Department of Mathematics
TUM School of Computation, Information and Technology
Technical University of Munich

Explainable Comparative Analysis of Modern Survival Models and Patient Similarity Networks:

A Case Study for Cancer Patients

Niklas Kiermeyer

Thesis for the attainment of the academic degree

Master of Science

at the TUM School of Computation, Information and Technology of the Technical University of Munich

Supervisors:
Prof. Fabian Theis and Dr. Bastian Rieck

Advisors:
Prof. Jens Kleesiek and Dr. Julius Keyl

Submitted:
Munich, 11. August 2023

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

[illegible]

Munich, 11. August 2023 Niklas Kiermeyer

Acknowledgments

I am immensely grateful to Dr. Bastian Rieck for his supervision of my thesis, coupled with his invaluable insights, ideas and constant support. I would like to thank Dr. Julius Keyl for his unwavering assistance in providing medical feedback and suggestions for my models. Additionally, I would like to express my gratitude to Prof. Jens Kleesiek for making this collaboration possible and for his valuable input, which contributed greatly to the success of my thesis. Finally, I would like to thank Prof. Fabian Theis for being my thesis supervisor.

Abstract

Hospitals play a crucial role in providing access to valuable medical data, making it essential to leverage this information in collaboration with physicians. Lung cancer is the focus of this thesis, in which special techniques are used to extract knowledge from a high-dimensional dataset provided by the University Hospital Essen. Considering that it is the deadliest cancer worldwide, this analysis is of great importance. Therefore, we explore different approaches that can help doctors gain additional insights into the intrinsics of this disease.

By modeling the risk of lung cancer patients experiencing an event such as treatment failure or death, it is possible to understand the mechanics of this disease and identify potential treatment options and their effectiveness. Thus, we compare three popular survival methods based on this real-world dataset, namely: Random Survival Forests, DeepSurv and XGBSE. We find all of these methods to exhibit a similar performance, which is surprising because they rely on different architectures and methodologies. In addition, we show that the use of Random Survival Forests in combination with SHAP values, which can be used to generate interpretable representations, provide clinicians with valuable insights into feature interactions and their importance, but also allow for model validation.

In addition to the survival analysis, which serves as a baseline for comparing different models, another important aspect is modeling when a patient is likely to fail to respond to treatment. If physicians know when to expect it, they can adjust the medication or change the treatment altogether to prolong the patient's survival. Our goal is to take advantage of patient-patient similarity graphs in combination with Graph Neural Networks to capture information from patients who are similar to each other to improve treatment failure time prediction. Given the absence of inherent connections between patients, we present a novel method to generate patient-patient similarity graphs using Unbalanced Optimal Transport on their laboratory values. By applying various Graph Neural Networks on these graphs, we highlight their effectiveness in predicting if treatment failure occurs within half a year or later. However, we also find that the predictive performance is still limited compared to a Multilayer Perceptron. Nonetheless, our approach warrants further investigation, particularly in subsets with higher overall similarity, given the diverse nature of lung cancers, which could lead to enhanced predictive performance.

In conclusion, this thesis emphasizes the importance of leveraging hospital data, underscores the significance of survival modeling, and exemplifies the necessity of exploring new approaches such as using Unbalanced Optimal Transport to construct patient-patient similarity graphs. Future research should seek to further optimize survival methods with an explainability integration and patient-patient similarity graphs, as both can directly contribute to the medical decision-making process and, more generally, advance medical research.

Contents

1 Introduction

Looking at Artificial Intelligence (AI) and related companies over the past few years, it seems like there is never-ending, exponentially growing progress in many areas. This is facilitated by hardware manufactures like NVIDIA, who produce the essentials to train AI models at a large scale. They provide Graphics Processing Units (GPUs), which offer higher efficiency compared to Central Processing Units (CPUs) and enable further scaling when combined collectively. Consequently, this technological advance has enabled the training on progressively larger datasets, significantly contributing to recent progress in model performance and success such as for Large Language Models (LLMs) like GPT-4, image generation models like Midjourney, and generative AI in general. This development naturally leads to the question of how these models can be applied to hospital data to advance medical research, which has already demonstrated its potential with models such as PAIGE and PubMedGPT [1, 2].

Although this progress in models and hardware creates several possibilities it also comes with many questions, such as the cost-benefit trade-off for training a model over multiple months in order to get incremental performance gains over an already existing model. This is also important from an environmental perspective, as running these models on server farms requires energy for cooling and operation. Is scaling alone the right approach? What about the impact of dataset quality, since this is already where bias could be introduced through imbalance. These are all questions that need to be considered for medical models such as survival models, as even incremental enhancements in predictive capabilities could potentially save lives, while imbalanced datasets could compromise the overall generalization performance.

One of the most important issues is the origin of the data used to train these models. Since this can lead to bias, it is important to ensure data quality. However, there are also concerns about data security, data ownership, and data privacy that need to be considered. For example, LLMs like PubMedGPT are trained on a large text corpus, but what if a patient does not want their medical record to be used for training and prefers the model to “unlearn” their information? Another notable case that underscores the importance of data privacy is the imposition of a 1.3 billion dollar fine on Meta for infringing European Union privacy laws by sending data to servers in the United States [3]. Additionally, because of the fast progress of AI, many legal concerns and loopholes have not yet been effectively addressed or legally defined by governments.

In the light of these aspects, a more general question arises: when will the pace of progress in AI decelerate and reach its peak and what areas are still to be explored? Focusing on the latter aspect, there is still a great deal of diversity, encompassing areas such as explainability, fairness, robustness, and the application of pre-existing models to previously unseen data. Numerous unexplored datasets exist within companies, institutes and hospitals that hold significant potential for discovering valuable insights. However, the applicability of generic out-of-the-box algorithms may be limited. Nevertheless, transfer learning—a technique that uses internal model parameters from existing models to train a custom model—may be a viable option, but is not always practical. In general there is not a one size fits all model to tackle all different problems which arise.

Hospitals, in particular, have a wealth of data with immense potential that has yet to be tapped. Their data comes from a variety of modalities, including MRI images, tissue samples such as Hematoxylin and Eosin stains used in histology, laboratory values, patient survey data, and more. Hospital data is of particular interest because they allow researchers to work hand-in-hand with the physicians who provide the data. This collaborative approach allows models to be tailored directly to specific needs and to leverage physician-induced medical knowledge by refactoring models based on medically relevant parameters.

1.1 Motivation

The dataset analyzed in this study was obtained from a collaboration with the University Hospital Essen (UHE), Germany, which specializes in oncology, transplantation, cardiovascular diseases, infectious diseases, immunology, neuroscience, and behavioral sciences [4]. Generally, clinical data in the UHE is stored in FHIR format—a standardized framework for exchanging electronic healthcare data—and can be easily retrieved through specific queries. In this particular dataset, we examined a cohort of 4320 individuals diagnosed with lung cancer. In 2020, lung cancer was the largest contributor to cancer deaths with a total of 1.8 million deceased worldwide [5]. Diagnosis includes imaging tests, sputum cytology and tissue samples, but detection is often delayed because symptoms tend to develop mostly when the tumor has already reached an advanced stage of growth [6, 7]. After initial diagnosis, the 5-year survival rate is approximately 23 percent—though this greatly depends on lung cancer type (NSCLC vs SCLC) and stage—and the chances of cure are highest if the cancer is detected at an early stage [8]. Treatment includes surgery, radiation and chemotherapy, however two-thirds of patients receive only palliative chemotherapy or radiation therapy, because metastases or compromised pulmonary function, render surgical intervention impossible [7]. However, since immunotherapy was approved as a first-line therapy for many patients with NSCLC several years ago, their chances of survival have improved significantly.

There are many risk factors for lung cancer, including environmental and occupational exposures, lifestyle factors, and chronic lung disease, but the most prevalent factor is still tobacco use [9]. The latter is also a feature of the dataset provided by UHE. In addition, we have unique information for each patient on more than 90 different laboratory values, molecular markers, disease classifications, surgical procedures, histological information, treatments, tumor stages, body composition information, and more. The dataset in this work is entirely tabular. With this wide range of features, we aim to take advantage of the diversity and depth of this dataset to provide a holistic study. Hence, it is essential to provide a detailed overview of the methods employed to analyse this dataset.

Furthermore, this is a real-world dataset, which in comparison to other clinical datasets, was not explicitly created for a clinical trial, thus encompassing a higher level of complexity and granularity. This unique nature has several advantages, such as a larger sample size which contributes to model performance and robustness. It also captures real-world complexity, allowing for a more comprehensive representation of diverse patient populations. In addition, real-world datasets are considered to be more representative, allowing for more accurate and generalizable conclusions. Their analysis allows physicians to explore multiple promising areas and identify potential domains for future clinical trials. However, it is important to recognize the limitations of real-world data because—unlike a controlled clinical trial—patient selection is not random. As a result, real-world data may have higher levels of bias and increased risk of confounding. Furthermore, it is more likely to have a higher prevalence of missing data, and overall, the data tends to be less comprehensive.

The methods applied in this study focus primarily on examining different facets of the dataset, including patient-patient similarity graphs and explainable survival analysis. One objective is to provide a detailed and comprehensive overview of recent state-of-the-art survival methods. Our goals are to evaluate their performance, obtain an understanding of feature importance and interactions, and benchmark their predictive power on a real-world dataset rather than clinical trial dataset. In this way, we try to gain an understanding of their robustness as well as their advantages and disadvantages. Adding to that, we aim to explain the decision-making process of such a survival model and provide new medical insights into the interactions between features for lung cancer patients.

Another objective is the creation of patient-patient similarity networks without prior knowledge of their connections and the comparison of their predictive power using Deep Learning (DL) methods. With this approach, we aim to leverage the collective knowledge within communities so that when performing a prediction task for a patient, the model relies primarily on information from the patient's nearest neighbors. In this way, we strive to provide a more individualized and less biased estimate, thus tailoring predictions to the specific circumstances of each patient. We further construct sparse patient-patient similarity networks and assess their performance in comparison to a naïve network generation approach.

These networks allow us to effectively capture relevant similarities between patients and use them to make predictions, while reducing the overall complexity of the model.

In general, we seek to understand disease patterns in lung cancer, uncover potential risk factors, facilitate data-driven decision-making, tailoring treatment, improving patient outcomes, and generally advancing medical research.

1.2 State-of-the-Art

Precision medicine, personalized treatment and the analysis of large scale medical datasets is not new, but few Machine Learning (ML) models are actually used on a daily basis by physicians. In 2021, for example, the first AI-based software named PAIGE for identifying potential cancerous regions on a prostate biopsy image, got market authorization by the United States Food and Drug Administration (FDA) [10]. It is designed not to replace pathologists, but to bolster the confidence in diagnosis [1]. Another example of the potential of carefully curated hospital quality data is the use of blood sample data combined with logistic regression to detect cancer four years before conventional diagnosis [11]. Additionally, this example showcases that exceptional performance does not necessarily require the newest Machine Learning model, underscoring the importance of exploring classical approaches as well.

To the best of our knowledge, there is no FDA-approved model for lung cancer detection based on imaging or laboratory values. Lung cancer research ranges from the design of nanoparticles for pulmonary mRNA delivery and genome editing, to precision medicine for patients, to new ways of detecting lung cancer [12, 13, 14]. Especially, with DLs success in imaging over the past decade, several methods have been proposed. For instance, by using deep learning-based algorithms it is possible to detect lung cancer from chest X-rays or Computed Tomography (CT) scans of the lungs—which are cross-sectional images—to classify suspected lung cancer and lung nodules with very high accuracy [15, 16]. In addition, image denoising models could be used in the future for patients undergoing CT scans. By exposing patients to lower levels of radiation, resulting in noisier images, these models can effectively denoise the images, thereby increasing patient safety while maintaining image quality [17]. This underscores the relevance of extracting and translating findings from such general settings into clinical data.

All of these previously mentioned DL methods work for images, but what about tabular datasets such as the one from the UHE? This type of dataset provides flexibility by accommodating many different types of data, offers transparency through comprehensible representation, and facilitates seamless tracking of variable changes over time. Furthermore, it offers several opportunities for exploration, including survival analysis, identification of similar patients, and predictive modeling of features within the dataset. Especially when predicting survival, physicians often rely on their subjective intuition and interpretation, which naturally limits accuracy and reproducibility [18]. Semi-parametric models such as the Cox proportional hazards model are among the classical statistical approaches, but have certain limitations. For example, they are unable to capture non-linear relationships between features and assume a constant effect of features over time. Other models which can capture non-linear relationships are Random Survival Forests (RSFs) and Neural Networks (NNs). RSFs and Random Forests (RFs) have been successfully employed across domains such as for clinical risk prediction, and length of hospitalization prediction of lung cancer patients [19, 20]. Furthermore, a wide range of NNs have been developed for survival analysis and more generally time-to-event analysis. Prominent examples include models such as DeepSurv, DeepHit, and MTLR, which have demonstrated successful applications [21, 22, 23]. For example, DeepSurv has been effectively used for survival prediction of oral cancer patients [24]. Moreover, in the realm of multimodal data encompassing data such as mRNA, miRNA, whole-slide images and clinical data, several methods exist that leverage information across modalities for survival prediction. One notable example for such a model is MultiSurv [18].

In summary, the use of clinical data is a large, growing and complex area of research that encompasses a number of different models and methodologies and offers many research opportunities. However, it is important to note that there is no single universal approach, as all models have their advantages and disadvantages depending on the task at hand.

2 Background

This Chapter focuses on introducing the mathematical principles required for building the models created in Chapter 3. The aim is for the reader to gain a thorough understanding of state-of-the-art techniques in practice as well as a solid understanding of the mathematical principles underlying them. After a general introduction to DL, its applications, and the format in which data is available, the following discussion presents a simple form of NNs with a detailed mathematical derivation on the training process of one. After laying the foundation for creating arbitrarily complex NNs, we derive Graph Neural Networks (GNNs), which in turn rely on NNs. Next, we introduce Optimal Transport (OT), a powerful tool in combination with GNNs, as shown by Tong et al. [25]. We then proceed with an introduction to survival analysis models and explainability.

2.1 Deep Learning

Although Deep Learning is a specific type of Machine Learning, it covers a wide variety of domains from recommender systems, Natural Language Processing (NLP), computer vision, to reinforcement learning, to name a few. Applications derived from these areas can, for example, recommend the next movie based on your interests, be a question and answer engine, create realistic looking images that don't exist, or learn to play Atari games better than human players do [26, 27, 28, 29].

In general, there are two training settings for DL, the supervised setting, where each data point X_i has K features and already comes with a label y_i, and the unsupervised case where no labels are given. In the classical supervised setting, DL uses a dataset D with a total of N samples $D = \{(X_i, y_i) \mid X_i \in \mathbb{R}^K, y_i \in \mathbb{R}, 1 \leq i \leq N\}$ and randomly splits it into a train, validation and test set. Usually, the dataset is first split into a train and test set with a ratio of 80% and 20%, respectively. In the following, the train set is randomly partitioned into two subsets, one of which is used for training and another, smaller one for validation. Training means that the models internal parameters are refined iteratively for each sample in the train set until a certain criterion, calculated over the validation set, is satisfied. The test set is only used for the final evaluation. This approach is very powerful because it provides an approximation of the models accuracy on new, unseen data, assuming that the train, validation and test set have the same underlying distribution. Thus, it is possible to infer y_{new} from X_{new}. The unsupervised case works in a similar way, except that now there are no labels y_i for the data.

2.1.1 Fundamentals and Techniques

The notation used in this work follows the conventions described in Higham and Higham [30]. The XOR dataset (see Figure 2.1) illustrates the capabilities of Multilayer Perceptrons (MLPs), a class of NNs, since they can handle nonlinear data dependencies, unlike traditional linear classifiers [31, 32]. The MLP approximates a target function $f^*(X) = y$ via a function $f(X, W) = y$. In total, an MLP consists of $l \in (1, \ldots, L)$ layers, each of which contains a different number of nodes $n^{[l]}$, also referred to as neurons.

The first layer is known as the input layer, followed by arbitrarily many hidden layers, and the last layer is the output layer. As input in the example depicted in Figure 2.2, the MLP takes $X_i \in \mathbb{R}^2$, which corresponds to the coordinates x_1 and x_2, and then applies an affine transformation described by a matrix of weights W and a scalar bias term b to the input, followed by a nonlinear activation function σ, and

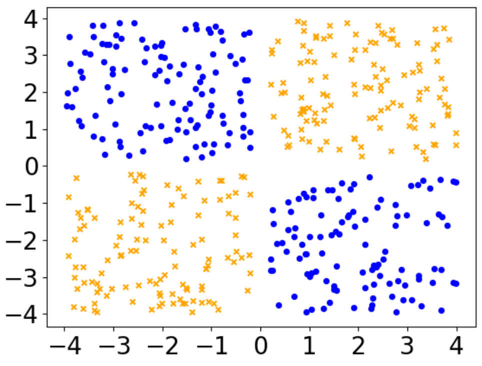

Figure 2.1 The XOR dataset is not linearly separable. It consists of two classes, which are depicted in blue (circles) and orange (crosses). It becomes linearly separable when the data points are transformed using a nonlinear function. Idea for Figure taken from Günnemann [32].

passes the result to the next layer l. One possible implementation of this network is shown in Figure 2.2, which defines $f(X, W)$ as

$$
\begin{aligned}
f(x, W) &= W^{[2]} a^{[1]} + b^{[2]} \\
&= W^{[2]} \sigma_0 \left(W^{[1]} a^{[0]} + b^{[1]} \right) + b^{[2]} \\
&= \begin{bmatrix} w_{00}^{[2]} & w_{10}^{[2]} \end{bmatrix} \times \sigma_0 \left(\begin{bmatrix} w_{00}^{[1]} & w_{10}^{[1]} \\ w_{01}^{[1]} & w_{11}^{[1]} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_0^{[1]} \\ b_1^{[1]} \end{bmatrix} \right) + b_0^{[2]},
\end{aligned}
\tag{2.1}
$$

where $w_{ij}^{[l]} \in \mathbb{R}$ describes the weight assigned between neuron i and neuron j in layer l, and $b_0^{[1]}, b_1^{[1]} \in \mathbb{R}$ the biases.

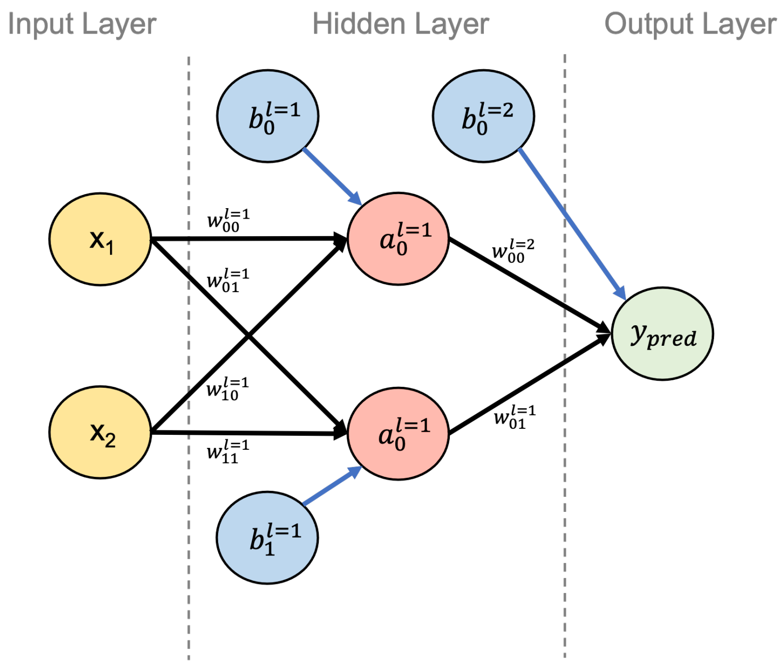

Figure 2.2 Multilayer Perceptron with one hidden layer. It takes two values x_1 and x_2 as input, transforms them via weights W, biases b and a nonlinear activation function σ into a hidden representation a, which is then used to for prediction. Idea for Figure taken from Goodfellow, Bengio, and Courville [31].

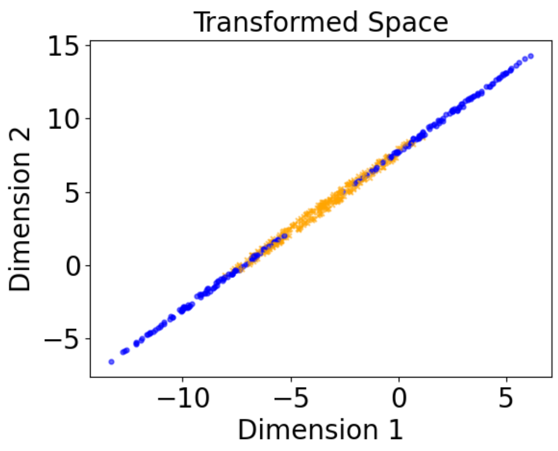

Figure 2.3 Visualized first layer output for the trained MLP (see Figure 2.2) of the input dataset (see Figure 2.1) after applying the activation function. Idea for Figure taken from Günnemann [32].

The MLP's output y_{pred} is iteratively refined by updating the weights and biases during training where the network attempts to minimize a predefined loss function for each sample in the training set.

After refining the weights over several epochs—where an epoch is defined as a training iteration that includes the entire train set, using each element once—it can be observed how the nonlinear activation function in the first layer transforms the coordinates of the XOR dataset into a space in which the data becomes linearly separable (see Figure 2.3).

Neural Networks

Neural Networks (NNs)—of which MLPs are a subclass—consist of multiple hidden layers containing an arbitrary amount of neurons. These networks can be customized with various modifications, such as non-fully connected neurons, shared weights between neurons, or the use of different activation functions within the NN. Furthermore, they provide state-of-the-art results in many areas such as computer vision or Natural Language Processing. Prominent examples of NN architectures include Convolutional Neural Networks, Recurrent Neural Networks, and Transformers.

In general, NNs are flexible models that can learn non-linear relationships, have the ability to generalize to new unseen data, and can also be scaled efficiently by training on GPUs.

Activation functions

The activation function σ is usually applied element-wise and introduces non-linearity to the output of the nodes [31]. Hence, it is capable of capturing complex patterns within the data. It takes the input and transforms it to a predefined range that depends on the activation function used.

Four commonly used activation functions are: Sigmoid activation, ReLU, LeakyReLU and ELU, as illustrated in Figure 2.4. Their usage depends on the specific problem, as each of them brings its own advantages and disadvantages. ReLU, for example, is computationally very efficient, but can suffer from the problem of dying ReLU, in which the output is set to zero for some neurons [33]. Leaky ReLU circumvents the latter by introducing an additional scalar hyperparameter a which avoids values smaller than zero from being entirely zero. The ELU is similar to the Leaky ReLU, but instead of having a fixed negative slope for $x \leq 0$, it decays exponentially, controlled by a hyperparameter α, typically set to 1. The sigmoid activations is computationally more complex and susceptible to vanishing or exploding gradients during NN optimization. However, since its output range is fixed, the result of the final layer can be rescaled to lie between zero and one, and can therefore be used for functions that require a probability as input, such as loss functions for classification.

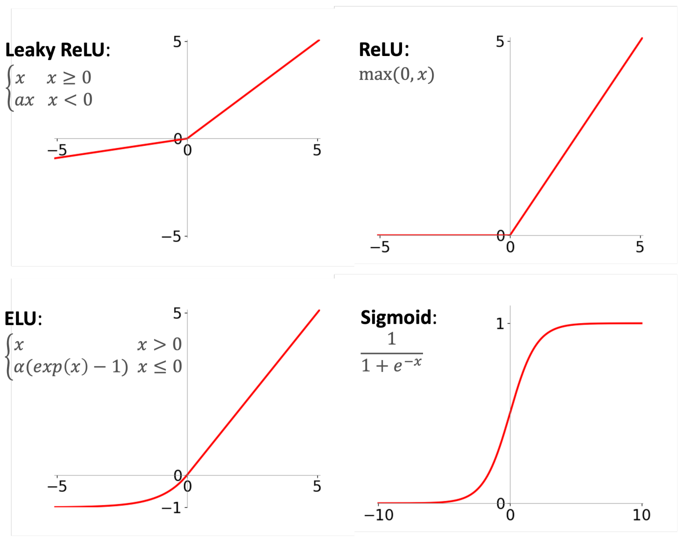

Figure 2.4 Visualization of various nonlinear activation functions commonly used in Neural Networks.

Loss function

The Loss function $\mathcal{L}$, also known as cost function, serves a crucial role in the training process of MLPs, and more generally for all NNs. It provides a measure of the quality of the model's prediction compared to the ground truth. Different loss functions can be used depending on the objective. In the following, two popular choices are discussed. In the case of the XOR example with N observations, the Mean Squared Error (MSE) loss function is used, where

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} ||y_i - f(X_i, W)||_2^2 = \frac{1}{N} \sum_{i=1}^{N} ||y_i - y_{\text{pred}_i}||_2^2. \tag{2.2}$$

In this case, due to the quadratic constraint, the further the network's prediction deviates from the actual solution, the more this error is penalized, and vice versa.

Given a classification task such as predicting whether a skin lesion is benign or not, the binary cross entropy loss functions with $y_i \in \{0, 1\}$ yields

$$\mathcal{L} = \sum_{i=1}^{N} -(y_i \log(p_{i,y_i}) + (1 - y_i) \log(1 - p_{i,y_i})), \tag{2.3}$$

where p_{i,y_i} is the networks' predicted probability of X_i belonging to class y_i. For the multi-class case with $y_i \in \{1, \dots, S\}$ for $S \in \mathbb{N}$, the formulation of $\mathcal{L}$ is adjusted to

$$\mathcal{L} = -\sum_{i=1}^{N} \log \frac{\exp(p_{i,y_i})}{\sum\limits_{s \in S} \exp(p_{i,s})}, \tag{2.4}$$

where $p_{i,s}$ represents the predicted probability of observation i belonging to class s.

Since the loss function is fully differentiable, the task of finding the optimal NN boils down to an optimization problem. Thus, the objective is to find the parameters θ that correspond to the global minimum of $\mathcal{L}(\theta)$:

$$\Theta^* = \arg\min_{\Theta} \mathcal{L}(\Theta). \tag{2.5}$$

Gradient Descent Techniques

Since the solution for high-dimensional optimization problems is often not as easy to compute, other methods are required to find the global minimum. One of these methods is gradient descent. The goal of gradient descent for NNs is to iteratively adapt the weights and biases to achieve the lowest possible loss. Because the gradient of a function indicates the direction of steepest ascent, it can be utilized to take a step of size τ in the opposite direction, that of steepest descent. There are several methods for gradient descent, two of which are very well known:

1. The gradient of the entire training data is calculated, which leads to a more accurate optimization, but involves a high computational overhead. This is described by the vanilla-gradient descent, which is represented as follows
$$\Theta \leftarrow \Theta - \tau \nabla_{\Theta} \mathcal{L}(X, y, \Theta), \tag{2.6}$$
where $\nabla_{\Theta}\mathcal{L}$ is the gradient ∇ of the models loss $\mathcal{L}$ with respect to its internal parameters Θ.

2. The second possibility is to work only with a randomly selected subset $\mathcal{S}$—also called a mini-batch—of the training data, which reduces computational cost but adds more noise to the update step. This process is known as stochastic gradient descent
$$\Theta \leftarrow \Theta - \frac{\tau}{|\mathcal{S}|} \sum_{j \in \mathcal{S}} \nabla_{\Theta} \mathcal{L}_j(X_j, y_j, \Theta). \tag{2.7}$$
The parameters Θ_t are updated iteratively for multiple epochs, where an epoch is defined such that every point in the training dataset was part of a mini-batch during training [34].

The step size τ is a positive scalar and also called the learning rate. By moving in the direction of the steepest descent, a local minimum is approached after a certain number of iterations. Gradient descent depends strongly on the initial value of the learning rate, i.e., if τ is chosen to small, the algorithm takes significantly longer to converge, may get stuck at saddle points or fail to escape a local minimum [31]. However, if the learning rate is too high, the global minimum might not be reached because it is overshot or oscillates above the perfect solution.

These different and well known problems are presented in Figure 2.4 for different learning rates τ_i and initial positions A, B, C and D. As Figure 2.5 shows, it is very important to choose and adapt the learning rate correctly. Thus, there are learning rate schedules to adjust τ iteratively, along with alternative algorithms that dynamically adjust the learning rate. The latter adapt the learning rate based on statistical information of past gradients to enable faster and more robust convergence. One of these algorithms is the Adam

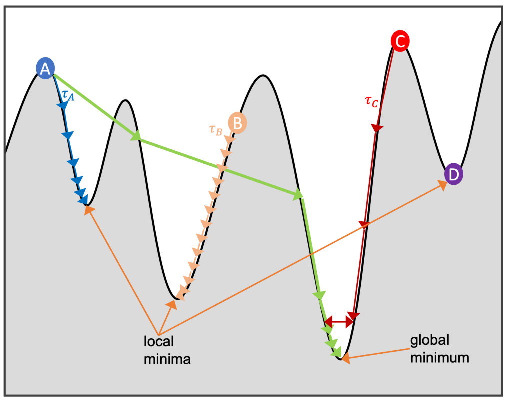

Figure 2.5 Visualization of the importance of appropriate learning rates τ_i for different starting points A,B,C and D. For low values of τ_i, the gradient descent algorithm takes longer to converge and is unable to escape local minima which is the case for A and B. C displays the problem when the learning rate is too large, as C then starts to oscillate above the minimum and thus cannot reach it. D is located in a local minimum, but its learning rate is so low that it is unable to escape it. The green line plots the optimal way for point A. Idea for Figure taken from Günnemann [32]

algorithm, which calculates the gradient for every datapoint and calculates the mean m and variance v over the entire batch, followed by the update

$$\begin{aligned} m &\leftarrow \beta_1 m + (1-\beta_1)\frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}}\nabla_\Theta \mathcal{L}(X_j, y_j), \Theta), \\ v &\leftarrow \beta_2 v + (1-\beta_2)\frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}}(\nabla_\Theta \mathcal{L}(X_j, y_j, \Theta))^2, \\ m' &\leftarrow \frac{m}{1-\beta_1^t}, \\ v' &\leftarrow \frac{v}{1-\beta_2^t}, \\ \Theta &\leftarrow \Theta - \frac{\tau}{\sqrt{v'}+\zeta}m', \end{aligned} \tag{2.8}$$

where β_1 and β_2 are hyperparameters controlling the decay rates of m and v, τ is the learning rate, ζ is a small positive scalar added for numerical stabilization [31, 35].

This provides almost all the necessary building blocks to train a NN of arbitrary size and connected components by replacing Θ with the weights and biases specific to the NN architecture. However, the only missing element is the gradient $\nabla_{W,b}$ of the loss function, which is derived in the following.

Backpropagation

Since all previously defined modules are differentiable and a composition of differentiable functions is also differentiable, the gradient of the loss function can be calculated using the chain rule. If the Neural Network is not fully connected, the gradient is only computed and propagated for connections that do exist. Thus, all the necessary tools are available to train a Neural Network end-to-end.

2.1.2 Regularization Strategies

In DL, the generalization ability of a model during training is assessed by evaluating the loss on the validation set for each epoch. During this evaluation, the validation data is passed through, for example, an MLP and the loss is computed without updating the weights of the model. If the validation loss increases after multiple epochs of training, this is an indication that the model is overfitting and a sign to stop the training. Overfitting occurs when the model begins to memorize noise in the training data and loses its ability to generalise to new, unseen data. To mitigate this problem, various regularization techniques have been proposed, such as L^1 and L^2 regularization, dropout, Batch Normalization (BN), early stopping, data augmentation and many more. In the following we describe the regularization methods, which were utilized in the context of this work.

Dropout

Dropout is a regularization technique for NNs, which for every training step, removes a random prespecified percentage of neurons along with its incoming and outgoing connections [36]. This reduces overfitting by preventing the model from relying too heavily on single neurons.

L^1 and L^2 Regularization

These techniques adapt the loss function by adding a regularization term that depends on the Neural Network's weights, which helps to control overfitting. L^2 regularization incorporates each weights squared value, denoted by $||W||_2^2$, over the entire NN, where $||\cdot||_2^2$ is the L^2 norm. This yields the modified loss function:

$$\mathcal{L}^{\text{new}} = \mathcal{L} + \lambda||W||_2^2. \tag{2.9}$$

In this equation, λ is the regularization strength, that controls the models complexity. L^2 regularization encourages the NN to have evenly distributed weight values, thereby preventing individual neurons from having excessively high values and thus a large influence. Hence, reducing model complexity and leading to improved generalization [31]. On the other hand, L^1 regularization plays a similar role as L^2 regularization, but encourages weights to be zero [37]. It can be represented as:

$$\mathcal{L}^{new} = \mathcal{L} + \lambda \|W\|_1, \tag{2.10}$$

where $\|W\|_1$ is the L^1 norm, which corresponds to the sum of the absolute values of each weight.

Batch Normalization

The introduction of Batch Normalization by Ioffe and Szegedy [38] in 2015 has significantly impacted NNs by fostering more stable gradient flow during training [39]. This technique makes the optimization landscape smoother and thus helps the training process [40]. It as an adaptive reparametrization method that normalizes the activation vector and thereby guarantees faster convergence [31]. For normalization it uses the first and second moments μ and ψ, which are computed layer by layer during training, based on the activation vector a of all elements n in a batch:

$$\begin{aligned} \mu^{[l]} &= \frac{1}{n}\sum_i a_i^{[l]}, \\ \psi^{2[l]} &= \frac{1}{n}\sum_i (a_i^{[l]} - \mu^{[l]})^2. \end{aligned} \tag{2.11}$$

Subsequently, for each neuron in every layer, its normalized activation $a_{i,norm}^{[l]}$ is obtained by

$$a_{i,norm}^{[l]} = \frac{a_i^{[l]} - \mu^{[l]}}{\sqrt{\psi^{2[l]} - \zeta}}, \tag{2.12}$$

where ζ is a constant used for numerical stability. Finally, the above results yield the neurons output at layer l

$$a_{i,final}^{[l]} = \gamma^{[l]} a_{i,norm}^{[l]} + \beta^{[l]}, \tag{2.13}$$

where $\gamma \in \mathbb{R}^L$ and $\beta \in \mathbb{R}^L$ denote layer specific trainable parameters that allow the model to learn the optimal distribution for each hidden layer [39].

2.1.3 Deep Learning Frameworks

Part of Deep Learning's success is due to efficient implementation of vector, matrix and tensor operations on GPU. Since their architecture makes them many times faster for this task than CPUs especially for matrix and tensor calculations, more complex algorithms can be modeled. There are several packages in Python that can offload operations directly to the GPU and provide the means to create DL algorithms. The two most popular are Tensorflow from Google with its high level API Keras and Pytorch from Meta. In the following work, Pytorch is used exclusively.

2.2 Deep Learning on Graphs

Graphs encode rich information that is everywhere in our world, ranging from social networks to structural information to knowledge graphs. However, extracting as well as encoding that information presents several challenges for researchers. Some of them include the heterogeneity of graphs, the irregularity of their structure, scalability for large graphs, robustness, and over-smoothing [34, 41]. Nonetheless, they have been used with great success in a variety of fields, ranging from chemistry, biology, and physics to recommendation and social sciences—from tasks such as predicting protein structures to recommending the next movie [42, 43, 44]. The purpose of this Chapter is to provide an overview of the intrinsic structure required to use graphs to encode information and to perform Deep Learning on them.

2.2.1 Notation and Preliminaries

This introduction largely follows Zhang, Cui, and Zhu [41] and Veličković [45]. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, then $\mathcal{V}$ is the set of its vertices $\{u_1, ..., u_N\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of $M = |\mathcal{E}|$ edges between vertices. The term vertex and node is equivalent and is used interchangeably. A graph can either be undirected, in which case $(u, v) \in \mathcal{E} \Leftrightarrow (v, u) \in \mathcal{E}$ or directed, where $(u, v) \in \mathcal{E}$ are ordered pairs describing a direct edge from node u to node v. In the remainder of this work, it is assumed that undirected graphs are used unless otherwise stated. We define the adjacency matrix A $\in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ as a square matrix, where each entry a_{uv} is set to 1 if node u has an edge connecting with node v and otherwise 0:

$$a_{uv} = \begin{cases} 1 & \text{if } (u, v) \in \mathcal{E}, \\ 0 & \text{if } (u, v) \notin \mathcal{E}. \end{cases} \tag{2.14}$$

Several extensions to the original graph representation A have been proposed which encode additional information such as weights or temporal information of nodes or edges [34]. Although this extended representation of A may be useful to fully encode the graph structure, for the remainder of this paper it is assumed that A is defined as in Equation 2.14 unless otherwise stated. Furthermore, each $u \in \mathcal{V}$ can be initialized with a feature vector $X_u \in \mathbb{R}^m$, which forms the node feature matrix $X \in \mathbb{R}^{|\mathcal{V}|\times m}$:

$$X = [X_1, X_2, \ldots, X_{|\mathcal{V}|}]^\top. \tag{2.15}$$

Additionally, the definitions above impose a node ordering and thus permutation invariance and equivariance have to be satisfied for any permutation matrix P and functions g and H defined as

$$\begin{aligned} g(PX, PAP^\top) &= g(X, A) \qquad (Invariance), \\ H(PX, PAP^\top) &= PH(X, A) \quad (Equivariance). \end{aligned} \tag{2.16}$$

This guarantees that even if the nodes and edges are permuted the result remains the same. All graph architectures must satisfy these conditions in order to perform Deep Learning on them. In addition, each node in a graph can aggregate information about its neighbors, forming the so-called neighborhood $\mathcal{N}$. The 1-step neighbors of node u, $\mathcal{N}_u$, are defined by:

$$\mathcal{N}_u = \{v \mid (u, v) \in \mathcal{E} \vee (v, u) \in \mathcal{E}\} \tag{2.17}$$

and the multiset of all neighbourhood features, $X_{\mathcal{N}_u}$:

$$X_{\mathcal{N}_u} = \{\{X_v \mid v \in \mathcal{N}_u\}\}. \tag{2.18}$$

Another important property of a graph is its degree matrix $D \in \mathbb{N}^{|\mathcal{V}|\times|\mathcal{V}|}$ which encodes how many incoming and outcoming edges exist for every node. Hence, $D_{ii} = \sum_j A_{ij}$. The Laplacian matrix, L, defined as $L = D - A_w$, where A_w is a weighted adjacency matrix is particularly effective for capturing connectivity structure of a graph. It is commonly used in techniques such as spectral clustering, which aims to partition a graph into subsets in a way that minimizes the number of edges between different parts [34].

2.2.2 Graph Neural Networks

Graph Neural Networks (GNNs) are of particular interest for the dataset provided by UHE (see Section 4.1) with outcome variables such as "treatment failure". This dataset can be used to create a graph where patients are encoded as nodes and edges are created based on patient-patient similarity (see Section 3.2.1). By applying GNNs to this patient-patient similarity graph, the time to treatment failure can be predicted. This approach offers several advantages, such as mitigating bias and leveraging knowledge within communities by considering only a patient's k-step neighbors for its prediction. Ultimately, this can help physicians make more informed healthcare decisions, such as changing treatment if early treatment failure is predicted.

Now that the basic concepts of graphs are defined, the necessary tools are available to explore GNNs in the following. We adopt the notation introduced in Xu et al. [46]. One of the paradigms that have made Graph Neural Networks so successful is what is known as message passing, in which a node incorporates features from its neighboring nodes. The aggregation is independent of the size of the graph, which makes GNNs highly adaptable in real-world scenarios [34]. Each node first learns about the features of other nodes in its neighborhood $\mathcal{N}_u$ and aggregates this information into its hidden representation $h_u^{[l]}$. Hence, after k-iterations of this aggregation process, a node captures the structural information of its k-hop network neighborhood (see Figure 2.6). Formally, this can be written as

$$a_u^{[l]} = \text{AGGREGATE}^{[l]}\left(\left\{h_v^{[l-1]} : v \in \mathcal{N}(u)\right\}\right), \quad h_u^{[l]} = \text{COMBINE}^{[l]}\left(h_u^{[l-1]}, a_u^{[l]}\right), \tag{2.19}$$

where $h_u^{[l]}$ is the feature vector of node u after the l-th iteration of the aggregation process. The choice of AGGREGATE$(\cdot)$ and COMBINE$(\cdot)$ functions is still an active area of research because there is a wide and growing variety of them. The type of the aggregation function leads to different architectures of GNNs. Most can be classified into three categories according to Bronstein et al. [47]: convolutional, attentional and message-passing.

Before delving deeper into the use of different aggregation functions in layer depth $l \geq 1$, the first layer $l = 0$ is instantiated as

$$h_u^{[0]} = X_u. \tag{2.20}$$

It is important to note, that the initialization of the first layer may differ across GNNs implementations. In the following subsection different GNN variants are explored as described in Dwivedi et al. [42] and Xu et al. [46].

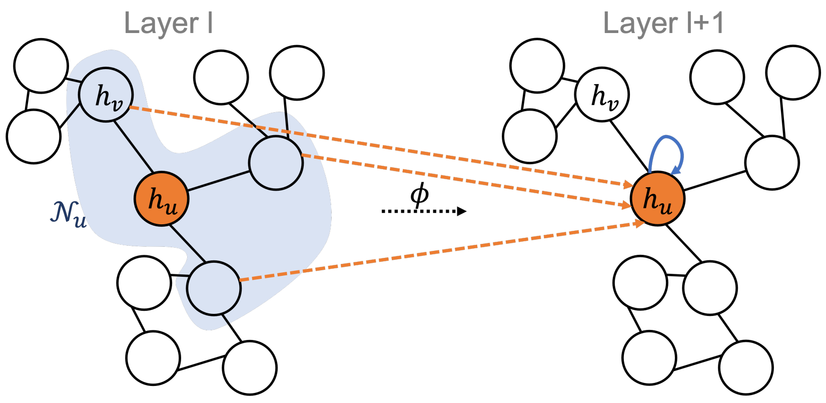

Figure 2.6 High level example of a message passing framework for a single node u. From layer l to layer l+1 of a GNN, u updates its hidden presentation $h_u^{[l+1]}$ with features from its neighboring nodes using an aggregation function ϕ, as indicated by the orange lines. In practice, for each layer, each node updates its hidden representation with features from its neighbourhood $\mathcal{N}$ separately. Idea for Figure taken from Günnemann [34] and Dwivedi et al. [42].

Graph Convolutional Networks

Graph Convolutional Networks (GCNs) combine the AGGREGATE and COMBINE steps by using element-wise mean pooling of the hidden features $h_u^{[l+1]}$ of nodes in combination with a learnable matrix W:

$$h_u^{[l]} = \text{ReLU}\left(W \cdot \text{MEAN}\left\{h_v^{[l-1]}, \forall v \in \mathcal{N}(u) \cup \{u\}\right\}\right). \tag{2.21}$$

Graph Attention Networks

The main aspect that distinguishes Graph Attention Networks (GATs) from models such as GCNs is that it uses dynamic weights e_{uv}^{kl}, also known as attention coefficients, instead of static weights. The idea behind this is that neighbors have different importance independent of their degree [48]. The attention weights e_{uv}^{kl} can then be learned by a NN. The network employs a multi-headed attention architecture with K heads to enhance its learning capacity, which is inspired by the transformer architecture. Therefore a node u updates its hidden representation for layer l by aggregating information from all its neighboring nodes in combination with learnable attention coefficients e_{uv}^{kl} and learnable weight matrices $W^{k,l}$, defined as

$$h_u^{[l+1]} = \text{CONCAT}_{k=1}^{K}\left(\text{ELU}\left(\sum_{v \in \mathcal{N}_u} e_{uv}^{k,l} W^{k,l} h_u^{[l]}\right)\right), \tag{2.22}$$

where the result of all K attention heads is then concatenated. For each head the attention coefficients e_{uv}^{kl} are set as:

$$\begin{aligned} e_{uv}^{k,l} &= \frac{\exp\left(\hat{e}_{uv}^{k,l}\right)}{\sum_{v' \in \mathcal{N}_u} \exp\left(\hat{e}_{uv'}^{k,l}\right)} \quad \text{and} \\ \hat{e}_{uv}^{k,l} &= W_{\text{att}}^{k,l} \text{LeakyReLU}\left(W^{k,l} \text{CONCAT}\left(h_u^{[l]}, h_v^{[l]}\right)\right). \end{aligned} \tag{2.23}$$

Graph Isomorphism Networks

Graph Isomorphism Networks (GINs) are motivated by the Weisfeiler-Leman (WL) test for graph isomorphism [45, 46]. The WL-test iteratively aggregates the labels of the nodes and their neighbourhoods, followed by a conversion to unique labels using perfect hashing. It is important to note that this test is not conclusive and no cubic time algorithm exists yet to solve the graph isomorphism problem. The WL algorithm resembles the GNN message passing framework. Its capability to differentiate graphs can be advantageous. GINs generalizes the WL test and thus reaches maximum discriminative performance among certain GNNs [46]. The update equation for node u at layer l is defined as:

$$h_u^{[l]} = \text{MLP}^{[l]}\left(\left(1 + \epsilon^{[l]}\right) \cdot h_u^{[l-1]} + \sum_{v \in \mathcal{N}(u)} h_v^{[l-1]}\right), \tag{2.24}$$

where ϵ is a trainable scalar and determines the importance of the source node compared to its neighbors.

2.2.3 Prediction Objectives

GNNs are used for prediction in a variety of task, among them classification, link prediction, node and graph classification, graph regression, cycle detection and other related fields [42]. The ensuing analysis gives particular attention to two tasks, namely edge prediction and node classification.

Node classification

One goal of GNNs is to determine the class of a node. For example, in the context of patients represented as nodes and medical conditions as edges, the objective is to identify the category to which a patient belongs, such as disease subtypes. First, several GNN layers are applied to the initial state of the graph so that nodes learn to incorporate features of their neighbors (see Figure 2.7). This is followed by shared classification function Γ, which independently obtains each nodes final hidden feature vector $h_u^{[L]}$ as input and then uses it for training and outputs its prediction for node u:

$$z_u = \Gamma(h_u^{[L]}). \tag{2.25}$$

Link prediction

The inference of the existence or properties of an edge, referred to as e_{uv}, between entities such as drugs and a disease is called edge or link prediction [45]. This is especially important for tasks such as patient-patient similarity analysis, where identifying similar patients in a graph can help avoid costly medical procedures and improve a patient's overall outcome. As shown in Figure 2.7, after L GNN layers, a classification function Γ takes the concatenated representation h_u, h_v of the two nodes as input and predicts whether the edge e_{uv} exists

$$z_{uv} = \Gamma(h_u^{[L]}, h_v^{[L]}, e_{uv}^{[L]}). \tag{2.26}$$

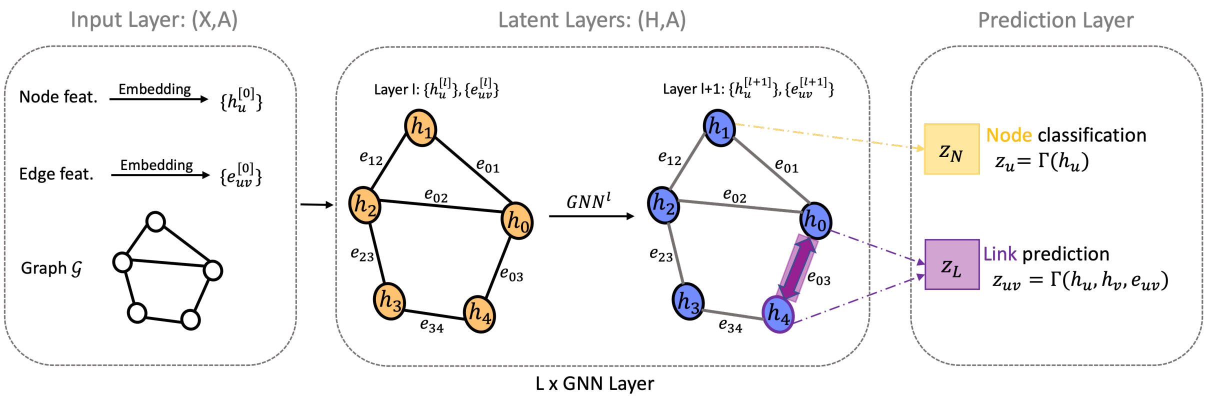

Figure 2.7 Overview of the internal architecture and prediction tasks of Graph Neural Networks. First, the graph $\mathcal{G}$ is encoded via its adjacency matrix A with node features $h_u^{[0]}$ and edge features $e_{uv}^{[0]}$. L layers of a GNN follow, in which the hidden representations of nodes $h_u^{[L]}$ are updated, as well as their edge weights $e_{uv}^{[L]}$, if given. Finally, the embeddings of the last layer $h_u^{[L]}$ and $e_{uv}^{[L]}$ are passed to a function Γ for node classification or link prediction. Idea for Figure taken from Dwivedi et al. [42] and Veličković [45].

2.3 Optimal Transport

Optimal Transport (OT), in particular OT maps, provide a unique approach to finding patient-patient similarities in a high-dimensional, noisy medical dataset such as the one in Section 4.1. The OT map identifies the most efficient way to "transport" the features of one patient to another, thereby providing a notion of similarity that can in turn be used to create edges for a patient-patient similarity graph (see Section 3.2.1).

A popular example motivating OT is the problem of a worker creating a certain shape from a pile of soil. Assuming the worker wants to minimize the total effort to save both energy and time, they want to find the "cheapest" way to move it. This can be mathematically modeled by considering the task of transforming all of the mass of one probability distribution to another, while minimizing the cost of transportation. The idea is to find an OT map that efficiently transports the mass from the source distribution to the target distribution, while ensuring that no mass is left behind.

Additionally, OT provides a way to measure the similarity between sets of points or probability distributions using the Wasserstein distance. It further aims to preserve the underlying geometry of the objects being transported, which in turn is useful for tasks such as data alignment [49, 50]. OT solvers became very attractive for Machine Learning as research made them more scalable [51], and has been an active area ever since, ranging from applications for single-cell genomics [52], to knowledge graphs [25], to generative models such as Wasserstein-GANs [53] or OT maps [54]. In the proceeding sections, we lay out the mathematical foundations of Optimal Transport, followed by a detailed explanation of how to solve for the optimal transportation matrix P^*, and an introduction to generalized versions of Optimal Transport.

2.3.1 Theoretical Foundations

Notation and Preliminaries

Given two datasets $\{(\mathbf{X}_i, a_i)\}_{i=1}^n$ and $\{(\mathbf{X}'_j, b_j)\}_{j=1}^m$, where $X \in \mathcal{X} \subset \mathbb{R}^d$ and $X' \in \mathcal{X}' \subset \mathbb{R}^d$ are two discrete finite spaces, we want to compute the distance between them [55]. The probability masses are defined as $\sum_{i=1}^n a_i = 1$ and $\sum_{j=1}^m b_j = 1$. The Monge-Kantorovich relaxation for OT defines the set of possible transport plans, also known as admissible couplings, between two discrete measures, $\mu = \sum_{i=1}^n a_i \delta_{X_i}$ and $\nu = \sum_{j=1}^m b_j \delta_{X_j}$ as

$$U(\mu, \nu) = \{P \in \mathbb{R}^{n \times m} : P\mathbf{1}_m = a, P^\top \mathbf{1}_n = b\}. \tag{2.27}$$

$\mathbf{1}_n$ denotes the n-dimensional vector of ones and δ_{X_i} is the Dirac distribution at position X_i, defined as:

$$\delta_{X_i} = \begin{cases} 1, & X = X_i, \\ 0, & \text{otherwise.} \end{cases} \tag{2.28}$$

Additionally, $a = (a_1, a_2, ..., a_n)^\top \in \mathbb{R}_+^n$ and $b = (b_1, b_2, ..., b_m)^\top \in \mathbb{R}_+^m$, where a and b are also referred to as left marginals and right marginals, respectively. This then gives the Kantorovich-OT formulation between two discrete measures

$$P^* = \min_{P \in U(\mu,\nu)} \langle P, C \rangle = \min_{P \in U(\mu,\nu)} \sum_{i=1}^n \sum_{j=1}^m p_{ij} c_{i,j}, \tag{2.29}$$

in which p_{ij} denotes the transportation plan P from source i to target j and c_{ij} represents the i,j-th element of the cost matrix C.

This discrete formulation of OT can be expressed in terms of a Linear Program (LP) and be solved in cubic time [56]. Minimizing the total cost $\langle P, C \rangle$ is subject to three constraints, the first requires that the entirety of the source is transported away, the second condition implies that the target receives everything it needs and the last inquires that the transportation plan must be positive [57].

Wasserstein Distance

The Wasserstein distance allows the comparison of non-overlapping singular distributions, such as discrete ones, and the quantification of their spatial shift [58]. Unlike classical distances, which are not even defined for discrete distributions, the Wasserstein distance allows a meaningful comparison. Defining the unit-cost C as a matrix with entries $c_{ij} = d(X_i, X_j)^p$ and using the the Optimal Transport map P^*, the p-Wasserstein distance is defined as

$$W_p(\mu, \nu) = (\langle P^*, C\rangle)^{1/p}, \tag{2.30}$$

wherein $d(\cdot)$ is a distance function such as the L^1 or L^2 distance [56]. The special case for the W_1 distance is also known as Earth Mover's Distance (EMD), which is an important measure of the distance between two probability distributions. It satisfies all the requirements of a metric space and thus defines a proper distance, given that the underlying ground distance is metric and the two distributions have the same integral [59]. In particular, the W_1 distance is typically more robust to the presence of noise and outliers as compared to the W_2 distance. Returning to the original example of moving two piles of soil, EMD is the minimum cost of converting one pile into the other using the Optimal Transport plan P^*, which is the solution to Equation 2.29.

2.3.2 Algorithms

Entropy regularized optimal transport

Since the naïve method required $O(\log(n)n^3)$ operations, Cuturi et al. introduced entropic regularization to perform the calculation in $O(n^2)$ [51]. Additionally, it was shown that some of the drawbacks of the classical OT, such as non-unique solutions, increased computation time for a large number of bins, and potential non-differentiability, are mitigated [57]. They add a relaxation term to the LP, thus enforcing less sparsity with a hyperparameter ϵ, promoting a smoother problem and allowing faster computation:

$$\begin{gathered}\arg\min_P \langle P, C\rangle + \epsilon H_P,\\ \text{subject to } P\mathbf{1}_m = a, P^\top \mathbf{1}_n = b, P \geq 0,\end{gathered} \tag{2.31}$$

where H_p is the entropic regularization term defined as

$$H_p = \sum_{ij} p_{ij} \log p_{ij}. \tag{2.32}$$

The condition of Equation 2.31 can then be incorporated by using Lagrange multipliers, resulting in the following optimization problem

$$L(P, \alpha, \beta) = \sum_{ij} p_{ij} c_{ij} + \epsilon \sum_{ij} p_{ij}(\log p_{ij}) + \alpha^\top (P\mathbf{1}_m - a) + \beta^\top (P^\top \mathbf{1}_n - b). \tag{2.33}$$

This can then be solved by taking the derivative with respect to P, which results in an expression for it, where only the Lagrangian multipliers are unknown, which is given by

$$P = \operatorname{diag}\left(\exp\left(\frac{\alpha}{\epsilon}\right)\right) \cdot \exp\left(-\frac{C}{\epsilon}\right) \cdot \operatorname{diag}\left(\exp\left(\frac{\beta}{\epsilon}\right)\right), \tag{2.34}$$

with the division operator applied element-wise. This has the form

$$P = \operatorname{diag}(u) \cdot K \cdot \operatorname{diag}(v), \quad K = \exp\left(-\frac{C}{\epsilon}\right), \tag{2.35}$$

where the unknown scaling vectors $u \in \mathbb{R}^n_+$ and $v \in \mathbb{R}^m_+$ must satisfy the conservation of mass conditions given by

$$a = u \odot (Kv) \quad \text{and} \quad b = v \odot (K^\top u), \tag{2.36}$$

Algorithm 1 Sinkhorn Algorithm

Input: Cost Matrix C, Regularization Parameter ϵ, Marginal Constraint Vectors u, v

Output: Optimal Transport Plan P^*

```
1:  procedure SINKHORN($C, \epsilon, u, v$)
2:      $K \leftarrow \exp(-\frac{C}{\epsilon})$
3:      $v \leftarrow \mathbf{1}_n$
4:      $u \leftarrow \mathbf{1}_m$
5:      repeat
6:          $u \leftarrow a/(Kv)$
7:          $v \leftarrow b/(K^T u)$
8:      until convergence
9:      $P^* \leftarrow \mathrm{diag}(u) \cdot K \cdot \mathrm{diag}(v)$
10: end procedure
```

where $\odot$ corresponds to the Hadamard product [58]. This can be solved by an algorithm called the Sinkhorn algorithm. It first assumes an initialization for the Lagrange multipliers and then iteratively updates these values as they depend on one another. When a stationary point is reached, the algorithm stops.

Unbalanced Optimal Transport

Unbalanced Optimal Transport (UOT) is a robust extension of OT that extends the set of admissible couplings $U(\mu, \nu)$ so that the marginal constraints do not have to be satisfied [60]. This allows the algorithm to discard irrelevant samples and not to transport them, thereby for example reducing the influence of outliers. Formally, this can be written as

$$\begin{aligned} P^* = \arg\min_P \langle P, C\rangle + \epsilon \cdot H_p + \rho_1 \cdot \mathrm{KL}(P\mathbf{1}, a) + \rho_2 \cdot \mathrm{KL}(P^\top \mathbf{1}, b), \\ \text{subject to } P \geq 0, \end{aligned} \tag{2.37}$$

with KL being the Kullback-Leibler divergence and ρ an unbalancedness parameter that controls the extent to which the marginal constraint can be violated. The Kullback-Leibler divergence is defined as

$$\mathrm{KL}(\mu \mid \nu) = \begin{cases} \sum_{z \in Z} \mu(z) \log\left(\frac{\mu(z)}{\nu(z)}\right) - \mu(z) + \nu(z) & \text{if } \mu, \nu > 0, \\ \infty & \text{else,} \end{cases} \tag{2.38}$$

where Z is a discrete and finite space, such as $X \times X'$. Divergences are a natural concept as a measure of dissimilarity or discrepancy between probability distributions and often used in the context of OT.

In general, UOT provides a flexible approach that can handle imbalanced datasets which is often the case in real-world scenarios. Similar to the Sinkhorn Algorithm, the unbalanced optimization problem can be efficiently solved with the generalized Sinkhorn-Knopp matrix scaling algorithm. For further mathematical details, please refer to the Chizat et al. [61], as it is beyond the scope of this work.

2.4 Survival Analysis

Building models for survival analysis is paramount for gaining insight into disease progression, features affecting survival likelihood (see Section 4.4.2), potential cures or medications for specific diseases, and exposure risks. In particular, for data collected by hospitals, such as that provided by UHE (see Section 4.1), it is crucial to support healthcare decisions and medical intuitions with mathematical models. In the following, the format of survival data is presented, succeeded by an introduction to various metrics used for evaluation, and finally, a detailed mathematical overview of survival models is given.

Survival data consists of N individuals with d features encoded in matrix $X \in \mathbb{R}^{N \times d}$, a time-to-event vector $T \in \mathbb{R}_+^N$ and an event indicator vector $E \in \{0,1\}^N$. If e_i equals zero, it indicates that the event did not occur until time t_i, and in this case the individual is considered to be right censored. For example, t_i and e_i could be the time an individual was in a medical trial and whether or not they survived, or the duration of an individual's specific treatment and if their outcome was successful or not.

Survival analysis typically uses two fundamental functions: the survival function $S(t)$ and the hazard function $\lambda(t)$, defined as

$$S(t) = P(t_i > t) \tag{2.39}$$

and

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{S(t) - S(t + \Delta t)}{\Delta t \cdot S(t)}. \tag{2.40}$$

The function $S(t)$ gives the probability that an individual will survive beyond time t. In contrast, $\lambda(t)$ represents the probability that an individual will not survive a small time period Δt, given that they have already survived up to time t.

Survival analysis requires special methods because standard approaches such as regression do not take into account the fact that data is censored and usually discard the censored data and thus potentially useful information [21]. Several survival methods exist, such as Kaplan-Meier estimators, Cox Proportional Hazards Models (CPHs), Accelerated Failure Time Models, Random Survival Forests, and Deep Learning based approaches. In addition to these methods and this particular type of data, specific metrics are needed to evaluate them.

2.4.1 Metrics

Concordance Index

The Concordance Index, also called C-Index or Harrell's Concordance Index, is a commonly used metric for evaluating right censored data and can be seen as a generalization of the area under the ROC curve [62]. It is a measure of the extent to which the predicted risk scores accurately reflect the observed sequence of events [63]. A C-index of 0.5 means the model performs as well as random prediction, and 1 represents perfect prediction. It is obtained by considering all comparable pairs of individuals (i,j), i.e., which of them (i or j) experienced an event first, and counting the number of concordant and discordant pairs, including edge cases for time or risk score ties. A comparable pair (i,j) is considered to be concordant if the individual with the higher assigned risk experienced the event earlier, otherwise it is discordant. It can be expressed as

$$\hat{C} = \frac{\sum_{i=1}^{N} \delta_i \sum_{j=i+1}^{N} [I(t_i < t_j)(1 - \delta_j)I(t_i = t_j)][I(r_i > r_j) + \frac{1}{2}I(r_i = r_j)]}{\sum_{i=1}^{N} \delta_i \sum_{j=i+1}^{N} [I(t_i < t_j) + (1 - \delta_j)I(t_i = t_j)]}, \tag{2.41}$$

where δ_i is a binary variable that is one if the event for individual i occurred by t_i, and zero if censoring happened. Furthermore, t_i is the survival time of individual i, r_i is the predicted risk score, and $I(\cdot)$ is an indicator function that returns one if its argument is true and zero if not. The advantage of this measure is that it incorporates information from censored patients even though the event of interest did not occur. This is possible because survival to censoring is known. For example, one patient may outlive another even if censored, which is taken into account for the Concordance Index, providing insightful information that may be lost in other metrics. If there are no ties in time and predicted risk, the equation simplifies to

$$\hat{C} = \frac{\sum_{i=1}^{N} \delta_i \sum_{j=i+1}^{N} I(t_i < t_j) I(r_i > r_j)}{\sum_{i=1}^{N} \delta_i \sum_{j=i+1}^{N} I(t_i < t_j)}. \tag{2.42}$$

Note that when a survival model produces multiple risk scores over time, the average risk for each individual is used to calculate the C-Index.

Brier Score

The Brier Score (BS) is another metric commonly used in survival analysis. It is the extension of the Mean Squared Error at time t for right censored survival data [64]. Because it is not a ranking metric like the Concordance Index, it directly accounts for the models prediction and can thus be used for model calibration and discrimination. The formula is expressed as

$$\mathrm{BS}^c(t) = \frac{1}{n} \sum_{i=1}^{N} \left(I(t_i \leq t \wedge \delta_i = 1) \frac{(0 - \hat{\pi}(t|X_i))^2}{\hat{G}(t_i)} + I(t_i > t) \frac{(1 - \hat{\pi}(t|X_i))^2}{\hat{G}(t)} \right), \tag{2.43}$$

where X_i denotes the d dimensional feature vector of individual i, N is the total number of individuals, $\hat{\pi}(t(X_i))$ represents the estimated likelihood of individual i remaining event-free up to time t, and $\hat{G}(t)$ gives the inverse probability of the censoring weight that was determined by the Kaplan-Meier estimator [65].

The Integrated Brier Score (IBS) is an important extension since it compresses the Brier Scores over multiple time points into a single scalar value. Thus, by considering all time points from the minimum time, $t_{\min}$, to the maximum time, $t_{\max}$, the IBS provides a holistic assessment of a model's predictive power throughout the time span of interest. It is defined as

$$\mathrm{IBS} = \int_{t_{\min}}^{t_{\max}} \mathrm{BS}^c(t) d\omega(t), \tag{2.44}$$

where $\omega(t) = t/t_{max}$ is a weighting function [66]. Generally, a lower Integrated Brier Score suggests a superior predictive performance of the model.

2.4.2 Survival Methods

Kaplan-Meier Estimator

One of the simplest models to estimate survival is the Kaplan-Meier estimator. It works by decomposing T into smaller time steps and thus estimating the probability of survival till time t by

$$\hat{S}(t) = \prod_{i:t_i \leq t} \frac{n_i - d_i}{n_i}, \tag{2.45}$$

where n_i is the number of individuals at risk and d_i the number of individuals for whom the event e_i occurred at time t_i. This method has several limitations. First, it does not take other features into account, thus disregarding potentially helpful additional information [67]. Second, it assigns the same probability of survival to all individuals at a given time, neglecting inherent differences between individuals. This creates the need for more sophisticated models.

Cox Proportional Hazards Model

The Cox Proportional Hazards Model (CPH) offers an approach which takes the features X as input, as well as the events E and times T as defined in Section 2.4. It assumes that the hazard ratio for any two individuals remain proportional and do not vary with time [67, 68]. This assumption can be evaluated with statistical tests, and if a feature fails these tests, it can either be omitted or binned into smaller intervals to reduce the violation. CPH are defined as follows

$$\lambda(t|X_i) = \lambda_0(t)\exp(\beta_1 x_{i1} + \ldots + \beta_d x_{id}) = \lambda_0(t)\exp(\beta X_i), \tag{2.46}$$

where $\lambda_0(t)$ is the baseline hazard which is the same for all individuals, x_{id} represents the scalar value of feature d of patient i, and $\beta \in \mathbb{R}^d$ are the estimates for the effect of each feature in the dataset. The likelihood of the event being observed for subject i at time t_i in the Cox Proportional Hazards Model can be expressed as follows:

$$L_i(\beta) = \frac{\lambda(t_i|X_i)}{\sum_{j:T_j \geq T_i} \lambda(t_i|X_j)} = \frac{\exp(\beta X_i)}{\sum_{j:T_j \geq T_i} \exp(\beta X_j)}. \tag{2.47}$$

Assuming the individuals to be statistically independent, the coefficients can be estimated with an optimization algorithm which maximizes the total partial log likelihood function over β defined as

$$l(\beta) = \sum_{i:E_i=1} log(L_i(\beta)) = \sum_{i:E_i=1}\left(\beta \cdot X_i - \log \sum_{j:T_j \geq T_i} \exp(\beta X_j)\right). \tag{2.48}$$

Please refer to Efron [69] for the formula that includes time ties, as the formula presented here assumes the absence of them for simplicity. The exponentials of these estimates are called the Hazard Ratios (HRs). If $HR_{\beta_i} > 1$, this means that an individual has a higher risk of experiencing the event of interest, while if $HR_{\beta_i} < 1$ signifies the opposite scenario. For $HR_{\beta_i} = 1$, feature i has no effect on the outcome of interest. Since this CPH build on the proportional hazards assumption, which can be restrictive in certain scenarios, alternative methods have been proposed.

Tree Methods

In general, tree methods offer numerous beneficial properties such as scalability, robustness, interpretability, handling of mixed data types, the ability to capture nonlinear interactions, and more. For survival analysis, there exist special tree methods that take censoring into account, such as Random Survival Forests and XGBoost Survival Embeddings. Both are as ensemble methods, which can lead to better generalization results and less overfitting, and unlike Neural Networks, they provide an inherent understanding of feature importance. Furthermore, they are effective at handling high-dimensional input, which is common for hospital datasets (see Section 4.1).

This discussion first introduces the concept of trees in general, followed by a detailed explanation of tree methods specifically designed for survival analysis.

1. Decision Trees

Decision trees are hierarchical, nonlinear models that have been successfully used in many applications for both classification and regression problems. They have many desirable features, such as scale independence, interpretability, and work for numerical and categorical features. They consist of nodes, branches and leaves. A node represents a test of a feature that partitions the dataset, and a branch represents the different results of that partitioning and the distribution of the samples in that particular region [70]. Since it is too costly to determine the perfect tree, there are greedy top-down approaches that fit the tree, with the help of special heuristics. Two of such heuristics are the Gini-Index (L_G) and Entropy (L_H). Both measure how pure the class distribution of a possible feature distribution is at a node to determine which feature the node should split the dataset for. They are defined as follows:

$$\begin{aligned} L_G &= 1 - \sum_{i=1}^{c} p_i^2, \\ L_H &= - \sum_{i=1}^{c} p_i \log_2(p_i). \end{aligned} \tag{2.49}$$

In these equations p_i represents the proportion of samples belonging to class c for a specific node. In addition, due to the recursive way of growing a decision tree, it is important to find an appropriate stopping point. Thus, there exist several stopping criteria, such as reaching a maximum depth, achieving a certain accuracy on the validation set, branch purity, and more. The ends of a grown tree are referred to as terminal nodes. As for the drawbacks, complex decision trees are prone to overfitting, which is usually addressed by pruning the tree, and are high variance predictors, as small variations in the data can create a different tree [71]. Figure 2.8 shows a Decision Tree (DT) that was constructed using a dataset of 120 samples, where each sample has four features and belongs to one of three classes A, B and C. For this DT the Gini-Index was used as a measure of impurity. A Gini-Index of zero indicates that the distribution in a branch is pure, such as for the samples belonging to class A in the first split of this example.

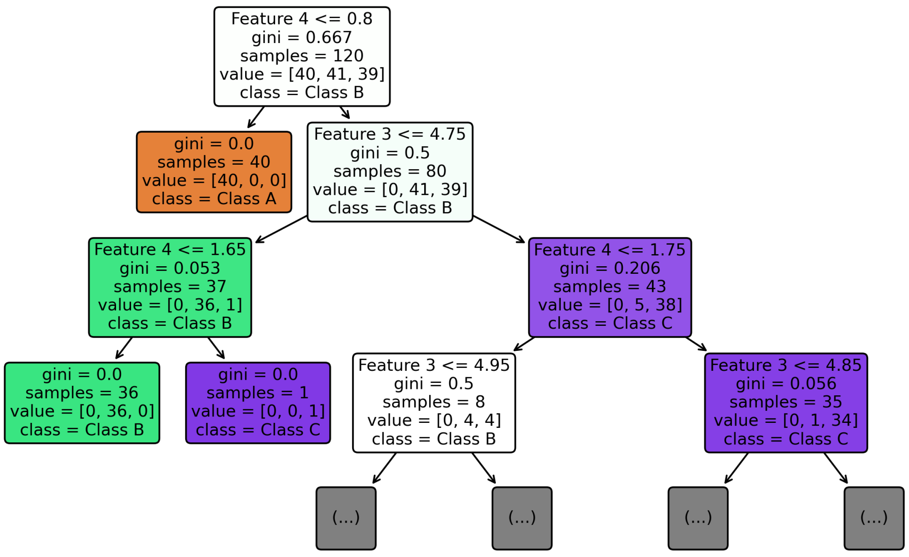

Figure 2.8 An illustration of a Decision Tree using a dataset with four different features and three outcomes (A,B, and C). Initially, the node is split based on the feature that maximizes the purity within the resulting branches. Using a heuristic, such as the Gini-index, feature 4 is found to be optimal for splitting the data. This is achieved by placing all data points whose feature 4 value is less than or equal to 0.8 in the left branch, while the remaining data points are assigned to the right branch. This procedure is repeated for all leaves, until the branch distribution becomes pure or meets a stopping criterion.

2. Random Forests

Random Forests (RFs) are a combination of multiple Decision Trees that use bootstrapping and random feature selection to build an ensemble of trees [72]. Hereby, the goal is to be robust against overfitting and to make better predictions. They use a technique called bagging, bootstrap aggregation, where multiple sub-datasets B_i are created by sampling with replacement from the original dataset, on which the DTs are then trained. This results in more uncorrelated trees than if the same dataset is used multiple times to build the forest, as well as less overall variance of the model [73]. In addition, for each node in the tree, the features used to evaluate split quality are randomly selected to reduce the correlation between the trees, also known as feature bagging. To infer the label of a new data point, it is passed through the RF. The final prediction is the most predicted class of all trees, or in the case of regression, the average. These mentioned procedures are shown in Algorithm 2.

3. Random Survival Forests

Random Survival Forests (RSFs) extend the concept of Random Forests by incorporating right-censored survival data [74]. In general, they follow the concepts developed for RFs, but have two major modifications: First, a new splitting rule to maximize survival differences between two branches, and second, a new risk heuristic, the ensemble Cumulative Hazard Function (CHF) [75]. Algorithm 3 summarizes the key points. Since the dataset contains censored data, the splitting rule has to account for the censoring. To simplify this scenario let's assume without loss of generality that we are at the root node and the data is not bootstrapped and nominal [74]. We also define

$$\{t_i \mid e_i = 1, t_i < t_j, \forall i < j\} \tag{2.50}$$

as the m unique death times of all patients in the training dataset which are used in the calculation. Suppose x_{ij} is feature j of individual i that we want to use to evaluate the quality of the split. This yields the left and right children of the node, $L = \{X_i \mid x_{ij} \le c\}$ and $R = \{X_i \mid x_{ij} > c\}$. Let

$$\begin{aligned} Y_{j,L} &= \#\{T_i \ge t_j, x_{ij} \le c\}, \\ Y_{j,R} &= \#\{Ti \ge t_j, x_{ij} > c\}, \end{aligned} \tag{2.51}$$

represent the number of individuals who are alive at time t_j in the child nodes, and let $d_{j,L}, d_{j,R}$ denote the number of events occurring in the respective leaves at time t_j. By defining the total number of individuals alive in the parent node at time t_j as Y_j, and the total number of individuals at risk—where the event occurred by time t_j as d_j

$$\begin{aligned} Y_j &= Y_{j,L} + Y_{j,R}, \\ d_j &= d_{j,L} + d_{j,R}, \end{aligned} \tag{2.52}$$

Algorithm 2 Random Forest

Input: Training Data X, Number of Trees N_t, Number of Randomly Selected Features N_f
Output: Classification or Regression Prediction

```
1: procedure RF(X, $N_t$, $N_f$)
2:     Generate k bootstrapped datasets $\{B_i | i \in \{1, \dots, k\}\}$ from X.
3:     Train: Build $N_t$ trees using the bootstrapped datasets and $N_f$ randomly selected features at each node.
4:     Predict: Pass test data through random forest and aggregate (e.g., mean) results for regression or count frequency for classification.
5: end procedure
```

Algorithm 3 Random Survival Forest

Input: Training Data X, Number of Randomly Selected Features N_f

Output: Cumulative Hazard Function (CHF)

```
1: procedure RSF(X, $N_f$)
2:     Generate k bootstrapped datasets $\{B_i | i \in \{1, \ldots, k\}\}$ from X.
3:     Grow tree for each $B_i$. Randomly select $N_f$ features at each node. Split on feature that maximizes
       survival difference between leaf nodes.
4:     Grow tree to full size until terminal node contains a minimum of $d_0 > 0$ unique deaths.
5:     Calculate the CHF for each tree. Average to derive the ensemble CHF.
6: end procedure
```

the log-rank split statistic can be expressed as follows

$$L(X, c) = \frac{\sum_{j=1}^{m} \left(d_{j,L} - Y_{j,L} \frac{d_j}{Y_j} \right)}{\sqrt{\sum_{j=1}^{m} \frac{Y_{j,L}}{Y_j} \left(1 - \frac{Y_{j,L}}{Y_j} \right) \left(\frac{Y_j - d_j}{Y_j - 1} \right) d_j}}. \tag{2.53}$$

The greater the magnitude of $L(X, c)$, the greater the survival difference between L and R, and thus, by definition, the better the split. Consequently, the optimal split is determined by finding the feature X^* and the split-value c^* that satisfies $\forall X, c : |L(X^*, c^*)| \geq |L(X, c)|$. The ensemble CHF is estimated through the following expression

$$H_h(t) = \sum_{t_{j,h} \leq t} \frac{d_{j,h}}{Y_{j,h}}, \tag{2.54}$$

where h represents terminal node to which an individual i is assigned when traversing the tree, and $t_{j,h}$ are the unique death times in h. The survival function estimate is given as

$$S_h(t) = \prod_{t_{j,h} \leq t} \left(1 - \frac{d_{j,h}}{Y_{j,h}} \right). \tag{2.55}$$

For prediction the RSF returns a mortality score which is defined as follows:

$$\mathcal{M}_i = \sum_{j=1}^{m} H_i(t_j). \tag{2.56}$$

The mortality score is scaled by the number of events, where $\mathcal{M}_i > \mathcal{M}_j$ represents a worse outcome for individual i compared to j. For example, if individual i has a mortality score of 200, this implies that if all other individuals had the same covariates, an average of 200 events is expected. Since this value can be interpreted as a measure of risk, it is used to calculate the Concordance Index.

4. Extreme Gradient Boosting

Extreme Gradient Boosting, also referred to as XGB, is a popular method known for its versatility in solving various ML problems. It is a tree ensemble method that uses additive training, where Decision Trees are built recursively, with subsequent trees improving on the predictions of previous trees [76]. Defining f to be the function that contains the tree's structure and outputs the terminal leaf scores this can be formulated as

$$\begin{aligned}
\hat{y}_i^{(0)} &= c \\
\hat{y}_i^{(1)} &= \tilde{f}_0(X_i) = \hat{y}_i^{(0)} + f_1(X_i) \\
&\vdots \\
\hat{y}_i^{(m)} &= \sum_{k=1}^{m} \tilde{f}_k(X_i) = \hat{y}_i^{(m-1)} + f_m(X_i),
\end{aligned} \tag{2.57}$$

with c being a random default prediction and $\hat{y}_i^{(k)}$ with $k \in \{1, \dots, m\}$ is the prediction of the k-th tree for data point i. Furthermore, the optimization goal is to minimize the residuals with respect to the following objective function:

$$\text{obj}^{(m)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(m)}) + \gamma\mathcal{T} + \frac{1}{2}\lambda \sum_{j=1}^{\mathcal{T}} w_j^2. \tag{2.58}$$

In this equation $l(\cdot,\cdot)$ is a custom loss function, which differs based on the objective at hand, $\mathcal{T}$ the number of terminal nodes in the tree, γ a factor encouraging pruning, λ the regularization strength and w_j the score vector associated with each terminal node. To accurately predict outcomes using survival data, special loss functions have to be employed to account for censoring such as the one based on the CPH model. Since there may not be a closed-form solution to this optimization problem, depending on the loss function, the second-order Taylor expansion is generally used to approximate it.

Minimizing this approximation with respect to w yields

$$w_j^* = -\frac{G_j}{H_j + \lambda}, \tag{2.59}$$

with G_j and H_j defined as

$$\begin{aligned}
G_j &= \sum_{i \in I_j} g_i, \\
H_j &= \sum_{i \in I_j} h_i,
\end{aligned} \tag{2.60}$$

where g_i is the first-order gradient of the loss function's Taylor expansion, h_i is its second-order gradient, and I_j denotes the set of indices of the data points assigned to the j-th leaf.

By then inserting w_j^* in the approximation of the loss function and performing some mathematical transformations, the best objective reduction obj* can be derived as

$$\text{obj}^* = -\frac{1}{2} \sum_{j=1}^{\mathcal{T}} \frac{G_j^2}{H_j + \lambda} + \gamma\mathcal{T}. \tag{2.61}$$

In this formulation a smaller score indicates a better tree structure. Lastly, a splitting heuristic is defined as follows:

$$\text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma. \tag{2.62}$$

This heuristic encodes the scores of both the new leaves and its parent, along with an additional regularization parameter γ that restricts the creation of new leaves if the gain is too small, implicitly pruning the tree.

5. XGBSE

XGBoost Survival Embeddings (XGBSE) is an approach that uses Extreme Gradient Boosting (XGB) in combination with a custom loss function accounting for censored data, to embed high-dimensional features in a lower-dimensional sparse space [77]. For each of the B trees used by XGB, denoted as $\mathrm{Tr}_{b\in B}$, a feature vector of zeros is defined, whose length is equal to the number of terminal nodes. Then, when an individual j with features X_j traverses through tree b and arrives at terminal node k, the k-th entry of the feature vector for that tree is set to 1:

$$\mathrm{Tr}_{b,j} = \mathrm{Tr}_b(X_j) = [0, 0, \dots, 1, 0, \dots, 0]^\top. \tag{2.63}$$

Now, all these one-hot-encoded vectors are concatenated into a single array for individual j, defined as:

$$\tilde{X}_j = \mathrm{Concat}(1, Tr_{1,j}, Tr_{2,j}, \dots, Tr_{B,j}). \tag{2.64}$$

Similarly, this is done for all other individuals in the dataset. These embeddings are utilized to predict the d_i/n_i term of the Kaplan-Meier formula with a logistic regression classifier. Therefore, this classifier is adjusted to exclude individuals who were censored before time t. The resulting set

$$N_u^t = \{j \mid t_j > t \text{ or } e_j = 1\}. \tag{2.65}$$

is then used to train the classifier. Generally, this is done for multiple user-defined discrete time points, where each time point has its own logistic regression classifier. Without loss of generality, let's consider the scenario for a single custom time point t. Logistic regression calculates the probability P of the binary outcome given the input feature $\tilde{X}$

$$P\left(e = 1 \,\middle|\, \tilde{X}\right) = \frac{1}{1 + \exp^{-\beta^T \tilde{X}}}, \tag{2.66}$$

where β is the vector of coefficients to be estimated, with its length equal to the number of elements in $\tilde{X}$. To find the optimal coefficients β^*, the negative log-likelihood must be minimized with respect to the model parameters β:

$$\beta^* = \arg\min_{\beta} \sum_{j \in N_u^t} \left[-e_j \log P(e_j|\tilde{X}_j) - (1 - e_j)\log(1 - P(e_j|\tilde{X}_j))\right]. \tag{2.67}$$

For multiple user-defined time points, a survival curve can be constructed for each individual by inputting the corresponding feature vector $\tilde{X}_j$ into the respective trained logistic regression models. Note that there is no closed-form solution for logistic regression, but β^* can be computed using methods such as gradient descent.

In general, XGBSE intends to use Extreme Gradient Boosted Trees to capture non-linearities and as a noise filter, since splits are only made on features with predictive power. Ultimately, the goal is a more robust and better calibrated estimate in combination with logistic regression. Furthermore, it can be extended to work with any classifier such as k-nearest-neighbors or even Neural Networks.

Deep Surv

Deep Surv is a method that models the interaction between patient features and treatment effectiveness in order to provide personalized treatment recommendations and support physicians in the decision-making process [21]. It has several desirable features, such as the fact that its internal Neural Network architecture can be adapted to each dataset, making it capable of capturing complex patterns. In addition, transfer learning is a viable option, where the weights of a trained NN can be loaded into another NN to potentially improve model performance on a different task. This is particularly interesting for the large variety of datasets available in hospitals, such as the one provided by UHE. In general, the scalability due to the efficient implementation on GPUs and the flexibility based on the architecture makes this method an attractive option.

It consists of a feed-forward NN with fully connected layers, non-linear activations and dropout. The network produces a scalar output value $\hat{h}_\Theta(\cdot)$, which represents the predicted risk function. The loss function is defined as the negative log partial likelihood

$$l(\Theta) = -\frac{1}{N_{E=1}} \sum_{i:E_i=1} \left(\hat{h}_\Theta(X_i) - \log \sum_{j:T_i>t} \exp^{\hat{h}_\Theta}(X_j) \right) + \lambda \, ||\Theta||_2^2 , \tag{2.68}$$

where $N_{E=1}$ is the number of events that occurred, Θ are the network weights, and λ is the regularization strength. Compared to Neural Networks in general, the main difference lies in the special loss function, which is specifically designed for censoring. This in turn allows any NN architecture to be used. Hence, the model is very flexible and adaptable.

Unlike the CPH where treatment interaction terms have to be explicitly added, it does not require pre-specification of these to calculate the recommender function

$$\mathrm{rec}_{l,k} = \log \left(\frac{\lambda(T; X_i | \tau = z)}{\lambda(T; X_i | \tau = k)} \right) = \hat{h}_{\Theta_z}(X_i) - \hat{h}_{\Theta_k}(X_i), \tag{2.69}$$

under the assumption that each treatment z of l treatment groups $\tau \in \{0, 1, \ldots, l-1\}$ has an independent hazard function with baseline hazard $\lambda_0(t)$

$$\lambda(t; X_i | \tau = z) = \lambda_0(t) \exp^{\hat{h}_{\Theta_z}(X_i)} . \tag{2.70}$$

This approach allows for personalized prediction by passing the patients feature vector through the Neural Network first in treatment group i and then in treatment group j. In the case for the event variable being death a $rec_{ij} > 0$ means that the patient has a higher risk-of-death in treatment group i compared to treatment group j and vice versa.

2.5 Explainability

Machine Learning models often seem like enigmatic black boxes that produce impressive results but offer little insight into how they got there. This lack of transparency raises several issues, including bias and accountability. However, by demystifying the decision-making process, numerous benefits can be unlocked, such as uncovering errors in the algorithm, detecting bias in the data, improving models, and even discovering knowledge, which ultimately leads to greater accountability. In the following section, we delve into one such method, namely Shapley values.

SHAP

SHapley Additive exPlanations (SHAP) is a game-theoretic method for explaining the output of mathematical models [78]. The game is replicating the model's outcome while the players are its features. SHAP values or Shapley values quantify the contribution of each player (feature) to the overall outcome of the game (model) [79]. It treats the model as a black box—so only the predictions are considered, nothing else—and employs different tools for explainability. A Shapley value is a measure for the importance of feature i given model f, traditionally defined as

$$\mathrm{SHAP}_i = \sum_{S \subset F \setminus \{i\}} \frac{|S|(|F| - |S| - 1)!}{|F|!} \left[f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S) \right] . \tag{2.71}$$

In this equation F represents the set of all features, x_s are the values of the input features in set S, and $| \cdot |$ denotes the cardinality of a set. The Shapley value for feature i is a weighted sum over all possible subsets S of features without i, considering the differences in predictions between the full feature set $S \cup \{i\}$ and the feature set S alone. In essence, SHAP values are used to capture the marginal effect of a feature to the model's prediction as well as their interaction effects. Moreover, the SHAP python package provides a set of tools including visualizations for understanding individual and global feature importance, based on combining many local explanations [80]. Numerous extensions with various trade-offs have been proposed to work with a wide range of Machine Learning models, as well as approximations for faster inference. Thus, this method enables fairness analysis, model debugging, feature interaction analysis, model comparison and generally, interpretability.

In our work, we use the model-agnostic permutation explainer, which can compute Shapley values for any model by systematically iterating over an entire permutation of the features in both the forward and backward directions [81]. However, it should be noted that this approach is very expensive from a computational point of view, since the number of permutations grows exponentially with the number of features.

3 Methods

This Chapter covers the detailed settings of the methods used and developed in this thesis and motivates the design choices. The goal is to provide a thorough understanding of the implementation details and the means to reproduce the results.

3.1 Survival Analysis

We compare the performance of three different algorithms commonly used in survival analysis to predict an individuals' risk over time: Random Survival Forests, DeepSurv and XGBSE (see Section 4.4.1). Furthermore, we train each survival model on two subsets for two different tasks: predicting the risk of treatment failure and predicting the risk-of-death for an individual. The subsets are created from the dataset provided by UHE (see Section 4.1). One contains features up to a maximum of 400 missing values (N/As) and the other one up to 2532 missing values. They are created by applying the preprocessing steps described in Section 4.3.

The training set is used to perform a stratified 5-fold Cross Validation (CV). This CV procedure splits the training data into five folds and ensures that the distribution of events in each fold is representative of the entire dataset. One fold is then retained as the test set used for prediction, while the model is trained on the remaining folds. This process is repeated until each of the five folds has been used once as a test set. We use the prediction of the models to compute their average C-Index. Additionally, the average IBS between the 10th and 90th percentile of the time-to-event variable T_{train} is calculated in 30-day increments. In general, we use CV to get a more robust estimate and to validate our approach. After the 5-fold CV, we train the models on the entire train set and evaluate their prediction performance on the hold-out test set. We also set a seed to ensure reproducibility and that all models are trained on the same splits.

For all RSFs we set the maximum depth of the tree to 50, use 1000 tree estimators, fix the minimum samples in a leaf to 5 and the minimum samples to split an internal node to 10. This is to reduce overfitting.

In the case of XGBSE, we use the default parameters of the package models, but set the maximum depth of the tree to 50 as for RSF, and use a Cox objective function, optimizing for the negative partial log-likelihood of the Cox Proportional Hazards, similar to that defined in Section 2.4.2.

Finally, we use DeepSurv with a NN consisting of 2 hidden layers with 32 neurons each and perform a search for the learning rate that maximizes the C-Index. We then use this learning rate for training.

3.2 Unbalanced Optimal Transport on Clinical Data

We use UOT to measure the "similarity" between a group of patients to itself. Therefore, we create a cost matrix C based on the euclidean distance between patients' laboratory values. Its entries represent the pairwise distances between patients. To constrain self-assignment an thus an identity matrix of P^*, we add a high cost on the diagonal of C. We further set the regularization parameters for UOT to $\epsilon = 0.005$ and $\rho_1, \rho_2 = 0.05$. In this way, we relax the problem to obtain a smoother solution and allow the algorithm not to transport every sample, as described in Section 2.3.2. The marginals, represented as a and b are set as uniform distribution on the number of patients. However, these distributions can be adjusted in future studies if more emphasis needs to be placed on certain patients or areas. Finally, the UOT map P^* is computed using the generalized Sinkhorn-Knopp algorithm for matrix scaling. An entry of P^* indicates how much mass from patient i is moved to patient j. We use the transported mass from patient i to patient j as an indicator of how "similar" two patients are, assuming that the more mass transported, the greater their similarity. By using unbalanced UOT instead of standard OT, we aim to relax the problem so that not all mass has to be transported. This creates the flexibility to exclude certain patients from transport if the associated costs are too high, as measured by a large L^2 distance.

To evaluate this method, we use the UOT embedding on a binary classification task. In addition, to conduct a comprehensive assessment of the UOT embeddings' significance on the classifier's performance, we train new classifiers using two additional datasets: the preprocessed laboratory values and the euclidean distance matrix among patients. In particular, we compare the extent to which the rich information of the laboratory values is preserved by P^* and the euclidean embedding (L^2 embedding). For classification, we use an MLP with two hidden layers, each comprising 64 neurons.

3.2.1 Patient-Patient Similarity Graphs with Unbalanced Optimal Transport

The UOT map P^* defined in the previous Section can then be used to create a patient-patient similarity graph. Each node corresponds to a patient, and the edges between patients are defined based on P^*. We assign an edge between patient i and patient j if the transported mass P^*_{ij} is greater than a custom threshold. For comparison, we create a second graph with edges between patients (i, j) if the euclidean distance of their lab values is below a certain threshold, ensuring that a smaller L^2 distance indicates a higher degree of similarity. Furthermore, each node is initialized with the non-laboratory features (see Section 4.1) of the corresponding patient.

Generally, by using graphs, we aim to leverage knowledge within communities to make a tailored prediction for each individual. In this way, we aim to mitigate bias by considering only a small neighborhood of similar patients for tasks such as node prediction. Additionally, the regularization parameters of UOT enable us to tune the sparsity of the solution, which in turn can yield faster training on graphs. Moreover, this approach can be considered an unsupervised method for detecting edges within a dataset when they are not predefined. In the future, these graphs could be used to cluster similar patients, which is a useful tool in the discovery process.

In order to assess these methods, we evaluate them on a binary prediction task. We use multiple GNN architectures for both, the L^2 and the UOT graph for node classification. In total, we evaluate their performance based on 3 different graph architectures, namely a GCN, a GAT with 8 attention heads and a GIN (see Listings A.1, A.2 and A.3). Both the GCN and the GAT employ a hidden dimension of size 64. However, for the GIN, we choose a hidden dimension of size 128 due to our observations that it yields superior results. Each model consists of 2 hidden layers that aggregate information for each node from its 2-hop neighborhood. In addition, we incorporate dropout and batch normalization techniques into the model to ensure stable training and prevent overfitting.

To assess whether encoding information in graphs and using a graph structure is preferable, we compare it to baseline model, namely an MLP (see Listing A.4). The MLP consists of two hidden layers, with each layer consisting of 64 neurons and uses dropout and batch normalization. Laboratory values and all other relevant patient characteristics are used as input to the MLP. For all methods we set the dropout rate to 35%. To guarantee a fair comparison, we perform a 5-fold Cross Validation on the training dataset, ensuring that each model is trained on the same folds. Finally, we train the models on the entire training set and evaluate them on the test set.

4 Data and Results

4.1 Lung Cancer Dataset

The dataset used in this analysis consists of 4320 patients, all suffering from lung cancer. In addition, up to 404 characteristics were recorded for each of them.

Laboratory Values, ICD Codes, and Patient Characteristics

92 laboratory values (`LAB_###`) were collected prior to individual therapy, ranging from erythrocyte and granulocyte counts to potassium and urea levels. Furthermore, each patient is described with 90 binary International Classification of Disease (`ICD_###`) codes, where each value is a separate characteristic. `ICD_###` codes provide a global standard for recording and analysing health data, enabling consistent comparison and interpretation of mortality and morbidity information, while ensuring compatibility for various purposes beyond statistics [82]. Age and sex are also characteristics in the dataset. Information is also available on whether a patient is a smoker and how many pack-years—packs of cigarettes per day times years of smoking—they have.

OPS, Therapy, Diagnosis, and Cancer Types

The next most prominent features in the dataset are the Operation and Procedure Classification System (`OPS_###`) values. In this dataset 51 binary features are used to indicate which operational procedures were performed. Generally, it is important to note that the dataset does not encompass all existing `OPS_###` categories. In addition, 31 dichotomous features `ctx_###` describe which type of intravenous therapy and another 17 binary features with prefix name `rezept_###` indicate which oral drugs were administered. Besides, 12 different binary features depict the histological diagnosis (`Histology_###`) of lung cancer, ranging from adenocarcinoma to glioblastoma and squamous cell carcinoma, to name a few. Furthermore, 9 dichotomous covariables (`Metastasis_###`) indicate whether a metastasis has occurred in a certain part of the body, for example in organs such as the liver, the lymph nodes, the brain or in the bones.

Moreover, 8 numerical clinical characteristics (`Clinical_###`) representing different body measurements, including arterial blood oxygen saturation, body temperature, heart rate, systolic blood pressure, diastolic blood pressure, height, weight and Body Mass Index (BMI), were recorded 2 weeks before treatment. Lung cancers (`Cancertype_###`) are classified into NSCLC (Non-Small Cell Lung Cancer) and SCLC (Small Cell Lung Cancer). Additionally, only sarcomas—which arise from the malignant transformation of connective tissue cells—were observed as secondary malignancies within the dataset.

Cancer Staging and Body Composition

The categorical feature `TNM_Stage`—following the internationally recognized TNM classification of malignant tumors—has 5 possible outcomes: 0 indicates no cancer, 1 represents small cancer that has not spread, 2 stands for cancer that has grown but not spread, 3 suggests larger cancer and possible spread to surrounding tissue, and 4 implies cancer that has spread from its site of origin to at least one other tissue. `TNM_T` describes the size of the tumor from 1 (small) to 4 (large), `TNM_N` indicates whether the cancer was found in 0 (none) to 3 (many) lymph nodes , and `TNM_M` signifies whether metastasis occurred (1) or not (0). There are a total of six different numerical Body Composition Analysis (`BCA_###`) characteristics, such as muscle and bone volume, which provide information about the patient's body composition. These measurements, derived from CT images, were taken a maximum of 2 months prior to treatment, using an automated workflow.

Genes and Biomarkers

The ECOG (Eastern Cooperative Oncology Group performance status) feature is a scale of functional ability ranging from 0 to 5, where 0 means full ability, 1-3 increasingly limited functional ability, 4 total disability, and 5 death [83]. Another feature is MSI, which stands for microsatellite instability. Microsatellites are short, repetitive DNA sequences within the genome that encode information such as whether or not two individuals are closely related, and the instability results from propagated DNA replication errors. Cancers with high MSI have been shown to have a better prognosis and response to immunotherapy [84]. Another binary feature present in the dataset is the Epidermal Growth Factor Receptor (EGFR), which is assigned a value of one if it is present. EGFR is a transmembrane protein that regulates epithelial tissue development and homeostasis. It is a driving force in lung cancer tumorigenesis, and is used as a biomarker for tumor resistance [85]. KRAS (Kirsten rat sarcoma viral oncogene homologue) is a gene—encoded as a binary feature in the dataset—that is considered to be a common gene driver for human cancers [86]. It also plays an active role in current research for therapies targeting KRAS mutations in KRAS-driven cancers. The binary feature, called TP53, is a gene that encodes tumor protein 53, which regulates cell division by preventing the cell from multiplying or growing too quickly [87]. The covariate Programmed Death-Ligand 1 Tumor Proportion Score (PD-L1_TPS) is numeric and encodes the Tumor Proportion Score (TPS) of the PD-L1 transmembrane protein, which is highly expressed in a variety of malignancies, including lung cancer [88, 89]. The PD-L1_TPS is commonly used as a selection marker for immunotherapy [90]

Outcome Variables

The outcome variables of the dataset are death_observed, i.e., whether a patient died or was censored. This is paired with the OS (Overall Survival) variable, which indicates how many days an individual was in the cohort before they died or got censored. Another potential outcome variable is whether treatment failure has occurred encoded as TTF_Event. It is 0 if the patient was censored or 1 if the patient died or received another treatment. This variable is paired with the Time to Treatment Failure variable TTF, which indicates the number of days a patient received this one treatment. Traditionally, observed death has been the most commonly used outcome for time-to-event analysis of patients. However, a new and growing trend is the use of treatment failure as an alternative measure. This analysis examines both. A summary of all features with their definition, number of features, and data type is displayed in the supplementary Table A.3.

4.2 Descriptive Analysis

The goal of this Section is to provide a first impression of the dataset, how features are distributed and what questions are worth evaluating.

As with most real-world datasets, the data is not perfect because many factors affect the data generation process, such as the quality of laboratory equipment, the honesty of patients in surveys, human error in data collection, N/As, and more. Therefore, it is necessary to find these errors and mitigate their impact on the overall performance of the model. This is done by first obtaining an overview of the dataset, including its statistical parameters for each characteristic. In addition, the data is examined for coherence by generating plots showing the relationship between outcome variables and selected covariates.

In total 1744 female and 2576 male patients are in the dataset. Death was observed for 2753 individuals, while the remaining 1567 were censored. Notice that the TTF_Event has more observations where the event took place (TTF_event = 1) than the outcome variable death_observed (see Table 4.1). This is coherent, because a failure of treatment is not necessarily to be set equal with death. Examining the histograms depicting the outcome variables OS and TTF in Figure 4.1, it is noticeable that most of the observations are concentrated within the first 1000 days and exhibit a rapid exponential decline as time progresses. From a medical perspective, it is to be expected that the likelihood of survival declines with time. However, this also depends on several other factors, such as when treatment began and the stage of the tumor. In addition, to check for consistency, potential outcome variables of interest such as Overall

	death_observed	TTF_Event	gender_female
0	1567	1215	2576
1	2753	3105	1744

Table 4.1 Summary of outcome variables for the lung cancer dataset provided by University Hospital Essen. In total 1567 patients were censored in the dataset, whereas 2753 died. Additionally, for 3105 out of 4320 patients treatment was not successfull after a certain time and 1215 were censored. The dataset contains 60% men and 40% women.

Survival (`OS`) and whether a death was observed are shown in Figure 4.2. This graph shows a patient's Overall Survival on the y-axis, categorized by varying `ECOG` scores, and is color-coded to distinguish between patient death and censoring. The shapes are the kernel density estimates, and the width of each bin has been scaled by the number of observations in that bin. Interpretation shows that lung cancer patients with a high `ECOG` score, i.e. with low functional capacity, have a lower `OS` than patients with a lower score. This observation aligns with the medical point of view.

Additionally, of the 4320 patients present in the dataset, there exist both categorical features and numerical features with many N/As (see Figures A.1 and A.2). To get a more detailed view of all N/As in the dataset, Tables A.5 and A.4 are to be consulted.

Generally, this poses the question on how to incorporate such features into the models. Since most of the survival models do not work with missing values, the features containing missing values have to be either dropped or processed in order to keep them. Simply dropping them from the model removes potential valuable information for the model. Various techniques exist for coping with N/As. One of them are polar encodings, which ensure equidistance between missing and non-missing values, providing an approach to incorporate missingness in attributes that aims to let Decision Trees choose how missing values should be split [91]. Another option are imputation techniques, where missing values are replaced by a statistical summary derived from the available non-missing values of a feature. The choice of technique depends on the context and requirements. In addition, a reasonable threshold for their presence in a column must be

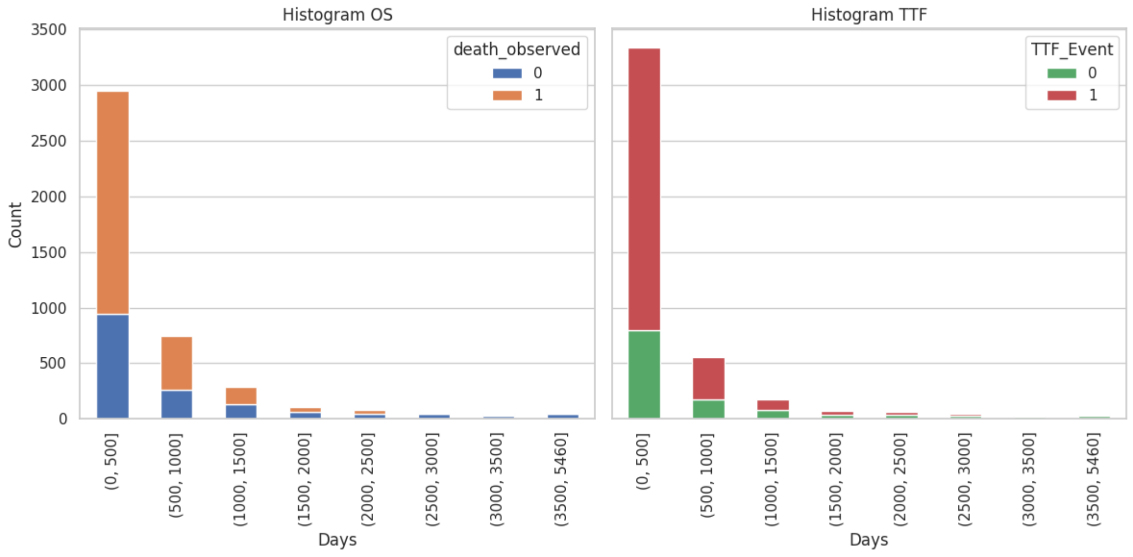

Figure 4.1 Histograms of the outcome variables Overall Survival and Time to Treatment Failure. Both are color-coded with their corresponding binary event variable. Most patients survived up to 500 days and were then either censored, which means `death_observed` is set to 0 or died. The `OS` in days decreases exponentially with time. For the `TTF` histogram similar behaviour is shown. `TTF` has more early events occurred (`TTF_Event` = 1) compared to `death_observed`. This observation is consistent because treatment failure does not necessarily mean the death of a patient, and it is reasonable to assume that patients may have undergone multiple treatments.

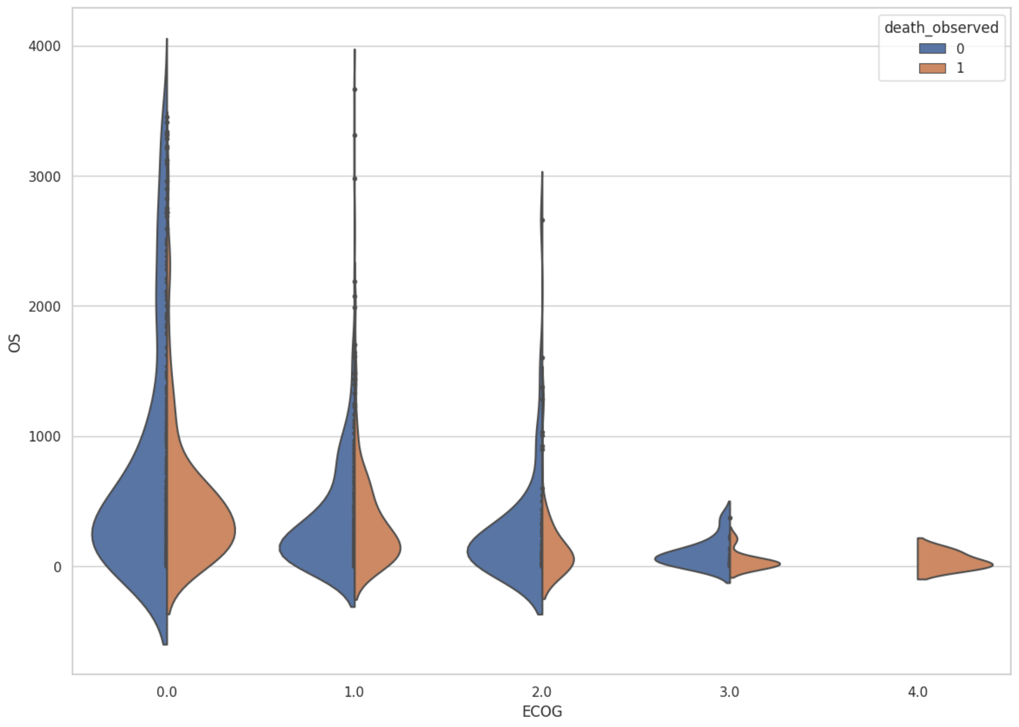

Figure 4.2 Overall Survival of patients in days, depending on their ECOG score and whether they were censored or died. The violins are the kernel density estimates, and the width of each bin was scaled by the number of observations in that bin. ECOG is a score used to assess functional ability of cancer patients. The higher the score, the less able patients are to take care of themselves, corresponding to advanced cancer progression. Therefore, it is medically expected that patients with a higher ECOG score will have a lower Overall Survival, which is confirmed by the plot. Note, that for an ECOG score of 4, no individuals were censored and all died within a short period of time. The results show that patients with lower functional ability (higher ECOG scores) tend to have shorter survival.

chosen. This is a critical consideration because an excessive number of missing values can be a source of bias and distortion in the overall analysis.

This descriptive overview serves to provide a first impression of the outcome variables in the dataset, including their distributions. Furthermore, it highlights potential shortcomings of the covariates, such as the presence of missing values and class imbalance. For a comprehensive summary of statistical parameters and variable definitions, please refer to Tables A.3, A.4 and A.5.

4.3 Data Preprocessing

This Section describes the data preprocessing steps that are carried out prior to feeding the data into the models. Note that this preprocessing step is the same for all models. Since we are working with a subset that includes only lung cancer patients from a larger pan-cancer dataset of UHE, we first remove the `Cancer_###` features. However, it is important to note that this variable remains of interest for future analyses, as it can be easily integrated into models and the preprocessing pipeline.

Secondly, all columns containing only N/As or only zeros are removed, as these columns do not provide any meaningful information. Next, we apply a filter to all categorical feature columns with binary outcomes: `OPS_###`, `ICD_###`, `Cancertype_###`, `Metastasis_###`, `rezept_###`, `ctx_###`, and `Histology_###.` If one of these columns has less than 100 entries set to 1, which corresponds to about 2% of the individuals in the dataset, it is removed. Following a discussion with our domain scientists, we then combine the Body Composition Analysis (BCA) values in the following way to create new features:

$$\begin{aligned}
\texttt{BCA\_muscle\_bone} &= \frac{\texttt{BCA\_muscle}}{\texttt{BCA\_bone}},\\
\texttt{BCA\_imat\_tat} &= \frac{\texttt{BCA\_imat}}{\texttt{BCA\_tat}},\\
\texttt{BCA\_imat\_muscle} &= \frac{\texttt{BCA\_imat}}{\texttt{BCA\_muscle}},\\
\texttt{BCA\_sat\_vat} &= \frac{\texttt{BCA\_sat}}{\texttt{BCA\_vat}}.
\end{aligned} \tag{4.1}$$

This step is taken based on previous research findings of UHE. Subsequently, the original `BCA_###` columns are dropped. After this, all columns with more N/As than a custom amount will be dropped. In our analysis, this amount is N/A ≤ 400, which is about 10% of the data, or N/A ≤ 2532, where we try to keep more features in the dataset, including `BCA_###` values, all of which have a total of 2532 missing values. This results in a total of 83 features for N/A $\leq$ 400 and 130 features for N/A $\leq$ 2532. Removing features with too many N/As serves several purposes: reducing the dimensionality of the dataset, enhancing the computational speed of models, and improving the interpretability of models by reducing the potential noise introduced by imputation.

A filter is then applied to all laboratory values, which checks if the values are more than 4 times a standard deviation away from the columns mean. These values are then set to N/A to be imputed later on. Thus, we lower the risk of biasing our models, because outliers can have a big effect on the training and the weights. Based on the objective at hand, the dataset can be partitioned into male and female subsets, and separate training and test sets are created. Employing an 80-20 split, we allocate 80% of the samples for training and reserve the remaining 20% for testing. Within the training set, we then use 80% to create the final training set and 20% for validation. The variables `death_observed`, `OS`, `TTF_Event`, and `TTF` serve as outcome variables.

Since the data is not longitudinal, potential data leakage can be easily avoided by imputing N/As separately for each set. Imputation is performed by replacing all N/As of a column with the respective column mean. The numerical features of the training, validation, and test datasets are then standardized by subtracting the mean and scaling to unit variance. The mean and variance used for scaling are derived from the training set features to ensure consistent scaling across all sets.

4.4 Survival Analysis

In this Section, we evaluate different survival models for predicting the risk-of-death or treatment failure. After identifying the differences in performance, we evaluate one of them using Shapley values and highlight potential use cases for clinicians, such as providing an interpretable representation of risk prediction. In addition, we identify features with high predictive power that can offer further insight into lung cancer in general and create new research opportunities.

4.4.1 Model Comparison

To ensure a consistent comparison among these methods, we used the same training, test, and validation sets for all models. Across all the methods described in Section 3.1, there is a consistent level of performance when using `death_observed` as the event variable, as shown in Table 4.2. All models are able to correctly rank the risk for individuals with over 70% accuracy indicated by a Concordance Index greater than 0.7. For N/A $\leq$ 400 DeepSurv is the best performing model with a C-Index of 0.707, whereas for N/A $\leq$ 2532 XGBSE performs best with a Concordance Index of 0.722. This showcases that model performance is dependent on the number of imputed missing values. Comparing the Integrated Brier Scores, RSFs achieve the best result on the test set as well as for the 5-fold CV.

Table A.1 shows that with treatment failure as the event variable, a RSF is the best performing model for N/A $\leq$ 400 with a Concordance Index of 69.4%. Contrarily, for N/A $\leq$ 2532 DeepSurv achieves the best result as measured by a Concordance Index of 0.707 and IBS of 0.162. This underscores that for time-to-event analysis, which typically takes only survival as the event, treatment failure is an alternative worth evaluating.

Additionally, across both datasets RSFs perform the best according to the 5-CV IBS. However, it is important to note that there is no uniquely superior model, as model performance tends to differ as measured by the C-Index depending on the dataset and specific event variable being analyzed. It strikes that in general, the more missing values we allow a feature to have, and to be replaced by the corresponding column mean, the better the overall performance of the models across all methods. A reason for that can be that effectively more information is kept in the dataset, leading to better results due to more variability. However, additional bias can be introduced through imputation which can inflate the performance metric. Hence, it is essential to evaluate the robustness and explainability of the models.

It is also important to mention, that training time of these models differs. One the one hand, training for XGBSE and RSFs takes longer compared to DeepSurv, but can be accelerated by parallelizing the compu-

Model	C-Index	IBS	5-CV C-Index	5-CV IBS
N/A $\leq$ 400				
DeepSurv	**0.707**	0.178	0.702 ± 0.019	0.179 ± 0.009
RSF	0.704	**0.177**	0.704 ± 0.023	**0.178 ± 0.009**
XGBSE	0.705	0.181	**0.705 ± 0.024**	0.181 ± 0.008
N/A $\leq$ 2532				
DeepSurv	0.714	**0.175**	0.701 ± 0.016	0.180 ± 0.009
RSF	0.717	**0.175**	0.723 ± 0.015	**0.173 ± 0.008**
XGBSE	**0.722**	0.178	**0.724 ± 0.022**	0.178 ± 0.007

Table 4.2 Survival model comparison measured by Concordance Index and Integrated Brier Score for `death_observed` as event variable. N/A smaller or equal than 400 indicates that only columns with at most 400 N/A's were kept to be imputed, while removing columns with more. Similarly, this was done for columns with up to 2532 N/A's. All model achieve comparable results and there is no single model which outperforms the others. Model performance generally increases when keeping features with many N/A values. However, it is crucial to treat this observation with caution, as it is influenced by the extensive imputation that took place.

tation of the trees. On the other hand, DeepSurv requires more considerations about network architecture and learning rate, and does not work out-of-the-box like the other two methods. We also found that for the NN of DeepSurv defined in Section 3.1, increasing the complexity by adding more layers or increasing the capacity by adding more nodes per layer did not improve the overall performance as measured by the C-Index.

4.4.2 Explainability with Random Survival Forests

After showing that RSFs have a similar or better performance than comparable state-of-the-art methods, we use their Shapley values to find important features which drive the models decision-making process. Our aim is to gain a better medical understanding of the features that are associated with a higher risk-of-event in lung cancer.

We create a total of six datasets for two N/A thresholds using the preprocessing steps outlined in Section 4.3 by keeping the entire dataset as well as splitting it into male and female individuals. These subsets are used to find gender specific differences in feature importance. All subsets are then partitioned into a train and test set.

All models have a comparable performance for the event variables `death_observed` and `TTF_Event` (see Tables 4.3 and A.2). It is striking that the RSF containing only male individuals performs the best for both events as measured by Concordance Index and IBS. This could be because male individuals have a higher sample size (see Table 4.1), so that their features have a higher variability and yield a more robust estimate. With `death_observed` as the event variable, all models achieve a C-Index greater than 0.7, indicating that the models are able to correctly rank individuals over 70% of the time. Consistently, we observe an improvement in model performance when a larger number of features are retained in the dataset.

Overall, the RSFs show good predictive power across all datasets, confirming their effectiveness. However, as their C-Index and IBS shows, there is still room for refinement, which could be achieved through more feature engineering, fine-tuning of hyperparameters, using subsets of the dataset, creating new ensemble methods and improving overall data quality.

Model	C-Index	IBS	5-CV C-Index	5-CV IBS
N/A $\leq$ 400				
All	0.704	0.175	**0.706 ± 0.013**	0.176 ± 0.007
Female	0.704	0.172	0.676 ± 0.012	0.188 ± 0.010
Male	**0.720**	**0.169**	0.703 ± 0.011	**0.171 ± 0.008**
N/A $\leq$ 2532				
All	0.716	0.175	**0.724 ± 0.015**	0.171 ± 0.008
Female	0.720	0.170	0.705 ± 0.019	0.184 ± 0.008
Male	**0.737**	**0.163**	0.721 ± 0.017	**0.166 ± 0.006**

Table 4.3 Random Survival Forest results with `death_observed` as event variable on the entire dataset, as well as on subsets filtered for male and female individuals. N/A smaller or equal than 400 indicates that only columns with at most 400 N/A's were kept to be imputed, while removing columns with more. Similarly, this was done for columns with up to 2532 N/A's. In general, we observe that model performance tends to improve as more columns are retained in the dataset. However, it is crucial to treat this observation with caution, as it is influenced by the extensive imputation that took place. The RSF on the male subset performs the best measured by the C-Index and IBS.

Shapley Value Analysis

Using the prediction of the RSFs on the test set, we calculate the SHAP values of the features for each individual with a permutation explainer. The magnitude of a SHAP value indicates to what extend a feature influences the event risk for an individual. Furthermore, a positive sign of the SHAP value indicates that the event risk is increased, whereas a negative SHAP value means a decreased risk. In the following analysis, each dot represents one person in the test set, with the corresponding feature values displayed on a continuous color scale. If a feature has only two colors, it implies that this feature has only binary entries, where the ones are color-coded red and the zeros are in blue.

A physician can now use these values to find relevant features that have a positive or negative impact on the patient's predicted risk. In addition, interaction effects between covariates can be identified based on the model's predicted risk score, providing valuable insight.

1. Feature Importance Comparison between Male and Female Subset

Figure 4.3 shows the top 20 features ranked by their influence for two RSFs, one with the male subset and the other with the female subset. `death_observed` was used as the event variable and N/A ≤ 400. Both models share many of the top 20 most important features, such as the laboratory values of `granulocytes`, `erythrocytes` and `hemoglobin`, but not all, like patient `age`, `MCV` levels or `GGT` values.

For the model containing only male individuals, `LDH` has the largest influence on the predicted risk, while for females it is the `lymphocyte` percentage in the blood. For both it is medically intuitive that they are important for risk-of-event prediction since `LDH` is an enzyme that is released during tissue damage and `lymphocytes` are known to play a role in the immune systems defense against diseases. In each case, there is no trend in whether high or low values indicate an increased or decreased risk-of-event. This can be seen as an indicator that more detailed patient level analysis is required. In addition, all individuals who received `Cisplatin` as a treatment (red color) have a negative SHAP value and thus it decreases the risk of an death, while those who did not receive this treatment have an increased risk-of-event (blue color). However, men treated with `Carboplatin` have a higher risk-of-death than those who did not receive it. At first glance, this seems counterintuitive, since one would expect the treatment to improve a patient's condition. This behavior could be explained by potential confounders, other medical decisions that influenced this treatment that are not encoded in the dataset, or by insufficient predictive power of the model. This analysis aims to compare feature importance between genders and to offer physicians insights for potential future research.

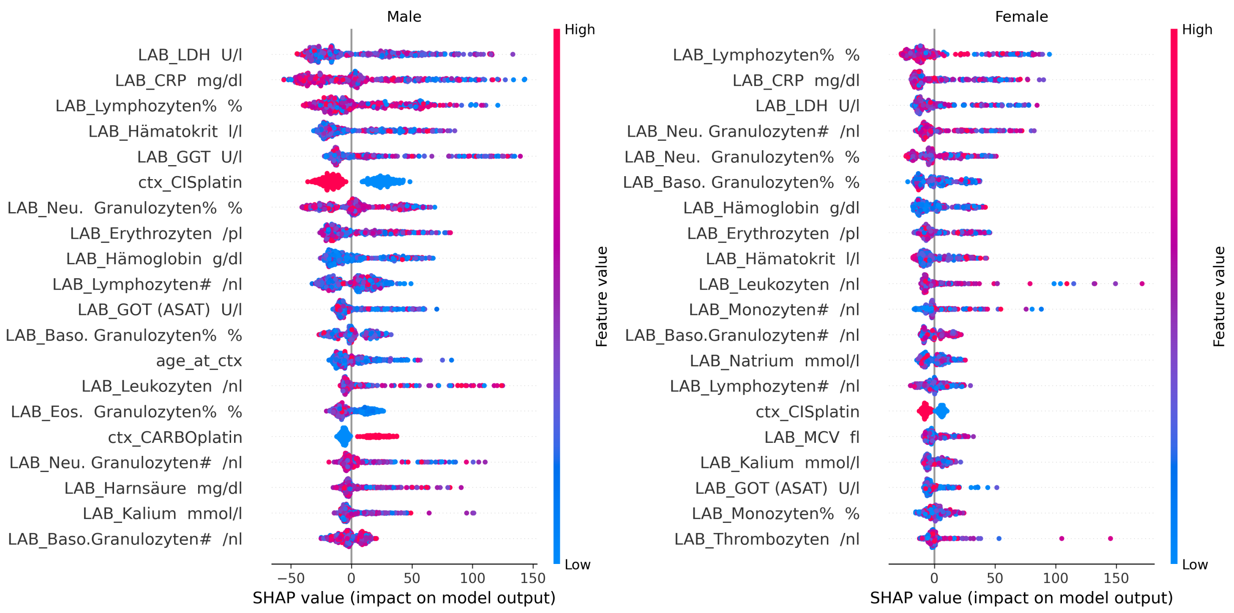

Figure 4.3 Top 20 most important features, as measured by SHAP values, for a Random Survival Forest based on male and female data with `death_observed` as the event variable. Every point is a feature of a patient from the test set. While the ranking of importance of features differs between the male and female models, there is considerable overlap, as many important features appear in both models.

2. Feature Importance Comparison of Complete Dataset

The subsequent in-depth analysis focuses on a RSF that was trained on the entire dataset with the event variable `death_observed` and N/A ≤ 400. Since this dataset is larger, the predictions also become more robust. In this way, an understanding of feature importance across `genders` can be gained and how they relate to the risk-of-death from lung cancer.

Similar to the previous analysis, the laboratory values of `LDH` and `lymphocytes` are the most important and third most important features. However, there is no trend if a high or a low feature value increases the predicted risk. This may also mean that a more sophisticated examination of these characteristics is required. For the second most important feature, the `CRP` value, however, a clear trend is evident. The color gradient for `CRP` changes slowly from light blue to dark red, indicating that a low `CRP` value lowers the risk-of-death (negative SHAP value), while a high `CRP` value increases the risk-of-death (positive SHAP value). To put it in a medical context, `CRP` has long been regarded as a minimal invasive measure of an ongoing inflammatory response [92]. A high value indicates a strong ongoing response and is therefore coherent with a higher risk-of-death, which further validates our model.

Laboratory values of `monocytes` express a similar behaviour. For an increasing amount of `monocytes` in the blood the SHAP value rises from negative (decreased event risk) to positive (increased event risk). Medically, it is to be expected that `monocytes` are among essential features, as they have proven to be important regulators of cancer progression and development as described in Olingy, Dinh, and Hedrick [93]. However, for further investigation, it is important to consider confounding factors in this context, as monocytes are a subclass of leukocytes, which are also a feature of the dataset. This is just one example of a general difficulty of this dataset, as many features are potentially correlated. It underscores the importance of considering potential interactions between variables during interpretation and collaborating with physicians towards this objective.

Another important observation is that there are many individuals with low `leukocytes` per nanoliter blood values who have a very high positive impact on the models predicted risk, thus an increased risk-of-death. To put this in the medical perspective, a low number of `leukocytes` means fewer `leukocytes` doing their crucial job of immune response, which is vital for cancer patients.

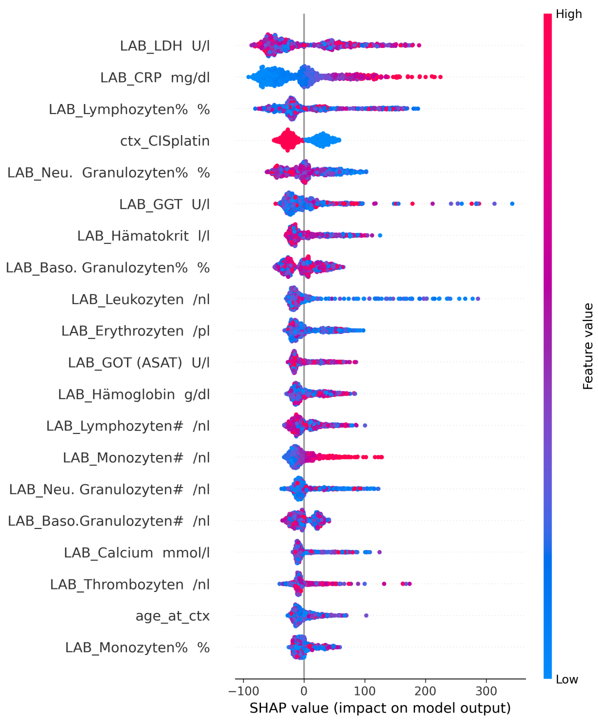

Figure 4.4 Top 20 most important features, as measured by SHAP values, for a Random Survival Forest with `death_observed` as the outcome variable.

Overall, this stresses the importance of interpretability of a model's predictions, as this can increase confidence, serve as a coherence check, and reveal new insights. Nonetheless, it is essential to acknowledge potential challenges, such as confounding and model limitations.

2.1 Feature Interaction Analysis

In the following the effects between different features are further examined. Figure 4.5 shows the relationship between the SHAP values of the standardized `age` of an individual with a color indicating if the person is a female (red) or male (blue). Generally, a low `age` decreases the risk-of-death, whereas for a higher `age` the event risk increases strongly. Examining the differences in event risk between `genders` reveals a similar risk behavior with increasing `age`.

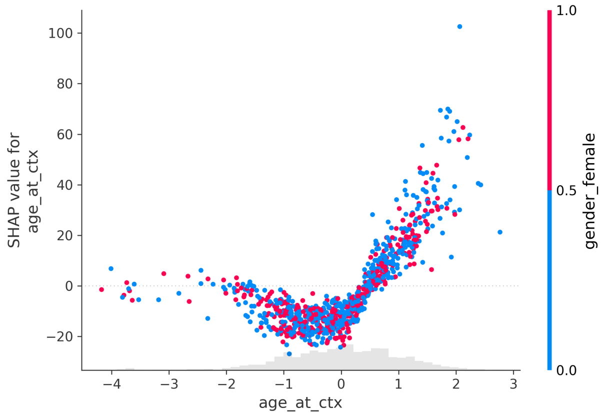

Figure 4.5 SHAP dependency plot of `age` color-coded with gender. As `age` increases, so does the risk-of-death for a patient. There is no clear trend as to whether this is different for male or female patients.

We also investigated if the two most important variables `CRP` and `LDH` for the RSF exhibit differences across `genders`. We find no clear indications that the risk value for these features varies significantly (see Figure 4.6). Additionally, although not depicted in this plot, our evaluation reveals that no trend across different `age` groups is apparent for both features.

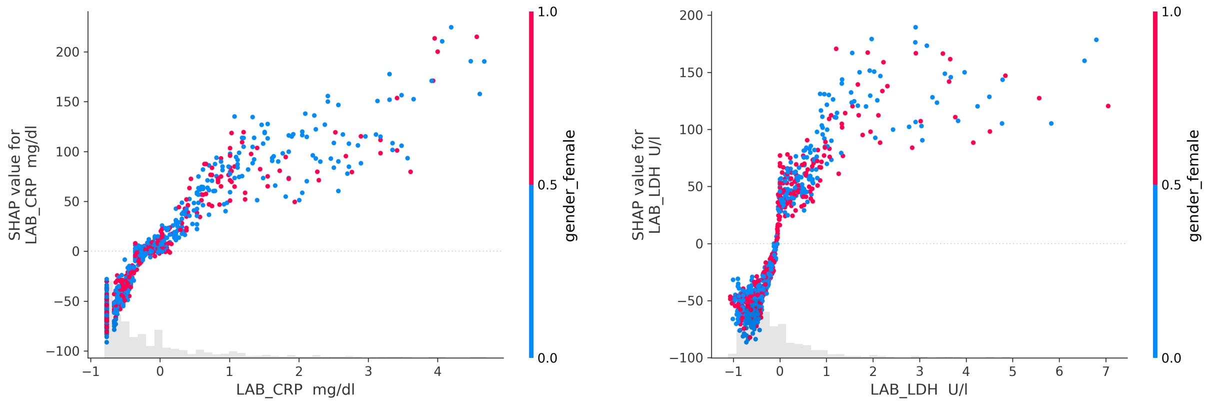

Figure 4.6 SHAP dependency plot for `CRP` and `LDH` lab values. Higher levels of `CRP` or `LDH` lead to an increased risk-of-death, while low levels decrease the risk. There is no difference between the sexes.

Furthermore, we find strong interactions between the laboratory `MCHC` values with `LDH` values as well as `CRP` values with `neutrophil granulocyte` counts per nanoliter blood. Figure 4.7 shows the dependencies between them. For low as well as high `MCHC` values the event risk differs across individuals. However, when overlaying it with the `LDH` values it becomes apparent, that all individuals who got a high SHAP score assigned to their `MCHC` value, exhibit a high `LDH` score and for those with a low score a low

LDH. Additionally, upon analysing the relationship between CRP values with `neutrophil granulocyte` counts, we observed distinct patterns. Initially, lower CRP values are associated with a general decrease in the risk-of-event. However, within the low CRP range, individuals with high `neutrophil granulocyte` counts had a smaller reduction in the risk-of-event compared to those with low counts. Conversely, higher CRP values are associated with an increased risk-of-event, with a greater increase observed in individuals with lower `neutrophil granulocyte` counts.

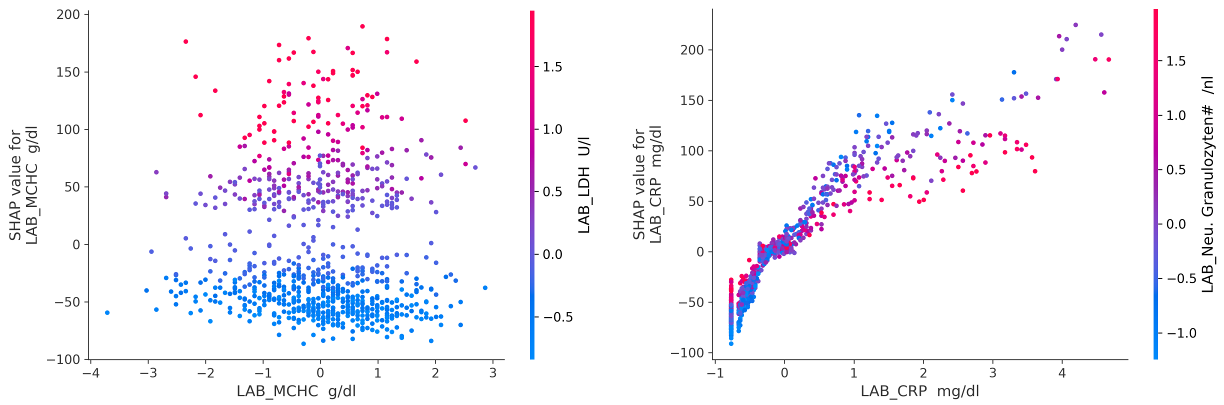

Figure 4.7 SHAP dependency plot for MCHC and CRP, color-coded by the most variable feature. The risk-of-death is evenly distributed for all MCHC values, but when color-coded with LDH, it shows that all individuals with a high SHAP value for MCHC also have a high LDH value. This can be an indicator of a potential interaction or confounding. For CRP an interaction with `neutrophil granulocyte` counts can be observed.

The above examples are intended to illustrate the power of such an analysis. In addition, there are numerous features with many more interactions in the dataset that cannot be mentioned here as this is beyond the scope of this thesis. It is left to the physicians at UHE, with whom we are working hand in hand, to further evaluate the results and identify potential interactions worthy of separate studies in the future.

3. Feature Importance for other Datasets and Event Variables

When using `death_observed` as the event variable with N/A $\leq$ 2532 we found different values being among the top 20 most important for a RSF risk prediction. This is to be expected since we now effectively incorporate more features and thus more information. As before LDH, CRP and `lymphocytes` features are still the most important, but they are followed by a binary feature for liver metastasis. The SHAP values indicate that for people who have a metastasis in the liver, an increased risk-of-death is to be expected which aligns with the medical intuition. Furthermore, other metastasis features in areas such as the bones or brain are among the top 20. However, even if the model predictions are consistent with medical expectations, it is important to treat these results with caution because many values were imputed.

For `TTF_Event` as the outcome variable, we generally observe more treatments under the 20 most important features such as `Cisplatin`, `Carboplatin` and `Vinorelbin`. This observation is of great scientific interest because it shows that the model is able to capture the importance of different treatments regarding treatment failure. In addition, LDH, CRP and `lymphocytes` are among the top 4 features for the model with N/A $\leq$ 400 and N/A $\leq$ 2532. For the latter one, we also find the feature `TNM_M` and `TNM_Stage` among the top 20. Both their SHAP values align with the medical expectation that a higher value indicates a higher risk of treatment failure for a patient, as it means that the patient was already in a medically worse condition before treatment began.

4. Summary

All in all the models findings align with initial medical expectations, but there are several factors that need to be considered. First, the data is imbalanced, for example the amount of people who received certain chemotherapy treatments differs, as shown in Table A.5. This can bias the model and limit generalization ability. Second, each patient receives treatment tailored to their individual needs, and factors that played a role in the physician's decision to administer a particular drug may not be included in the dataset. In addition, potential confounding factors may not have been taken into account, and association between features does not necessarily imply causation.

Nevertheless, such models can assist clinical decision-making, in order to reduce a patients overall risk, such as death or treatment failure. Another potential use case is to apply these models to identify promising biomarkers that may indicate, for example, whether a patient is likely to develop cancer, or that could be used to tailor treatment for patients who already have the disease.

Ultimately, this approach may provide transparency to patients about how a physician arrived at a particular treatment decision, thereby creating greater trust and accountability. However, this new accountability could also have subsequent legal implications when it comes to why a physician did not provide a certain treatment to a patient even though the model would recommend it. All of these considerations must be taken into account when developing a final model for building trust between patients and physicians.

4.5 Unbalanced Optimal Transport on Clinical Data

In this Section, we evaluate the effectiveness of the methods developed in Section 3.2. In particular, we use a subset of the lung cancer dataset provided by UHE (see Section 4.1) based on all individuals who had a treatment failure event (`TTF_Event` = 1) at a given time `TTF`. By focusing on individuals who experienced treatment failure, prediction objectives become more meaningful for non-survival models, since for the censored individuals (`TTF_Event` = 0), their actual `TTF` may have been longer than the recorded value in the dataset. This results in a total of 3105 individuals in the subset.

For the binary classification problem, we define the outcome as follows: individuals who experienced a treatment failure in the first 189 days ($\texttt{TTF} \leq 189$), which accounts for 1562 patients, are labeled as class 1, while the remaining 1543 patients are labeled as class 2. We decided to use the median of `TTF` as the threshold for class creation, so that a default classifier would achieve an accuracy of 50%. The L^2 and UOT embeddings are then calculated based on this subset, with the standard preprocessing steps outlined in Section 4.3, where all columns with more than 400 missing values were dropped. This ultimately results in 28 laboratory features on the basis of which the embeddings are computed.

As shown in Table 4.4, the performance of the MLP for a 5-fold CV remains consistent across the different embeddings as input, in terms of accuracy and F1 score. The maximum accuracy reached is 65.73% for a baseline MLP which uses the standard preprocessed train data. It should also be noted that, for a fair comparison, the MLP only uses the 28 laboratory features of individuals from the UHE dataset as input, since the UOT and L^2 embeddings are also calculated based on them. The L^2 and UOT MLP classifiers achieve almost the same accuracy and F1 score as the baseline classifier. In general, the UOT classifier shows less variation in prediction accuracy and F1 score compared to the other two classifiers.

Overall, this indicates that predicting whether or not treatment failure will occur within approximately six months is a challenging task. However, the embeddings still contain enough information to produce comparable results. It is reasonable to expect some performance degradation for the MLP with L^2 and UOT embedding as input, since some information is effectively removed during the embedding process.

Method	**Accuracy**	**F1 Score**
MLP Classifier Baseline	**65.73 ± 1.70**	**65.49 ± 1.72**
MLP Classifier L2	65.02 ± 1.59	64.17 ± 2.03
MLP Classifier UOT	64.51 ± 0.77	64.36 ± 0.79

Table 4.4 The Table displays the accuracy and the F1 score in % for an MLP classifier based on different embeddings. The baseline consists of the preprocessed laboratory values, which perform the best as measured by both metrics. The MLPs for UOT and L^2 embeddings perform only marginally worse.

When comparing the average class prediction performance of the different classifiers for the 5-fold cross-validation, neither method is clearly better than the other (see Figure 4.8), as expected from Table 4.4. Furthermore, our analysis reveals that the MLP using the L^2 embeddings tends to incorrectly assign more individuals from class 1 than from class 2. Conversely, the baseline and UOT MLPs display a more balanced classification, distributing individuals more evenly across classes.

In conclusion, as this comparison highlights, both laboratory data and their respective embeddings possess some predictive power for the task at hand. However, it is important to recognize that the prediction objective remains challenging, suggesting that further research and refinement is needed to make more accurate predictions. Additionally, we extended our analysis to a larger subset that includes more features, limiting the number of missing values per feature to a maximum of 2532. However, this expansion did not significantly improve the performance of the models.

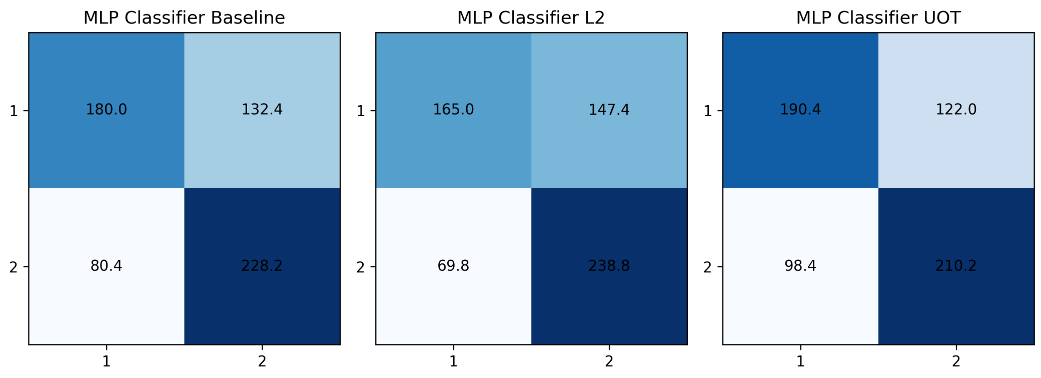

Figure 4.8 Average confusion matrices based on a 5-fold CV of MLP classifiers with different embeddings as input. For the L^2 embedding the MLP tends to classify more individuals to class 2 compared to the other two embeddings.

4.5.1 Patient-Patient Similarity Graphs with Unbalanced Optimal Transport

In this Section, we explore whether encoding the information in a graph structure improves the performance of predicting the treatment failure time class. To build a graph, we create an edge between individual i and j, if the corresponding entry in the embedding is bigger or smaller than some threshold. For the L^2 embedding we create an edge (i,j) if its entry is smaller than 0.13. For the UOT embedding (P^*) however, an edge is created if the entry is bigger than 0.006. We choose these thresholds so that almost all of the 3105 patients have at least one connection to another patient in the graph in order to obtain meaningful associations. Furthermore, these thresholds help to reduce the number of edges in the graph while ensuring that there is still a large sample size of patients available for analysis. Overall, this results in a graph for the UOT embedding with a total of 50983 edges containing all 3105 patients as nodes. In contrast, the graph created from the L^2 embedding encodes only 3098 patients as nodes with a total of 3469316 edges. To ensure a fair comparison, we only use patients that are present in both graphs, thus removing unique nodes.

In general, we observe that the graph constructed using the L^2 embedding has a significantly higher node degree compared to the graph generated based on P^*. This can affect the network structure and overall performance. In addition, increasing the threshold for L^2 quickly isolates many nodes, but the total number of edges remains relatively high.

Figure 4.9 illustrates the connected components of the graph created from P^*, with nodes color-coded according to their class. We observe that a patient has no more than 70 edges to other patients. The degree histogram reveals that most of the patients have a link—implying similarity—to between 30 and 50 of the other patients. The degree rank plot provides further insight in the connectivity distribution of the patients. It ranks the node degrees in descending order and shows that for the graph based on P^* the node degree exhibits a sharp decline between 70 and 50. Thereafter, the connectivity of patients gradually decreases, with a greater decline at lower degrees, meaning that there are fewer patients who are “similar” to only a few other patients. In contrast, the graph for the L^2 embedding has a substantial number of outgoing edges for each patient, reaching as high as 2900 (see Figure A.3). This results in a very dense graph that is difficult to visualize effectively. Furthermore, the degree rank plot reveals that the first 2.500 patients have degrees of up to 2.000, while the connectivity of the remaining patients shows a steep and rapid decline.

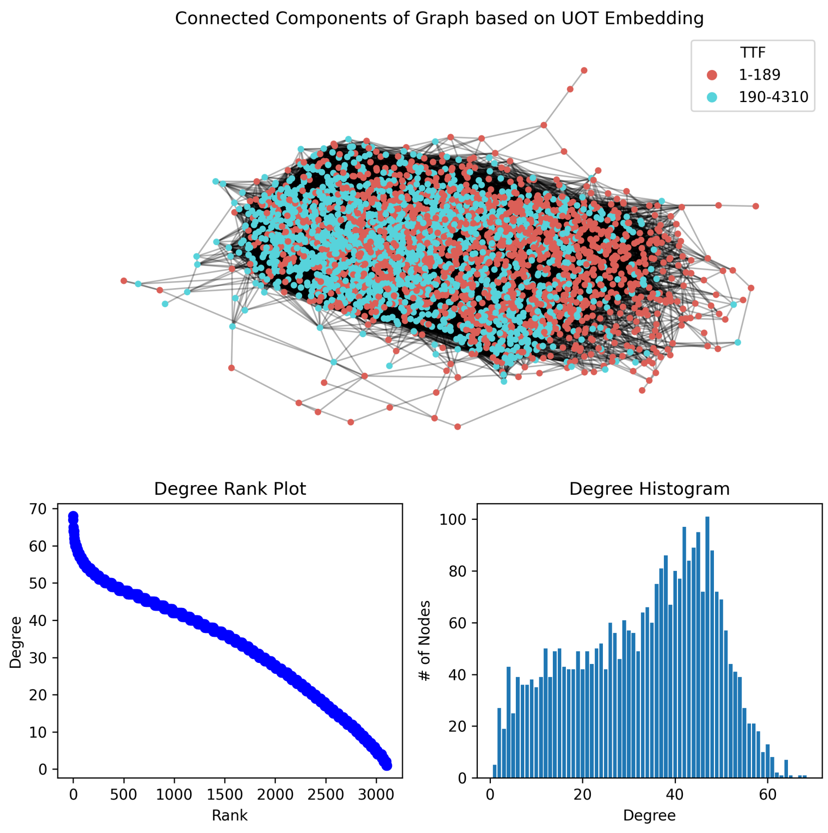

Figure 4.9 Connected components of a patient-patient similarity graph based on the Unbalanced Optimal Transport map. A node represents a patient and an edges link similar patients. The `Time to Treatment Failure` class for each patient is shown by the color of each node.

In summary, constructing a graph using the UOT map results in a sparser representation compared to building a graph based on the L^2 embedding. Moreover, the choice of thresholds and, in the case of UOT, also the choice of regularization parameters affects the overall connectivity of the graph and must therefore be tailored to the problem at hand.

As described in Section 3.2.1, we proceed to encode each node with its non-laboratory features, comprising a total of 55 features for N/A ≤ 400. These features consist mainly of the categorical features `ICD_###`, `OPS_###`, and intravenous therapy types. To compare whether encoding the information in a graph structure is preferable, we also train a MLP as the baseline with the combined laboratory and non-laboratory features.

Table 4.5 provides a comprehensive summary of the models' accuracy achieved during a 5-fold CV on the training set. The results indicate that the UOT approach outperforms the L^2 embedding. Among all models, the GAT based on the graph with the UOT embedding has the best performance with an average accuracy of 64.7%. Nevertheless, the MLP demonstrates only a marginally lower mean accuracy of 64.35%, while at the same time having a lower variability of accuracy. In general, all GNNs based on the graph created from the L^2 embeddings perform worse in terms of mean and deviation of accuracy over the 5-folds. It can also be observed that GCNs performs worse than GATs but better than GINs.

This shows that a sparse graph can give better results by aggregating more localized information and potentially capturing specific patterns. Moreover, the results also indicate that a dense L^2 graph, where a node's two-hop neighbors make up a significant portion of the entire graph, often fails to take advantage of the knowledge inherent in communities. The results suggest that using graphs can lead to a marginal performance gain over the baseline MLP. However, since this is a graph created using a real-world dataset, the data may be too complex and the sample size too small for the graph to work as intended. In the future, its predictive power should be evaluated using a subset of the dataset that already has a higher intrinsic

Model	UOT	L2	Baseline
GAT	**64.70 ± 3.47**	54.21 ± 6.36	-
GCN	62.18 ± 4.51	57.03 ± 5.61	-
GIN	60.82 ± 0.99	54.11 ± 4.03	-
MLP	-	-	64.35 ± 1.79

Table 4.5 Model accuracy in % for a 5-fold Cross Validation for various GNN architectures using different embeddings compared to an MLP.

notion of similarity. For example, sub-graphs could be created based on patients with Non-Small Cell Lung Cancer (NSCLC) and Small Cell Lung Cancer (SCLC) to provide a more meaningful representation and potentially better predictive results.

Training Time Analysis

One aspect that highlights the power of sparse graphs is their impact on model training time, since the number of edges directly affects it. As expected, the graph based on the UOT embedding trains faster than the one based on the L^2 embedding for a 5-fold CV on an NVIDIA A100 GPU (see Table 4.6). Additionally, both GCN and GIN have a quicker training process relative to the one of the MLP. This scaling behavior is especially interesting for larger graphs with more nodes. It has the potential to provide faster training compared to an MLP and with a better graph architecture and improved edges, similar or even superior results could be achieved.

Model	UOT	L2	Baseline
GCN	1.06 ± 0.34	5.24 ± 0.03	-
GAT	2.73 ± 0.06	120.24 ± 0.06	-
GIN	**0.82 ± 0.04**	6.13 ± 0.06	-
MLP	-	-	1.12 ± 0.07

Table 4.6 Model training time in seconds on a NVIDIA A100 GPU.

Evaluation of Model Performance on the Test Set

After obtaining the best results for UOT-based graphs, we proceed to train the models on the entire training set and evaluate their performance on the test set. Contrary to the CV results, we find that the graph methods do not perform better than the baseline MLP (see Table 4.7). It has an accuracy of 64.03%, closely followed by the GAT, which achieved an accuracy of 62.9%. This result further demonstrates that a graph-based approach does not guarantee superior results. Given the modest performance of current methods, using a more homogeneous dataset, such as a subset of NSCLC patients, and building a graph with their UOT map promises potential improvement in predictive accuracy beyond the baseline. Due to time constraints, this could not be investigated here, but should be followed up in future studies.

Model	UOT	Baseline
GAT	62.90	-
GCN	61.45	-
GIN	60.97	-
MLP	-	**64.03**

Table 4.7 Accuracy of models on test set.

Summary

Overall, the UOT graph generation method developed in this work did not lead to a substantial improvement in prediction accuracy. However, it showed better performance, resulting in higher prediction accuracy and shorter training times, compared to a naïve graph based on the L^2 embedding. For future applications, the UOT graph generation method is promising. It creates edges that are not intrinsically given, lowers computational complexity, and holds the potential for superior outcomes when combined with GNNs. Moreover, its performance could be further improved when applied to more homogeneous subsets.

In general, using a graph structure can still have several advantages. In fact, a graph can be used to identify potential clusters of similar patients, such as those who have failed treatment after the same period of time. This, in turn, could be studied in more detail to find patterns. Moreover, representing patient information as a graph can simplify data complexity, especially when dealing with large and diverse datasets. Another aspect worth exploring for the future is the use of graphs with pre-defined edges between patients, such as those based on molecular markers. This provides the opportunity to train graphs for edge prediction, which can identify edges from a set of patients to a new patient. As a result, unnecessary interventions or painful procedures can potentially be avoided by taking advantage of the network's insight.

4.6 Lessons Learned and Failed Methods

In research, there are many methods, models, and ideas that fail or are not further followed upon. These lessons are often neglected in scientific papers. However, they are critical to the learning process. The purpose of this Section is to provide insights into failed approaches and to offer an opportunity to learn from, refine, and reevaluate them for future efforts.

First, we evaluated several strategies for dealing with N/As. We tried using the median instead of the mean for imputation, but this did not produce better results. Another technique we explored for dealing with missing values were polar encodings, as introduced by Lenz, Peralta, and Cornelis [91]. These gave comparable and sometimes better results than the mean imputation method when training DTs on built-in datasets from the sklearn package. However, on our lung cancer dataset, RSFs using polar encodings did not perform better compared to those using mean imputation. One reason for that could be the high dimensionality and diversity of the dataset provided by UHE.

We made initial efforts to determine sex using laboratory values, but have not pursued this project further due to time constraints. One reason is that it requires careful consideration of variables such as laboratory values like BMI and hemoglobin, which are already known to differ between the sexes. Eliminating such features may yield interesting results on characteristics that have not yet been shown to be gender-specific. For example, these results could then be used for gender-specific precision medicine.

In addition, several other survival models were trained, such as XGB trees with a cox survival objective function, Cox Proportional Hazards models with elastic net penalties, a probability mass function method called Neural Multi-Task Logistic Regression, and a Piecewise Constant hazard (PC Hazard) model. Since they did not provide better results as measured by the Concordance Index compared to DeepSurv, RSF and XGBSE, they were not used in further analysis.

The dataset can also be split into many sub-datasets to measure the impact of different groups on the event variables, e.g. individuals diagnosed with NSCLC and individuals diagnosed with SCLC. Models such as RSFs, combined with the explanatory power of Shapley values, can be used to compare and assess the importance of features within subgroups. We have trained several such models, but to avoid going beyond the scope of this work, we leave the evaluation to the clinicians.

We also attempted to create patient-patient similarity networks with OT transport maps, but found UOT embeddings to be superior. This can be explained by the fact that UOT does not force all mass to be transported from the source to the target, so no mass needs to be transported when patients are dissimilar.

Furthermore, we used a method from topological data analysis, the Kepler mapper, to build patient-patient similarity graphs. This method creates a graph of connected clusters of patients. Based on the underlying clustering algorithm, the number of clusters does not necessarily have to be specified which makes this approach very flexible. We then used the clusters and their connections to build a patient level graph. Each patient is connected to every other patient within their respective cluster, as well as to all patients in the other clusters that have been linked. We then performed the same node classification task on the resulting graph as we did on the graphs based on the UOT and L^2 embeddings. While this approach performed better than the L^2 graph, it did not outperform the UOT graph. The better performance of the Kepler graph compared to the L^2 graph may be attributed to the fact that its edges are more meaningful and to the reduced graph density. However, it did not achieve better node classification results than the UOT graph, which could be caused by the inherent complexity of the data. Since the results were not superior to the UOT graph, we did not pursue it further. Nevertheless, it should be subjected to future evaluation since many hyperparameters such as the clustering algorithm, the choice of lenses, the overlap and the number of cubes affect the resulting Kepler graph.

Additionally, we did not get an improvement as measured by node classification accuracy when using weighted edges based on the transported mass of the UOT map or the cost of the L^2 embedding. This can be due to several reasons, such as suboptimal edge quality, the complexity of the prediction objective, or the fact that the transported mass might not capture the real underlying relationship between patients. For future research, where the edges may already be given by the design of the data, edge weighting may still be an area worth exploring to give more weight to certain edges from a patient, as it is natural to assume that patients are not all equally similar to other patients.

We also evaluated whether a certain node degree achieved a higher accuracy for node classification, but could not find a clear trend. It is possible that the sample size of nodes at certain degrees was too small. This may be a worthwhile field for future research to determine whether predictive accuracy improves for patients with fewer or more edges, and to assess the impact of edge weighting.

In general, we have shown numerous promising areas for further research, spanning from imputation schemes, gender-specific medicine and alternative survival models to topological data analysis techniques, special graph architectures, and beyond.

5 Conclusion

In summary, we have provided a detailed mathematical overview of the foundations necessary to perform survival analysis, compute Optimal Transport maps, and do Deep Learning on Graph Neural Networks using a dataset comprising lung cancer patients from the University Hospital Essen.

We have further shown that modern methods for survival analysis such as DeepSurv, XGBSE and Random Survival Forests provide a good risk-of-event estimate for lung cancer patients. However, it is important to note that there is no one single model that clearly outperforms the others. In addition, RSFs together with SHAP values yield a powerful approach to provide an explainable risk prediction for patients. Integrating these methods into the medical decision-making process may allow for more personalized treatment and provide new insights into the importance of features in determining a patient's risk-of-event. By leveraging this combination and accounting for potential confounding, physicians can make a more informed treatment decision, increase explainability to the patient with less subjective influence, and discover new potential areas for clinical trials. As a result, this approach can lead to greater trust between patient and physician.

Moreover, we have created patient-patient similarity graphs based on the distances between patients' laboratory features and shown that they can be used for tasks such as node prediction, particularly for predicting treatment failure within six months or later. In this context, we proposed a novel method to construct sparse patient-patient similarity graphs with the use of Unbalanced Optimal Transport maps. We have demonstrated that they achieve better predictive results and faster training with Graph Neural Networks when compared to a naïve graph created from the L^2 distance between patients. In comparison to an MLP, which was used as a baseline, we find that the graphs do not necessarily yield a better prediction accuracy. This could be due to the heterogeneity and complexity of the underlying dataset, the unsupervised way edges are created, the amount of edges per patient, the lack of relevant node-level features, limitations in the graph structure, or more generally the dataset size.

Beyond that, we outlined additional opportunities of patient-patient similarity graphs such as finding clusters of similar patients which can enable personalized treatment approaches. In addition, these graphs can be used to predict edges between patients, potentially providing insight into disease progression or indicating the presence of additional diseases that can be medically tested for.

In general, leveraging graphs creates the possibility for a more localized approach to decision-making with clinical data, which can provide many desirable properties, such as scalability, personalized recommendations, patient-patient similarity discovery, and more. However, we have shown that creating meaningful edges within a graph remains challenging, especially when no "natural" edges exist, and thus must be inferred from certain features within the dataset using methods such as our UOT edge generation method.

Finally, we give an overview of the methods that either failed, did not achieve the desired level of performance, or could not be explored further due to time constraints, to provide insight into our decision-making and to advance research in general.

5.1 Applicability of Methods

The methods developed in this context can be applied to all time-to-event data including but not limited to outcome variables such as treatment failure or death. They are especially interesting for hospitals, as they can be easily integrated, provided that the data is stored in a structured format. By employing methods such as DeepSurv, XGBSE or RSF to large, diverse and hospital-quality datasets, medical decision-making can be improved and new potential areas for future clinical trials can be discovered. In addition, these methods can be continually refined as new data becomes available. However, except for DeepSurv, which can reuse its pretrained weights, tree-based methods require retraining to incorporate new data points into the model. From a computational point of view, this proves to be ineffective. Nevertheless, this problem can be mitigated by building the trees in parallel.

Additionally, many factors play a vital role in the performance of these models, such as the number of missing values in the dataset, feature imbalance, as well as method design and the causal context of the prediction objectives. Especially, in real-world datasets for which potential feature interactions were not directly considered, physicians must carefully account for potential confounding effects.

Utilizing graphs for prediction tasks on clinical data presents greater complexity in contrast to survival methods, since it requires careful data preprocessing and sophisticated models. We have shown that prediction tasks on them remain challenging. Our edge generation method based on the UOT map showed promising results in terms of edge density, computational speed for GNNs, and performance. It can be easily integrated for other datasets and should therefore be further investigated to realize the full potential of graph-based predictive modeling in clinical settings. Moreover, due to its versatility, UOT also holds great promise in areas such as anomaly detection, disease progression modeling, predictive modeling of unbalanced datasets, and more.

Furthermore, numerous aspects are still to be explored, for instance, edge purity could be evaluated based on different features such as a patient's metastasis information or biological markers to gain a better understanding of the graph structure and similarities between patients. Overall, the use of graphs for clinical data remains an area for future exploration.

5.2 Outlook

Based on our study, several new questions and opportunities emerged that are worth evaluating in future research. While the dataset provided by UHE is not inherently longitudinal, exploring it across multiple time points could help to gain a more comprehensive understanding of its features. A survival model can then be trained for each time point and tested to see how the predicted risk changes over time for each individual and how accurate those predictions are. In addition, the incorporation of SHAP values for explainability allows tracking the development of feature importance over time. This can shed light on the longitudinal dynamics of the features of lung cancer patients. In addition, it can potentially provide insight into which factors, such as laboratory values, are ultimately responsible for the death of a patient.

Another potential approach is to use the k-1 time points to construct an ensemble of models that correct the errors of their previous models, similar to XGBoost. This condensed knowledge can then be used to predict survival at the final time point, which can provide higher predictive accuracy. However, due to the longitudinal nature of the data, model building requires more thought to avoid potential data leakage.

UOT can be used to track patients over time and find similar patients. For example, this approach can help monitor the development of disease symptoms and serve as an "early warning system" or provide earlier treatment options, especially when two patients are found to be similar, but one exhibits certain symptoms and the other does not yet. In addition, for each time point a patient-patient similarity graph could be created and then be used to track how similarities between patients change over time. This might provide important information about the rate of disease progression in patients. In addition, with information such as treatment success/failure, changes in patient-patient similarity can be tracked in the graph and its communities to understand the effect of a particular treatment. Another potential application would be to adapt the method proposed in Tong et al. [25] to use a patient's features, such as lab values, as signals on a patient-patient similarity graph or knowledge graph. It has been shown that these embeddings can be used to identify potential clusters and clinically significant overlaps among patient diagnoses that may not be directly apparent.

Generally, to further refine patient-patient similarity, a human-in-the-loop strategy is a viable option, where similarity predictions are refined based on feedback from clinicians.

Furthermore, the methods employed in this dataset can be easily applied to bigger datasets like a pan-cancer dataset. This will provide more insight into cancer in general. For example, an understanding of how cancers are expressed differently, such as in lab values, can be gained. In addition, our UOT approach can also provide a notion of similarity between individuals with different cancers, identifying cancers that are more similar than others. These insights could in turn motivate future clinical trials.

In summary, this outlook highlights the importance of continued research and potential opportunities in this area and paves the way for future explorations. By relentlessly pursuing and evaluating different approaches, we have the potential to significantly improve the lives of cancer patients and make progress towards a potential cure through a better understanding of the disease.

A Appendix

A.1 Supplementary Figures

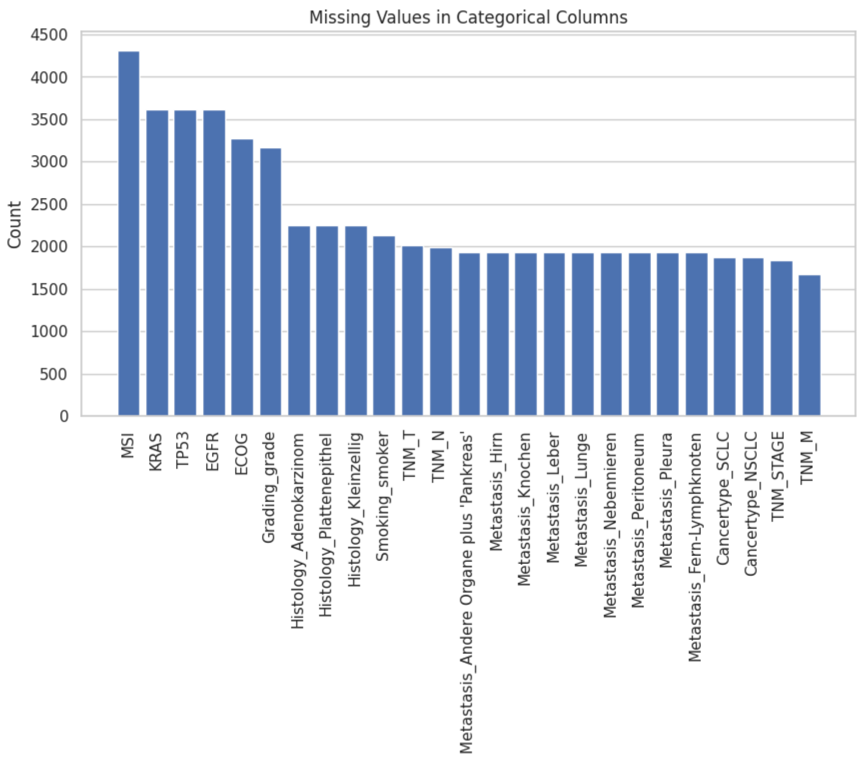

Figure A.1 Histogram of all missing values for categorical features in the dataset. The median, range and standard deviation are shown for each feature in the Table A.5.

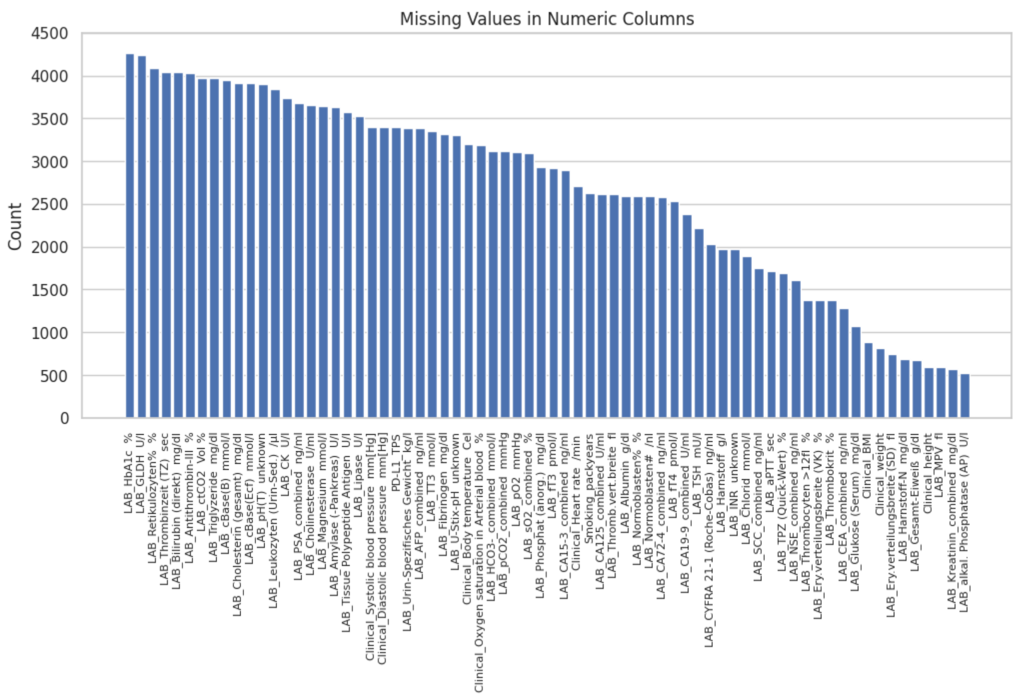

Figure A.2 Histogram of missing values for numeric features in the dataset. Only features with more than 500 N/As are displayed. For all missing values Table A.4 is to be consulted.

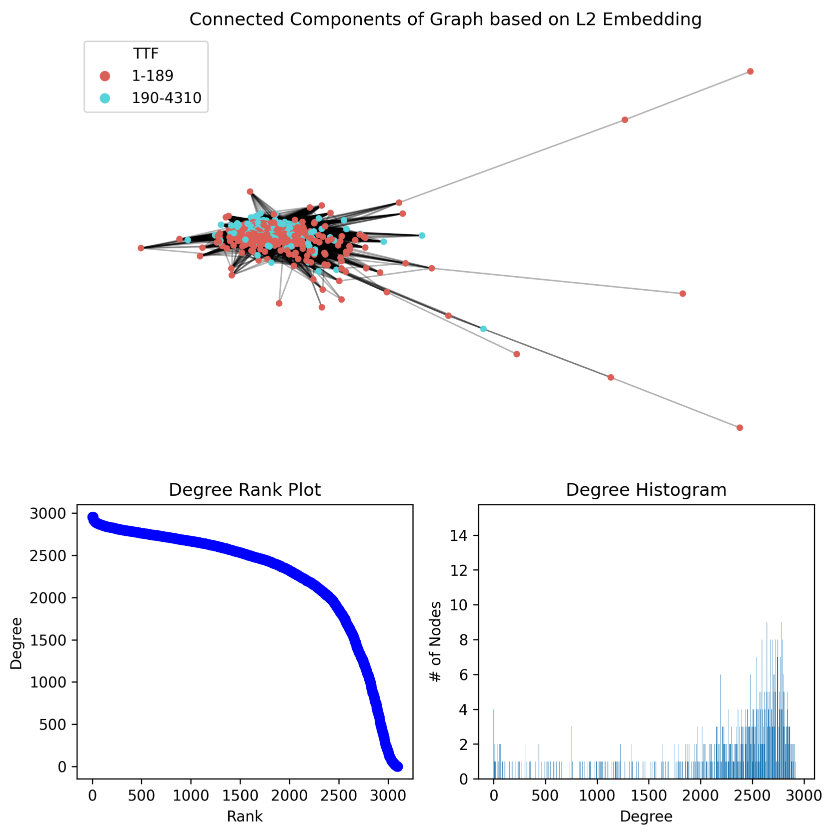

Figure A.3 Connected components of a patient-patient similarity graph based on the L^2 embedding. A node represents a patient and an edges link similar patients. The `Time To Treatment Failure` class for each patient is shown by the color of each node. The visualization of the graph appears too dense, due to the excessive number of edges.

A.2 Supplementary Tables

Model	C-Index	IBS	5-CV C-Index	5-CV IBS
N/A $\leq$ 400				
DeepSurv	0.693	0.171	0.681 ± 0.020	0.175 ± 0.006
RSF	**0.694**	**0.167**	0.686 ± 0.014	**0.171 ± 0.005**
XGBSE	0.689	0.171	**0.686 ± 0.013**	0.174 ± 0.005
N/A $\leq$ 2532				
DeepSurv	**0.707**	**0.162**	0.685 ± 0.015	0.172 ± 0.006
RSF	0.704	0.164	**0.704 ± 0.010**	**0.168 ± 0.004**
XGBSE	0.703	0.168	0.701 ± 0.011	0.172 ± 0.005

Table A.1 Survival model comparison measured by Concordance Index and Integrated Brier Score for `TTF_Event` as event variable. N/A smaller or equal than 400 indicates that only columns with at most 400 N/A's were kept to be imputed, while removing columns with more. Similarly, this was done for columns with up to 2532 N/A's. All model achieve comparable results and there is no single model which outperforms the others. Model performance generally increases when keeping features with many N/A's.

Model	C-Index	IBS	5-CV C-Index	5-CV IBS
N/A $\leq$ 400				
All	0.696	0.165	0.688 ± 0.008	0.170 ± 0.011
Female	0.672	0.169	0.655 ± 0.025	0.183 ± 0.014
Male	**0.705**	**0.163**	**0.694 ± 0.005**	**0.163 ± 0.009**
N/A $\leq$ 2532				
All	0.706	0.163	0.706 ± 0.013	0.167 ± 0.013
Female	0.696	0.168	0.678 ± 0.028	0.179 ± 0.017
Male	**0.733**	**0.154**	**0.710 ± 0.006**	**0.161 ± 0.007**

Table A.2 Random Survival Forest comparison with `TTF_Event` as event variable on the entire dataset, as well as on subsets filtered for male and female individuals. N/A smaller or equal than 400 indicates that only columns with at most 400 N/A's were kept to be imputed, while removing columns with more. Similarly, this was done for columns with up to 2532 N/A's. In general, we observe that model performance tends to improve as more columns are retained in the dataset. However, it is crucial to treat this observation with caution, as it is influenced by the extensive imputation that took place. The RSF on the male subset performs the best measured by the C-Index and IBS.

A.2.1 Definition of Dataset Features

Variable	Meaning	Features	Type
`LAB_###`	Pre-Treatment Laboratory Values	92	Numeric
`ICD_###`	International Classification of Diseases	90	Binary
`Cancer_###`	Cancer Diagnosis	60	Binary
`OPS_###`	Classification for Encoding Operations, Procedures, and General Medical Measures [94]	51	Binary
`ctx_###`	Intravenous Therapy Type	31	Binary
`rezept_###`	Oral Medication Type	17	Binary
`Histology_###`	Histological Diagnosis of Cancer Types	12	Binary
`Metastasis_###`	Metastasised Organs	9	Binary
`BCA_###`	Body Composition Analysis	6	Numeric
`Clinical_###`	Body Statistics 14 Days before Treatment	8	Numeric
`Cancertype_###`	Type of Cancer (CCC, HCC, NSCLC, SCLC, Sarcoma)	5	Binary
`TNM_###`	Classification for Cancer Staging [95]	4	Categorical
`age_at_ctx`	Age at Start of Therapy	1	Numeric
`gender_female`	Gender of Patient	1	Binary
`Grading_grade`	Cancer Stages from In-Situ to Metastatic	1	Categorical
`Smoking_smoker`	Smoking Status	1	Binary
`Smoking_packyears`	Cigarette Packs per Day $\times$ Smoking Years	1	Numeric
`ECOG`	Performance Status Scale [83]	1	Categorical
`MSI`	Microsattelite Instability [84]	1	Binary
`EGFR`	Epidermal Growth Factor Receptor [85]	1	Binary
`KRAS`	Kirsten Rat Sarcoma Viral Oncogene Homologue [86]	1	Binary
`TP53`	Gene Enabling Tumor Suppressor Protein 53 [87]	1	Binary
`PD-L1_TPS`	PD-L1 Tumor Proportion Score [89]	1	Numeric
`TTF`	Time to Treatment Failure	1	Numeric
`TTF_Event`	Treatment Failure	1	Binary
`OS`	Overall Survival	1	Numeric
`death_observed`	Death Observed	1	Binary

Table A.3 Definitions of features of the dataset provided by University Hospital Essen.

A.2.2 Statistical Summary of Features

Numerical Variables

Feature	#N/As	Median	Range	SD
LAB_Eos. Granulozyten# /nl	309	0.12	100.62	1.66
LAB_Leukozyten /nl	24	8.73	108.89	5.27
LAB_GPT (ALAT) U/l	88	22.0	519.0	33.37
LAB_Lymphozyten# /nl	302	1.42	6.47	0.72
LAB_MCHC g/dl	24	33.5	10.7	1.19
LAB_Baso.Granulozyten# /nl	300	0.03	2.3	0.05
LAB_GOT (ASAT) U/l	69	21.0	579.0	33.36
LAB_Hämatokrit l/l	24	0.38	0.37	0.05
LAB_Lymphozyten% %	302	16.6	68.6	9.28
LAB_Bilirubin (gesamt) mg/dl	156	0.4	13.2	0.49
LAB_MCV fl	85	88.1	57.5	5.71
LAB_Erythrozyten /pl	24	4.3	4.79	0.6
LAB_Hämoglobin g/dl	85	12.6	12.4	1.83
LAB_Natrium mmol/l	100	139.0	39.0	3.49
LAB_MCH pg	24	29.5	21.0	2.33
LAB_Gesamt-Eiweiß g/dl	672	6.8	5.18	0.7
LAB_CRP mg/dl	96	1.8	53.0	5.5
LAB_Monozyten% %	302	8.3	42.0	3.54
LAB_GGT U/l	157	42.0	2706.0	197.4
LAB_Eos. Granulozyten% %	309	1.4	93.3	3.25
LAB_Neu. Granulozyten% %	309	71.3	97.0	11.89
LAB_Baso. Granulozyten% %	300	0.4	7.6	0.35
LAB_Neu. Granulozyten# /nl	309	6.13	91.66	4.4
LAB_Kalium mmol/l	99	4.5	7.7	0.52
LAB_Glukose (Serum) mg/dl	1075	101.0	565.0	47.95
LAB_LDH U/l	77	247.0	5775.0	296.74
LAB_Harnstoff g/l	1970	0.33	2.25	0.17
LAB_Thrombozyten /nl	24	292.5	1060.0	121.93
LAB_Monozyten# /nl	302	0.71	3.63	0.37
LAB_Calcium mmol/l	202	2.3	2.66	0.17
LAB_Ery.verteilungsbreite (SD) fl	743	45.0	62.7	6.71
LAB_Thrombocyten >12fl %	1382	23.4	49.4	7.13
LAB_MPV fl	595	9.8	6.1	0.88
LAB_Harnstoff-N mg/dl	688	15.1	105.0	7.67
LAB_Thromb.vert.breite fl	2612	10.9	14.8	1.91
LAB_Ery.verteilungsbreite (VK) %	1377	14.0	19.6	2.07
LAB_Thrombokrit %	1372	0.29	0.96	0.11
LAB_Normoblasten% %	2588	0.0	17.7	0.67
LAB_Normoblasten# /nl	2588	0.0	1.05	0.05
LAB_Magnesium mmol/l	3647	0.84	0.85	0.12
LAB_TSH mU/l	2217	1.14	34.25	2.39
LAB_Harnsäure mg/dl	155	5.1	19.7	1.73
LAB_fT4 pmol/l	2532	16.1	41.6	3.19
LAB_Tissue Polypeptide Antigen U/l	3574	90.0	9012.0	630.89
LAB_CYFRA 21-1 (Roche-Cobas) ng/ml	2035	4.0	1127.7	39.68
LAB_TT3 nmol/l	3351	1.65	6.84	0.45

Continued on next page

Table A.4 – continued from previous page

Feature	#N/As	Median	Range	SD
LAB_TPZ (Quick-Wert) %	1691	104.0	109.0	18.14
LAB_aPTT sec	1718	26.2	141.8	5.12
LAB_INR unknown	1968	1.0	3.34	0.19
LAB_fT3 pmol/l	2920	4.6	15.7	0.98
LAB_Fibrinogen mg/dl	3315	501.0	892.0	186.63
LAB_cBase(Ecf) mmol/l	3915	1.4	30.0	3.74
LAB_pO2 mmHg	3111	75.0	184.0	17.53
LAB_ctCO2 Vol %	3973	58.8	60.1	8.08
LAB_cBase(B) mmol/l	3947	1.6	26.9	3.09
LAB_pH(T) unknown	3902	7.45	0.32	0.05
LAB_Antithrombin-III %	4035	99.0	120.0	16.7
LAB_alkal. Phosphatase (AP) U/l	522	92.0	1117.0	91.7
LAB_Lipase U/l	3527	30.0	2714.0	102.79
LAB_CK U/l	3739	57.0	743.0	75.45
LAB_Chlorid mmol/l	1892	103.0	56.0	4.33
LAB_HbA1c %	4264	6.8	8.1	1.74
LAB_Urin-Spezifisches Gewicht kg/l	3389	1.01	0.06	0.01
LAB_U-Stix-pH unknown	3308	5.5	4.5	0.81
LAB_Leukozyten (Urin-Sed.) /µl	3845	6.0	4482.0	331.91
LAB_Albumin g/dl	2598	4.2	3.2	0.45
LAB_Triglyzeride mg/dl	3969	121.0	616.0	79.43
LAB_Cholesterin (gesamt) mg/dl	3915	188.0	333.0	45.4
LAB_Amylase (-Pankreas) U/l	3636	26.0	679.0	44.5
LAB_Phosphat (anorg.) mg/dl	2935	3.4	6.1	0.64
LAB_Cholinesterase U/ml	3651	7.4	17.3	2.4
LAB_Retikulozyten% %	4093	1.16	15.34	1.3
LAB_Thrombinzeit (TZ) sec	4046	16.8	156.4	13.18
LAB_Bilirubin (direkt) mg/dl	4044	0.19	9.7	1.1
LAB_GLDH U/l	4245	4.8	391.2	55.27
LAB_CEA_combined ng/ml	1283	4.0	88767.6	1738.85
LAB_CA19-9_combined U/ml	2377	14.3	279031.0	7839.33
LAB_CA125_combined U/ml	2617	36.0	27928.0	938.01
LAB_AFP_combined ng/ml	3383	2.4	30676.1	1002.07
LAB_SCC_combined ng/ml	1754	1.0	306.0	10.44
LAB_CA15-3_combined ng/ml	2899	21.0	3961.0	180.01
LAB_NSE_combined ng/ml	1609	25.4	12880.6	450.78
LAB_CA72-4_combined ng/ml	2584	2.4	9506.8	370.77
LAB_PSA_combined ng/ml	3685	0.88	66.4	4.56
LAB_sO2_combined %	3097	95.4	91.3	10.7
LAB_pCO2_combined mmHg	3113	35.0	51.4	5.26
LAB_HCO3-_combined mmol/l	3114	25.4	26.3	2.5
LAB_Kreatinin_combined mg/dl	568	0.75	38.58	0.71
Clinical_Oxygen saturation in Arterial blood %	3186	96.0	64.0	3.44
Clinical_Body temperature Cel	3201	36.4	6.5	0.55
Clinical_Heart rate /min	2709	81.0	116.0	15.26
Clinical_Systolic blood pressure mm[Hg]	3397	121.0	218.0	18.68
Clinical_Diastolic blood pressure mm[Hg]	3397	73.0	153.0	11.38
age_at_ctx	0	63.68	73.01	10.12

Continued on next page

Table A.4 – continued from previous page

Feature	#N/As	Median	Range	SD
`Clinical_height`	599	1.72	0.43	0.09
`Clinical_weight`	818	75.0	90.0	16.23
`BCA_bone`	2532	30.42	58.39	4.76
`BCA_muscle`	2532	68.63	94.42	14.53
`BCA_sat`	2532	73.75	356.31	43.87
`BCA_vat`	2532	42.51	141.89	25.5
`BCA_imat`	2532	14.66	55.82	7.71
`BCA_tat`	2532	138.73	472.42	67.33
`PD-L1_TPS`	3397	5.0	100.0	33.6
`Smoking_packyears`	2631	40.0	240.0	24.36
`Clinical_BMI`	886	24.84	33.26	4.62
`TTF`	0	196.0	5459.0	651.9
`OS`	0	277.0	5459.0	722.97

Table A.4 Table containing all numeric features with their respective N/A counts, median values, range and standard deviation.

Categorical Variables

Feature	#N/As	Category Frequencies
ICD_B95	0	0: 4270, 1: 50
ICD_B96	0	0: 4258, 1: 62
ICD_C32	0	0: 4310, 1: 10
ICD_C41	0	0: 4318, 1: 2
ICD_C56	0	0: 4318, 1: 2
ICD_C76	0	0: 4299, 1: 21
ICD_C77	0	0: 3287, 1: 1033
ICD_C78	0	0: 3789, 1: 531
ICD_C79	0	0: 3488, 1: 832
ICD_D41	0	0: 4300, 1: 20
ICD_D50	0	0: 4293, 1: 27
ICD_D61	0	0: 4257, 1: 63
ICD_D62	0	0: 4251, 1: 69
ICD_D63	0	0: 4233, 1: 87
ICD_D64	0	0: 4245, 1: 75
ICD_D68	0	0: 4270, 1: 50
ICD_D69	0	0: 4258, 1: 62
ICD_D70	0	0: 4257, 1: 63
ICD_E03	0	0: 4173, 1: 147
ICD_E05	0	0: 4267, 1: 53
ICD_E11	0	0: 4009, 1: 311
ICD_E66	0	0: 4101, 1: 219
ICD_E78	0	0: 4017, 1: 303
ICD_E79	0	0: 4248, 1: 72
ICD_E83	0	0: 4291, 1: 29
ICD_E86	0	0: 4193, 1: 127
ICD_E87	0	0: 4066, 1: 254
ICD_E89	0	0: 4278, 1: 42
ICD_F10	0	0: 4278, 1: 42
ICD_F17	0	0: 3778, 1: 542
ICD_F32	0	0: 4271, 1: 49
ICD_G40	0	0: 4262, 1: 58
ICD_G47	0	0: 4249, 1: 71
ICD_G62	0	0: 4261, 1: 59
ICD_G81	0	0: 4277, 1: 43
ICD_G93	0	0: 4278, 1: 42
ICD_I10	0	0: 3598, 1: 722
ICD_I11	0	0: 4240, 1: 80
ICD_I20	0	0: 4214, 1: 106
ICD_I25	0	0: 3960, 1: 360
ICD_I26	0	0: 4256, 1: 64
ICD_I34	0	0: 4262, 1: 58
ICD_I35	0	0: 4268, 1: 52
ICD_I48	0	0: 4178, 1: 142
ICD_I49	0	0: 4276, 1: 44
ICD_I50	0	0: 4170, 1: 150
ICD_I70	0	0: 4166, 1: 154

Continued on next page

Table A.5 – continued from previous page

Feature	#N/As	Category Frequencies
ICD_I80	0	0: 4271, 1: 49
ICD_J18	0	0: 4234, 1: 86
ICD_J38	0	0: 4270, 1: 50
ICD_J44	0	0: 3768, 1: 552
ICD_J45	0	0: 4282, 1: 38
ICD_J90	0	0: 4207, 1: 113
ICD_J91*	0	0: 4226, 1: 94
ICD_J96	0	0: 4106, 1: 214
ICD_K21	0	0: 4247, 1: 73
ICD_K29	0	0: 4241, 1: 79
ICD_K56	0	0: 4306, 1: 14
ICD_K57	0	0: 4290, 1: 30
ICD_K59	0	0: 4276, 1: 44
ICD_K74	0	0: 4299, 1: 21
ICD_K80	0	0: 4288, 1: 32
ICD_K83	0	0: 4312, 1: 8
ICD_M54	0	0: 4253, 1: 67
ICD_N13	0	0: 4310, 1: 10
ICD_N17	0	0: 4286, 1: 34
ICD_N18	0	0: 4181, 1: 139
ICD_N28	0	0: 4289, 1: 31
ICD_N39	0	0: 4260, 1: 60
ICD_N40	0	0: 4249, 1: 71
ICD_R06	0	0: 4158, 1: 162
ICD_R10	0	0: 4273, 1: 47
ICD_R11	0	0: 4206, 1: 114
ICD_R13	0	0: 4255, 1: 65
ICD_R16	0	0: 4305, 1: 15
ICD_R18	0	0: 4300, 1: 20
ICD_R26	0	0: 4264, 1: 56
ICD_R50	0	0: 4252, 1: 68
ICD_R52	0	0: 4148, 1: 172
ICD_R53	0	0: 4194, 1: 126
ICD_R59	0	0: 4257, 1: 63
ICD_R63	0	0: 4259, 1: 61
ICD_R64	0	0: 4269, 1: 51
ICD_R77	0	0: 4235, 1: 85
ICD_Z12	0	0: 4315, 1: 5
ICD_Z85	0	0: 4142, 1: 178
ICD_Z90	0	0: 4170, 1: 150
ICD_Z93	0	0: 4311, 1: 9
ICD_Z95	0	0: 4079, 1: 241
gender_female	0	0: 2576, 1: 1744
TNM_T	2017	0: 18, 1: 295, 2: 433, 3: 523, 4: 1034
TNM_N	1994	0: 524, 1: 240, 2: 874, 3: 688
TNM_M	1679	0: 982, 1: 1659
TNM_STAGE	1842	1: 48, 2: 158, 3: 613, 4: 1659
Histology_Adenokarzinom	2255	0: 1179, 1: 886

Continued on next page

Table A.5 – continued from previous page

Feature	#N/As	Category Frequencies
Histology_Glioblastom	2255	0: 2065
Histology_Astrozytom	2255	0: 2065
Histology_Neuroendokrin	2255	0: 2016, 1: 49
Histology_Plattenepithel	2255	0: 1670, 1: 395
Histology_Kleinzellig	2255	0: 1731, 1: 334
Histology_Melanom	2255	0: 2064, 1: 1
Histology_Duktal	2255	0: 2063, 1: 2
Histology_Mesotheliom	2255	0: 2062, 1: 3
Histology_Leiomyosarkom	2255	0: 2060, 1: 5
Histology_Ewing	2255	0: 2064, 1: 1
Histology_Liposarkom	2255	0: 2065
Cancertype_CCC	1616	0: 2704
Cancertype_HCC	1616	0: 2704
Cancertype_NSCLC	1873	0: 486, 1: 1961
Cancertype_SCLC	1873	0: 1961, 1: 486
Grading_grade	3175	1: 13, 2: 253, 3: 670, 4: 209
ECOG	3280	0: 465, 1: 454, 2: 98, 3: 20, 4: 3
Cancertype_Sarkom	2251	0: 2016, 1: 53
Metastasis_Andere Organe plus 'Pankreas'	1936	0: 2161, 1: 223
Metastasis_Fern-Lymphknoten	1936	0: 2196, 1: 188
Metastasis_Hirn	1936	0: 1959, 1: 425
Metastasis_Knochen	1936	0: 1857, 1: 527
Metastasis_Leber	1936	0: 2056, 1: 328
Metastasis_Lunge	1936	0: 2014, 1: 370
Metastasis_Nebennieren	1936	0: 2082, 1: 302
Metastasis_Peritoneum	1936	0: 2359, 1: 25
Metastasis_Pleura	1936	0: 2162, 1: 222
TP53	3618	0: 310, 1: 392
KRAS	3621	0: 481, 1: 218
EGFR	3617	0: 607, 1: 96
MSI	4316	0: 2, 1: 2
OPS_1-100	0	0: 4271, 1: 49
OPS_1-204	0	0: 4277, 1: 43
OPS_1-275	0	0: 4216, 1: 104
OPS_1-420	0	0: 4309, 1: 11
OPS_1-422	0	0: 4293, 1: 27
OPS_1-424	0	0: 4307, 1: 13
OPS_1-559	0	0: 4316, 1: 4
OPS_1-610	0	0: 4262, 1: 58
OPS_1-611	0	0: 4264, 1: 56
OPS_1-620	0	0: 4145, 1: 175
OPS_1-630	0	0: 4268, 1: 52
OPS_1-632	0	0: 4298, 1: 22
OPS_1-661	0	0: 4302, 1: 18
OPS_1-710	0	0: 4262, 1: 58
OPS_1-901	0	0: 4269, 1: 51
OPS_3-992	0	0: 4295, 1: 25
OPS_5-010	0	0: 4185, 1: 135

Continued on next page

Table A.5 – continued from previous page

Feature	#N/As	Category Frequencies
OPS_5-015	0	0: 4201, 1: 119
OPS_5-021	0	0: 4230, 1: 90
OPS_5-155	0	0: 4316, 1: 4
OPS_5-156	0	0: 4309, 1: 11
OPS_5-399	0	0: 4186, 1: 134
OPS_5-401	0	0: 4280, 1: 40
OPS_5-402	0	0: 4299, 1: 21
OPS_5-403	0	0: 4304, 1: 16
OPS_5-469	0	0: 4311, 1: 9
OPS_5-501	0	0: 4317, 1: 3
OPS_5-511	0	0: 4309, 1: 11
OPS_5-541	0	0: 4302, 1: 18
OPS_5-852	0	0: 4306, 1: 14
OPS_5-870	0	0: 4313, 1: 7
OPS_5-894	0	0: 4280, 1: 40
OPS_5-895	0	0: 4280, 1: 40
OPS_5-900	0	0: 4304, 1: 16
OPS_5-903	0	0: 4297, 1: 23
OPS_5-983	0	0: 4290, 1: 30
OPS_5-984	0	0: 4122, 1: 198
OPS_5-988	0	0: 4267, 1: 53
OPS_5-989	0	0: 4317, 1: 3
OPS_8-137	0	0: 4309, 1: 11
OPS_8-522	0	0: 3725, 1: 595
OPS_8-526	0	0: 4316, 1: 4
OPS_8-527	0	0: 4020, 1: 300
OPS_8-528	0	0: 4033, 1: 287
OPS_8-529	0	0: 3981, 1: 339
OPS_8-530	0	0: 4312, 1: 8
OPS_8-542	0	0: 4198, 1: 122
OPS_8-543	0	0: 4061, 1: 259
OPS_8-701	0	0: 4256, 1: 64
OPS_8-800	0	0: 4292, 1: 28
OPS_8-831	0	0: 4109, 1: 211
OPS_8-83b	0	0: 4238, 1: 82
OPS_8-900	0	0: 4157, 1: 163
OPS_8-902	0	0: 4311, 1: 9
OPS_8-910	0	0: 4314, 1: 6
OPS_8-919	0	0: 4281, 1: 39
OPS_8-925	0	0: 4232, 1: 88
OPS_8-930	0	0: 4237, 1: 83
OPS_8-931	0	0: 4175, 1: 145
Smoking_smoker	2136	0: 115, 1: 2069
ctx_CISplatin	0	0: 2009, 1: 2311
ctx_PACLitaxel	0	0: 2975, 1: 1345
ctx_DOCEtaxel	0	0: 4267, 1: 53
ctx_Pemetrexed	0	0: 3500, 1: 820
ctx_Pembrolizumab (MK3475)	0	0: 3959, 1: 361

Continued on next page

Table A.5 – continued from previous page

Feature	#N/As	Category Frequencies
ctx_CARBOplatin	0	0: 3236, 1: 1084
ctx_Ifosfamid	0	0: 4279, 1: 41
ctx_DOXOrubicin (-HCl)	0	0: 4267, 1: 53
ctx_Atezolizumab (MPDL3280A)	0	0: 4208, 1: 112
ctx_Bevacizumab	0	0: 4273, 1: 47
ctx_PACLitaxel-Nanopartikel	0	0: 4307, 1: 13
ctx_Durvalumab (MEDI4736)	0	0: 4233, 1: 87
ctx_Oxaliplatin	0	0: 4314, 1: 6
ctx_Folinsäure (aus Calciumfolinat)	0	0: 4312, 1: 8
ctx_5-Fluorouracil	0	0: 4305, 1: 15
ctx_VinORELBin (aus -tartrat)	0	0: 3948, 1: 372
ctx_MitoMYcin	0	0: 4316, 1: 4
ctx_Dacarbazin	0	0: 4318, 1: 2
ctx_Gemcitabin	0	0: 4145, 1: 175
ctx_VinCRIStin (-sulfat)	0	0: 4297, 1: 23
ctx_Cyclophosphamid	0	0: 4298, 1: 22
ctx_Epirubicin (-HCl)	0	0: 4316, 1: 4
ctx_Nivolumab	0	0: 4193, 1: 127
ctx_Irinotecan (-HCl 3Wasser)	0	0: 4305, 1: 15
ctx_Melphalan	0	0: 4320
ctx_Ipilimumab	0	0: 4318, 1: 2
ctx_Topotecan	0	0: 4259, 1: 61
ctx_Trastuzumab	0	0: 4317, 1: 3
ctx_PEG liposomales Doxorubicin (-HCl) (Caelyx®)	0	0: 4317, 1: 3
ctx_Cetuximab	0	0: 4316, 1: 4
ctx_etoposid	0	0: 3639, 1: 681
rezept_Capecitabin	0	0: 4315, 1: 5
rezept_Dabrafenib	0	0: 4315, 1: 5
rezept_Erlotinib	0	0: 4221, 1: 99
rezept_Fulvestrant	0	0: 4319, 1: 1
rezept_Imatinib	0	0: 4319, 1: 1
rezept_Lenvatinib	0	0: 4320
rezept_Letrozol	0	0: 4318, 1: 2
rezept_Lomustin	0	0: 4319, 1: 1
rezept_Nintedanib	0	0: 4317, 1: 3
rezept_Pazopanib	0	0: 4320
rezept_Sorafenib	0	0: 4320
rezept_Sunitinib	0	0: 4316, 1: 4
rezept_Tamoxifen	0	0: 4319, 1: 1
rezept_Temozolomid	0	0: 4310, 1: 10
rezept_Trametinib	0	0: 4315, 1: 5
ctx_other	0	0: 4291, 1: 29
rezept_other	0	0: 4176, 1: 144
TTF_Event	0	0: 1215, 1: 3105
death_observed	0	0: 1567, 1: 2753

Table A.5 Table containing a summary of all categorical variables, including their N/A counts and respective category frequencies.

A.3 Code Listings

```python
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch_geometric.nn import GCNConv


class GCN(torch.nn.Module):
    """Graph Convolutional Network"""
    def __init__(self, dim_in, dim_h, dim_out):
        super().__init__()
        self.gcn1 = GCNConv(dim_in, dim_h)
        self.batch_norm1 = nn.BatchNorm1d(dim_h)
        self.gcn2 = GCNConv(dim_h, dim_out)
        self.batch_norm2 = nn.BatchNorm1d(dim_out)
        self.optimizer = torch.optim.Adam(self.parameters(), lr=0.01,
           weight_decay=2e-3)
        self.dp_rate = 0.35

    def forward(self, x, edge_index):
        h = F.dropout(x, p=self.dp_rate, training=self.training)
        h = self.gcn1(h, edge_index).relu()
        h = self.batch_norm1(h)
        h = F.dropout(h, p=self.dp_rate, training=self.training)
        h = self.gcn2(h, edge_index)
        h = self.batch_norm2(h)
        return h, F.log_softmax(h, dim=1)
```

Listing A.1 GCN architecture for treatment failure time classification.

```python
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch_geometric.nn import GINConv


class GIN(torch.nn.Module):
    """Graph Isomorphism Network"""
    def __init__(self, dim_in, dim_h, dim_out):
        super(GIN, self).__init__()
        self.gin1 = GINConv(nn.Sequential(
            nn.Linear(dim_in, dim_h),
            nn.ReLU(),
            nn.Linear(dim_h, dim_h)
        ))
        self.batch_norm1 = nn.BatchNorm1d(dim_h)
        self.gin2 = GINConv(nn.Sequential(
            nn.Linear(dim_h, dim_h),
            nn.ReLU(),
            nn.Linear(dim_h, dim_out)
        ))
        self.batch_norm2 = nn.BatchNorm1d(dim_out)
        self.optimizer = torch.optim.Adam(self.parameters(), lr=0.01,
           weight_decay=2e-3)
        self.dp_rate = 0.35

    def forward(self, x, edge_index):
        h = F.dropout(x, p=self.dp_rate, training=self.training)
        h = self.gin1(h, edge_index).relu()
        h = self.batch_norm1(h)
        h = F.dropout(h, p=self.dp_rate, training=self.training)
        h = self.gin2(h, edge_index)
        h = self.batch_norm2(h)
        return h, F.log_softmax(h, dim=1)
```

Listing A.2 GIN architecture for treatment failure time classification.

```
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch_geometric.nn import GATv2Conv


class GAT(torch.nn.Module):
    """Graph Attention Network"""
    def __init__(self, dim_in, dim_h, dim_out, heads=8):
        super().__init__()
        self.gat1 = GATv2Conv(dim_in, dim_h, heads=heads)
        self.batch_norm1 = nn.BatchNorm1d(dim_h * heads)
        self.gat2 = GATv2Conv(dim_h * heads, dim_out, heads=1)
        self.batch_norm2 = nn.BatchNorm1d(dim_out)
        self.optimizer = torch.optim.Adam(self.parameters(), lr=0.01,
           weight_decay=2e-3)
        self.dp_rate = 0.35

    def forward(self, x, edge_index):
        h = F.dropout(x, p=self.dp_rate, training=self.training)
        h = self.gat1(h, edge_index)
        h = F.elu(h)
        h = F.dropout(h, p=self.dp_rate, training=self.training)
        h = self.gat2(h, edge_index)
        h = self.batch_norm2(h)
        return h, F.log_softmax(h, dim=1)
```

Listing A.3 GAT architecture for treatment failure time classification.

```
import torch
import torch.nn as nn
import torch.nn.functional as F


class MLP(torch.nn.Module):
    """Multi-Layer Perceptron"""
    def __init__(self, dim_in, dim_h, dim_out):
        super().__init__()
        self.linear1 = torch.nn.Linear(dim_in, dim_h)
        self.batch_norm1 = torch.nn.BatchNorm1d(dim_h)
        self.dropout1 = torch.nn.Dropout(0.35)
        self.linear2 = torch.nn.Linear(dim_h, dim_h)
        self.batch_norm2 = torch.nn.BatchNorm1d(dim_h)
        self.dropout2 = torch.nn.Dropout(0.35)
        self.linear3 = torch.nn.Linear(dim_h, dim_out)
        self.optimizer = torch.optim.Adam(self.parameters(), lr=0.01,
           weight_decay=2e-3)

    def forward(self, x):
        h = F.relu(self.linear1(x))
        h = self.batch_norm1(h)
        h = self.dropout1(h)
        h = F.relu(self.linear2(h))
        h = self.batch_norm2(h)
        h = self.dropout2(h)
        h = self.linear3(h)
        return h, F.log_softmax(h, dim=1)
```

Listing A.4 MLP architecture for treatment failure time classification.

A.4 Source Code Access

The project's source code is available on GitHub: https://github.com/aidos-lab/PSIM/tree/main. The repository contains the data preprocessing scripts, implemented algorithms and the experimental setups used in this work. Access to the repository can be provided upon request.

Glossary

C-Reactive Protein Proteins in the blood plasma that are elevated in association with inflammatory response. 39, 40, 41

Carboplatin Chemotherapy medication used to treat various cancers. 38, 41

Cisplatin Chemotherapy medication used to treat cancers. 38, 41

Concordance Index A metric designed to evaluate censored data. 19, 20, 24, 29, 36, 37, 48

ECOG Eastern Cooperative Oncology Group performance status. 32, 33

Erythrocytes Red blood cells. 38

Gamma Glutamyltransferase Enzyme that plays an important role in liver function and detoxification. 38

Granulocytes A type of white blood cells that is important for the body's immune response. 38

Hematoxylin and Eosin stain One of the major tissue stains used in histology. 1

Hemoglobin Oxygen-transport protein present in red blood cells. 38

Lactate Dehydrogenase Enzyme which is released during tissue damage. 38, 39, 40, 41

Leukocytes White blood cells. 39

Lymphocytes A type of white blood cells involved in the body's immune response. 38, 39

MCHC Mean Corpuscular Hemoglobin Concentration. 40

Mean Corpuscular Volume Measure of average volume of a red blood cell used for classification of blood anemia. 38

Monocytes Largest type of white blood cells. 39

Neutrophil Granulocyte Predominant subtype among granulocytes. 40, 41

Vinorelbin Chemotherapy medication used to treat various cancers. It is commonly used for non-small-cell lung cancer. 41

Acronyms

AI Artificial Intelligence. 1, 3

API Application Programming Interface. 11

BCA Body Composition Analysis. 31, 35

BMI Body Mass Index. 31

BN Batch Normalization. 10, 11

BS Brier Score. 20

CHF Cumulative Hazard Function. 23

CPH Cox Proportional Hazards Model. 3, 19, 21, 25, 27, 29, 48

CPU Central Processing Unit. 1, 11

CT Computed Tomography. 3

CV Cross Validation. 29, 36, 43, 45

DL Deep Learning. 2, 3, 5, 10, 11, 12, 19

DT Decision Tree. 22, 23, 24, 33, 48

EGFR Epidermal Growth Factor Receptor. 32

EMD Earth Mover's Distance. 17

FDA Food and Drug Administration. 3

GAT Graph Attention Network. 14, 30, 45, 46

GCN Graph Convolutional Network. 14, 30, 45, 46

GIN Graph Isomorphism Network. 14, 30, 45, 46

GNN Graph Neural Network. 5, 13, 14, 15, 30, 45, 47, 52

GPU Graphics Processing Unit. 1, 7, 11, 26

HR Hazard Ratio. 21

IBS Integrated Brier Score. 20, 29, 36, 37

ICD International Classification of Disease. 31, 35, 45

LLM Large Language Model. 1

LP Linear Program. 16, 17

ML Machine Learning. 3, 5, 16, 24, 27

MLP Multilayer Perceptron. 5, 7, 8, 10, 30, 43, 45, 46, 51

MSE Mean Squared Error. 8, 20

MSI Microsattelite Instability. 32

N/A Missing Value. 29, 32, 33, 35, 36, 37, 38, 39, 41, 48

NLP Natural Language Processing. 5, 7

NN Neural Network. 3, 5, 7, 8, 9, 10, 11, 14, 21, 26, 27, 29, 37

NSCLC Non-Small Cell Lung Cancer. 2, 46, 48

OPS Operation and Procedure Classification System. 31, 35, 45

OT Optimal Transport. 5, 16, 17, 18, 29, 48

PD-L1_TPS Programmed Death-Ligand 1 Tumor Proportion Score. 32

RF Random Forest. 3, 23

RSF Random Survival Forest. 3, 19, 21, 23, 29, 36, 37, 38, 39, 40, 41, 48, 51

SCLC Small Cell Lung Cancer. 2, 46, 48

SHAP SHapley Additive exPlanations. 27, 41, 51

TPS Tumor Proportion Score. 32

UHE University Hospital Essen. 2, 3, 13, 19, 26, 29, 35, 41, 43, 48, 51, 52

UOT Unbalanced Optimal Transport. 18, 29, 30, 43, 45, 46, 47, 48, 51, 52, 53

WL Weisfeiler-Leman. 14

XGB Extreme Gradient Boosting. 24, 25, 26, 48

XGBSE XGBoost Survival Embeddings. 21, 25, 26, 29, 36, 48, 51

List of Figures

List of Tables

Bibliography

[1] *Home*. en. `https://paige.ai/`. Accessed: 2023-6-10. Mar. 2022.

[2] *Stanford CRFM Introduces PubMedGPT 2.7B*. en. `https://hai.stanford.edu/news/stanford-crfm-introduces-pubmedgpt-27b`. Accessed: 2023-8-2.

[3] A. Satariano. "Meta Fined $1.3 Billion for Violating E.U. Data Privacy Rules". en. In: *The New York Times* (May 2023).

[4] *Startseite*. de. `https://www.uk-essen.de/`. Accessed: 2023-6-9. Oct. 2021.

[5] *Cancer*. en. `https://www.who.int/news-room/fact-sheets/detail/cancer`. Accessed: 2023-6-9.

[6] *Lung cancer*. en. `https://www.mayoclinic.org/diseases-conditions/lung-cancer/diagnosis-treatment/drc-20374627`. Accessed: 2023-6-9. Mar. 2022.

[7] *Lungenkarzinom*. de. `https://www.amboss.com/de/wissen/Lungenkarzinom`. Accessed: 2023-6-9.

[8] R. L. Siegel et al. "Cancer statistics, 2023". en. In: *CA Cancer J. Clin.* 73.1 (Jan. 2023), pp. 17–48.

[9] B. C. Bade and C. S. Dela Cruz. "Lung Cancer 2020: Epidemiology, Etiology, and Prevention". en. In: *Clin. Chest Med.* 41.1 (Mar. 2020), pp. 1–24.

[10] Office of the Commissioner. *FDA Authorizes Software that Can Help Identify Prostate Cancer*. en. `https://www.fda.gov/news-events/press-announcements/fda-authorizes-software-can-help-identify-prostate-cancer`. Accessed: 2023-6-10.

[11] X. Chen et al. "Non-invasive early detection of cancer four years before conventional diagnosis using a blood test". en. In: *Nat. Commun.* 11.1 (July 2020), p. 3475.

[12] B. Li et al. "Combinatorial design of nanoparticles for pulmonary mRNA delivery and genome editing". en. In: *Nat. Biotechnol.* (Mar. 2023).

[13] M. Tsuboi et al. "Overall Survival with Osimertinib in Resected EGFR-Mutated NSCLC". en. In: *N. Engl. J. Med.* (June 2023).

[14] D. Mathios et al. "Detection and characterization of lung cancer using cell-free DNA fragmentomes". en. In: *Nat. Commun.* 12.1 (Aug. 2021), p. 5060.

[15] A. Shimazaki et al. "Deep learning-based algorithm for lung cancer detection on chest radiographs using the segmentation method". en. In: *Sci. Rep.* 12.1 (Jan. 2022), p. 727.

[16] P. Nguyen et al. "Active Semi-Supervised Learning via Bayesian Experimental Design for Lung Cancer Classification Using Low Dose Computed Tomography Scans". en. In: *NATO Adv. Sci. Inst. Ser. E Appl. Sci.* 13.6 (Mar. 2023), p. 3752.

[17] Y. Lei, J. Zhang, and H. Shan. "Strided Self-Supervised Low-Dose CT Denoising for Lung Nodule Classification". en. In: *Phenomics* 1.6 (Dec. 2021), pp. 257–268.

[18] L. A. Vale-Silva and K. Rohr. "Long-term cancer survival prediction using multimodal deep learning". en. In: *Sci. Rep.* 11.1 (June 2021), p. 13505.

[19] B. Alsinglawi et al. "An explainable machine learning framework for lung cancer hospital length of stay prediction". en. In: *Sci. Rep.* 12.1 (Jan. 2022), p. 607.

[20] S. Wongvibulsin, K. C. Wu, and S. L. Zeger. “Clinical risk prediction with random forests for survival, longitudinal, and multivariate (RF-SLAM) data analysis”. en. In: *BMC Med. Res. Methodol.* 20.1 (Dec. 2019), p. 1.

[21] J. L. Katzman et al. “DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network”. en. In: *BMC Med. Res. Methodol.* 18.1 (Feb. 2018), p. 24.

[22] C. Lee et al. “DeepHit: A deep learning approach to survival analysis with competing risks”. In: *Proc. Conf. AAAI Artif. Intell.* 32.1 (Apr. 2018).

[23] S. Fotso. “Deep Neural Networks for Survival Analysis Based on a Multi-Task Framework”. In: (Jan. 2018). arXiv: `1801.05512 [stat.ML]`.

[24] D. W. Kim et al. “Deep learning-based survival prediction of oral cancer patients”. en. In: *Sci. Rep.* 9.1 (May 2019), p. 6994.

[25] A. Tong et al. *Embedding Signals on Knowledge Graphs with Unbalanced Diffusion Earth Mover’s Distance.* 2021.

[26] S. Zhang et al. “Deep Learning Based Recommender System: A Survey and New Perspectives”. In: 52.1 (2019). ISSN: 0360-0300.

[27] T. Brown et al. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems.* Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[28] L. Yang et al. “Diffusion Models: A Comprehensive Survey of Methods and Applications”. In: *ArXiv* abs/2209.00796 (2022).

[29] L. Chen et al. “Decision Transformer: Reinforcement Learning via Sequence Modeling”. In: *Advances in Neural Information Processing Systems.* Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 15084–15097.

[30] C. F. Higham and D. J. Higham. *Deep Learning: An Introduction for Applied Mathematicians.* 2018.

[31] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* `http://www.deeplearningbook.org`. MIT Press, 2016.

[32] S. Günnemann. *Lecture notes on Machine Learning.* 2021.

[33] A. K. Bhoi et al., eds. *Bio-inspired Neurocomputing.* Springer Singapore, 2021.

[34] S. Günnemann. *Lecture notes on Machine Learning on Graph and Sequential Data.* 2021.

[35] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization.* 2014.

[36] G. E. Hinton et al. “Improving neural networks by preventing co-adaptation of feature detectors”. In: (July 2012). arXiv: `1207.0580 [cs.NE]`.

[37] A. Oppermann. *Regularization in Deep Learning — L1, L2, and Dropout.* en. `https://towardsdatascience.com/regularization-in-deep-learning-l1-l2-and-dropout-377e75acc036`. Accessed: 2023-6-25. Feb. 2020.

[38] S. Ioffe and C. Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *CoRR* abs/1502.03167 (2015). arXiv: `1502.03167`.

[39] J. Huber. *Batch normalization in 3 levels of understanding.* 2023. URL: `https://towardsdatascience.com/batch-normalization-in-3-levels-of-understanding-14c2da90a338`.

[40] S. Santurkar et al. *How Does Batch Normalization Help Optimization?* 2018.

[41] Z. Zhang, P. Cui, and W. Zhu. *Deep Learning on Graphs: A Survey.* 2018. eprint: `arXiv:1812.04202`.

[42] V. P. Dwivedi et al. “Benchmarking Graph Neural Networks”. In: (2020). eprint: `arXiv:2003.00982`.

[43] J. Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. en. In: *Nature* 596.7873 (Aug. 2021), pp. 583–589.

[44] F. Monti, M. M. Bronstein, and X. Bresson. *Geometric Matrix Completion with Recurrent Multi-Graph Neural Networks.* 2017. eprint: arXiv:1704.06803.

[45] P. Veličković. *Everything is Connected: Graph Neural Networks.* 2023. eprint: arXiv:2301.08210.

[46] K. Xu et al. *How Powerful are Graph Neural Networks?* 2018. eprint: arXiv:1810.00826.

[47] M. M. Bronstein et al. "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges". In: *CoRR* abs/2104.13478 (2021). arXiv: 2104.13478.

[48] M. Labonne. *Graph Attention Networks: Self-Attention for GNNs.* 2022.

[49] D. Alvarez-Melis, T. S. Jaakkola, and S. Jegelka. "Structured Optimal Transport". In: (Dec. 2017). arXiv: 1712.06199 [stat.ML].

[50] J. Lee et al. "Hierarchical Optimal Transport for Multimodal Distribution Alignment". In: (June 2019). arXiv: 1906.11768 [stat.ML].

[51] M. Cuturi. "Sinkhorn Distances: Lightspeed Computation of Optimal Transport". In: *NIPS.* 2013.

[52] L. V. Eyring et al. "Modeling Single-Cell Dynamics Using Unbalanced Parameterized Monge Maps". In: *bioRxiv* (2022). eprint: https://www.biorxiv.org/content/early/2022/10/05/2022.10.04.510766.full.pdf.

[53] M. Arjovsky, S. Chintala, and L. Bottou. *Wasserstein GAN.* 2017. eprint: arXiv:1701.07875.

[54] L. Rout, A. Korotin, and E. Burnaev. *Generative Modeling with Optimal Transport Maps.* 2021. eprint: arXiv:2110.02999.

[55] M. Yamada et al. *Approximating 1-Wasserstein Distance with Trees.* 2022. eprint: arXiv:2206.12116.

[56] A. Williams. *A Short Introduction to Optimal Transport and Wasserstein Distance.* 2020. URL: https://alexhwilliams.info/itsneuronalblog/2020/10/09/optimal-transport/ (visited on 11/01/2023).

[57] F. Hamprecht. *Lecture notes in Computer Vision: Foundations (Summer Term 2020).* URL: https://hci.iwr.uni-heidelberg.de/people/fhamprec (visited on 11/01/2023).

[58] G. Peyré and M. Cuturi. "Computational Optimal Transport". In: (2018). eprint: arXiv:1803.00567.

[59] R. Fisher. *The Earth Mover's Distance.* 2023. URL: https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/RUBNER/emd.htm.

[60] T. Séjourné, G. Peyré, and F.-X. Vialard. "Unbalanced Optimal Transport, from Theory to Numerics". In: (Nov. 2022). arXiv: 2211.08775 [stat.ML].

[61] L. Chizat et al. "Scaling Algorithms for Unbalanced Transport Problems". In: (July 2016). arXiv: 1607.05816 [math.OC].

[62] *C-index.* en. https://square.github.io/pysurvival/metrics/c_index.html. Accessed: 2023-6-11.

[63] E. Longato, M. Vettoretti, and B. Di Camillo. "A practical perspective on the concordance index for the evaluation and selection of prognostic time-to-event models". en. In: *J. Biomed. Inform.* 108 (Aug. 2020), p. 103496.

[64] *Evaluating Survival Models — scikit-survival 0.21.0.* en. https://scikit-survival.readthedocs.io/en/stable/user_guide/evaluating-survival-models.html. Accessed: 2023-6-13.

[65] *sksurv.metrics.brier_score — scikit-survival 0.21.0.* en. https://scikit-survival.readthedocs.io/en/stable/api/generated/sksurv.metrics.brier_score.html. Accessed: 2023-6-13.

[66] *sksurv.metrics.integrated_brier_score — scikit-survival 0.21.0*. en. https://scikit-survival.readthedocs.io/en/stable/api/generated/sksurv.metrics.integrated_brier_score.html. Accessed: 2023-6-13.

[67] *Deep Learning for Survival Analysis*. en. https://humboldt-wi.github.io/blog/research/information_systems_1920/group2_survivalanalysis/. Accessed: 2023-6-16.

[68] Wikipedia contributors. *Proportional hazards model*. https://en.wikipedia.org/w/index.php?title=Proportional_hazards_model&oldid=1156138408. Accessed: NA-NA-NA. May 2023.

[69] B. Efron. "The Efficiency of Cox's Likelihood Function for Censored Data". In: *J. Am. Stat. Assoc.* 72.359 (Sept. 1977), pp. 557–565.

[70] *Machine Learning*. de. https://www.cs.cit.tum.de/daml/lehre/wintersemester-202122/machine-learning/. Accessed: 2023-6-14.

[71] *What is a Decision Tree*. en. https://www.ibm.com/topics/decision-trees. Accessed: 2023-6-14.

[72] J. Thorn. *Random Forest Explained*. en. https://towardsdatascience.com/random-forest-explained-7eae084f3ebe. Accessed: 2023-6-13. Sept. 2020.

[73] Wikipedia contributors. *Random forest*. https://en.wikipedia.org/w/index.php?title=Random_forest&oldid=1153432547. Accessed: NA-NA-NA. May 2023.

[74] *Random Survival Forests*. en. https://www.randomforestsrc.org/articles/survival.html. Accessed: 2023-6-13.

[75] H. Ishwaran et al. "Random survival forests". In: (Nov. 2008). arXiv: 0811.1645 [stat.AP].

[76] *Introduction to Boosted Trees — xgboost 1.7.6 documentation*. en. https://xgboost.readthedocs.io/en/stable/tutorials/model.html. Accessed: 2023-6-21.

[77] D. Vieira et al. *XGBoost Survival Embeddings: improving statistical properties of XGBoost survival analysis implementation*. Version 0.2.3. 2021.

[78] S. Lundberg. *shap: A game theoretic approach to explain the output of any machine learning model*. en.

[79] S. Mazzanti. *SHAP Values Explained Exactly How You Wished Someone Explained to You*. en. https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30. Accessed: 2023-6-20. Jan. 2020.

[80] S. M. Lundberg et al. "From Local Explanations to Global Understanding with Explainable AI for Trees". en. In: *Nat Mach Intell* 2.1 (Jan. 2020), pp. 56–67.

[81] *Permutation explainer — SHAP latest documentation*. en. https://shap.readthedocs.io/en/latest/example_notebooks/api_examples/explainers/Permutation.html. Accessed: 2023-6-30.

[82] *International Classification of Diseases (ICD)*. en. https://www.who.int/standards/classifications/classification-of-diseases. Accessed: 2023-5-23.

[83] *ECOG Performance Status Scale*. 2023. URL: https://ecog-acrin.org/resources/ecog-performance-status/.

[84] A. Bloom. *What is microsatellite instability?* 2023. URL: https://www.mdanderson.org/cancerwise/what-is-microsatellite-instability-MSI.h00-159617067.html.

[85] S. Sigismund, D. Avanzato, and L. Lanzetti. "Emerging functions of the EGFR in cancer". en. In: *Mol. Oncol.* 12.1 (Jan. 2018), pp. 3–20.

[86] L. Huang et al. "KRAS mutation: from undruggable to druggable in cancer". en. In: *Signal Transduct Target Ther* 6.1 (Nov. 2021), p. 386.

[87] *TP53 gene*. en. https://medlineplus.gov/genetics/gene/tp53/. Accessed: 2023-5-23.

[88] Wikipedia contributors. *PD-L1*. https://en.wikipedia.org/w/index.php?title=PD-L1&oldid=1145463897. Accessed: NA-NA-NA. Mar. 2023.

[89] D. B. Doroshow et al. "Programmed Death-Ligand 1 Tumor Proportion Score and Overall Survival From First-Line Pembrolizumab in Patients With Nonsquamous Versus Squamous NSCLC". en. In: *J. Thorac. Oncol.* 16.12 (Dec. 2021), pp. 2139–2143.

[90] E. J. de Ruiter et al. "Comparison of three PD-L1 immunohistochemical assays in head and neck squamous cell carcinoma (HNSCC)". en. In: *Mod. Pathol.* 34.6 (June 2021), pp. 1125–1132.

[91] O. U. Lenz, D. Peralta, and C. Cornelis. "Representing missing values through polar encoding". In: (Oct. 2022). arXiv: 2210.01905 [cs.LG].

[92] P. C. Hart et al. "C-Reactive Protein and Cancer-Diagnostic and Therapeutic Insights". en. In: *Front. Immunol.* 11 (Nov. 2020), p. 595835.

[93] C. E. Olingy, H. Q. Dinh, and C. C. Hedrick. "Monocyte heterogeneity and functions in cancer". en. In: *J. Leukoc. Biol.* 106.2 (Aug. 2019), pp. 309–322.

[94] *OPS*. 2023. URL: https://www.bfarm.de/EN/Code-systems/Classifications/OPS-ICHI/OPS/_node.html.

[95] *TNM Classification of Malignant Tumours*. 2023. URL: https://www.uicc.org/what-we-do/sharing-knowledge/tnm.