AGH University of Krakow

Faculty of Computer Science, Electronics and Telecommunications

Engineering Thesis

Machine learning for prediction of neonatal morbidity assessed by hemodynamic echocardiography and laboratory blood tests

Uczenie maszynowe w przewidywania patologii rozwoju noworodków na podstawie badań hemodynamicznych serca i testów krwi

Author: **Michał Jan Tajak**
Academic Degree: **ICT Studies**
Supervisior: **dr hab. inż. Paweł Kułakowski prof. AGH**

Kraków, 2025

Contents

List of Figures v

List of Tables vii

List of Listings ix

1. Introduction 1

2. Hemodynamic Echocardiography Data in Neonatology 3
2.1. The Essence of Cardiac Hemodynamic Testing . . . 3
2.2. Machine Learning Models for Neonatal Data . . . 3
2.3. Examples of Applications of Machine Learning Models in Neonatology . . . 6

3. Preparation of Dataset and Research Scenarios 9
3.1. Dataset . . . 9
3.2. Data Dictionary . . . 9
3.3. Data Processing . . . 12
3.4. Scenario 1 - Echo-Cardiological Prediction . . . 13
3.4.1. Change of Data Type . . . 13
3.4.2. Analysis of Cull Samples . . . 14
3.4.3. Visualization of Cleaned Data . . . 15
3.5. Scenario 2 - Predicting Newborn Development . . . 16
3.5.1. Change of Data Type . . . 16
3.5.2. Analysis of Cull Samples . . . 18
3.5.3. Visualization of Cleared Data . . . 18

4. Implementation Of Machine Learning Algorithms 21
4.1. Scenario 1: Echo-Cardiological Prediction . . . 21
4.1.1. Results Summary . . . 32
4.2. Scenario 2: Predicting Newborn Development . . . 33
4.2.1. Results Summary . . . 50

5. Conclusion 53
5.1. Summary . . . 53

5.2. Opportunities for Further Development 54
5.3. Application of Research 55

Bibliography **57**

List of Figures

2.1. Example image showing the results of LV strain, *source: Jagielllonian University Medical College* 4

3.1. Different cross sections of the heart used in the project [7] 9
3.2. Comparison of two types of images provided to the dataset. 13
3.3. The structure of the data used in Scenario 1, on the left are the input data and on the right are the labels. 14
3.4. Matrix showing the correlations between the various input data parameters. 15
3.5. Presentation of sample histograms for classification data (left side) and regression data (right side). 16
3.6. The structure of the data used in Scenario 2, on the left are the inputs from both input categories, these sets have been combined, on the right are the labels. 17
3.7. Matrix showing correlations between various input data parameters from Category 2. 19
3.8. Presentation of sample histogram for regression data. 19

4.1. Detailed visualization of the results for the best-predicted label of the linear regression model. 23
4.2. Detailed visualization of the results for the best-predicted label of the KNN model. 24
4.3. Results for the validation and training set for the label *LV GLS*. 27
4.4. Detailed results for one of the best-predicted Gradient Boosting model labels. 30
4.5. Detailed results for one of the best-predicted Multilayer Perceptron model labels. 32
4.6. Prediction results for regression data labels of the linear regression model. . . 35
4.7. Confusion matrix for the label *P27.1* (left side) and the label *H35.1* (right side). 36
4.8. Prediction results for regression data labels of the KNN model. 39
4.9. Confusion matrix for the label *P27.1* (left side) and the label *H35.1* (right side). 39
4.10. Prediction results for regression data labels of the random forest model. . . . 42

4.11. Confusion matrix for the label *P27.1* (left side) and the label *H35.1* (right side). 42
4.12. Prediction results for regression data labels of the gradient boosting model. . 45
4.13. Confusion matrix for the label *P27.1* (left side) and the label *H35.1* (right side). 45
4.14. Prediction results for regression data labels of the MLP model. 48
4.15. Confusion matrix for the label *P27.1* (left side) and the label *H35.1* (right side). 49
4.16. Confusion matrix for the label *P27.1* (left side) and the label *H35.1* (right side). 50

List of Tables

4.1. MAPE results for each fold and mean MAPE for each feature. 22
4.2. Prediction results of individual labels for the linear regression model for the best folds. 22
4.3. Results for each fold and MAPE averages for the KNN model. 24
4.4. Prediction results of individual labels for the KNN model along with selected hyperparameters. 25
4.5. Results for each fold and MAPE averages for the random forest model. . . . 26
4.6. Prediction results of individual labels for the random forest model along with selected hyperparameters. 27
4.7. Results for each fold and MAPE averages for the gradient boosting model. . 28
4.8. Prediction results and best parameters for individual labels for Gradient Boosting model. 29
4.9. Results for each fold and MAPE averages for the neural network model. . . . 31
4.10. Prediction results and best parameters for individual labels for Multilayer Perceptron model. 31
4.11. Comparison of mean MAPE scores for linear regression, KNN, Random Forest, Gradient Boosting and MLP by trained labels. 32
4.12. Comparison of best MAPE results from folds for linear regression, KNN, Random Forest, Gradient Boosting and MLP by trained labels. 33
4.13. Results for each fold and MAPE averages for ridge regression model on regression data. 34
4.14. Results for each fold and f1_score averages logistic regression model classification data. 35
4.15. Results for each fold and MAPE averages for KNN model for regression data. 37
4.16. Results for each fold and f1_score averages KNN model classification data. . 37
4.17. Prediction results of individual labels for the KNN model along with selected hyperparameters for regression data. 38
4.18. Prediction results of individual labels for the KNN model along with selected hyperparameters for classification data. 38
4.19. Results for each fold and MAPE averages for random forest model for regression data. 40
4.20. Results for each fold and F1 score averages random forest model classification data. 41

4.21. Prediction results of individual labels for the random forest model along with selected hyperparameters for regression data. 41
4.22. Prediction results of individual labels for the random forest model along with selected hyperparameters for classification data. 41
4.23. Results for each fold and MAPE averages for gradient boosting model for regression data. 44
4.24. Results for each fold and f1 score averages gradient boosting model classification data. 44
4.25. Prediction results of individual labels for the gradient boosting model along with selected hyperparameters for regression data. 44
4.26. Detailed results for fold 5 of the gradient boosting model along with hyperparameters for classification data. 44
4.27. Results for each fold and MAPE averages for MLP model for regression data. 47
4.28. Results for each fold and F1 score averages MLP model classification data. . 47
4.29. Prediction results of individual labels for the MLP model along with selected hyperparameters for regression data. 47
4.30. Prediction results of individual labels for the MLP model along with selected hyperparameters for classification data. 48
4.31. Results for each fold and f1 score averages isolation forest model classification data. 49
4.32. Prediction results of individual labels for the isolation forest model along with selected hyperparameters for classification data. 50
4.33. Comparison of mean MAPE scores for linear regression, KNN, random forest, gradient boosting, MLP by trained labels. 51
4.34. Comparison of mean f1 scores for logistic regression, KNN, random forest, gradient boosting, MLP, isolation forest by trained labels 51

5.1. Average MAPE scores for all models for Scenario 1. 54
5.2. Average MAPE (for regression data) and f1 (for classification data) scores for all models for Scenario 2. 54

List of Listings

4.1. Representation of hyperparameters tuned during KNN model training. . . . 23
4.2. Representation of hyperparameters tuned during Random Forrest model training. . . . 26
4.3. Representation of hyperparameters tuned during Gradient Boosting model training. . . . 28
4.4. Representation of hyperparameters tuned during Multilayer Perceptron model training. . . . 30
4.5. Representation of hyperparameters tuned during logistic regression model training for classification data. . . . 34
4.6. Representation of hyperparameters tuned during KNN model training for regression data. . . . 36
4.7. Representation of hyperparameters tuned during KNN model training for classification data. . . . 37
4.8. Representation of hyperparameters tuned during random forest model training for regression data. . . . 40
4.9. Representation of hyperparameters tuned during random forest model training for classification data. . . . 40
4.10. Representation of hyperparameters tuned during gradient boosting model training for regression data. . . . 43
4.11. Representation of hyperparameters tuned during gradient boosting model training for classification data. . . . 43
4.12. Representation of hyperparameters tuned during MLP model training for regression data. . . . 46
4.13. Representation of hyperparameters tuned during MLP model training for classification data. . . . 47
4.14. Representation of hyperparameters tuned during isolation forest model training for classification data. . . . 49

1. Introduction

In recent years, the development of neonatology has contributed to an increase in the survival rate of premature babies. Unfortunately, in parallel with an earlier birth, babies often experience various health complications, which include respiratory as well as cardiac problems. Tests such as LV strain (left ventricular strain), which measure the deformation of the left ventricular muscle during the cardiac cycle, or blood tests can provide information on the health of premature babies. However, these tests require a high level of competence when they are performed, at the same time being time-consuming, which can translate into difficulties in making treatment decisions.

The dynamic development of machine learning algorithms allows medicine to approach diagnosis in new ways, to look for connotations between initially unrelated tests as well as to predict complications. Machine learning, because of its ability to process data, identify patterns or create predictive models, would allow, in the case of successful models, to help doctors with initial diagnoses and decision-making. Using a good quality model along with the medical knowledge of doctors, for example, would allow them to decide which children need more in-depth testing. Such an approach could streamline doctors' work, and in the case of similarities between successive diseases in premature babies, this could translate into systematized treatment.

This thesis aims to discuss the application of various machine learning methods in predicting pathologies in neonatal development. The dataset was provided by the Neonatology Clinic of the Jagiellonian University Medical College, consisting of laboratory blood tests and cardiac hemodynamic studies of the clinic's patients. The data was first reviewed. Values were extracted from the images and combined with the provided dataset. Subsequently, after consultation with doctors, the features to be used for the study were selected. After that, the collection was divided according to the recommendations into two neonatal scenarios. After the split, blemished samples were removed or modified. Then, with the division into scenarios, specific machine learning models (focusing mainly on shallow learning models) were learned for parameter prediction. The results for each model are shown in graphs and tables. The study conducted has allowed providing clues to predict parameters for both Scenario 1 - Echo-Cardiological Prediction and Scenario 2 - Predicting Newborn Development. The results suggest that further work could lead to more precise models despite the fact that the current ones have not reached full effectiveness. This is particularly evident in Scenario 1, where the predicted values for the left and right ventricles are good enough that they could potentially be implemented to help physicians in the near future.

In the following sections, the various stages of the study are meticulously discussed along with the results obtained. Chapter 2 describes the theoretical issues necessary to understand the work. The essence of cardiac hemodynamic studies, which are crucial in the diagnosis of cardiac function in newborns, is explained. The machine learning models used for data analysis are then presented. Finally, examples illustrating the application of machine learning models in neonatology are presented. Chapter 3 was devoted to the description and processing of the data provided. At the outset, the provided collection was described. Then the significance of the various features used in the study was explained. Subsequently, the data was divided into two scenarios. Finally, the steps of data processing and data cleaning to maintain the integrity of the collection were discussed. Chapter 4 describes in detail the process of training the various machine learning models. The learning took place for both scenarios, where successive models were trained according to an established scheme that first analyzed the selection of hyperparameters, and then presented the results in the form of tables and graphs. At the end of each scenario, a summary and comparison of the average results of each model was done. Chapter 5 summarizes the results from both scenarios. Conclusions from the study and possible problems for further work are presented. Finally, directions for further research and suggestions for further research are discussed.

2. Hemodynamic Echocardiography Data in Neonatology

2.1. The Essence of Cardiac Hemodynamic Testing

Hemodynamic echocardiography assesses cardiac function in a non-invasive way. It is one of the key tests in assessing cardiovascular function in premature and newborn infants. One of the main such parameters is LV strain (Left Ventricular Strain). Based on the study "Deformation imaging and rotational mechanics in neonates: a guide to image acquisition, measurement, interpretation, and reference values [1].", LV strain provides very sensitive indicators of cardiac systolic function. This makes possible to detect, more in advance, super-subtle changes in cardiac functioning, which translate into earlier diagnosis and quicker response in treatment. A photo of the results illustrating the exemplary study is shown in Fig 2.1.

In the figure, three different projections, namely A2C, A3C, and A4C, can be seen. Based on the work "How to do it? Speckle-tracking echocardiography [2].", we can learn that this is a natural division, as the different views allow for a more detailed analysis of different segments of the left ventricle. For each value, the average contraction should be about -20% (on average, the myocardium should shrink by 20% with each contraction). The two-chamber view (A2C) allows the left ventricle (LV) and left atrium (LA) of the heart to be depicted in the longitudinal plane, which allows for deformations in the lower and anterior parts of the heart to be assessed. The three-chamber view (A3C) also includes the left ventricle and left atrium, additionally showing the ascending aorta (LVOT). This view allows for assessment of valve function and flow through the ascending aorta. Finally, the four-chamber view (A4C) is the most versatile, as it includes a view of not only the left ventricle and left atrium but also the right ventricle and right atrium. It is the main reference in the evaluation of LV strain, as it shows all four cavities.

2.2. Machine Learning Models for Neonatal Data

In presenting the models and their main advantages and disadvantages, it will be noted that most of them are shallow models. This is due to the fact that neonatology data tends to be small sets, as relatively few premature babies are born. When we contrast the size

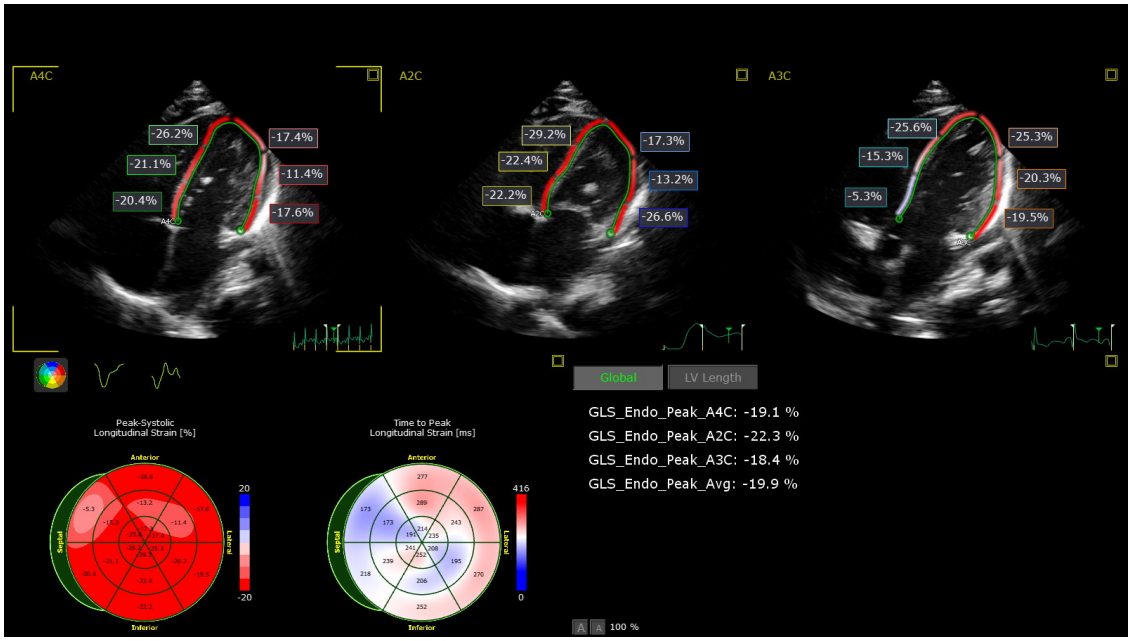

Figure 2.1.: Example image showing the results of LV strain, *source: Jagielllonian University Medical College*

of the set, especially with data such as that described above, one can conclude that deep models could have problems here. In addition, shallow models, because of their greater interpretability, might also allow one to see how the models divide the individual data. Such an approach would allow doctors to learn more about the data based on the breakdowns and better tailor treatment. Therefore, the summary of models used is as follows:

- **Linear Regression** — A basic machine learning algorithm that involves fitting a rectilinear function to data to predict values. It assumes that the data have a linear relationship with each other. Linear regression is simple to implement and easy to understand. It works quickly and for a small amount of data if it is somehow linearly dependent then it can continue to be effective. In our case, of the disadvantages, the model can be sensitive to outliers and due to the large number of features juxtaposed with a small number of samples, overfitting can occur[3].

- **Logistic Regression** — A basic machine learning algorithm, in which a variable is represented as a value of 1 or 0 (yes or no). It is mainly used for classification data and usually comes in tandem with linear regression (which predicts regression values). Logistic regression is simple to implement, easy to understand and also works quickly. It handles classification data well but can have problems with the small dataset provided[3].

- **KNN** — K-Nearest Neighbors is a machine-learning algorithm that estimates a label based on the values of nearest neighbours found in a training set. KNN works with both classification and regression data. It is easy to implement. We can tune a large number of hyperparameters such as *n neighbors* responsible for the number of neighbors or *p* specifying how the distance is counted. This allows the model to be flexible with respect to the data. It requires well-scaled data, and can become inferior in performance if the set increases. Can be sensitive to outliers[3].

- **Random Forest** — A set of decision trees that are independently learned. Decision-making involves collecting predictions from each tree, verifying which value will be the most popular in the entire forest (in the case of classification data) or selecting the average value for the entire forest (in the case of regression data). Then such an estimate represents the entire ensemble. Random forest works with both classification and regression data. Relatively resistant to overfitting. Due to the large number of hyperparameters responsible for the size of a single tree or the number of trees in a forest, it can also be very flexible. Due to its computational complexity, training the model can be time-consuming. The model can have problems with values whose representation is small (for example, outliers)[3].

- **Gradient Boosting** — is a machine learning method that uses decision trees to create them iteratively. Each subsequent non-first tree model is learned from the differences between the values predicted by the predecessor and the actual values. Such a procedure is designed to minimize the predecessor's errors and ultimately create a single strong predictor altogether. It comes in versions for classification data GradientBoostingClassifier as well as for regression data GradientBoostingRegressor. Gradient boosting works with both classification and regression data. It has similar advantages and disadvantages to random forest. It has similar hyperparameters as random forest enriched with those responsible for learning rate or controls against over-fitting. This makes the model even more flexible and efficient. The main disadvantage is the high computational complexity, which can translate into long training times. In addition, the model can be sensitive to outliers and their small representation[3].

- **MLP** — a multi-layer perception is a machine-learning model based on an artificial neural network consisting of several layers of perceptions. In this model, the neurons in each layer are connected to each neuron in the next layer, which means that the network is fully connected. MLP is the only one that does not count as a shallow model. It handles both classification and regression data well. It has many hyperparameters that help tune the model and determine the size of the neural network. It handles complex data well and is very flexible. The main problem can be a very small data set, as MLP needs a large number of samples to learn effectively. In addition, it is difficult to interpret the various decisions of the model because it operates on the principle of

a "black box" or hidden layer whose structure and decision-making can be difficult to understand[3].

- **Isolation Forest** — is an algorithm based on decision trees, which is mainly used to detect anomalies. In our case, it will only be applied to classification data. It works similarly to other forest-based models. The difference is that it isolates anomalies faster, plus outliers have a shorter path length than normal ones. Then, based on these lengths, it is assessed which group the sample falls into (the shorter the path, the higher the probability of an anomaly). Isolation forest is a model that is fast to train and versatile. It may be less effective with a small data set. In addition, it is difficult to interpret its results, which translates into a lack of explanatory power[3].

In addition, models may encounter the following problems due to the characteristics of this type of medical data (collected during a daily routine in a particular clinic):

- **Data quality**: due to the collection of a large number of features from different types of surveys, the data may somehow be incomplete or cull for entire surveys, which may translate into the removal or modification of samples in an already small dataset.
- **Small data set**: due to the large number of features and the small number of samples, models may have problems with overfitting due to such combination. The variety of individual feature values can result in difficulties with qualitative predictions.
- **Class balance problem**: most of the samples in the collection are of newborns who are not sick. Therefore, most samples will be close to the correct value. The problem may be the number of samples of children who have some complications, which usually translates into outliers in particular features. For this reason, the models may have trouble predicting outliers regarding anomalies.
- **Data ambiguity**: giving a particular example, in the set there could be a feature like *dni hospitalizacji*, which accounts for the length spent in the hospital by the child. The examinations for individual children are not done at the same intervals (for example, a healthy child will have one examination at a time when a sick child has already had three) which can translate into problems in predicting this feature.

2.3. Examples of Applications of Machine Learning Models in Neonatology

Nowadays, machine learning models are very often used in neonatology. The article "Artificial Intelligence for Automatic Measurement of Left Ventricular Strain in Echocardiography [4]" discusses an attempt to automatically measure GLS in echocardiography images using

deep learning. This text demonstrates an attempt to automate the examination of Left Ventricular (LV) Strain and minimize the time required for analysis. In addition, it presents an approach to a similar problem we have in Scenario 1, where we try to predict outcomes for the left ventricle. Very often, in scientific papers related to neonatology, we may encounter an attempt to detect fatal diseases as early as possible. Examples of such papers are "Machine Learning Models for Predicting Neonatal Mortality: A Systematic Review [5]" and "Enhancing the accuracy in predicting infant mortality using logistic regression in comparison with decision trees with support vector machine [6]" where both attempt to predict infant mortality. In the case of the first text, neural networks or random forests are used. The second paper focuses on using shallow learning models such as decision trees, logistic regression, or support vector machines (SVM). Analysis of both papers indicates that these models are suitable for predicting the risk of death in infants, especially neural networks, logistic regression, or decision trees.

3. Preparation of Dataset and Research Scenarios

3.1. Dataset

The dataset used in this article comes from the Jagiellonian University Medical College in Krakow. It consists of medical data including general clinical data, blood parameters, cardiac parameters, and echo heart of premature babies born at different weeks of gestation. The excel file provided contains 170 samples and 155 features. In addition, each sample should be provided with one image describing the examination of the left ventricle of the heart in three different projections, which can be seen in Fig. 3.1. In the end, 167 images were provided.

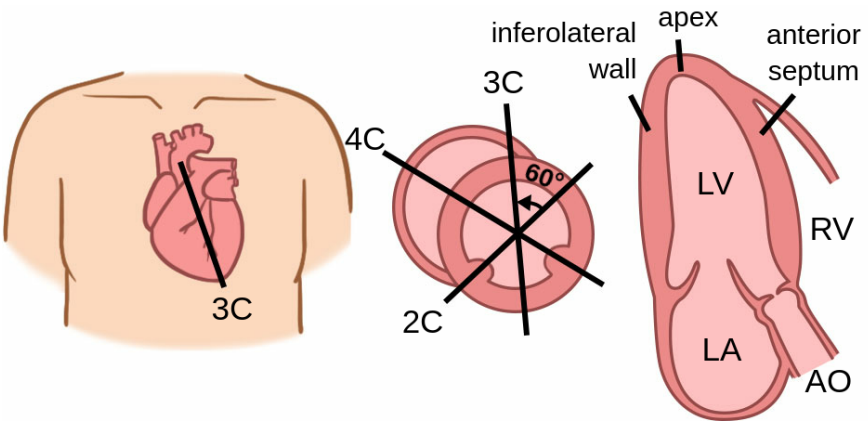

Figure 3.1.: Different cross sections of the heart used in the project [7]

3.2. Data Dictionary

In the next stage of the work, selected data features will be used after consultation with a specialist in the field, Dr. Agnieszka Ochoda-Mazur. Due to the way they will be used, they have been divided into several different categories, grouped as follows:

- **Category 1** — contains health and clinical parameters, concerning both the baby and the mother, such as data related to the course of pregnancy, or data on the baby's development and potential infections.
- **Category 2** — contains cardiac parameters such as data on left atrial, left and right ventricular heart function or echocardiographic data.
- **Category 3** — contains parameters related to the baby's health after birth, such as developmental data, diagnostic data or data on hospitalization time.
- **Category 4** — contains detailed parameters read from the provided images. These relate to echocardiography.

After explaining each category, one can proceed to discuss the features in them as follows:

- Category 1
  - **DM (0-no, 1-diet, 2-insulin)** — Diabetes Mellitus, refers to the incidence of diabetes in the mother during pregnancy where: 0 is no incidence, 1 is diabetes controlled with diet, 2 is diabetes treated with insulin [8].
  - **GA** — Gestational age, the time measured in weeks from the last menstrual period of the mother of the baby until delivery, helps assess the baby's development stage[9].
  - **BW** — Birth weight, indicates the weight of the baby at birth calculated in grams or pounds [10].
  - **BW (pc)** — Percentile of the child's birth weight relative to the standard chart [10].
  - **BW (Z Score)** — A measurement comparing the baby's weight at birth at a given gestational age against the average for that population, including standard deviations [10].
  - **Apgar 1'** — The system of assessing the condition of the newborn at 1 minute of life takes into account such parameters appearance, pulse, grimace, activity and breathing, each of them can get a score from 0 to 2 [11].
  - **Apgar 5'** — The system of assessing the condition of the newborn at 5 minutes of life takes into account such parameters appearance, pulse, grimace, activity and breathing, each of them can get a score from 0 to 2 [11].
  - **P00.0** — ICD-10 code indicating the condition of the newborn caused by factors such as infections and a maternal history of disease during pregnancy [12].
  - **Ht** — The percentage of red blood cells present in the blood helps to determine ischemia or redness in the newborn and the ability to carry oxygen [13].

  - **Hb** — Protein responsible for oxygen transport [14].
  - **Rbc** — Measurement indicating the concentration of red blood cells [15].
  - **CRP** — Protein released as a result of inflammation, often used to identify infection in newborns[16].
  - **Il6** — A cytokine that, when elevated, often signals sepsis or inflammation [17].

- Category 2
  - **LV A2C** — Represent view of the left ventricle (LV) during the two-chamber apical echocardiography view (see Fig. 3.1) [18].
  - **LV A3C** — Represent view of the left ventricle (LV) during the view of three-chamber apical echocardiography (see Fig. 3.1) [18].
  - **LV A4C** — Represent view of the left ventricle (LV) during the apical echocardiography view of four chambers (see Fig. 3.1) [18].
  - **LV GLS** — Global longitudinal strain indicates how the heart muscle contracts during the cardiac cycle[18].
  - **LASr ED** — Left Atrial Strain rate during early diastole [19].
  - **LAScd ED** — Left Atrial Conduit Strain during early diastole [19].
  - **LASct ED** — Left Atrial Contractile Strain during early diastole [19].
  - **LASr AC** — Left atrial strain rate during early systole [19].
  - **LAScd AC** — Left Atrial Conduit Strain during early systole [19].
  - **LASct AC** — Left Atrial Contractile Strain during early systole [19].
  - **RVFWSL** — Right Ventricular Free Wall Strain Longitudinal is a measure of the strain or shortening of the right ventricular free wall along its longitudinal axis during systole [18].
  - **RV4CSL** — Right Ventricular 4-Chamber Strain Longitudinal, where the pathways deform in the apical 4-chamber view of the RV. These are important for assessing RV systolic function, especially in right ventricular failure [18].

- Category 3
  - **HC - WYPI** — Head circumference (HC) parameter determining the size of the child's head measured around the largest part of the head, in this case the measurement took place during the day of the child's discharge [20].
  - **HC (pc) - WYPI** — HC Percentile indicates how a child's head size compares to a normalized population of the same age, in our case a week [20].

- **HC (Z Score) - WYPI** — Z-Score is a statistical measure that determines the standard deviation of a given child's head size from the population average [20].
- **P27.1** — ICD-10 code referring to "bronchopulmonary dysplasia (BPD) arising in the perinatal period" [21].
- **H35.1** — ICD-10 code referring to "retinopathy of prematurity (ROP)." This condition involves abnormal growth of blood vessels in the retina [22].
- **dni hospitalizacji** — Days of Hospitalization, determines the number of days a patient spends in the hospital.

A description of the features pertaining to Category 4 will be discussed in Section 3.3.

3.3. Data Processing

First, it was decided to extract data from images to combine and arrange them with values in an excel file. This was initially attempted using the pytesseract [23] library for the Python programming language, which uses optical character recognition to extract text from images. Unfortunately, due to the different types of images (see Figure 3.2), their lack of systematization (a different way of writing image names) and reading efficiency, it was decided to create a separate excel file and transcribe the data manually. It contains the patient's ID, the date of the examination, and 18 parameters read from the images, which were eventually transcribed manually. The data were transcribed from the images from left to right in a clockwise direction (see the right side of Fig. 3.2). To further simplify the breakdowns, the parameters read from the images were placed in a separate data category, the fourth as follows:

Category 4

All features describe the different points at which the heart muscle contracts and diastases. They are given in negative values suggesting how much the heart is contracting in this part (by how much percentage). For a healthy child, these values should oscillate around a value of -20% at all measured points. Each section has 6 such points, and their names and the way they are numbered look like this:

- **columns A2C 1 to A2C 6** — describe 6 different points of the left ventricular myocardium during apical two-chamber echocardiography.
- **columns A3C 1 to A3C 6** — describe 6 different points of the left ventricular myocardium during apical three-chamber echocardiography.
- **columns A4C 1 to A4C 6** — describe 6 different points of the left ventricular myocardium during apical four-chamber echocardiography.

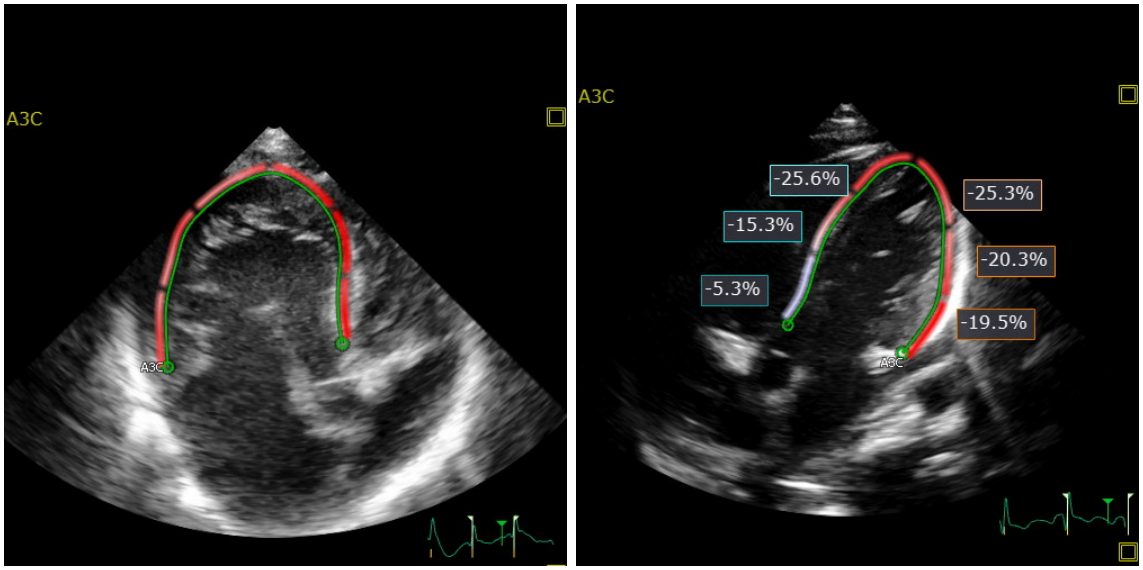

Figure 3.2.: Comparison of two types of images provided to the dataset.

After obtaining all possible data, it was decided to perform the division of the rest of the work into two scenarios suggested by specialists. A detailed description of each scenario can be found in Section 3.4 for Scenario 1 “Echo-cardiological prediction” and Section 3.5 for Scenario 2 “Predicting newborn development”.

3.4. Scenario 1 - Echo-Cardiological Prediction

Scenario 1 ”Echo-cardiological prediction” uses the data classified in Category 1, which describes the general data of the child and blood tests as input. In contrast, the data in Category 2 describes the echocardiography tests as labels. We can observe the structure in the diagram of Fig.3.3. Such an approach, in the case of high-quality model performance, would allow physicians to estimate echocardiographic results and, based on this, diagnose which premature infants should be sent for additional, more thorough testing and for which such testing is not necessary.

3.4.1. Change of Data Type

In order to further process the data, first, the data types for both input data (Category 1) and labels (Category 2) were checked. The first group consists of integer (int), floating point, and object type data, which are *P00.0*, *CRP* and *Il6*. The second group of data consists of float type data only. Ultimately, we only need data of type int and float, so the data of

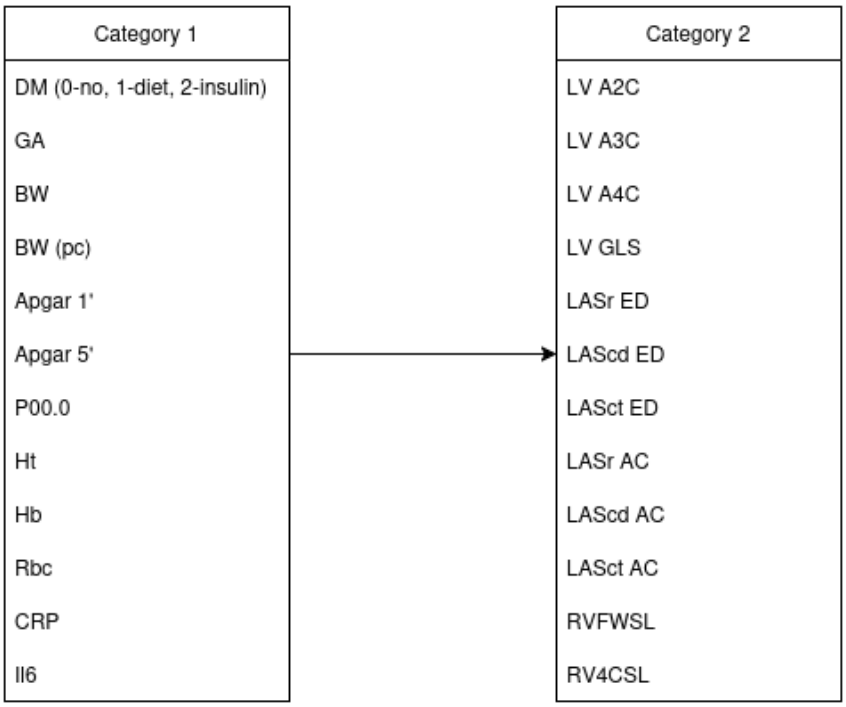

Figure 3.3.: The structure of the data used in Scenario 1, on the left are the input data and on the right are the labels.

type object has been rewritten. For the *P00.0* parameter, the values were changed from "T" (meaning true) and null to zero-one values. For the columns *CRP* and *Il6*, a minority sign was observed when the parameter did not exceed the values of 5 and 1.5, respectively. Therefore, after reconciliation, they were changed to the value of 1 for both cases. Such a procedure allowed us to convert these features into data of type int or float.

3.4.2. Analysis of Cull Samples

In the next stage of data processing, it was decided to remove or supplement the missing samples. After checking these samples, it was noticed that all of them are highly fussy. That is, if they have deficiencies it is, for example, in all parameters related to echocardiography (Category 2). Therefore, it was decided to remove all the fussy samples. After performing the above procedures, the dataset decreased from the initial 170 samples to 89, giving us a change of about 48%.

3.4.3. Visualization of Cleaned Data

In the last stage of data processing for Scenario 1, it was decided to represent the input data using a confusion matrix (see Fig. 3.4). As can be seen, three pairs of highly correlated parameters. This means that with more samples available, and with the approval of specialists, we could get rid of three features out of six. In addition, a pair plot visualization from the *seaborn* [24] library was also performed on the input data, with which you can see the histograms of the individual columns (see Fig. 3.5) on the main diagonal. Due to the large size and number of plots, it was decided to present single histograms for classification and regression data. For the classification data, the distributions of the individual parameters were similar to each other. For example, the histogram is on the left side of Fig. 3.5. The most common value is 0; the others, such as 1 or 2 (if any) are rare because they indicate the occurrence of a particular disease or infection. This suggests that most premature babies are healthy or have no major complications. The regression data also have a similar structure between them. An example graph for this structure is illustrated on the right side of Fig. 3.5. The data resemble a near-normal distribution with a slight skewness on the left side. Such a structure also highlights the frequent occurrence of healthy children, which oscillates around the value inherent in the feature.

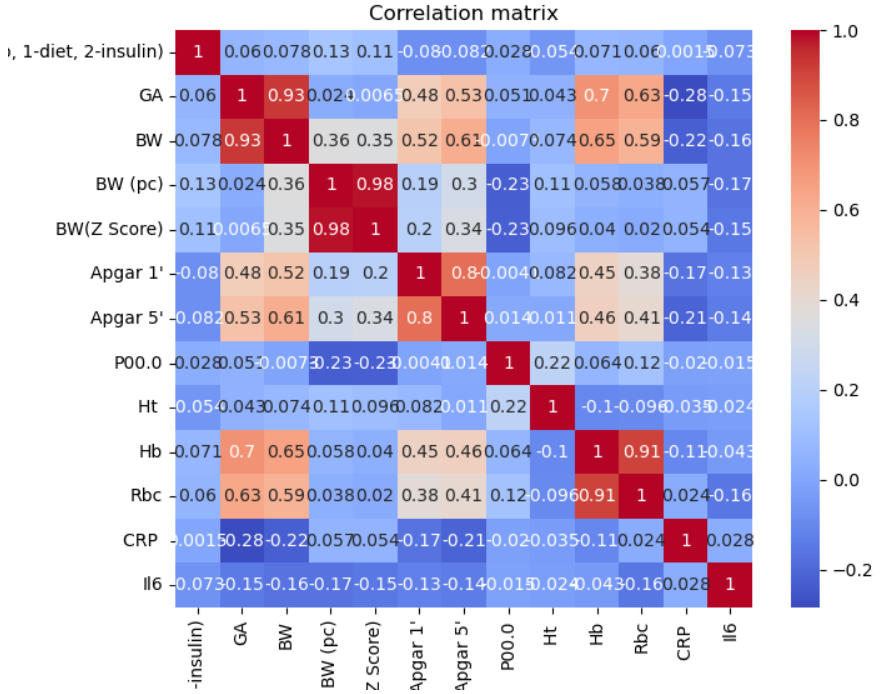

Figure 3.4.: Matrix showing the correlations between the various input data parameters.

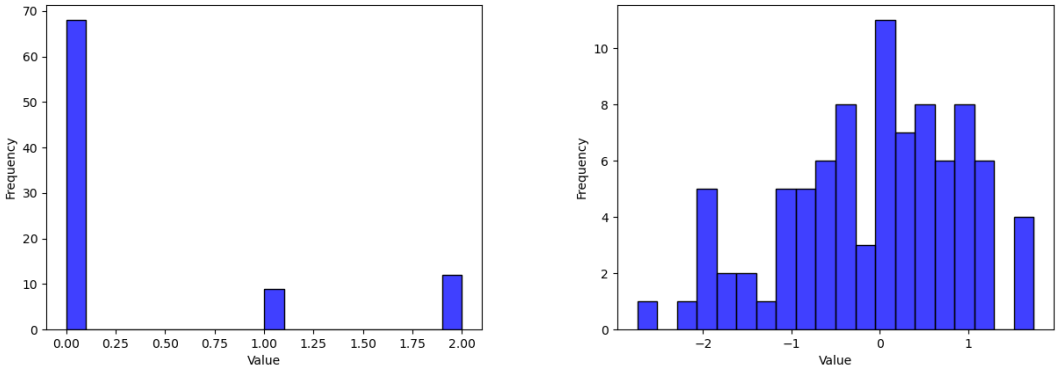

Figure 3.5.: Presentation of sample histograms for classification data (left side) and regression data (right side).

3.5. Scenario 2 - Predicting Newborn Development

Scenario 2 "Predicting newborn development" uses data read from the provided images, in addition to data from the excel file. In this situation, the input data are features found in Category 2 and Category 4, both of which describe parameters related to echocardiography. On the other hand, the labels in this scenario are Category 3 data describing the child's head size, time spent in the hospital, and two parameters describing the presence of bronchopulmonary or retinal problems, respectively. A diagram has been created to visualize the breakdown of the data (see Fig.3.6). With an effectively trained model for this scenario, it would be possible to predict visual or respiratory problems based on echocardiography. In addition, by predicting the length of hospitalization, doctors could better adjust the form of therapy and estimate the child's progress.

3.5.1. Change of Data Type

In this case, the same procedure was used as in Scenario 1. After loading the input data (Category 2 and Category 4) and labels (Category 3), the type of data present in each parameter was checked. In the case of the input data, all features are of float type. However, in the case of labels, there are features with data types float and object. The object type features are *P27.1* and *H35.1*, which will be changed to data type int or float. To change the data type, it was decided to proceed in the same way as for the *P00.0* column from Scenario 1, that is, replacing the "T" (meaning true) and null values with zero-one integers. Such an operation allowed changing the data type object to int.

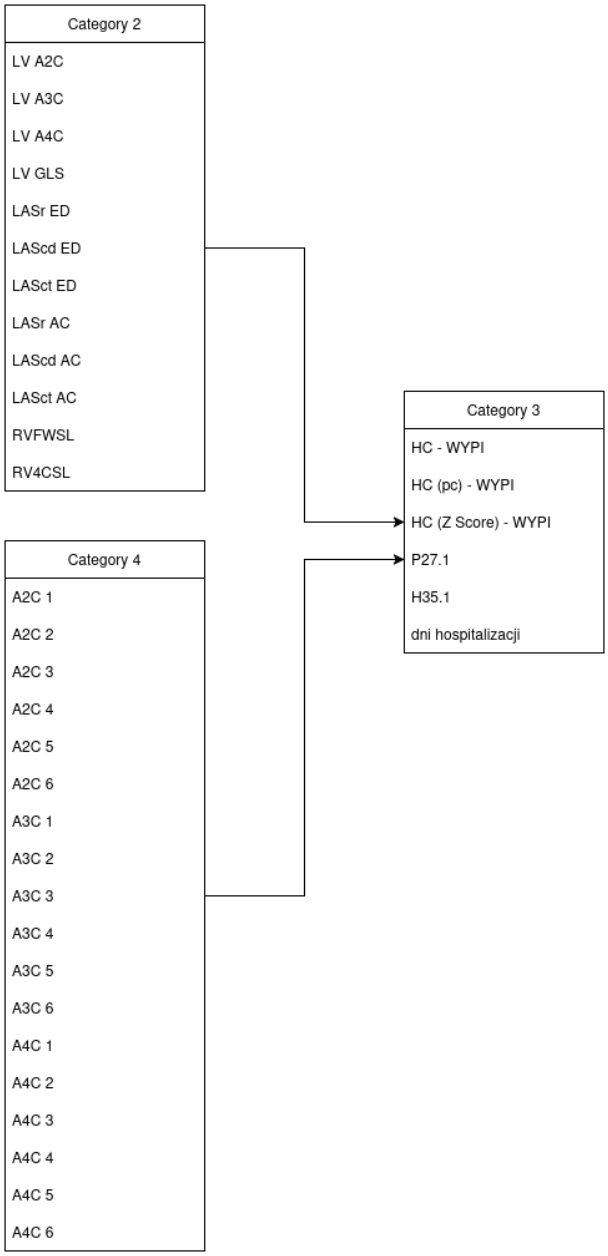

Figure 3.6.: The structure of the data used in Scenario 2, on the left are the inputs from both input categories, these sets have been combined, on the right are the labels.

3.5.2. Analysis of Cull Samples

The next part proceeded to remove or add to the samples. First, all samples that did not have data from the images were removed. Then the remaining cull samples were checked for their degree of culling. As in Scenario 1, feature groups were missing examples of all features for echocardiography data. Therefore, it was decided to remove all the cull samples. After applying the above procedures, we were left with 142 samples out of the initial 170, which shows that our collection decreased by about 16%. Compared to the first scenario, the decrease in the number of samples is noticeably smaller.

3.5.3. Visualization of Cleared Data

After cleaning the data, it was decided to create a confusion matrix for the input data from Category 2 (see Fig. 3.7), the pictorial representation of the data from Category 4 was omitted, as these parameters in the averaged version are in the columns *LV A2C*, *LV A3C* and *LV A4C*. The matrix shows a very strong correlation between the last two features. In addition, it would be possible for a larger set to reduce the size from the features by the parameter *LV GLS* because the features *LV A3C* and *LV A4C* have a very strong correlation with this parameter. Of course, performing the reverse operation would be possible but removing the features *LV A3C* and *LV A4C* due to their relevance in terms of connotation with images is not.

As with Scenario 1, a pair plot matrix was made containing histograms of individual features on the main diagonal. Due to the large size of the matrix and therefore poor readability, it was decided to present a sample distribution for one of the features. All features contain regression data resembling a histogram with a symmetrical normal distribution with a slight leftward bias, as shown in Fig. 3.8. This arrangement can translate into difficulties when teaching some models, such as random forests, because most of the data is collected near the mean value.

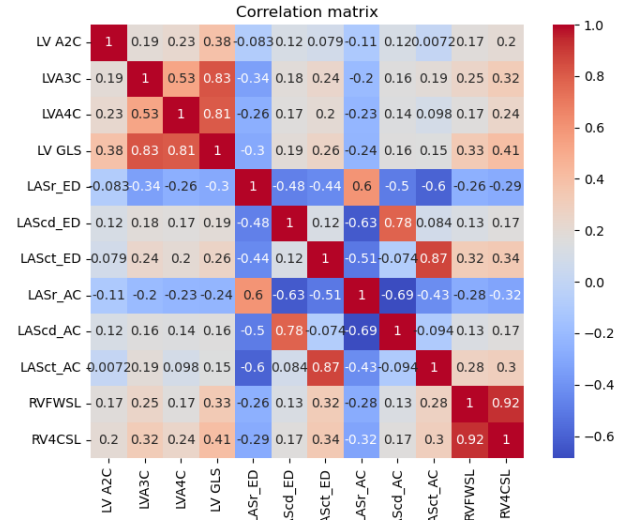

Figure 3.7.: Matrix showing correlations between various input data parameters from Category 2.

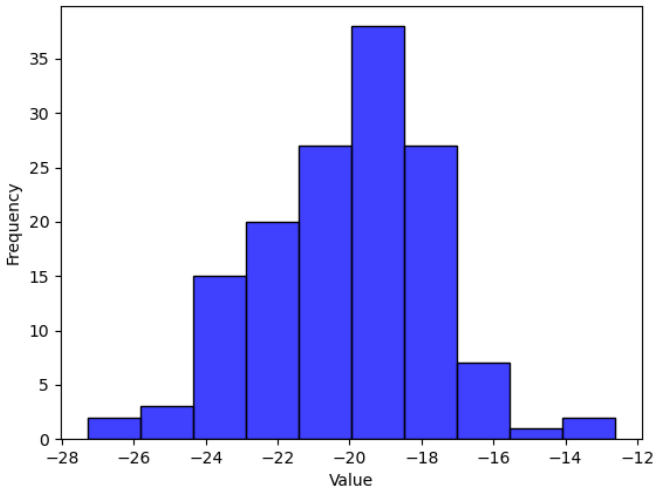

Figure 3.8.: Presentation of sample histogram for regression data.

4. Implementation Of Machine Learning Algorithms

This chapter will highlight the various methods used in the learning process that were used in both versions of the scenario. They have been discussed in detail with a presentation of their advantages and disadvantages in Section 2.2. Due to the small dataset for both scenarios, it was decided to use four shallow learning models and only one model belonging to the deep learning category. In the case of a larger dataset, it would most likely be advisable to use more deep learning models.

4.1. Scenario 1: Echo-Cardiological Prediction

To begin with, before training any models, decisions were made to divide the available data (in this case 89 samples) into three parts: training, validation, and test subsets. It was also determined in what proportions this would be done according to a pre-written order of 60, 20, and 20. The input data was assigned to the X variable, while the labels were assigned to the y variable. In addition, 5-fold cross-validation was used for each model to assess the stability of the model in all combinations. Each of the tested models had the MAE (mean absolute error) and MAPE (mean absolute percentage error) metrics determined after making predictions on the validation and test sets. In addition to the metrics, scatter plots were displayed showing the juxtaposition of predicted data (highlighted in red or blue, depending on whether it was a validation or test set) and actual data (highlighted in green).

Linear Regression The first model used in the data analysis is linear regression. Tuning of hyperparameters was not used here because using regularization such as ridge or lasso made the model oscillate around the mean value. The model was trained for each of the 12 labels (derived from Category 2); the MAPE values for the 5 validations of the test set are shown in Table 4.1.

The table shows that the model performs well in most folds except for 3, which deviates very strongly from the others, suggesting the difficulty of fitting the model acutely to such a layout. For the rest of the folds, the values are similar, suggesting that the model performs fairly stably in the remaining cases. The best results for each feature are shown in Table 4.2.

Label name	MAPE fold 1	MAPE fold 2	MAPE fold 3	MAPE fold 4	MAPE fold 5	MAPE mean
LV A2C	23.32%	13.56%	1171.92%	19.15%	17.50%	268.49%
LV A3C	12.96%	10.15%	211.19%	17.45%	14.99%	53.35%
LV A4C	15.46%	11.30%	529.70%	16.70%	19.49%	118.73%
LV GLS	10.54%	9.41%	292.45%	13.80%	12.11%	67.46%
LASr ED	29.57%	32.51%	461.14%	22.37%	33.16%	155.15%
LAScd ED	42.84%	56.03%	126.79%	32.50%	59.76%	63.78%
LASct ED	89.87%	33.18%	120.01%	41.96%	29.11%	62.22%
LASr AC	23.76%	35.25%	590.88%	28.35%	19.06%	39.06%
LAScd AC	33.79%	43.91%	801.65%	33.32%	50.19%	52.17%
LASct AC	76.99%	28.50%	124.33%	33.84%	25.23%	57.58%
RVFWSL	21.84%	13.09%	463.05%	22.54%	24.35%	89.77%
RV4CSL	21.75%	15.97%	514.56%	20.45%	17.42%	57.03%

Table 4.1.: MAPE results for each fold and mean MAPE for each feature.

Label name	MAE	MAPE (%)
LV A2C	2.83	13.56%
LV A3C	2.33	10.15%
LV A4C	2.49	11.30%
LV GLS	2.09	9.41%
LASr ED	7.92	22.37%
LAScd ED	10.44	42.84%
LASct ED	5.10	33.18%
LASr AC	10.77	35.25%
LAScd AC	6.94	33.79%
LASct AC	3.79	28.50%
RVFWSL	3.41	13.09%
RV4CSL	3.25	15.97%

Table 4.2.: Prediction results of individual labels for the linear regression model for the best folds.

Comparing it with the results for the average value in Table 4.1 a big difference is noticeable, which emphasizes the diversity of samples in each folder. In addition, the diversity of results between features is noticeable, which can be divided into easier (those whose names start with LV and RV) and more difficult (those with a name starting with LAS) to predict. The simpler features achieve MAPE values below 20, while the more difficult ones do the opposite. When we look at the best-predicted feature *LV GLS* and its more detailed graphs found in Fig. 4.1.

For this feature, the model on the validation set predicts values within a larger range, as can be seen on the left side of the Fig. 4.1. For the test samples, the model predicts values

closer to the mean. In addition, the model does not handle strong outliers, such as sample eight of the graph on the right side of Fig.4.1.

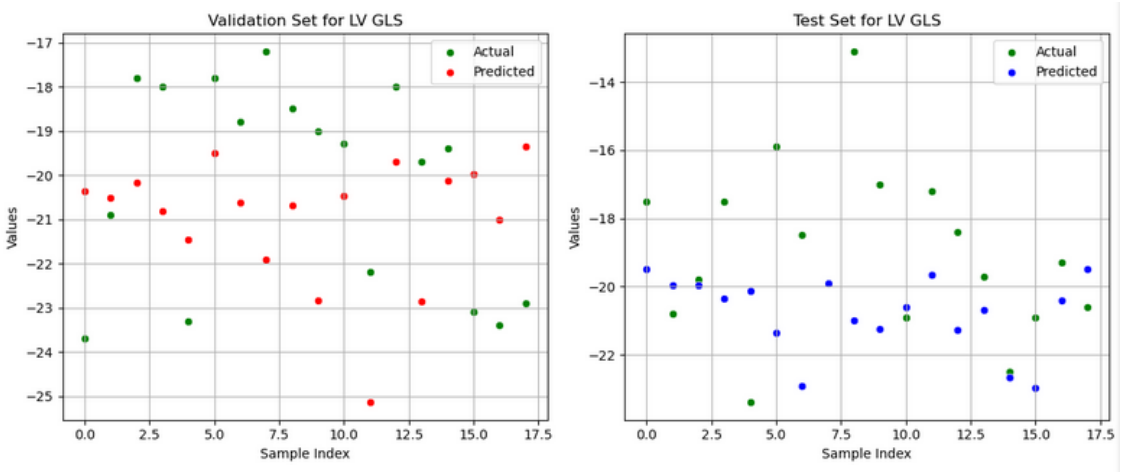

Figure 4.1.: Detailed visualization of the results for the best-predicted label of the linear regression model.

KNN The second trained model is k-nearest neighbors. For this model, hyperparameter tuning was performed, as illustrated in Listing 4.1. It was decided to go for low values of the hyperparameter *n_neighbors* to increase precision, but this may translate into overfitting. Table 4.3 displays the results for each cross-validation of the KNN model.

```
param_grid = {
    'n_neighbors': [2, 3, 4, 5],
    'weights': ['uniform', 'distance'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
    'p': [1, 2]
}
```

Listing 4.1.: Representation of hyperparameters tuned during KNN model training.

For most features, the model achieves comparable results, as can be seen in the column on average values for all cross validations. The exception is again fold 3, where for *LASct ED* and *LASct AC* the model has very bad predictions. The KNN model results for individual labels are generally better (have lower MAPE values) compared to the linear regression model.

Label name	MAPE fold 1	MAPE fold 2	MAPE fold 3	MAPE fold 4	MAPE fold 5	MAPE mean
LV A2C	22.48%	15.80%	17.52%	20.60%	16.28%	18.94%
LV A3C	13.02%	10.10%	16.17%	18.64%	10.83%	13.15%
LV A4C	15.33%	10.39%	15.96%	14.99%	13.43%	13.62%
LV GLS	10.62%	9.99%	12.62%	14.97%	12.09%	11.66%
LASr ED	27.18%	16.90%	19.31%	25.32%	33.18%	24.78%
LAScd ED	34.86%	44.14%	26.88%	33.96%	36.29%	35.43%
LASct ED	82.59%	24.77%	57.39%	52.01%	24.55%	48.06%
LASr AC	26.98%	18.93%	19.12%	29.16%	23.31%	23.70%
LAScd AC	34.39%	37.67%	40.34%	34.80%	32.89%	36.42%
LASct AC	70.03%	22.66%	48.63%	42.15%	21.10%	36.71%
RVFWSL	21.45%	17.26%	43.12%	24.49%	26.66%	26.60%
RV4CSL	23.23%	22.37%	30.67%	20.49%	22.24%	23.80%

Table 4.3.: Results for each fold and MAPE averages for the KNN model.

Table 4.4 illustrates the best results for each feature along with the tuned hyperparameters. The first thing noticed is a much smaller split between more difficult and easier-to-predict features. The KNN model tries to match each feature with hyperparameters, as can be seen in the *n_neighbors* and *p* columns. The model for most of the features achieves a value of MAPE lower than 20%, suggesting that it handles the data well, and only in a single case does it achieve a value above 30%. The best predicted feature in terms of MAPE is *LV GLS*, which was looked at more closely, as shown in Fig. 4.2.

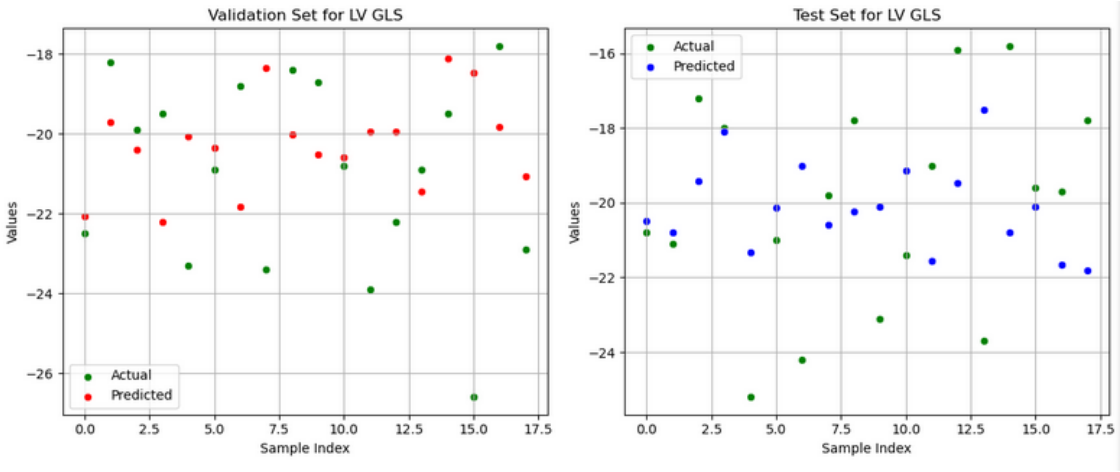

Figure 4.2.: Detailed visualization of the results for the best-predicted label of the KNN model.

The validation and test subsets perform comparably with the data provided. In the case of the test set, trend detection is noticeable, as well as very close predictions that almost coincide with the actual values, as can be seen in the first two samples. In the case of more

Label name	MAE	MAPE (%)	Algorithm	n_neighbors	p	Weights
LV A2C	3.24	15.80%	auto	4	2	uniform
LV A3C	2.32	10.10%	auto	5	1	uniform
LV A4C	2.28	10.39%	auto	5	2	uniform
LV GLS	2.18	9.99%	auto	5	1	uniform
LASr ED	6.12	16.90%	auto	3	1	uniform
LAScd ED	5.05	26.88%	auto	5	2	uniform
LASct ED	3.94	24.55%	auto	5	1	uniform
LASr AC	5.49	18.93%	auto	4	2	uniform
LAScd AC	5.86	32.89%	auto	2	1	uniform
LASct AC	2.88	21.10%	auto	5	1	uniform
RVFWSL	4.10	17.26%	auto	5	1	uniform
RV4CSL	4.08	20.49%	auto	5	2	distance

Table 4.4.: Prediction results of individual labels for the KNN model along with selected hyperparameters.

outliers, the model further fails to cope with them.

Random Forest The third model considered is the random forest. In the case of this algorithm, there are many hyperparameters, of which we have chosen the following for regularization purposes (we can find a detailed description of each hyperparameter in the detailed description of the scikit-learn library [25])

- *n_estimators* — number of trees in the forest.
- *min_samples_split* — minimum number of samples for further division.
- *min_samples_leaf* — determines the minimum number of samples in a leaf.
- *max_depth* — maximum tree depth.

The values of the hyperparameters that were used for the adjustment are presented in Listing 4.2.

A total of 240 combinations were tested, which, with 12 labels, translated into training in the neighborhood of 30 minutes on a 12-core processor. The most different values used for training had the hyperparameter *n_estimators* in order to more easily adapt to the varying data in each feature. The MAPE results for the different folds are shown in Table 4.5.

The table shows that the model has similar performance between different folds for most characteristics. From this, it follows that random forest performs well with the entire found set regardless of whether the test set is once easier to predict or more difficult. The only

```
param_grid = {
    'n_estimators': [1600, 1800, 2000, 2200, 2400],
    'min_samples_split': [2, 3, 4, 5],
    'min_samples_leaf': [1, 2, 3, 4],
    'max_depth': [2, 3, 4]
}
```

Listing 4.2.: Representation of hyperparameters tuned during Random Forrest model training.

Label name	MAPE fold 1	MAPE fold 2	MAPE fold 3	MAPE fold 4	MAPE fold 5	MAPE mean
LV A2C	23.08%	13.61%	11.07%	17.99%	14.31%	16.01%
LV A3C	11.95%	8.98%	14.38%	17.76%	10.47%	12.71%
LV A4C	15.04%	13.78%	15.05%	16.68%	12.87%	14.68%
LV GLS	10.28%	11.08%	12.04%	13.96%	10.32%	11.54%
LASr ED	27.05%	20.39%	22.41%	22.28%	29.64%	24.35%
LAScd ED	34.90%	47.24%	26.11%	34.38%	27.25%	33.97%
LASct ED	84.50%	27.00%	63.77%	42.56%	18.82%	47.33%
LASr AC	26.24%	16.56%	18.10%	30.05%	19.01%	21.99%
LAScd AC	31.48%	39.95%	34.12%	34.13%	29.78%	33.89%
LASct AC	72.16%	22.68%	52.30%	33.73%	16.57%	39.49%
RVFWSL	23.29%	16.79%	35.74%	22.39%	21.56%	23.96%
RV4CSL	23.62%	18.90%	25.61%	20.14%	21.03%	21.86%

Table 4.5.: Results for each fold and MAPE averages for the random forest model.

labels that do not meet this rule are *LASct ED* and *LASct AC*, which outperform the others in fold 1 and fold 3, which also translates into an average value. The best predicted folds along with the MAE value and the best hyperparameters are presented in Table 4.6.

Looking at the results, one can see the variety of adapted hyperparameters. For this reason, it took several hours (in around the value of 5) to train this model, because we first had to test several sets with different values of hyperparameters. A particular problem was to determine the number of estimators (trees), and it began by testing values ‘50‘. Values ranging from ‘50‘ to ‘5000‘ were tested, finally deciding on a range from ‘1600‘ to ‘2400‘ with intervals of ‘200‘. This was the optimal value, since setting a larger value did not improve the results, but only increased the learning time. In addition, it was noted that the MAPE for all features did not exceed the value of 30%, which, compared to the non-significant differences from the average value from Table 4.5, allows us to conclude that the model performs well for most features. To further interpret the results, the graphs for the *LV GLS* label (see Fig. 4.3) are shown with the previous models.

For the test set, the model mostly oscillates around the mean value and only in two cases

Label Name	MAE	MAPE (%)	Max Depth	Min Samples Leaf	Min Samples Split	n Estimators
LV A2C	2.29	11.07%	3	4	2	2400
LV A3C	2.12	8.98%	2	4	2	1600
LV A4C	2.32	12.87%	2	4	2	2000
LV GLS	2.21	10.28%	2	1	2	1600
LASr ED	7.55	20.39%	2	4	2	2400
LAScd ED	5.03	26.11%	2	4	2	1800
LASct ED	3.08	18.82%	2	4	2	2400
LASr AC	5.05	16.56%	2	4	2	2400
LAScd AC	4.64	29.78%	2	4	2	1600
LASct AC	2.28	16.57%	3	4	2	2400
RVFWSL	4.34	16.79%	2	3	2	2400
RV4CSL	3.75	18.90%	2	1	3	2400

Table 4.6.: Prediction results of individual labels for the random forest model along with selected hyperparameters.

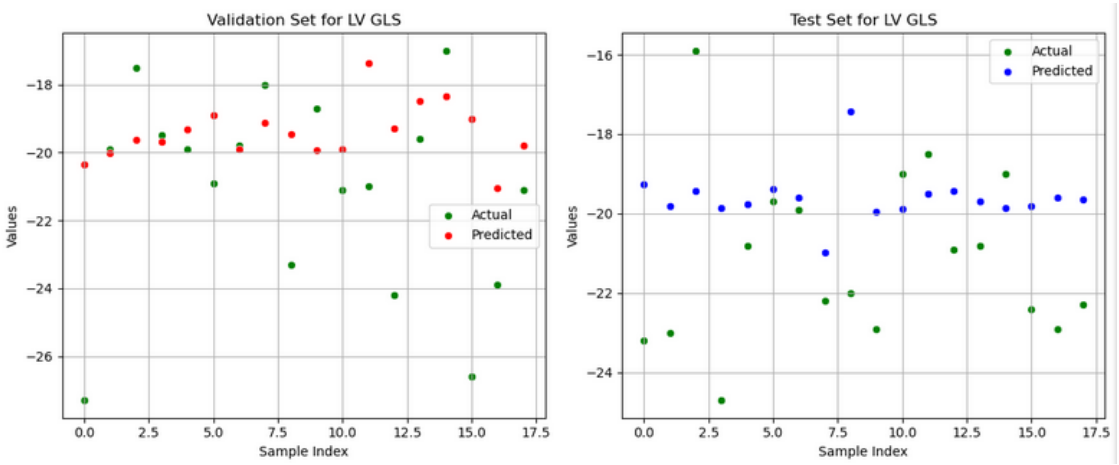

Figure 4.3.: Results for the validation and training set for the label *LV GLS*.

predicts a value outside the -20 to -18 range. For outliers, the model is not able to predict them well. This type of fit may suggest the need for more outlier samples in the learning set.

Gradient Boosting As the fourth model used for training is gradient boosting. Initially, due to the similarity of the model to the previously discussed random forest algorithm, it was decided to use the same dictionary with hyperparameters. The only difference was the addition of the hyperparameter *learning_rate* responsible for the tree's contribution to the learning process. This approach gave worse results at first, but after gradually increasing the number of estimators, the results began to improve. It was also noted that decreasing the learning rate while increasing the number of trees made the model perform better and

better. Subsequently, the number of hyperparameters was reduced by those pertaining to the single tree itself, leaving only *max_depth* and adding regularization *subsample*. In the final version, the dictionary containing the parameters of the tuned hyperparameters can be observed in Listing 4.3.

```
param_grid = {
    'n_estimators': [500, 1000, 2000],
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [3, 4, 5],
    'subsample': [0.5, 0.6, 0.8],
}
```

Listing 4.3.: Representation of hyperparameters tuned during Gradient Boosting model training.

The dictionary is the final version that gave the most effective results. Versions with a larger number of estimators, such as 10000 with a much lower learning rate of 0.001 or 0.0001, were also tested, but such parameters did not translate into better results. They only increased the learning time of the model, and setting the learning rate too low translated into a lack of training of the model (the predicted values were always the average). The final results for each fold can be seen in Table 4.7.

Label name	MAPE fold 1	MAPE fold 2	MAPE fold 3	MAPE fold 4	MAPE fold 5	MAPE mean
LV A2C	23.74%	16.21%	17.70%	20.74%	15.62%	18.80%
LV A3C	14.49%	10.91%	16.19%	17.96%	11.80%	14.27%
LV A4C	16.87%	15.30%	17.48%	16.91%	12.98%	15.91%
LV GLS	11.92%	13.13%	14.12%	14.93%	10.57%	12.93%
LASr ED	29.35%	21.56%	35.24%	25.95%	32.85%	28.99%
LAScd ED	41.66%	45.30%	35.55%	34.83%	31.22%	37.71%
LASct ED	95.96%	29.64%	67.09%	48.83%	30.28%	54.36%
LASr AC	26.97%	20.99%	29.89%	35.52%	26.22%	27.92%
LAScd AC	37.58%	38.59%	42.89%	33.23%	31.09%	36.67%
LASct AC	82.19%	23.95%	55.83%	39.88%	25.32%	45.43%
RVFWSL	26.19%	19.29%	38.13%	20.11%	29.75%	26.70%
RV4CSL	26.33%	21.00%	29.60%	19.30%	21.86%	23.61%

Table 4.7.: Results for each fold and MAPE averages for the gradient boosting model.

As with the random forest model, the gradient boosting model does not cope with the labels *LASct ED* and *LASct AC* in folds 1 and 3, where it has very high MAPE values. For the rest of the features, the model obtains comparable values between folds, which can also be seen from the result of the average value, which does not deviate strongly from the other

values. For further analysis, Tables 4.8 were created with the best predicted folds for each feature, along with information on MAE and hyperparameters.

Label name	MAE	MAPE (%)	learning rate	max depth	n estimators	subsample
LV A2C	2.94	15.62%	0.01	3	500	0.5
LV A3C	2.47	10.91%	0.01	3	500	0.6
LV A4C	2.37	12.98%	0.1	4	2000	0.5
LV GLS	1.92	10.57%	0.01	3	500	0.6
LASr ED	8.06	21.56%	0.01	3	500	0.5
LAScd ED	6.28	31.22%	0.1	5	500	0.8
LASct ED	4.87	29.64%	0.01	3	500	0.5
LASr AC	6.25	20.99%	0.01	5	500	0.8
LAScd AC	5.18	31.09%	0.01	3	500	0.6
LASct AC	3.32	23.95%	0.01	5	500	0.5
RVFWSL	4.88	19.29%	0.01	4	500	0.5
RV4CSL	4.01	19.30%	0.1	4	2000	0.6

Table 4.8.: Prediction results and best parameters for individual labels for Gradient Boosting model.

The model for the best values in most cases achieves MAPE values below 30%. In addition, the model struggles to match individual features, as can be seen in the differences in hyperparameter values between labels. As the MAE and MAPE results are comparable to previous models, it was stated to display detailed graphs for the label *LV GLS*, which was used in previous cases. The results of these graphs are shown in Fig. 4.4.

For both the test set and the validation set, the model tries to match the actual data, which can translate into worse results compared to the random forest. The test set performs poorly with outliers, while it is able to predict the value well with the rest. However, it still lacks a good enough representation of outliers for the model to at least come close to the actual value. Currently, in the case of a strong outlier, the model predicts a value close to the mean.

MLP The last model used for training is the only one that can be classified in the deep learning category. Therefore, the hyperparameters are different from the ones for the previous models, and they are as follows:

- *hidden_layer_sizes* — determines the number of neurons in the hidden layers.
- *activation* — determines the activation function (when the perceptron gives 1 to be 0).
- *learning_rate_init* — determines the speed of learning.
- *alpha* — determines the regularization factor.

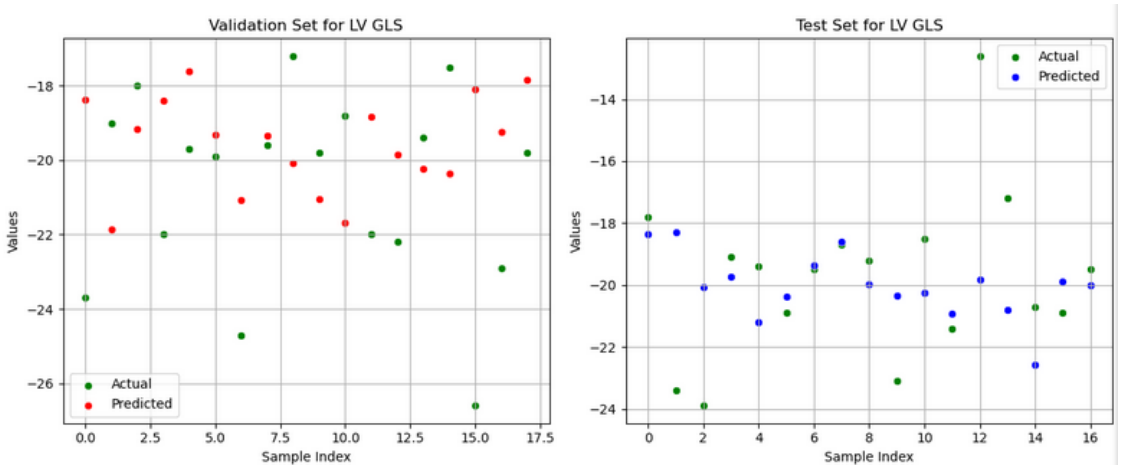

Figure 4.4.: Detailed results for one of the best-predicted Gradient Boosting model labels.

- *max_iter* — determines the maximum number of training epochs (iterations).

The dictionary with trained hyperparameters can be observed in Listing 4.4.

```
    param_grid = {
        'hidden_layer_sizes': [(32,), (64,)],
        'activation': ['relu'],
        'learning_rate_init': [0.01],
        'alpha': [0.0001],
        'max_iter': [2000]
    }
```

Listing 4.4.: Representation of hyperparameters tuned during Multilayer Perceptron model training.

In this case, the dictionary does not consist of a large number of possible combinations due to the small size of the dataset. Therefore, the limitation is more limited to the number of perceptrons in the hidden layer. The MAPE results for individual folds along with the average value can be found in Table 4.9.

The model does not handle the provided data well as can be seen from the average values. In addition, it is noticeable that there are large disparities between successive folds in most of the labels, which makes it difficult for the model to fit a variety of test sets. Even for the *LV GLS* label, there is a noticeable discrepancy between the best predicted folder (about

Label name	MAPE fold 1	MAPE fold 2	MAPE fold 3	MAPE fold 4	MAPE fold 5	MAPE mean
LV A2C	21.47%	19.04%	51.74%	17.27%	43.55%	30.62%
LV A3C	14.72%	15.86%	27.44%	31.56%	43.55%	26.63%
LV A4C	22.19%	19.52%	28.65%	27.88%	17.69%	23.18%
LV GLS	15.08%	12.31%	23.43%	21.65%	21.17%	18.73%
LASr ED	39.19%	32.21%	62.11%	33.18%	82.11%	49.76%
LAScd ED	48.00%	81.35%	91.77%	43.31%	147.04%	82.29%
LASct ED	100.70%	53.62%	134.90%	59.77%	88.85%	87.57%
LASr AC	39.35%	36.71%	92.28%	34.04%	56.39%	51.75%
LAScd AC	51.87%	79.78%	110.66%	49.70%	118.52%	82.11%
LASct AC	79.74%	49.44%	96.60%	50.35%	64.49%	68.12%
RVFWSL	31.15%	30.66%	93.76%	24.55%	55.29%	47.08%
RV4CSL	30.90%	33.60%	92.20%	31.27%	35.18%	44.63%

Table 4.9.: Results for each fold and MAPE averages for the neural network model.

12%) and the worst (about 23%). To further illustrate the results, Table 4.10 was created showing the MAE, MAPE, and hyperparameters of the best-predicted folds for each label.

Label name	MAE	MAPE (%)	activation	alpha	hidden_layer_sizes	learning_rate_init	max_iter
LV A2C	3.07	17.27%	relu	0.0001	(32,)	0.01	2000
LV A3C	3.44	15.86%	relu	0.0001	(32,)	0.01	2000
LV A4C	3.29	17.69%	relu	0.0001	(32,)	0.01	2000
LV GLS	2.53	12.31%	relu	0.0001	(32,)	0.01	2000
LASr ED	10.84	32.21%	relu	0.0001	(64,)	0.01	2000
LAScd ED	8.08	43.31%	relu	0.0001	(64,)	0.01	2000
LASct ED	7.90	53.62%	relu	0.0001	(64,)	0.01	2000
LASr AC	10.14	34.04%	relu	0.0001	(64,)	0.01	2000
LAScd AC	7.79	49.70%	relu	0.0001	(32,)	0.01	2000
LASct AC	6.29	49.44%	relu	0.0001	(32,)	0.01	2000
RVFWSL	6.02	24.55%	relu	0.0001	(32,)	0.01	2000
RV4CSL	6.25	30.90%	relu	0.0001	(32,)	0.01	2000

Table 4.10.: Prediction results and best parameters for individual labels for Multilayer Perceptron model.

Compared to the previous results, the model stands very much at a disadvantage. For more than half of the labels, the MAPE value exceeds 30%, and for three it oscillates around the 50% value. This just shows that the model cannot cope with the provided data. The hyperparameters, due to the small dataset, are also unable to help improve the results. To illustrate the more detailed information once again, the label *LV GLS* was used (see Fig. 4.5).

For the test set, the model performs worse at predicting values close to the mean compared to other models. In addition, the MLP tries to predict values that are strong outliers, which even in the case of one sample succeeds, but is not the norm. In summary, this model is very erratic, as it is capable of being very wrong despite well-predicted samples.

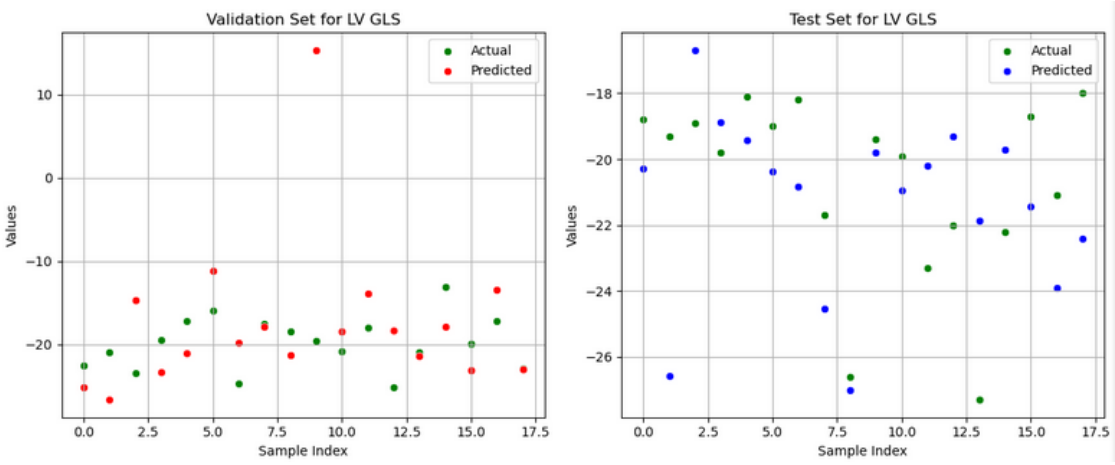

Figure 4.5.: Detailed results for one of the best-predicted Multilayer Perceptron model labels.

4.1.1. Results Summary

In order to compare and summarize the results for all trained models, a Table 4.11 was created containing the average MAPE value for the test sets.

Label Name	Linear Regression (%)	KNN (%)	Random Forest (%)	Gradient Boosting (%)	MLP (%)
LV A2C	268.49	18.94	16.01	18.80	30.62
LV A3C	53.35	13.15	12.71	14.27	26.63
LV A4C	118.73	13.62	14.68	15.91	23.18
LV GLS	67.46	11.66	11.54	12.93	18.73
LASr ED	155.15	24.78	24.35	28.99	49.76
LAScd ED	63.78	35.43	33.97	37.71	82.29
LASct ED	62.22	48.06	47.33	54.36	87.57
LASr AC	39.06	23.70	21.99	27.92	51.75
LAScd AC	52.17	36.42	33.89	36.67	82.11
LASct AC	57.58	36.71	39.49	45.43	68.12
RVFWSL	89.77	26.60	23.96	26.70	47.08
RV4CSL	57.03	23.80	21.86	23.61	44.63

Table 4.11.: Comparison of mean MAPE scores for linear regression, KNN, Random Forest, Gradient Boosting and MLP by trained labels.

Based on the table, the labels can be divided into three groups, which respectively start with the letters *LV* (first 4 lines), *LAS* (next 6 lines) and *RV* (last 2 lines). The models deal with each of the different groups in different ways. Linear regression performs the worst of all the models, as can be seen from the high MAPE scores, even though it performs well in the best-predicted folds, as illustrated by Table 4.12. This shows that the model is unstable and sensitive to a variety of data. The MLP model performs better than linear regression but has noticeable problems with labels starting with the letters *LAS*, where for most labels the MLP

model achieves MAPE values of more than 50%. The other models, namely KNN, random forest, and gradient boosting, achieve better results and are comparable among themselves. Each of the three models performs best with labels of *LV* and *RV* types. In contrast, for most *LAS* labels, the models perform noticeably worse, as shown by the results.

Label Name	Linear Regression (%)	KNN (%)	Random Forest (%)	Gradient Boosting (%)	MLP (%)
LV A2C	13.56	15.80	11.07	15.62	17.27
LV A3C	10.15	10.10	8.98	10.91	15.86
LV A4C	11.30	10.39	12.87	12.98	17.69
LV GLS	9.41	9.99	10.28	10.57	12.31
LASr ED	22.37	16.90	20.39	21.56	32.21
LAScd ED	42.84	26.88	26.11	31.22	43.31
LASct ED	33.18	24.55	18.82	29.64	53.62
LASr AC	35.25	18.93	16.56	20.99	34.04
LAScd AC	33.79	32.89	29.78	31.09	49.70
LASct AC	28.50	21.10	16.57	23.95	49.44
RVFWSL	13.09	17.26	16.79	19.29	24.55
RV4CSL	15.97	20.49	18.90	19.30	30.90

Table 4.12.: Comparison of best MAPE results from folds for linear regression, KNN, Random Forest, Gradient Boosting and MLP by trained labels.

Table 4.12 shows a summary of the best results for each label. In the case of this comparison, all models except the MLP model represent comparable results, which show that the linear regression model does well with stable folds and underperforms with problematic ones. In the case of the MLP model, when comparing both the results from Table 4.11 and Table 4.12 it is noticeable that it performs worse than the other three models, which may suggest that the MLP model needs more data to train a more complex algorithm. For the KNN, random forest, and gradient boosting models, the results in both Tables 4.11 and 4.12 are very similar, which may indicate that the dataset has too few samples with outliers, making the models unable to achieve better values. However, of the three, the KNN and gradient boosting models should be tested in further stages of research due to more frequent attempts to adjust for outliers.

4.2. Scenario 2: Predicting Newborn Development

Here, the initial approach was the same as for Scenario 1. The available dataset (142 samples) was divided into three subsets: training, validation, and test sets, which have 60%, 20%, and 20% of the original set, respectively. The input data were assigned to the X variable and the labels to the y variable. In addition, 5-fold cross-validation was applied for each model. Due to the presence of regression and classification data, two ways of evaluating the models were determined. For regression data, MAE and MAPE metrics were used, as in Scenario 1. In addition, scatter plots were made showing a comparison between predicted

(plotted in red or blue) and actual values (plotted in green). In the case of classification data, the f1_score parameter was used as an indicator, along with a plot of the confusion matrix. In addition, the isolation forest algorithm was used to detect anomalies.

Linear Regression and Logistic Regression The first trained models were linear regression and logistic regression. The use of two models is due to the fact that the first model is suitable for regression data, while the second model is suitable for classification data. The following models used in this scenario have versions for both types of data. Ridge regularization with the hyperparameter *alpha* set to 1 was used to train the regression data, while the set of hyperparameters illustrated in Listing 4.5 was used for the classification data.

```
param_grid_lr_cla = {
    'C': [5, 10, 15],
    'solver': ['liblinear', 'saga'],
    'max_iter': [1000, 2000, 3000]
}
```

Listing 4.5.: Representation of hyperparameters tuned during logistic regression model training for classification data.

The results for each fold were presented in Table 4.13 (regression) and in Table 4.14 (classification).

In the case of regression data (see Table 4.13), only for the *HC - WYPI* label does the model take values below 10% MAPE for each fold. This shows that the model performs well with just this group of data. The other labels reach values exceeding 80% MAPE, suggesting that the model cannot handle this data. This is particularly evident in the label *HC (Z Score) -WYPI*, which has the highest average MAPE value, additionally having large differences in values between successive folds. In order to better illustrate the regression data, graphs were created that included predicted values as well as actual values. This is rearranged in Fig. 4.6 and refers to the label with the best MAPE result - *HC - WYPI*.

Label name	MAPE fold 1	MAPE fold 2	MAPE fold 3	MAPE fold 4	MAPE fold 5	MAPE mean
HC - WYPI	6.73%	5.40%	5.24%	5.99%	6.91%	6.05%
HC (pc) -WYPI	156.11%	86.94%	105.63%	93.41%	127.40%	113.90%
HC (Z Score) -WYPI	245.53%	458.23%	526.20%	276.85%	146.19%	330.60%
dni hospitalizacji	94.21%	277.58%	212.21%	106.93%	275.51%	193.29%

Table 4.13.: Results for each fold and MAPE averages for ridge regression model on regression data.

For samples with values close to the mean, the linear regression model does well, predicting values with a difference of a few hundredths. For outliers, the model does poorly in predicting trends, as can be seen in sample number 3 from the test set (see Fig. 4.6). Unfortunately, the model does not cope with very outliers, such as sample 21, where the predicted value differs from the actual value by almost 6 points.

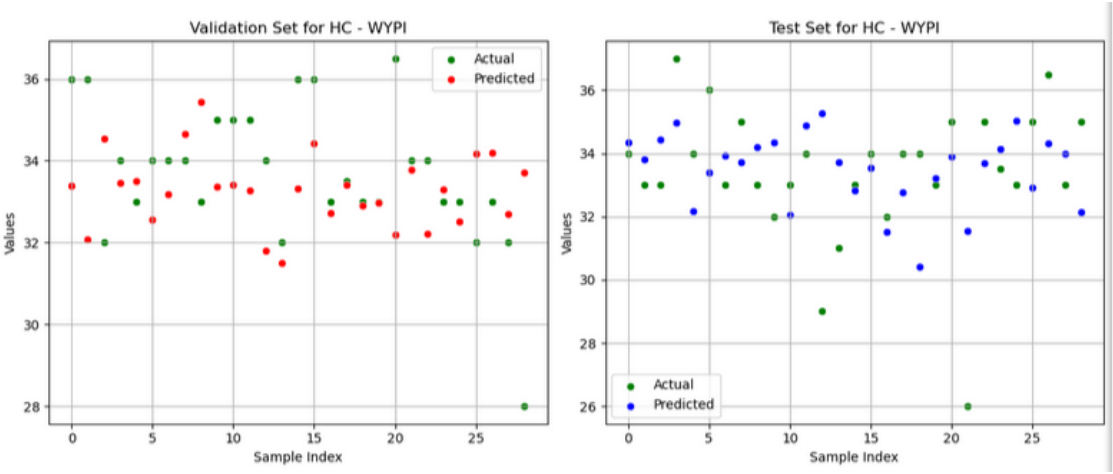

Figure 4.6.: Prediction results for regression data labels of the linear regression model.

In the case of classification data (see table 4.14), problems are noticeable with the label *H35.1* for which the average value is 0. In the case of the feature *P27.1*, the model is able to obtain a value other than 0 for four folds.

Label name	F1 Score fold 1	F1 Score fold 2	F1 Score fold 3	F1 Score fold 4	F1 Score fold 5	F1 Score mean
P27.1	0.17	0.22	0.18	0	0.29	0.172
H35.1	0	0	0	0	0	0

Table 4.14.: Results for each fold and f1_score averages logistic regression model classification data.

For the classification data for both available labels, confusion matrices were calculated for the test set, as shown in Fig. 4.7. On the left is the matrix for the label *P27.1*, while on the right is the matrix for the label *H35.1*. For a better comparison, the matrices are for fold 2, where the f1 score for the label *P27.1* is 0.17 while that for *H35.1* is 0. In both cases, the logistic regression model does not do well in predicting the value of 1, classifying most samples as 0. The difference in model accuracy is due to the number of samples with a value of 1, where for the matrix on the right there is one such sample and one misclassified as 1. For the matrix on the left, there are exactly six samples with a value of 1, but you can see that one was predicted correctly. Both graphs suggest that the classification model is not doing too well with the data for both labels.

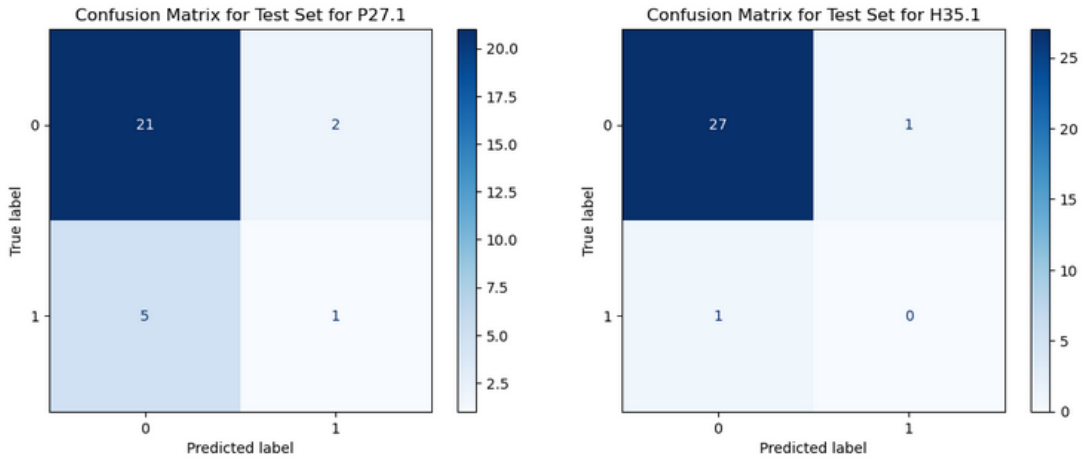

Figure 4.7.: Confusion matrix for the label *P27.1* (left side) and the label *H35.1* (right side).

KNN The second model trained is the k-nearest neighbors algorithm, which already has versions for classification and regression data. Hyperparameter tuning was carried out for this model, where for regression data this is illustrated by Listing 4.6, and for classification data by Listing 4.7.

```
param_grid_knn_reg = {
    'n_neighbors': [2, 3, 5, 7, 9],
    'weights': ['uniform', 'distance'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
    'p': [1, 2]
}
```

Listing 4.6.: Representation of hyperparameters tuned during KNN model training for regression data.

For the regression data, a more extensive dictionary was used due to the variety of data in each feature. For classification data, a simpler dictionary was used because, for example, increasing the hyperparameter *n_neighbors* to a value of 7 caused problems in predicting the correct values of 0, without improving the result for a value of 1.

MAPE results for individual folds and specific labels are shown in Table 4.15 for regression data and in Table 4.16 for classification data.

The table on regression data shows that the model again performs well with the label *HC - WYPI*. The MAPE values for this feature are below 10% for each fold, suggesting high

```
param_grid_knn_cla = {
    'n_neighbors': [3, 4],
    'weights': ['uniform', 'distance'],
    'p': [1, 2]
}
```

Listing 4.7.: Representation of hyperparameters tuned during KNN model training for classification data.

Label name	MAPE fold 1	MAPE fold 2	MAPE fold 3	MAPE fold 4	MAPE fold 5	MAPE mean
HC - WYPI	5.57%	4.40%	4.78%	4.63%	6.39%	5.55%
HC (pc) -WYPI	64.84%	41.46%	56.69%	69.84%	64.58%	59.78%
HC (Z Score) -WYPI	604.34%	189.65%	149.43%	124.05%	151.22%	243.94%
dni hospitalizacji	59.03%	74.12%	65.47%	65.97%	73.13%	67.74%

Table 4.15.: Results for each fold and MAPE averages for KNN model for regression data.

predictive accuracy of the model. The other labels show better performance than in the case of the linear regression model. The labels *HC (pc) -WYPI* and *dni hospitalizacji*, although they reach values above 50% in most cases, can be said to be stable, as individual folds reach comparable values. Further noticeable is the problem with the label *HC (Z Score) -WYPI*, which for all folds, reaches MAPE values above 100%. In one case, even exceeding 600% MAPE.

Label name	F1 Score fold 1	F1 Score fold 2	F1 Score fold 3	F1 Score fold 4	F1 Score fold 5	F1 Score mean
P27.1	0.18	0.60	0	0.18	0	0.192
H35.1	0	0	0	0	0	0

Table 4.16.: Results for each fold and f1_score averages KNN model classification data.

For the classification data (see table 4.16), accuracy similar to the logistic regression model is noticeable. Again, the model has a problem with the label *H35.1*, where it is not able to predict well a rare sample with a value of 1. For the feature *P27.1* the result is better because in three folds we see a value other than 0.

To further visualize the model results for each label, Tables 4.17 and 4.18 were created, where the best-conducted folds and their hyperparameters are shown for both regression and classification data. For the regression data, the hyperparameter *n_neighbors* reached values of 7 or 9, suggesting that the model is more stable to the most frequent data but less flexible to outliers. The MAPE value for *HC (Z Score)-WYPI* even for the best fold, further exceeds 100% but is paired with a very low MAE of less than 1. This combination suggests that the values for this label are very small. This problem is illustrated in Fig. 4.8.

Label name	MAE	MAPE (%)	Algorithm	n_neighbors	p	Weights
HC - WYPI	1.4699	4.40%	auto	9	2	distance
HC (pc) -WYPI	17.2553	41.46%	auto	7	1	distance
HC (Z Score) -WYPI	0.7456	124.05%	auto	9	1	distance
dni hospitalizacji	23.8980	65.47%	auto	7	2	uniform

Table 4.17.: Prediction results of individual labels for the KNN model along with selected hyperparameters for regression data.

It can be seen from the graphs that the set of values for this label is very small, and in the case of this test set (see the right graph of Fig. 4.8) ranges from -2 to 1. Therefore, it can be concluded that any prediction with even a small difference from the actual value strongly affects the final MAPE result. In the case of a very outlier sample, such as the one with a value of 6 in the validation set (see the left graph in Fig.4.8), a large difference between the actual value and the correct value strongly affected the final MAE and MAPE result.

Label name	F1 Score	n_neighbors	p	Weights
P27.1	0.6	4	1	distance
H35.1	0	4	1	uniform

Table 4.18.: Prediction results of individual labels for the KNN model along with selected hyperparameters for classification data.

In the case of Table 4.18 on classification data, there is a noticeable similarity in the hyperparameters, which differ only in the value in *weights*. To further illustrate the data, confusion matrices are presented in Fig. 4.9 for both labels with the highest accuracy rate. There is a big difference in the matrix on the left side referring to the label *P27.1* compared to the logistic regression model. The KNN model achieves 50% efficiency on samples with a value of 1 (three samples well predicted). On the matrix on the right side pertaining to the label *H35.1* there is a problem even trying to predict a value of 1 by classifying all samples as a value of 0.

Random Forest The next model considered is the random forest built of decision trees. For this model, there are multiple hyperparameters, of which for regularization purposes we chose the same ones described in Scenario 1. Again, two dictionaries were created: one for regression data, which is described in Listing 4.8, and one for classification data, which is described in Listing 4.9.

For both dictionaries, the structure and choice of hyperparameters are similar. The main difference is the number of trees used in the forest (*n_estimators*). For the regression data, due to its greater complexity, values of 500 and 1000 were used. Larger values such as 2000

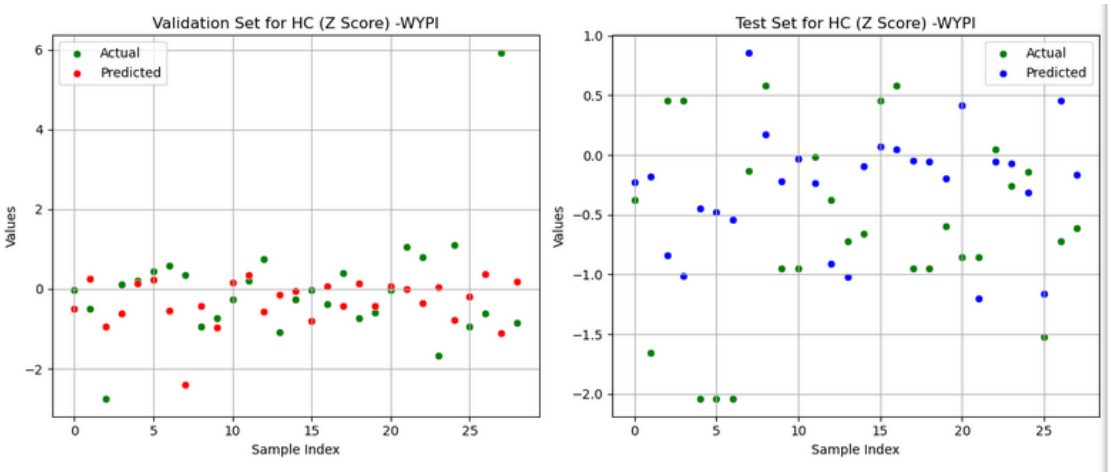

Figure 4.8.: Prediction results for regression data labels of the KNN model.

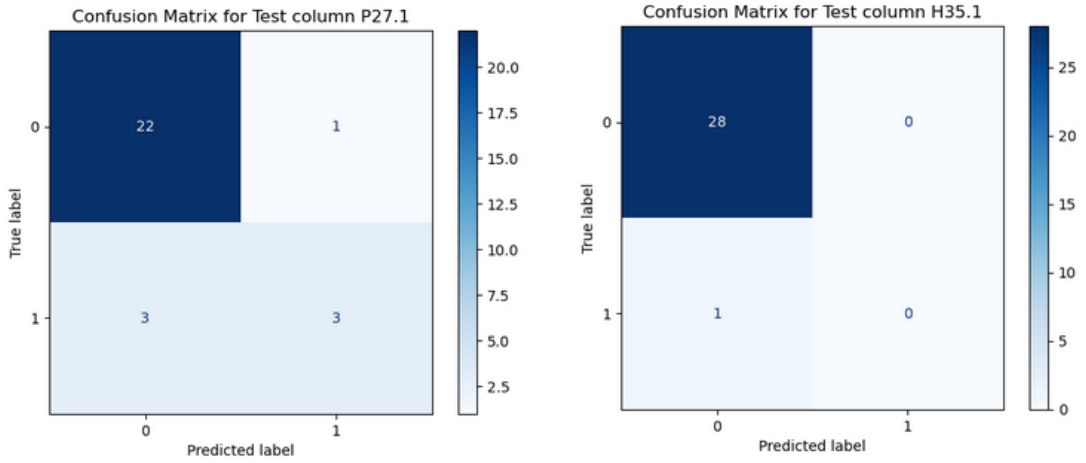

Figure 4.9.: Confusion matrix for the label *P27.1* (left side) and the label *H35.1* (right side).

or 3000 were also tried, but as in Scenario 1, this did not translate into better results. For classification data, values of 300, 500, and 700 were used, although the model mainly used the value of 300, suggesting less complexity of the set.

As in previous models, two tables were created: one responsible for regression data (see Fig. 4.19), where the quality indicator of individual folds is the MAPE parameter, and another responsible for classification data (see Fig. 4.20), where the quality indicator of individual folds is the f1 score parameter.

For regression data, the similarity of results to the KNN model is noticeable. MAPE values for individual folds are comparable to KNN, and the trends for individual labels are

```
param_grid_reg = {
    'n_estimators': [500, 1000],
    'min_samples_split': [2, 3, 4],
    'min_samples_leaf': [1, 2],
    'max_depth': [3, 4, 5]
}
```

Listing 4.8.: Representation of hyperparameters tuned during random forest model training for regression data.

```
param_grid_cla = {
    'n_estimators': [300, 500, 700],
    'min_samples_split': [2, 3, 4],
    'min_samples_leaf': [1, 2],
    'max_depth': [3, 4, 5]
}
```

Listing 4.9.: Representation of hyperparameters tuned during random forest model training for classification data.

Label name	MAPE fold 1	MAPE fold 2	MAPE fold 3	MAPE fold 4	MAPE fold 5	MAPE mean
HC - WYPI	5.48%	4.62%	4.48%	4.86%	6.54%	5.60%
HC (pc) -WYPI	63.01%	40.53%	54.81%	55.17%	62.00%	55.10%
HC (Z Score) -WYPI	589.68%	220.33%	150.05%	103.77%	162.19%	245.60%
dni hospitalizacji	49.05%	77.41%	55.99%	63.87%	84.26%	66.99%

Table 4.19.: Results for each fold and MAPE averages for random forest model for regression data.

identical. Again, the label *HC - WYPI* performs best and stands out from the rest of the regression data. For the other labels, although the trends are similar to KNN, the differences between the best and worst predicted folds are larger, suggesting that the random forest model, although it achieves smaller values for the best folds, is more sensitive to a variety of data.

For classification data, the situation with results for individual folds is similar to the KNN model. The difference lies mainly in the results for the label *P27.1*, where the random forest model has worse overall results. In addition, only at two folds is it able to obtain a value other than 0.

Again, in order to illustrate the best-predicted folds, two tables were created, the first for

Label name	F1 Score fold 1	F1 Score fold 2	F1 Score fold 3	F1 Score fold 4	F1 Score fold 5	F1 Score mean
P27.1	0	0.25	0	0	0.44	0.138
H35.1	0	0	0	0	0	0

Table 4.20.: Results for each fold and F1 score averages random forest model classification data.

regression data (see Table 4.21) containing MAE, MAPE, and the hyperparameters used, and the second for classification data (see Table 4.22) containing the f1 score parameter and hyperparameters.

Label Name	MAE	MAPE (%)	Max Depth	Min Samples Leaf	Min Samples Split	n Estimators
HC - WYPI	1.5448	4.48%	3	1	4	500
HC (pc) -WYPI	16.8663	40.53%	4	1	2	500
HC (Z Score) -WYPI	0.7031	103.77%	4	2	2	1000
dni hospitalizacji	20.4373	49.05%	3	2	2	500

Table 4.21.: Prediction results of individual labels for the random forest model along with selected hyperparameters for regression data.

Label Name	F1 score	Max Depth	Min Samples Leaf	Min Samples Split	n Estimators
P27.1	0.44	5	1	4	300
H35.1	0	3	1	2	300

Table 4.22.: Prediction results of individual labels for the random forest model along with selected hyperparameters for classification data.

The best results for regressive data suggest a slight improvement over the KNN model. The variety of hyperparameters selected is noticeable. Even though the *HC (Z Score) - WYPI* label achieves a value above the MAPE 100%, it still performs better by several percentage points compared with the KNN model, which instead has a better average value from this label. For this reason, the label *HC (Z Score) -WYPI* was again checked in more detail, as shown in Fig. 4.10. The graph for the test set (see the right side of the Fig. 4.10) shows that the model performs well with values close to the mean. For a set with a small number of outliers, this can translate into better random forest results. However, for a more heterogeneous set, these results would be worse.

For the classification data (see table 4.22), the results for both labels show similarity to the KNN model. This is also confirmed by the confusion matrix for both cases (see figure 4.11). The random forest model for the label *P27.1* well predicted two samples out of four for the value 1, with a 50% success rate. There is a noticeable problem with the *H35.1* feature, where a value of 0 was predicted for all samples.

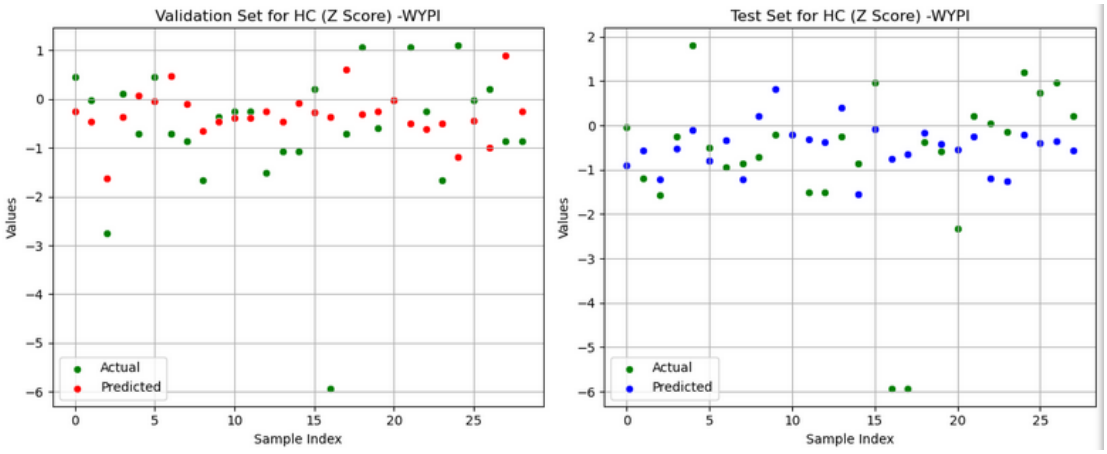

Figure 4.10.: Prediction results for regression data labels of the random forest model.

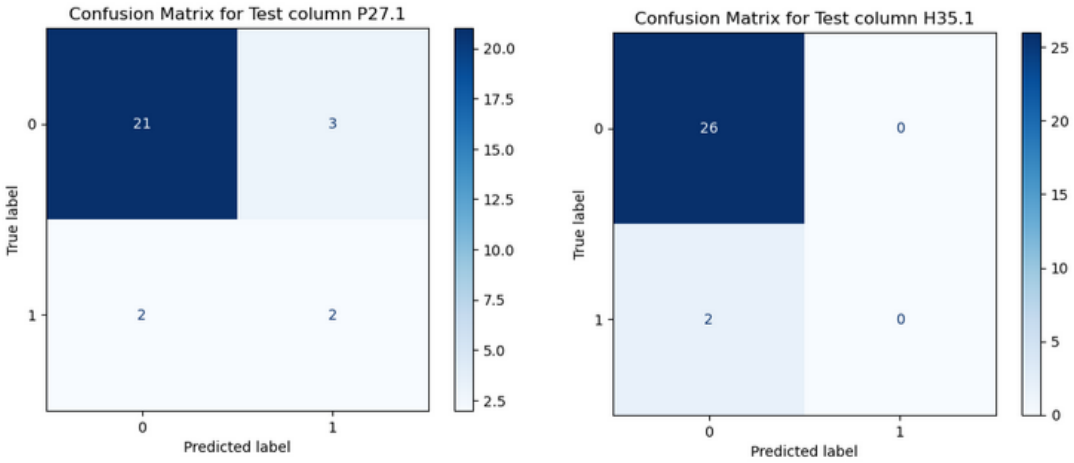

Figure 4.11.: Confusion matrix for the label *P27.1* (left side) and the label *H35.1* (right side).

Gradient Boosting The next model used for training was the gradient boosting. As with Scenario 1, it was decided to use a similar dictionary as with the random forest model. Due to the startlingly worse results, it was decided to remove the hyperparameters *min_samples_split* and *min_samples_leaf* replacing them with *learning_rate* and *subsample*. Compared to the random forest model, by the fact of adding the hyperparameter *learning_rate*, the value of the hyperparameter *n_estimators* was also increased, finally stopping at a value of 4000 for regression data as well as 2000 for classification data. In the final version, the dictionaries containing the hyperparameters were placed in Listing 4.10 for regression data and in Listing 4.11 for classification data.

```
param_grid_reg = {
    'n_estimators': [1000, 2000, 4000],
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [3, 4, 5, 6],
    'subsample': [0.8, 0.9]
}
```

Listing 4.10.: Representation of hyperparameters tuned during gradient boosting model training for regression data.

```
param_grid_cla = {
    'n_estimators': [500, 1000, 2000],
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [3, 4, 5],
    'subsample': [0.7, 0.8]
}
```

Listing 4.11.: Representation of hyperparameters tuned during gradient boosting model training for classification data.

For both dictionaries with hyperparameters, the number of possible combinations was about 50. This was to allow the model to be more flexible with respect to individual data. The main difference between the two dictionaries lies in the hyperparameter *n_estimators*, where the range for the regression data is higher, ranging from 1000 to 4000, while for the classification data it ranges from 500 to 2000. To represent the results of the training, a table 4.23 was created for the regression data containing the MAPE results for each fold, and a table 4.24 was created for the classification data containing the f1 score results.

The Table 4.23 for regression data repeats a pattern that is already well known, whether from the KNN or random forest model. Again, one can see results that are comparable between the mentioned models, which may suggest that the models cannot improve their results, for example, due to data heterogeneity or lack of sufficient representation of outliers.

Similar problems to previous models were observed with the classification data, as described in 4.24. The label *P27.1* looks better than in the other models, having only one fold predicted to be 0. Unfortunately, for the feature *H35.1* the same problem of predicting any value of 1 well still occurs.

In the case of Table 4.25, which contains the best MAE and MAPE values from each label, it is noticeable that there is a lot of diversity in the hyperparameters between successive rows.

Label name	MAPE fold 1	MAPE fold 2	MAPE fold 3	MAPE fold 4	MAPE fold 5	MAPE mean
HC - WYPI	5.91%	4.60%	5.01%	4.68%	6.65%	5.37%
HC (pc) -WYPI	68.11%	43.55%	53.73%	56.52%	68.06%	57.59%
HC (Z Score) -WYPI	643.24%	234.91%	139.53%	106.12%	181.18%	260.20%
dni hospitalizacji	55.96%	73.15%	59.62%	65.88%	80.46%	67.81%

Table 4.23.: Results for each fold and MAPE averages for gradient boosting model for regression data.

Label name	F1 Score fold 1	F1 Score fold 2	F1 Score fold 3	F1 Score fold 4	F1 Score fold 5	F1 Score mean
P27.1	0.20	0.36	0.22	0	0.46	0.248
H35.1	0	0	0	0	0	0

Table 4.24.: Results for each fold and f1 score averages gradient boosting model classification data.

This is due to the diversity of values in the sets, where in the case of a label like *HC (Z Score) -WYPI* there is a very low range as can be seen from the low MAE. On the other hand, there is a label such as *HC (pc) -WYPI*, where the MAE reaches a value of about 18 and the MAPE is more than 2 times lower. Comparing in detail the label *HC (Z Score) -WYPI* to the previously discussed models, we can see that the MAE and MAPE values are comparable. However, when we look at the graphs (see Fig. 4.12) especially the one on the right regarding the test set, some difference is apparent. Compared to the random forest model, the gradient boosting model tries to adapt to the data more though predicting values in a wider range than -1 to 1.

Label name	MAE	MAPE (%)	learning rate	max depth	n estimators	subsample
HC - WYPI	1.5382	4.60%	0.01	3	4000	0.9
HC (pc) -WYPI	18.1224	43.55%	0.05	3	2000	0.9
HC (Z Score) -WYPI	0.6378	106.12%	0.05	4	4000	0.8
dni hospitalizacji	21.7606	59.62%	0.01	3	1000	0.9

Table 4.25.: Prediction results of individual labels for the gradient boosting model along with selected hyperparameters for regression data.

Label name	F1 score	learning rate	max depth	n estimators	subsample
P27.1	0.46	0.01	6	5000	0.8
H35.1	0	0.001	5	5000	0.9

Table 4.26.: Detailed results for fold 5 of the gradient boosting model along with hyperparameters for classification data.

The different hyperparameters are noticeable in the case of the 4.26 table, which contains

detailed results for the classification data from Fold 5. It is observed that despite using the same number of trees (5000), the *learning rate* is different. After comparing the confusion matrix for the two labels (see Fig. 4.13), it can be seen that there is a big difference between them. In the case of the data from label *P27.1*, the model predicted well three out of four samples with a value of 1, which means a 75% success rate. This does not translate into the best f1 score as the KNN model performed better in this regard. Given the data from the *H35.1* label, the model tried twice to predict the value of 1, but did so incorrectly.

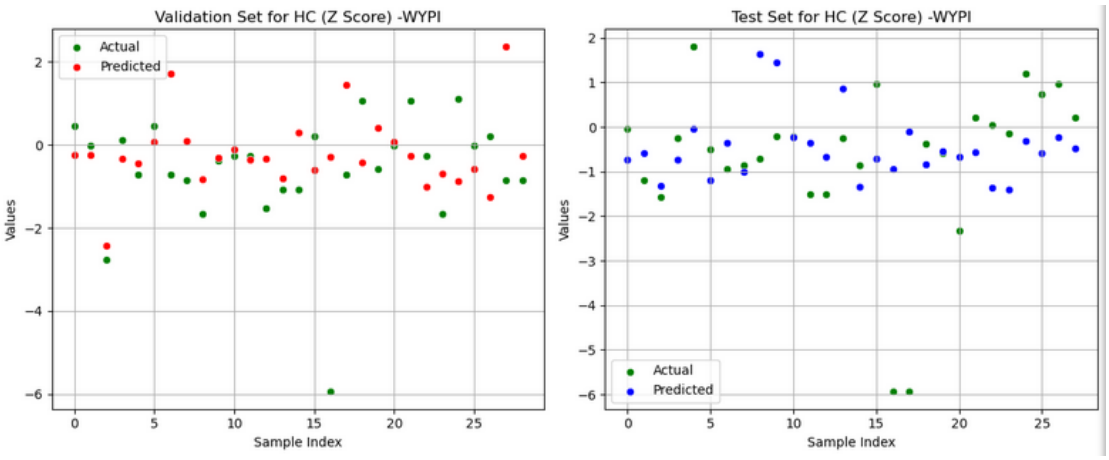

Figure 4.12.: Prediction results for regression data labels of the gradient boosting model.

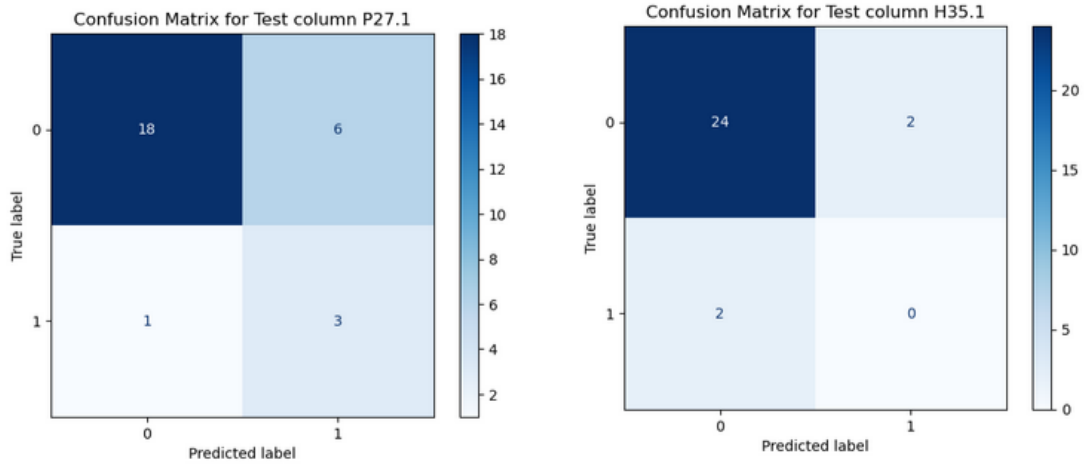

Figure 4.13.: Confusion matrix for the label *P27.1* (left side) and the label *H35.1* (right side).

MLP The last model used for training is the multi-layer perceptron (MLP). Due to the fact that this model is categorized as deep learning, the hyperparameters used for training have changed and are as follows:

- *activation* — neuron activation function.
- *alpha* — L2 regularization value.
- *hidden_layer_sizes* — structure of hidden layers in neural networks.
- *learning_rate* — is responsible for the rate of learning.
- *solver* — optimizer responsible for updating weights during training .

The hyperparameters used to tune the model are illustrated in Listing 4.12 for regression data and Listing 4.13 for classification data. The main difference is the hyperparameter *hidden_layer_sizes*, which in the case of classification data always has one hidden layer differing in the number of neurons.

```
param_grid_reg = {
    'hidden_layer_sizes': [(32,), (64,), (64, 32)],
    'activation': ['relu', 'tanh'],
    'solver': ['adam', 'lbfgs'],
    'alpha': [0.0001, 0.001],
    'learning_rate': ['constant', 'invscaling']
}
```

Listing 4.12.: Representation of hyperparameters tuned during MLP model training for regression data.

In Table 4.27, responsible for presenting the results for each fold of the regression data, the similarity to previous models is observable. MLP performs best with the label *HC - WYPI* but compared to previous models, it is inferior. Again, problems with the other features are noticeable. The biggest problem is with the label *HC (Z Score) -WYPI*, for which the model is not stable and cannot achieve similar results on individual folds. For the other two labels *HC (pc) -WYPI* and *dni hospitalizacji*, the model is stable but achieves poor values around 60 and 70 percent MAPE on average.

As presented in table 4.28 with classification data, the model performs comparably to its predecessors in terms of score. On average, it performs worse than either the KNN or gradient boosting model in terms of label *P27.1*.

```
param_grid_cla = {
    'hidden_layer_sizes': [(64,), (128,), (32,)],
    'activation': ['relu', 'tanh'],
    'solver': ['adam', 'lbfgs'],
    'alpha': [0.0001, 0.001, 0.01, 0.1],
    'learning_rate': ['constant', 'invscaling']
}
```

Listing 4.13.: Representation of hyperparameters tuned during MLP model training for classification data.

Label name	MAPE fold 1	MAPE fold 2	MAPE fold 3	MAPE fold 4	MAPE fold 5	MAPE mean
HC - WYPI	9.20%	6.06%	4.75%	8.89%	7.99%	7.38%
HC (pc) -WYPI	62.04%	51.17%	53.95%	53.17%	63.37%	56.74%
HC (Z Score) -WYPI	645.16%	212.44%	153.17%	113.77%	152.25%	255.36%
dni hospitalizacji	59.15%	78.01%	69.00%	73.53%	85.93%	73.12%

Table 4.27.: Results for each fold and MAPE averages for MLP model for regression data.

Label name	F1 Score fold 1	F1 Score fold 2	F1 Score fold 3	F1 Score fold 4	F1 Score fold 5	F1 Score mean
P27.1	0	0.31	0	0.43	0	0.148
H35.1	0	0	0	0	0	0

Table 4.28.: Results for each fold and F1 score averages MLP model classification data.

In a more detailed Table 4.29 containing the best-predicted fold along with the parameters, one can see the variety of hyperparameters. This means that the model tried to fit the data for different features. The results for each label are comparable to the other models, so in order to get a better insight into the predictions, graphs were created with the results for the label *HC (Z Score) -WYPI* as shown in Fig. 4.14.

Again, the problem with outliers is notable. In general, the prediction pattern itself resembles that observed for the random forest model. That is, the predicted values oscillate close to the mean value.

Label name	MAE	MAPE (%)	activation	alpha	hidden_layer_sizes	learning_rate	solver
HC - WYPI	1.5823	4.75%	tanh	0.001	(32,)	constant	lbfgs
HC (pc) -WYPI	21.2924	51.17%	relu	0.0001	(64,)	constant	adam
HC (Z Score) -WYPI	0.6838	113.77%	relu	0.001	(32,)	invscaling	adam
dni hospitalizacji	32.8203	59.15%	tanh	0.001	(64, 32)	constant	lbfgs

Table 4.29.: Prediction results of individual labels for the MLP model along with selected hyperparameters for regression data.

Label name	F1 score	activation	alpha	hidden_layer_sizes	learning_rate	solver
P27.1	0.43	relu	0.0001	(128,)	constant	adam
H35.1	0	relu	0.0001	(64,)	constant	adam

Table 4.30.: Prediction results of individual labels for the MLP model along with selected hyperparameters for classification data.

Table 4.30 refers to the best predicted folds with parameters for classification data. There is a noticeable similarity in the fitting of the hyperparameters, where the only difference is in the number of neurons for which the hyperparameter *hidden_layer_sizes* is responsible. As the results are comparable to previous models, a confusion matrix of test sets was created, which is shown in Fig. 4.15.

In the case of the *P27.1* label, the model, out of ten samples with a value of 1, is able to predict three well, giving it an efficiency of 30%. In terms of f1 score, it performs similarly to the gradient boosting model. In the case of the *H35.1* label, the model predicts a value of 0 for all samples.

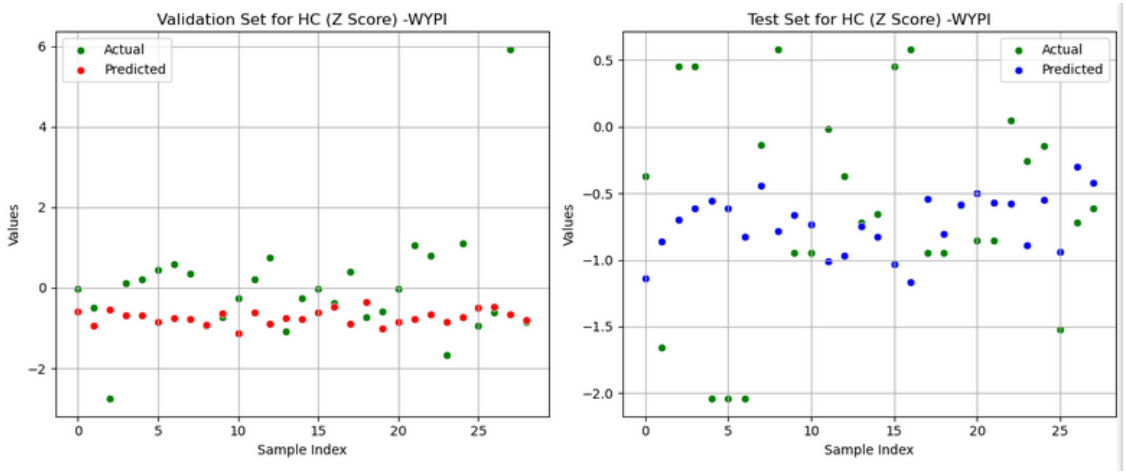

Figure 4.14.: Prediction results for regression data labels of the MLP model.

Isolation Forest Due to unsatisfactory results for most classification models, it was decided to train an anomaly detection model, namely isolation forests. The hyperparameters for this model are illustrated in Listing 4.14.

In addition to the well-known hyperparameters such as *n_estimators* or *max_samples*, a new parameter *contamination* appears. It has been set as a fixed value, as it affects the number of predicted values of 1. If this parameter has a low value (for example, 0.01), it may

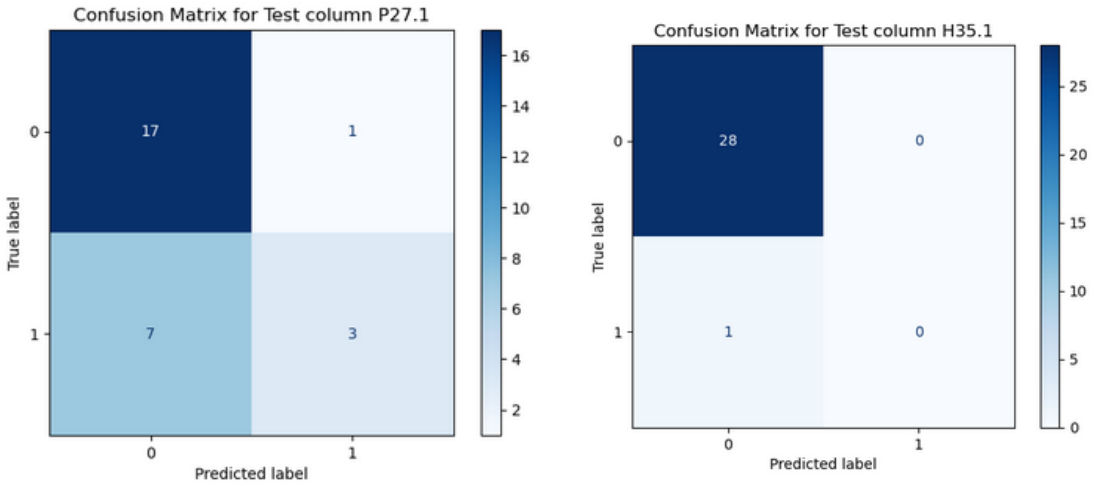

Figure 4.15.: Confusion matrix for the label *P27.1* (left side) and the label *H35.1* (right side).

```
param_grid = {
    'n_estimators': [50, 100, 150, 200],
    'max_samples': [0.8, 0.9],
    'contamination': [0.3],
}
```

Listing 4.14.: Representation of hyperparameters tuned during isolation forest model training for classification data.

result in not detecting any values of 1, which in this case are anomalies. The results for each fold are presented in Table 4.31.

Label name	F1 Score fold 1	F1 Score fold 2	F1 Score fold 3	F1 Score fold 4	F1 Score fold 5	F1 Score mean
P27.1	0.58	0.29	0	0.29	0.25	0.282
H35.1	0.12	0	0	0.22	0.14	0.096

Table 4.31.: Results for each fold and f1 score averages isolation forest model classification data.

It is noticeable that there is a difference in the average score between the two labels by about 0.2. In the case of the *P27.1* label, the average score is better, which suggests better model quality. In order to illustrate the hyperparameters for a single fold, Tables 4.32 were created.

Label name	F1 score	n_estimators	max_samples	contamination
P27.1	0.58	50	0.8	0.3
H35.1	0.14	50	0.8	0.3

Table 4.32.: Prediction results of individual labels for the isolation forest model along with selected hyperparameters for classification data.

Since the next table does not provide an answer, a confusion matrix was created for both cases, which illustrates Fig. 4.16. It was noted that the model for both labels used the same hyperparameters. The isolation forest model predicts the values of 1 very well, especially for the label *P27.1*. Out of nine samples with a value of 1, the model is able to predict seven well, which is about 77% effective. For the *H35.1* label, the model is able to predict the value of 1 with an efficiency of 50%. This is a departure from the other models because the isolation forest was the only one able to predict the value of 1 well.

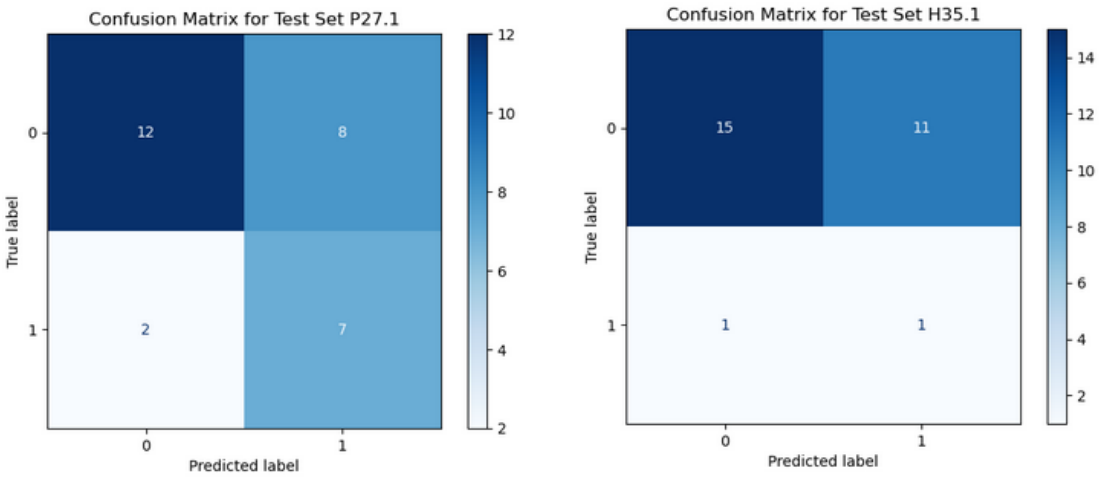

Figure 4.16.: Confusion matrix for the label *P27.1* (left side) and the label *H35.1* (right side).

4.2.1. Results Summary

In order to compare and summarize the results for all trained models, a table 4.33 was created containing the average MAPE value for the regression data. For the classification data, a 4.34 table was created containing the average accuracy value.

In the case of regression data, the high performance of all models against the label *HC - WYPI* is observable. The linear regression model deviates at almost all labels (except

the one mentioned above) achieving worse values than other models. KNN, random forest, gradient boosting, and MLP models achieve similar performance with all labels. The feature with which the models perform the worst is *HC (Z Score) -WYPI*. This is due to the small range of data (most data are in the range from -1.5 to 1.5) which results in high precision of the predictions. In addition, outlier data, if they occur, reach values around 5 or 6, which causes problems in their prediction, and with it, a high MAPE. Although KNN, random forest, gradient boosting, and MLP models achieve comparable values, some differences are notable when looking in detail. The gradient boosting model focuses on fitting the data, which sometimes pays off with a worse MAPE value. The random forest and MLP models are the least flexible, which translates into better results with easier samples (similar values between each other), but work less well when dealing with diverse data. The KNN model can be classified somewhere between the previously mentioned.

Label Name	**Linear Regression (%)**	**KNN (%)**	**Random Forest (%)**	**Gradient Boosting (%)**	**MLP (%)**
HC - WYPI	6.05%	5.55%	5.60%	5.37%	7.38%
HC (pc) -WYPI	113.90%	59.78%	55.10%	57.59%	56.74%
HC (Z Score) -WYPI	330.60%	243.94%	245.60%	260.20%	255.36%
dni hospitalizacji	193.29%	67.74%	66.99%	67.81%	73.12%

Table 4.33.: Comparison of mean MAPE scores for linear regression, KNN, random forest, gradient boosting, MLP by trained labels.

Label Name	**Logistic Regression**	**KNN**	**Random Forest**	**Gradient Boosting**	**MLP**	**Isolation Forest**
P27.1	0.172	0.192	0.138	0.248	0.148	0.282
H35.1	0	0	0	0	0	0.096

Table 4.34.: Comparison of mean f1 scores for logistic regression, KNN, random forest, gradient boosting, MLP, isolation forest by trained labels .

For classification data, all models achieve comparable accuracy values. For the label *H35.1*, only the isolation forest model was able to achieve a value other than 0 for f1 score. The problem with this feature is due to the fact that there are very few samples with a value of 1 in the set by which each model has a problem even with an incorrect prediction of a value of 1. On the other hand, the models perform better with the label *P27.1*, where samples with a value of 1 are more frequent. In this case, we can observe at least one well-predicted sample with a value of 1 in each model. Of all the models, the gradient boosting and isolation forest models perform best with the *P27.1* label, as they were the only ones able to exceed the average value of 0.2 for f1 score. In addition, the KNN and isolation forest models achieved the best value for a single fold around 0.6. Although the results for both labels look poor, for the *P27.1* feature, all models attempted to predict at least one value of 1 well. For the *H35.1* label, only the isolation forest model was able to achieve an f1 score value other than

0. Based on this information, it can be concluded that obtaining more samples with a value of 1 (especially for the *H35.1* label) should translate into better results for each model.

5. Conclusion

5.1. Summary

In order to summarize the results for both scenarios, Tables 5.1 for Scenario 1 and Tables 5.2 for Scenario 2 were created.

First, we will discuss the classification data that occurs only in Scenario 2. The models do not cope with the label *H35.1* when predicting values of 1, where samples with these values are rare. The test sets contained only 1 to 3 samples with a value of 1, and the rest had a value of 0. This caused the models to assign a value of 0 to all samples, rarely attempting to predict a value of 1, even incorrectly. Only the isolated forest was able to predict the value of 1 for this label well. The situation changes for the *P27.1* label, where there are more samples with a value of 1. The models still have problems predicting the value of 1, but the results are better here, as they were able to predict the value of 1 at least once for each model. Despite a score greater than 0, it was still the case that each model had at least one fold that reached a value of 0 for the f1 score measure. In addition, the disparity in results between folds may suggest a lack of sufficient representation of samples with values of 1 in the training sets. Of all the models, the best performers were gradient boosting and isolation forest, which were the only ones to exceed 0.2 for the f1 score measure. To summarize this data type, the models completely underperforms for the label *H35.1* and for the feature *P27.1* they perform poorly. Therefore, drawing conclusions and observations from the above-mentioned problems, it would be necessary for further research to obtain a larger number of samples with a value of 1 for both labels. This approach should translate into overall better results for each model.

For regression data, we are able to compare the results between both analyzed scenarios. The linear regression model performs the worst with the data. The best-predicted label is *HC - WYPI* from Scenario 2, which is the only one that achieves less than 10% MAPE for each model. Almost all labels from Scenario 1 have values less than 50% MAPE for the best models such as KNN, random forest, and gradient boosting. For Scenario 2, the remaining labels have very poor results above 50% MAPE for all models except linear regression. The label with the worst results above 200% MAPE is *HC (Z Score) -WYPI*. Comparing Tables 5.1 and 5.2, it can be concluded that Scenario 1 performs better overall than Scenario 2.

The outcomes predicted in Scenario 1 for the left ventricle (the names starting with LV) and right ventricle (starting with RV) have good enough results for KNN, random forest,

Label Name	Linear Regression (%)	KNN (%)	Random Forest (%)	Gradient Boosting (%)	MLP (%)
LV A2C	268.49%	18.94%	16.01%	18.80%	30.62%
LV A3C	53.35%	13.15%	12.71%	14.27%	26.63%
LV A4C	118.73%	13.62%	14.68%	15.91%	23.18%
LV GLS	67.46%	11.66%	11.54%	12.93%	18.73%
LASr ED	155.15%	24.78%	24.35%	28.99%	49.76%
LAScd ED	63.78%	35.43%	33.97%	37.71%	82.29%
LASct ED	62.22%	48.06%	47.33%	54.36%	87.57%
LASr AC	39.06%	23.70%	21.99%	27.92%	51.75%
LAScd AC	52.17%	36.42%	33.89%	36.67%	82.11%
LASct AC	57.58%	36.71%	39.49%	45.43%	68.12%
RVFWSL	89.77%	26.60%	23.96%	26.70%	47.08%
RV4CSL	57.03%	23.80%	21.86%	23.61%	44.63%

Table 5.1.: Average MAPE scores for all models for Scenario 1.

and gradient boosting models that they could be used in a commercial setting. The hardest to be predicted are some of the outliers, which are well illustrated in Fig. 4.2 or Fig. 4.4.

In the case of Scenario 2, despite a good result for one label *HC - WYPI* the others perform much worse. The models have the largest problem with the label *HC (Z Score) -WYPI*, as most of the values are in the range from -1.5 to 1.5, and some outliers can achieve values about -5 or 6, which translates into very bad MAPE results.

Label Name	Linear/Logistic Regression	KNN	Random Forest	Gradient Boosting	MLP	Isolation Forest
HC - WYPI	6.05%	5.55%	5.60%	5.37%	7.38%	-
HC (pc) -WYPI	113.90%	59.78%	55.10%	57.59%	56.74%	-
HC (Z Score) -WYPI	330.60%	243.94%	245.60%	260.20%	255.36%	-
dni hospitalizacji	193.29%	67.74%	66.99%	67.81%	73.12%	-
P27.1	0.172	0.192	0.138	0.248	0.148	0.282
H35.1	0	0	0	0	0	0.096

Table 5.2.: Average MAPE (for regression data) and f1 (for classification data) scores for all models for Scenario 2.

5.2. Opportunities for Further Development

Although the scores for both scenarios do not provide concrete results that can be used in medical forecasting at this time, several conclusions can be drawn:

- Dataset — A larger dataset would be useful in a further study. It would be especially helpful if the dataset could be supplemented with data on sick children and their treatment, as outlier data is the most lacking. In addition, the data would be best collected in a structured manner, meaning that surveys should be done at equal intervals between each other, for example, every 14 days. Each premature baby should have a

specific set of tests done that are needed for Scenarios 1 and 2, so that when the data is cleaned, there would be no need to remove as many samples.

- Model selection — KNN, random forest and gradient boosting models, due to the best results for both scenarios, should be used first for further research. Because of the best flexibility with respect to the data, the gradient boosting model should be the primary choice. If a larger dataset could be collected then the MLP model or other deep models should also be considered as they are sometimes able to predict outliers well.

- Choice of scenario — Scenario 1, due to its better overall performance, should be considered as the first area of further research. In addition, good model performance for Scenario 1 could be more useful than for Scenario 2 because it would predict the results of the LV strain study. Consequently, it would be possible to use the model to tentatively determine which preterm infants should be sent for more detailed testing.

5.3. Application of Research

For the moment, none of the models from both Scenario 1 and 2 should be used in the hospital. However, for further research, it would be suggested to focus on Scenario 1, dividing it into two stages:

- first stage — would focus on training the model for labels for the left right ventricles, as these, if the model's performance for outliers improved, would allow the model to be implemented as a support for doctors to make initial predictions about a child's condition.

- second stage — implies the addition of left atrial labels to training, which Scenario 1 models do less well with. This would allow further research to be conducted on the entire Scenario 1 at the same time as having a simplified version that could already be tested.

In the case of Scenario 2, it would be useful to acquire more data, preferably structured, especially in terms of time spent in the hospital by the patient. Only after applying these assumptions would it be possible to predict potential illnesses or the length of a child's hospitalization, which is what Scenario 2 was designed to do.

Bibliography

[1] *Deformation imaging and rotational mechanics in neonates: a guide to image acquisition, measurement, interpretation, and reference values.* URL: https://www.nature.com/articles/s41390-018-0080-2.

[2] *How to do it? Speckle-tracking echocardiography.* URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC3860973/.

[3] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition.* O'Reilly Media, 2022.

[4] *Artificial Intelligence for Automatic Measurement of Left Ventricular Strain in Echocardiography.* URL: https://www.jacc.org/doi/full/10.1016/j.jcmg.2021.04.018.

[5] *Machine Learning Models for Predicting Neonatal Mortality: A Systematic Review.* URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC8887024/.

[6] *Enhancing the accuracy in predicting infant mortality using logistic regression in comparison with decision trees with support vector machine.* URL: https://pubs.aip.org/aip/acp/article-abstract/3193/1/020065/3319881/Enhancing-the-accuracy-in-predicting-infant?redirectedFrom=fulltext.

[7] *Apical 3-chamber view.* URL: https://theory.labster.com/apical_3_chamber_view/.

[8] *Diabetes.* URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC6424769/.

[9] *Gestational Age.* URL: https://www.ncbi.nlm.nih.gov/medgen/?term=gestational+age.

[10] *Birth Weight.* URL: https://www.ncbi.nlm.nih.gov/books/NBK304161/.

[11] *Apgar score.* URL: https://medical-dictionary.thefreedictionary.com/Apgar+score.

[12] *P00.0.* URL: https://icd.who.int/browse10/2019/en#/P00.0.

[13] *Hematocrit.* URL: https://www.thefreedictionary.com/Hematocrit.

[14] *Hemoglobin.* URL: https://www.thefreedictionary.com/hemoglobin.

[15] *rbc.* URL: https://medical-dictionary.thefreedictionary.com/Red+Blood+Cell+Count.

[16] *C-Reactive Protein*. URL: https://medical-dictionary.thefreedictionary.com/C-Reactive+Protein.

[17] *interleukin-6*. URL: https://encyclopedia2.thefreedictionary.com/Interleukin-6.

[18] *LV, RV*. URL: https://www.asecho.org/wp-content/uploads/2018/08/WFTF-Chamber-Quantification-Summary-Doc-Final-July-18.pdf.

[19] *LAS*. URL: https://www.acc.org/latest-in-cardiology/ten-points-to-remember/2018/08/10/11/01/assessment-of-left-ventricular-global-longitudinal-strain.

[20] *Head circumference*. URL: https://www.measurement-toolkit.org/anthropometry/objective-methods/simple-measures-head.

[21] *P27.1*. URL: https://icd.who.int/browse10/2019/en#/P27.1.

[22] *H35.1*. URL: https://icd.who.int/browse10/2019/en#/H35.1.

[23] *pytesseract*. URL: https://pypi.org/project/pytesseract/.

[24] *seaborn*. URL: https://seaborn.pydata.org/.

[25] *sklearn*. URL: https://scikit-learn.org/1.5/index.html.