables and Target

The dataset is not well balanced, there are only 32% of patients who die, which corresponds to 96 patients of 299. All the categorical variables are binary.

- **[TARGET] Death Event**: if the patient died during the follow-up period [1 → Yes]
- **Anaemia**: if the patient has Anaemia [1 → Yes]
- **High Blood Pressure (HBP)**: if the patient has hypertension [1 → Yes]
- **Diabetes**: if the patient has diabetes [1 → Yes]
- **Sex**: is the patient a man or a woman [1 → Man]
- **Smoking**: if the patient smokes or not [1 → Yes]

	Anaemia	**Diabetes**	**HBP**	**Sex**	**Smoking**	**Death Event**
Count	299	299	299	299	299	299
Unique	2	2	2	2	2	2
Top	No	No	No	Man	No	No
Freq.	170	174	194	194	203	203

Table 1: Summary statistics for the binary features and target feature.

Numerical variables

- **Age**: age of the patient (years)
- **Creatinine Phosphokinase (CPK)**: level of the CPK enzyme in the blood (mcg/L)
- **Ejection Fraction (EF)**: percentage of blood leaving the heart at each contraction (%)
- **Platelets**: platelets in the blood (platelets/mL)
- **Serum Creatinine (SC)**: level of serum creatinine in the blood (mg/dL)
- **Serum Sodium (SS)**: level of serum sodium in the blood (mEq/L)
- **Time**: follow-up period (days)

	Age	**CPK**	**EF**	**Platelets**	**SC**	**SS**	**Time**
Mean	60.83	581.84	38.08	263358.03	1.39	136.63	130.26
Std	11.89	970.29	11.83	97804.24	1.03	4.41	77.61
Min	40.00	23.00	14.00	25100.00	0.50	113.00	4.00
25%	51.00	116.50	30.00	212500.00	0.90	134.00	73.00
50%	60.00	250.00	38.00	262000.00	1.10	137.00	115.00
75%	70.00	582.00	45.00	303500.00	1.40	140.00	203.00
Max	95.00	7861.00	80.00	850000.00	9.40	148.00	285.00

Table 2: Summary statistics for the numerical features.

3.1 Relationship to Target

The following plots are the ones that we deem are the most relevant to predict the death event on the follow-up period after a heart failure. While none of these will predict the target feature by itself, we can clearly see there is some relationship between these and the target feature. For example, observing the distribution of the Serum Creatinine, we can see how the median for those who died is quite higher than the ones who did not die. As such, we can begin to think that having a large value of SC will have a negative impact on the chances of survival of a patient.

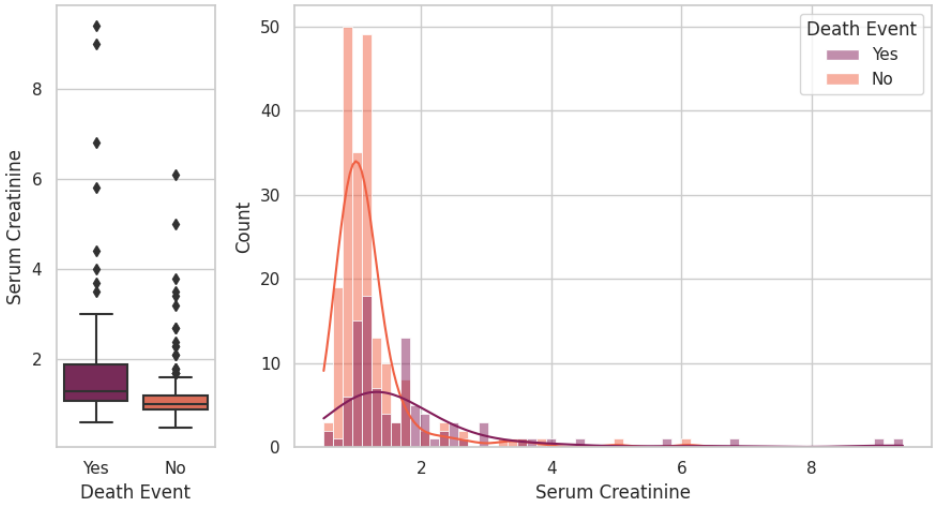

Figure 1: Boxplot and histogram of Serum Creatinine grouped by the death event.

The inverse can be said with the Ejection Fraction, it seems that higher values of EF represent a higher chance of survival compared to having lower values. Again, the relationship is not perfect and presents overlap.

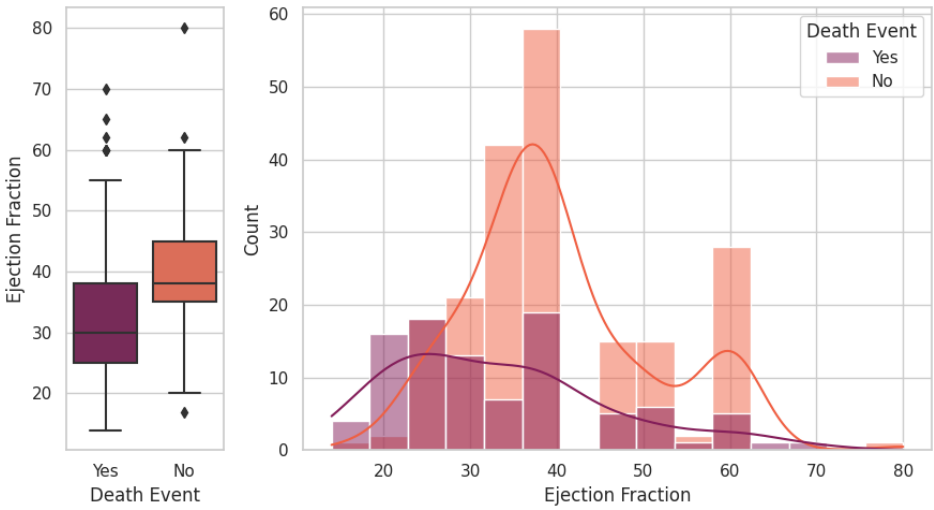

Figure 2: Boxplot and histogram of Ejection Fraction grouped by the death event.

3.2 Pre-processing

- **Missing Values**: we check the metadata to check if there are any values encoded to represent missing values, we also check the summary statistics to check if there are impossible/weird values that could represent a missing encoding. After thorough checking, there are no missing values in the presented database.

  Manually checking a subset of the data, there are various numerical values that exactly match the mean of its feature. For example, there are 11 values of Platelets that are exactly the mean of all the Platelets. This makes us believe that some kind of mean imputation on the data has been done beforehand, which was not described in the metadata.

- **Incoherent values**: while checking for missing values, we have checked to see if there are incoherent values, (negative values in values that can only accept positive values, etc.). The only incoherent value is one person's age which is not an integer, this value matches the mean of the attribute. We round it to the nearest integer.

- **Outliers**: in the presented graphs, we can definitely see that there are some variables that have values that we *could* consider as outliers. However, since we are not familiar with the medical terminology nor their concrete meaning, we rather leave these values in, instead of removing them. For example, in the following graph with CPK we can see the whiskers of the boxplot are around 1200, however is 8000 really that out of place to consider an outlier? Checking the other values outside of the IQR, we can see many values ranging from 2000 to 6000, as such 8000 does not really seem that out of place. Hence, since we are missing understanding of the features meanings, we decide to leave all values without removing any observations.

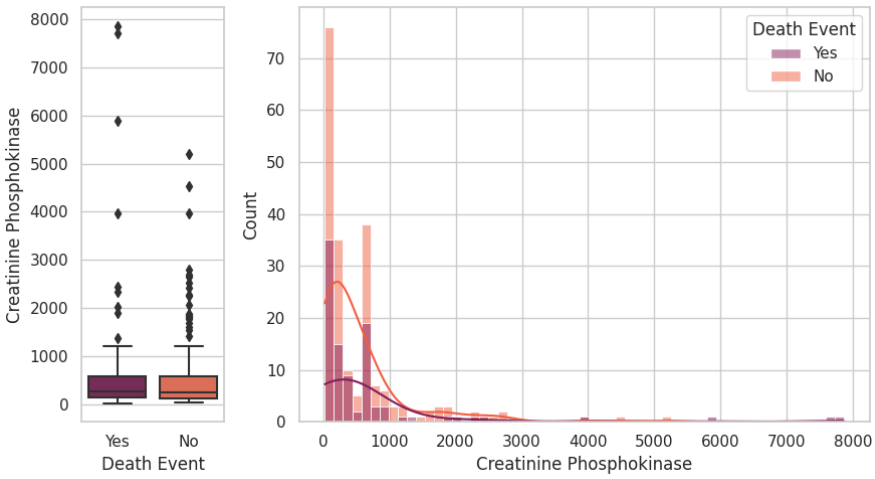

Figure 3: Boxplot and histogram of Creatinine Phosphokinase grouped by the Death Event.

– **Codification of variables**: as mentioned earlier, we have a few variables that are binary codified as 0-1. We leave these variables as they are, as in the end we would need to one-hot encode them, which would end with the same result.

– **Feature elimination**: we decide to eliminate Time, which is the follow-up of the patient in days. We can see how most patients who are going to die, die quite quickly in the first 100 days. As such, more intensive care could be given in the first few weeks after the Heart Failure to ensure the safety and health of the patients. While this previous analysis seems interesting, we think it is not very relevant in predicting if the patient will survive. The feature seems artificial to us, the impact of the follow-up period is not suitable to predict when a patient will die. We cannot increase or affect the follow-up period externally, its more related to when the data was obtained than an exogenous variable. Moreover, this feature presents a clear bias, as patients that die quickly will not be able to have longer follow-up periods.

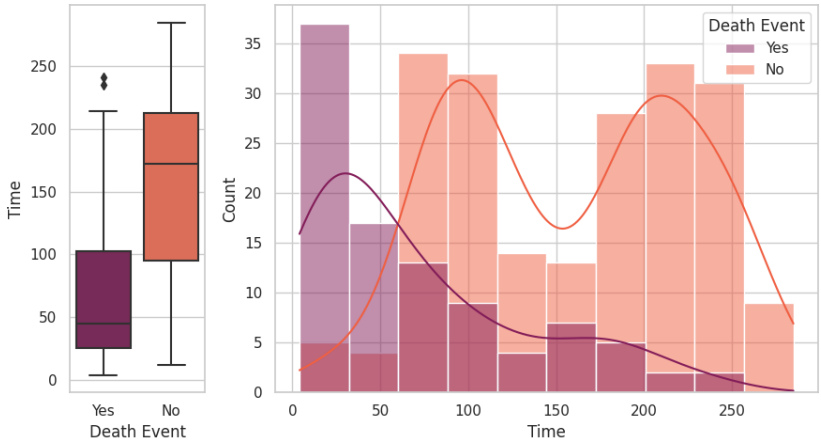

Figure 4: Boxplot and histogram of Time grouped by the death event.

– **Feature creation**: our knowledge of medicine is quite lacking, as such we decided not to try and create new variables with the ones given. We did try to categorize some numerical feature, however the results were insignificant most of the time or worse than leaving them as is.

– **Normalization**: we split the data into a few different normalization's, according to the needs of each of the machine learning algorithms. Concretely, *i*) no normalization, *ii*) standardization, and *iii*) normalization to the range have been applied. In all of these, we only normalize the numerical data, we ignore the ones that are semantically binary. We apply the normalization "fitted" from the train data to the test data.

– **Feature transformation**: after checking the normalized plots, we do not consider it necessary to correct any feature to correct skewness.

3.3 Clustering

In this section, we are going to try some different unsupervised learning algorithms and check if they can help us predict the target feature. We are doing this in an exploratory manner, we are not trying to concretely predict the death event, however we are trying to see if there is some structure in our data that could relate to the target feature.

Since we have features of different types, numeric and binary, we need to use a metric that can use both at the same time to compute distances with it. Concretely we use the Gower Similarity which was first defined by J. C. Gower [5]. It calculates the similarity between observations i and j by computing the average of partial similarities across all the features of the dataset. Partial similarities are defined for each data type, hence we get the similarity of numerical and binary features separately and then combine them by taking the average of the partial similarities. We wish to use this distance with hierarchical clustering [8], however GS is a distance that does not obey Euclidean geometry. So we use the square root of Gower Dissimilarity, $GD = \sqrt{1 - GS}$, which is in Euclidean space.

We present some of the results explored, however the profiling of the clusters does not seem to indicate anything useful to us. The values are way too mixed between labels for us to make a sensible interpretation of the clusters. Here we present an example of a scatterplot with EF and SC, we cannot clearly state any relationship between the labels and the features. Even when compared with the categorical values in barplots the results were not that significant. Moreover, the heatmap with the death event and the labels does not seem to present any significant relationship, other than random. Hence, the clustering process was not very successful in giving us new information about the data.

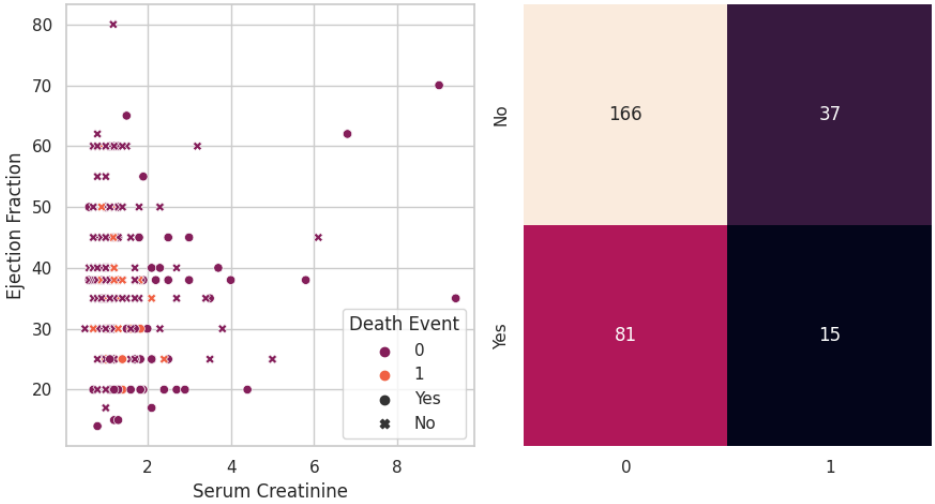

Figure 5: Results of the clustering. To produce the heatmap we assumed the label (x-axis) with the most values are from the people that survive. On the scatterplot, the shapes represent the target feature, while color represntes label from the clustering. There does not seem to be any clear profiling of each cluster.

3.4 Visualization

Before delving into predicting with machine learning, we try and visualize our data with various dimensionality reduction algorithms. Out of the ones tested, the only ones we deem relevant enough to present are the Principal Components Analysis (PCA) and the Linear Discriminant Analysis (LDA). There are some conceptual problems using these methods since we have to consider the binary data as numeric. There are corrections for a binary PCA as shown in Collins [3], however, since we are only focused in understanding our data, the results given seem alright.

The following graph is a biplot of the PCA, which contains all the data values transformed and plotted according to the first 2 Principal Components. There are also the features and their relative importance plotted as 'arrows'. The longer the arrow the more weight they have in the component. We can also see what features are correlated by checking how close their arrows are, for example Age and Serum Creatinine seem to be very related.

Since Age and Serum Creatinine are close to the positive axis of the first component (PC1), we can state that PC1 is strongly related to high values in these features. Analogously, we can state that high values of Serum Sodium and Creatinine Phosphokinase are more related to negative values in PC1. For example, higher values of Age and SC will result in values closer to 1, while the more SS and CPK the patient has will result in values closer to -1. While the previous statements are true, the relationship with each axis is not as perfect as we describe it, as there are other features affecting the component as well. The plot is quite an interesting result, as by representing each data point with its corresponding target feature there seems to exist a relationship with PC1. Values closer to 1 in PC1 seem to die more than values closer to -1. Hence, the older a patient is or the higher his SC is, the more chances he has to not survive the heart failure event.

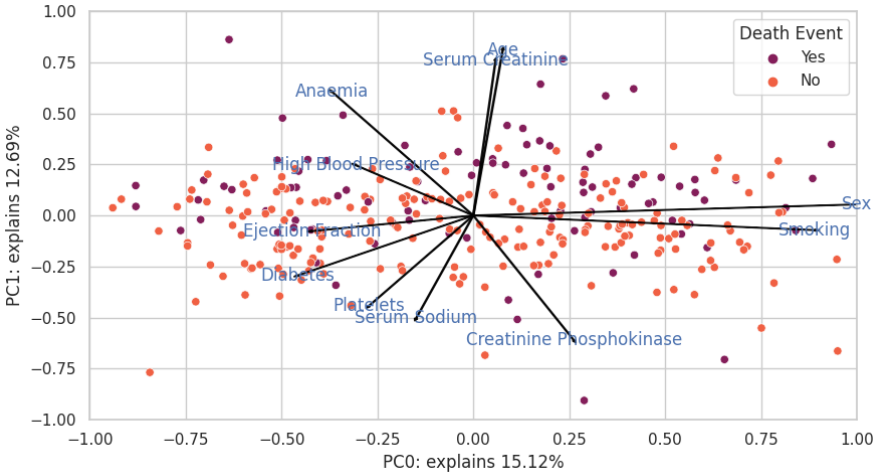

Figure 6: Biplot of the PCA with the first 2 components. The arrows represent each feature. We represent around 30% of the variance of the dataset with these components.

To round our visualization and introduce machine learning to classify the target feature, we use a Linear Discriminant Analysis (LDA) to reduce the dimensionality and plot the data with the maximum separability between classes. Again, we graph the biplot with the most important features to separate between the target value. Take into consideration that since our target feature is binary, the resulting plot would be 1-Dimensional. To visualize the results better, we added some jittering on the x-axis. The importance of each of the features is plotted skewed to the left to be able to see the data better. Again, the closer the feature is to 1 or -1 will result in more importance in separating the classes.

Here, we can again see how Serum Creatinine is the most important feature in order to discriminate, along with High Blood Pressure and if they have Anemia or Diabetes. All these values, the higher they are, or if they are true in case of Anemia and Diabetes, will result in higher chances of the person dying. Finally, if the person is male, the chances of the patient dying after heart failure seem to be much lower.

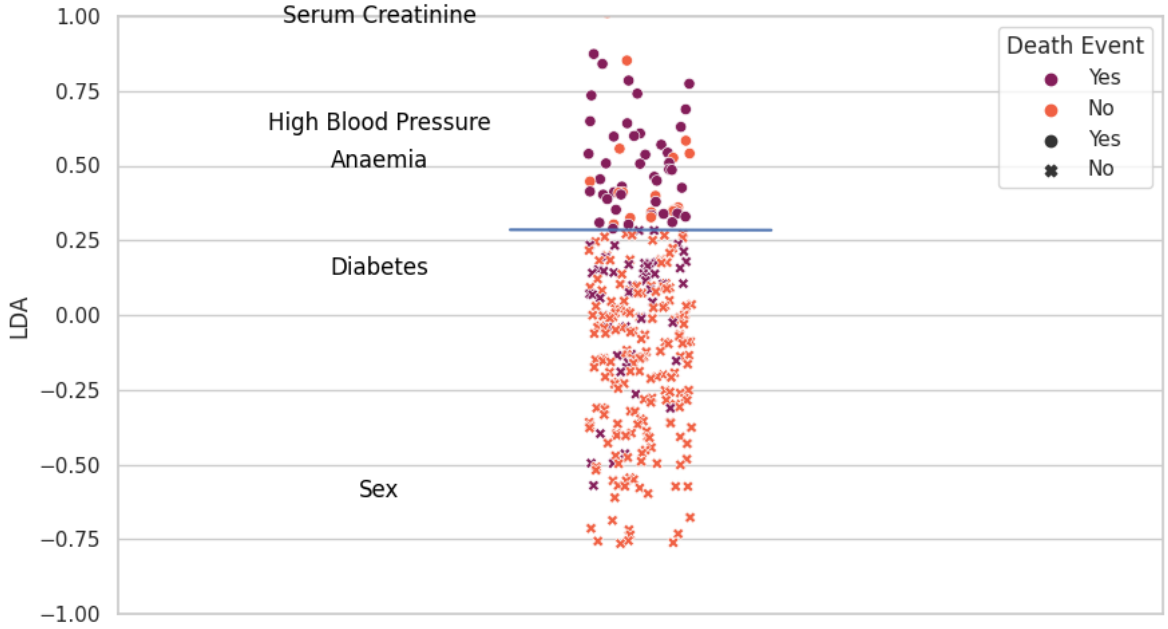

Figure 7: Linear Discriminant Analysis on the data. In blue, the decision boundary which separates where the LDA would predict each of the classes. The features with little importance are not plotted.

4 Modeling

In the following section, we describe the resampling protocol, evaluation metrics, models, and the results of our modeling.

4.1 Resampling Protocol

The first step is to split data into training and test sets. We decide to split the data 70-30 train-test. As such, from the 299 samples we will have 209 observations to train the models and 90 to evaluate the quality of our trained models. By virtue of this split, we can achieve a generalization error as close as possible to the error we would get using the model on completely new data. An important remark, whenever we scale the data, we apply the same scaling that has been "fitted" on the train partition on the test partition.

In order to tune the hyperparameters, we use grid search with k-fold cross-validation over the train partition and take the one that has the best recall.

Once the grid search is completed, we refit the model with the best hyperparameters found. In order to return an accurate estimate of how the model works, this refitting is done using k-fold cross-validation on the train partition as well. Afterwards, this same model is evaluated on the test partition.

4.2 Evaluation Metrics

We need to define our metrics before we start presenting results. Metrics will help us in quantifying the quality of the predictions, and choosing which model is better. Since our problem consists of classifying a binary class, we compare the amount of correct predictions the model returns versus the true event that happened. We will consider positive as 'Death Event' = 'Yes'. The following definitions and metrics are from [7].

We define:

- **True Positive (tp)**: a prediction that correctly indicates the presence of a condition. In this case, the model returns the patient will die ($\hat{y} = \text{Yes}$) and he did die ($y = \text{Yes}$).
- **True Negative (tn)**: a prediction that correctly indicates the absence of a condition. In this case, the model returns the patient will not die ($\hat{y} = \text{No}$) and he did not die ($y = \text{No}$).
- **False Positive (fp)**: a prediction that wrongly indicates the presence of a condition. In this case, the model returns the patient will die ($\hat{y} = \text{Yes}$) and he did not die ($y = \text{No}$).
- **False Negative (fn)**: a prediction that wrongly indicates the absence of a condition. In this case, the model returns the patient will not die ($\hat{y} = \text{No}$) and he did die ($y = \text{Yes}$).

Since the target feature is binary, these results translate to the confusion matrix.

$$Confusion\ Matrix = \begin{bmatrix} tn & fp \\ fn & tp \end{bmatrix}$$

With these basic defintions we can compute the following:

- **Accuracy**: the usual metric in classification problems, it represents the amount of instances that are correctly classified overall. In an unbalanced dataset this metric is not as useful. Consider a naïve classifier that always predicts that the patient will survive in our data, thus the accuracy for this classifier would be $203/299 = 0.68$ which seems like a decent result. However, the classifier would be useless in predicting people who die, which is the most relevant. We can calculate it as:

$$Accuracy = \frac{tp + tn}{tp + tn + fn + fp}$$

- **Balanced Accuracy**: the balanced accuracy normalizes the predictions of the tp and tn by the number of positives and negatives in the sample. In the naïve classifier described earlier, the balanced accuracy would return 0.5, which would be the same as predicting at random in a balanced dataset. We can calculate it as:

$$Balanced\ Accuracy = \left(\frac{tp}{tp + fn} + \frac{tn}{tn + fp} \right) /2$$

- **Precision**: the fraction of positive instances among all that are classified as positive. In the naïve classifier, we would get 0, which represents better that the model is not working properly. We can calculate it as:

$$Precision = \frac{tp}{tp + fp}$$

- **Recall**: the fraction of positive instances among all that are positive. In the naïve classifier, we would get 0, which represents better that the model is not working properly. We can calculate it as:

$$Recall = \frac{tp}{tp + fn}$$

- **F1**: the harmonic mean of the precision and recall, the closer the value is to 1, the better our classifier will be. We can calculate it as:

$$F1 = 2\frac{precision \cdot recall}{precision + recall}$$

– **AUC**: a receiver operating characteristic (ROC), is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied. It is created by plotting the fraction of true positives out of the positives versus the fraction of false positives out of the negatives, at various threshold settings. The Area Under the Curve (AUC) measures the area below the ROC curve and provides an aggregate measure of performance across all possible classification thresholds. The possible values are between 0.5 (random) and 1 (perfect classifier).

Given the nature of the problem, predicting people who will die after having a heart failure, it is more critical to identify all individuals who will die rather than misclassify some who will not. With this goal in mind, and since our data is a bit unbalanced, we consider the most important metric to be the recall of the positive class (Death Event = 'Yes'). We will cross validate maximizing this metric.

4.3 Models

We describe the models we have applied using Python's Scikit-Learn library [8] and the hyperparameters tuned in order to predict the death of the patient following a heart failure. Many of the information here is extracted from Scikit-Learn documentation [8], or the books *Pattern Recognition and Neural Networks* [9] and *Advanced Data Mining Techniques* [7]

Linear/Quadratic Models

– **Dummy Classifier** makes predictions that ignore the input features, returns the most frequent class label in the observed target feature. Will serves as a simple baseline to compare against other more complex classifiers.

– **Logistic regression** is a linear model for classification rather than regression. There is no real hyperparameter here, however we test the different available regularization penalty (`L1`, `L2`, `both`, `none`), and its regularization strength (`C`) while using the solver `saga`. We use the Standard Scaler for the numerical values as regularization makes the predictor dependent on the scale of the features.

– **Linear Discriminant Analysis** we already used an LDA before in this project, however this time we are going to use it in order to predict instead of just visualizing. There are no hyperparameters to tune, nor do we need any special normalization.

– **Quadratic Discriminant Analysis** similar to the LDA, however with a quadratic decision boundary, generated by fitting class conditional densities to the data using

Bayes' rule. We do not need any special normalization, however we can tune the regularization of the per-class covariance estimates.

- **K-Nearest Neighbors** is an instance-based learning method which does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the K nearest neighbors of each point.

  As such, by using distances, if we have features with very different orders of magnitude the larger ones will have greater effect on the predictions. So in order to fix this, we want to normalize the data such that no feature is overly represented just because of its magnitude. Hence, we scale the data between its minimum and maximum. There are quite a few hyperparameters to train, the amount of neighbors that vote K, if all points weigh equally or depend on the distance, and the distance metric to use for the tree, which by fixing at Minkowski and varying the Power parameter for the Minkowski metric we can get the other metrics.

- **Support Vector Machine** similar to the LDA and QDA, SVM's objective is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. It is highly recommended to standardize the data when using SVM. The main hyperparameters tuned are the regularization parameter C for the SVM, the kernel used, and the kernel parameters (linear: none, RBF: σ, polynomial: degree, sigmoid: σ). When using RBF or sigmoid as kernels the SVM is no longer a linear method, but non-linear.

Non-linear Models

- **Decision Tree** creates a model that predicts the value of the target variable by learning simple decision rules inferred from the data features. Since it is rule based, the scale of the values is irrelevant, therefore we do not need any scaling. There are quite a few hyperparameters, such as the function to measure the quality of a split, the maximum depth of the tree, the minimum number of samples required to split an internal node, and the minimum number of samples required to be at a leaf.

- **Random Forest** creates an ensemble of decision trees where each is built from a sample drawn with replacement (i.e. a bootstrap sample) from the training set. No scaling is needed and it has the same hyperparameters as the decision tree with a new one: the amount of trees in the forest.

- **Extremely Randomized Trees** similar to random forests, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of

these randomly-generated thresholds is picked as the splitting rule. It has the same hyperparameters as random forests.

- **Gradient Tree Boosting** it gives a prediction model in the form of an ensemble of decision trees. It is a generalization of boosting to arbitrary differentiable loss functions. The same hyperparameters as the decision tree can be found here, with a few new ones: the loss function (where using exponential will convert this to the AdaBoost), learning rate, the fraction of samples to be used for fitting the individual base learners, and the number of boosting stages to perform.
- **Neural Networks** concretely a Multi-Layer Perceptron (MLP) classifier where it can learn a non-linear function approximator for classification. MLP is sensitive to feature scaling, hence, we use a standard scaler. There are many hyperparameters to tune here, but the most relevant are the hidden layers of the model, the activation function, and the regularization.

4.4 Results

The following table summarizes all the evaluation metrics after the hyperparameters have been tuned. The presented results are the ones measured on the test data set, except the ones on the rightmost part of the table, which are the average results of the CV on the train set after finding the best parameters.

Model	Acc.	Pre.	Rec.	F1	AUC	B-Acc.	TN	FP	FN	TP	Acc. Tr.	Rec. Tr.
DC	0.677	0.000	0.000	0.000	0.500	0.500	61	0	29	0	0.679	0.000
LR	0.733	0.666	0.344	0.639	0.631	0.631	56	5	19	10	0.741	0.346
LDA	0.744	0.650	0.448	0.677	0.666	0.666	54	7	16	13	0.755	0.374
QDA	0.688	0.510	0.827	0.681	0.725	0.725	38	23	5	24	0.660	0.760
KNN	0.566	0.352	0.413	0.523	0.526	0.526	39	22	17	12	0.670	0.420
LSVM	0.722	0.625	0.344	0.629	0.623	0.623	55	6	19	10	0.732	0.389
KSVM	0.688	0.513	0.655	0.665	0.680	0.680	43	18	10	19	0.616	0.480
DT	0.722	0.576	0.517	0.672	0.668	0.668	50	11	14	15	0.741	0.583
RF	0.744	0.625	0.517	0.692	0.684	0.684	52	9	14	15	0.760	0.613
ET	0.677	0.500	0.379	0.603	0.599	0.599	50	11	18	11	0.755	0.492
GB	0.577	0.378	0.482	0.545	0.552	0.552	38	23	15	14	0.458	0.745
MLP	0.755	0.620	0.620	0.720	0.720	0.720	50	11	11	18	0.726	0.646

Table 3: Summary of the results evaluated on the test partition set of the data. True positive is set to be Death Event = 'Yes'. On the right we have the Validation Error after hyperparameter tuning.

Checking the accuracies on the test set, we see that most models return better results than the dummy classifier. However, before choosing the model with the highest accuracy,

remember that our data is unbalanced and it is more important to detect people who will die to try and prevent it. By fixating on accuracy, one would be quick to pick the MLP. The QDA, however, even though its accuracy is only slightly better than the dummy classifier, has the highest recall of any the models. As we stated too many times already, the recall is the most important metric to us for the negative socio-economical impact of having false negatives.

Taking a look at the other metrics, we can see how the balanced accuracy and AUC return their highest values with the QDA. Hence, fixing for the unbalance in the target feature, the QDA seems to be the best overall model for this data. And, over all the different decision thresholds the QDA will still outperform the other models.

Similarly to the analysis did in Chicco's work [2], the Random Forest and Decision Trees return some of the best accuracies and AUCs. In contrast, the GB and KNN seem to work much worse for us than it did for them. All the other results are quite comparable, although, unfortunately, they did not consider the QDA, nor SVM with polynomial kernels.

The following table summarizes the best hyperparameters found for each of the models.

Model	Parameters
LR	`C:1`, `l1_ratio:0.6`, `penalty:elasticnet`
QDA	`reg_param:0.8`
KNN	`metric:chebyshev`, `n_neighbors:3`, `weights:distance`
LSVC	`C:10`, `loss:hinge`, `penalty:l2`
KSVC	`C:0.1`, `kernel:poly`, `degree:2`
DT	`criterion:entropy`, `max_depth:7`, `min_samples_leaf:6`, `min_samples_split:2`
RF	`n_estimators:250`, `criterion:gini`, `max_depth:9`, `samples_leaf:2`, `samples_split:8`
ET	`n_estimators:200`, `criterion:gini`, `max_depth:None`, `samples_leaf:1`, `samples_split:4`
GB	`n_estimators:200`, `criterion:friedman_mse`, `learning_rate:0.8`, `loss:deviance` `max_depth:3`, `samples_leaf:1`, `samples_split:8`, `subsample:0.5`
MLP	`activation:tanh`, `alpha:1e-6`, `hidden_layers:(5,20,50)`, `learning_rate:constant`

Table 4: Summary of the best hyperparameters of the best models, there are other parameters that we have removed from this table, such as solvers, maximum number of iterations, etc. In DT, FR, ET, and GB the max features was best to 'None'.

4.5 Final Model

We want to first comment that no model seems to overfit the data. Checking the accuracies and recalls from the validation set, we can see there is no model where the metrics are way worse on the test set. Except perhaps the KNN and ET which do not seem to work as well on new data, however the results given are not horrible.

Overall, due to its high recall, we would consider the best model the QDA followed by the more balanced MLP. While the QDA worked adequately for what we consider important (detect all those who will die), it is classifying many of the patients as being positive, resulting in low precision. Even though precision can be fixed by repeating the tests to combine its probabilities, it is possible that medical professionals could consider the low precision of the model a hindrance or too expensive to repeat. Hence, if they would rather have a more balanced model we could pick to use the MLP. Overall, it depends on what would be the expert opinion, however, we think picking the QDA and maximizing the recall is the best choice.

We returned each of the metrics evaluated on the test partition, hence, we can compute the generalization error of any model as $err = 1 - Accuracy$. So, the generalized error for the QDA would be $err_{QDA} = 0.312$, and for the MLP $err_{MLP} = 0.245$.

To finish the modeling section we explain how the Logistic Regression and the Random Forest make their decisions. Even though the MLP and QDA returned better results, we explain the LR and RF since they are a bit more intuitive to understand. Moreover, the QDA is quite similar to the LDA which we already explained in the section 3, and we wish to give more variety.

4.5.1 Logistic Regression

Observing Figure 8 we see that the most important features are, in order: Ejection Fraction, Serum Creatinine, being a man, Serum Sodium, and High Blood Pressure. A negative coefficient means that higher value of the corresponding feature pushes the classification more towards the negative class (patient survives), and inversely for positive coefficients. We can also see that having Anaemia or high CPK increases the chances of dying, while age seems to only contribute in a very small factor towards the positive class. Smoking, Platelets, and Diabetes seem to have negligible impact on the patients survival.

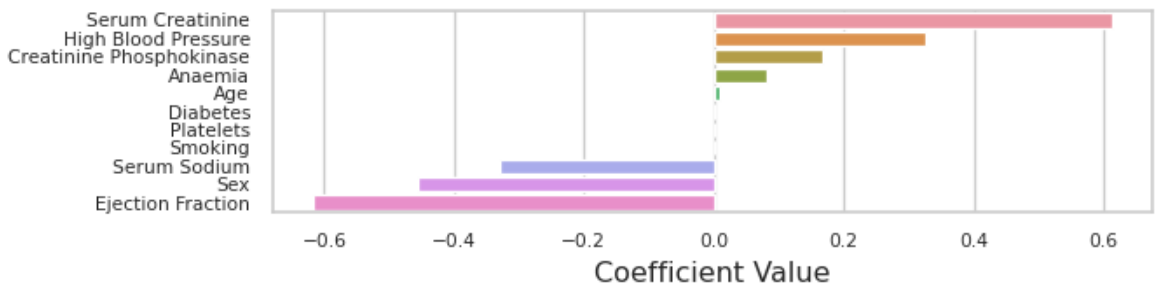

Figure 8: Weight of the coefficients of the Logistic Regression with the parameters described in 4.

Lets check some concrete values: observation 203, has the highest value of SC in the test set, was missclassified as someone who will die when in reality will not. This seems due to the fact that the EF, standarized, returns a negative value and as such the multiplication by its coefficient will return a high positive value and end up miss-classifying as positive. For observation 4, similar comments can be made, however this time the combination to positive worked well in predicting correctly, since we have large negative values of SS and EF, and a high value of SC we can see the combination of it all returned a probability very close to 1, as the model is very sure of this prediction.

Similarly, values 75 and 88 have negative values of SC, in consequence, we push the probabilities towards the negative class. We can also see the effect EF has on the odds. In observation 75, which has a negative value of EF, the probability is closer to 0.5 meaning it is not that sure, while in observation 88, which has a positive value of EF, the probability is closer to zero meaning it is quite sure about this prediction.

Its important to note, that while all the previously explained is true, we need to take into consideration **all** the non-zero coefficients of the model to calculate this probability. However, to simplify our explanation, we only give a sample of the 4 most important features.

Id	SC	CPK	SS	EF	Prediction	True	Probability
203	1.742044	-0.549613	-0.086239	-1.141948	'Yes'	'No'	0.720115
75	-0.647316	-0.562131	0.598215	-1.575258	'No'	'Yes'	0.385904
4	1.059370	-0.444250	-4.649264	-1.575258	'Yes'	'Yes'	0.929344
88	-0.647316	-0.523533	0.598215	0.157982	'No'	'No'	0.155066

Table 5: Some examples of observations we find interesting. The values are standarized.

4.5.2 Random Forest

While Decision Trees can be easily overfitted, the benefit of RF is that it can generalize better due to the ensemble. However, we have no way of individually checking each of the single models nor do weights exist for the features. To do an approximation, we compute the permutation feature importance which is the decrease in a model score when a single feature value is randomly shuffled.

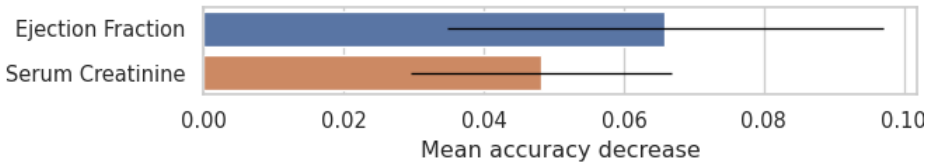

Figure 9: Mean decrease of the feature importance when removed in the RF, modeled with the parameters described in 4.

Checking Figure 9, we can see that the most important features are Ejection Fraction and Serum Creatinine, similar results to what was found in Chicco's work [2]. Comparing these with the Logistic Regression from the previous section, we can see that EF and SC are also the most relevant features, however the Logistic Regression seems to take into consideration a few more features as important. Overall, EF and SC seem to carry a lot of importance in order to correctly predict the defunction of a person.

We cannot recreate the same explanations that we did with the Logistic Regression when trying to explain how the Random Forest is classifying. With the LR we had definite coefficients, each with its weight. However, on the RF, we have 250 decision trees who are each voting for a class, which in hindsight should have been an even number to avoid ties. So to understand how it classifies, we have to give approximate answers with how the distributions move inside each predicted class. Observing the boxplots on Figure 10 we can clearly see on how higher values of SC are more likely to die, while high values of EF are more likely to survive. Similar to what was found on the Logistic Regression.

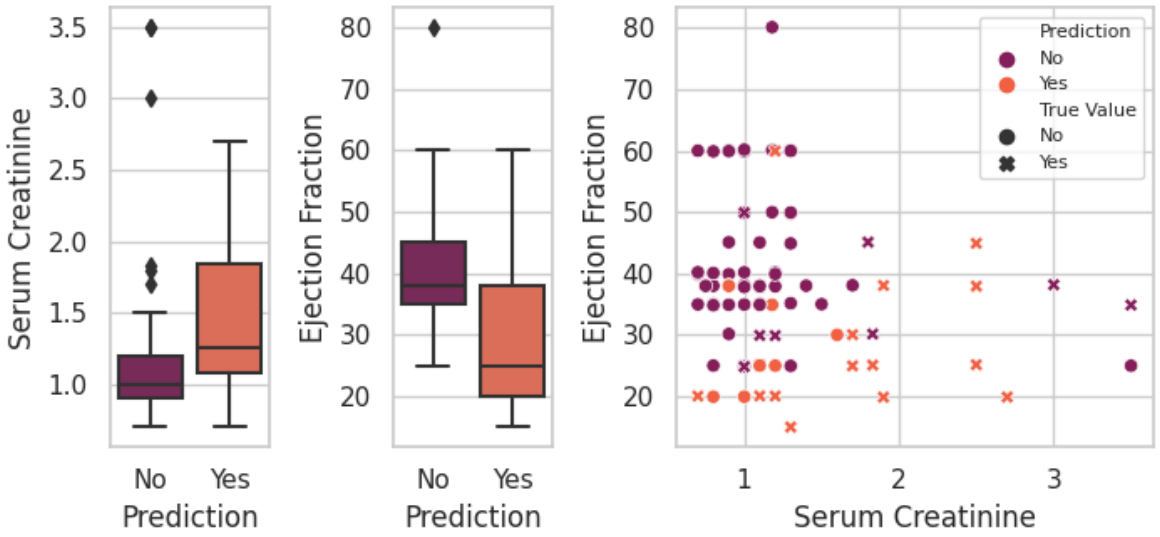

Figure 10: The two main features, and their distributions when plotted against the prediction, we can see how higher values of Ejection Fraction mostly correspond predicting 'No', while lower values correspond to predicting 'Yes'. The inverse can be said with the Serum Creatinine.

Looking at the scatterplot, we can see that we have two values that are predicted as negative around SC 3.5 and EF of 35. One is correctly classified, while the other is not. This miss-classified value is observation 35. Taking into consideration that higher SC seemed like higher probability of being positive, what happened? Taking a look at the train partition, we find a value very close to these two. Mr. 129, who did not die and had a SC of 3.4 and EF of 35. Thus, we can believe that the RF learned that values around that range should be all classified as negative. Some cases this is true, in others,

like poor Mr. 35, we did not prevent their death. However, this analysis is much more qualitative than exact, so take it with a grain of salt.

We present the the observations 4 and 275 as prototypical examples of what is described in Figure 10.

Id	Serum Creatinine	Ejection Fraction	Prediction	True
35	3.5	35	'No'	'Yes'
229	1.2	25	'Yes'	'No'
4	2.7	20	'Yes'	'Yes'
275	0.8	38	'No'	'No'
129*	3.4	35	'No'	'No'

Table 6: Some examples of values we find interesting. Marked with (*) values from the train set.

5 Conclusion

In this project, we have successfully applied machine learning classifiers in order to predict defunction on patients that had a heart failure. We applied a correct methodology to evaluate the results and compare them with previous studies on the same dataset. The final results are comparable to the ones presented in Chicco [2] in both evaluation metrics and feature ranking importance.

On the given data, it seems quite clear that elevated values of Serum Creatinine is the most relevant feature in predicting the patients death, while low levels of Ejection Fraction are related to their survival. The effect of Serum Creatinine was quite evident since the data exploration section. While the correlation with the target feature was not very high, when applying dimensionality reduction to the data with an LDA, we observed how Serum Creatinine could transform the data into something linearly separable by being the main feature to achieve this separability between classes.

After applying various machine learning algorithms on the pre-processed data, we found that the QDA and MLP worked the best. The QDA returned the highest recall overall, which is what we desired since we had an unbalanced dataset and the cost of missing positive cases was too high. The results seem a bit better than Chicco's work [2].

As a limitation of the present project, we have to report the small size of the dataset (299 patients): a larger dataset would have permitted us to obtain more reliable results. Additional information about the physical features of the patients (height, weight, body mass index, etc.) and their medical history would have been useful to detect additional risk factors for cardiovascular health diseases. Finally, we would rather not have the missing values already imputed, as we would have liked to use more sophisticated methods than mean imputation, which seems to be what was done.

As for our modeling, even though our hyperparameter search space is quite extensive, we would have wished to try with custom kernel functions when using the SVM or more custom neural networks (using TensorFlow or Pytorch). Nevertheless, seeing how we matched quite well the results presented in the literature, it seems doubtful that we can reach better results without more data. And more complex models without more data are bound to overfit.

Seeing how the PCA worked pretty well in separating the classes back in the data exploration section, we would also have liked to try the machine learning algorithms on the principal components. However, the training time of the grid search with its current hyperparameter search is already quite a monumental task, taking over 5 days of non-stop training to fit and tune models. Since the price of electricity is quite outrageous, we decided to rent a cloud server on Vultr [10] to train and evaluate our models.

6 References

[1] Tanvir Ahmad et al. "Survival analysis of heart failure patients: A case study". In: *PloS one* 12.7 (2017), e0181001.

[2] Davide Chicco and Giuseppe Jurman. "Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone". In: *BMC medical informatics and decision making* 20.1 (2020), pp. 1–16.

[3] Michael Collins, Sanjoy Dasgupta, and Robert E Schapire. "A Generalization of Principal Components Analysis to the Exponential Family." In: *Nips*. Vol. 13. 2001, p. 23.

[4] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[5] John C Gower. "A general coefficient of similarity and some of its properties". In: *Biometrics* (1971), pp. 857–871.

[6] Kochanek KD. "Mortality in the United States." In: *NCHS Data Brief, no 395.* (2019).

[7] David L Olson and Dursun Delen. *Advanced data mining techniques*. Springer Science & Business Media, 2008.

[8] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[9] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[10] *Vultr - The Infrastructure Cloud*. vultr.com. Accessed: 2021-12-27.