Gottfried Wilhelm Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik

Enhancing Interpretability of Machine Learning Models for Survival Analysis using Knowledge Graphs

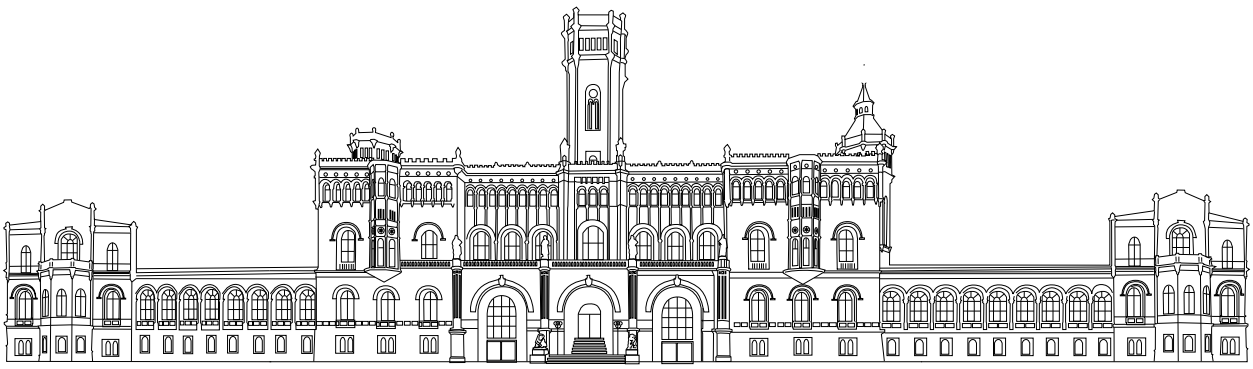
A thesis submitted in fulfillment of the requirements for the degree of
Master of Science (M. Sc.)

BY

Abdulrahman Arnous
Matriculation number: 10000594
E-mail: abdoar1995@gmail.com

First evaluator: Prof. Dr. Maria-Esther Vidal
Second evaluator: Prof. Dr. Sören Auer
Supervisor: M.Sc. Yashrajsinh Chudasama

December 29, 2023



Declaration of Authorship

I, Abdulrahman Arnous, declare that this thesis titled, 'Enhancing Interpretability of Machine Learning Models for Survival Analysis using Knowledge Graphs' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Abdulrahman Arnous

Signature: A.Arnous

Date: 29.12.2023, Hannover

Acknowledgements

I would like to express my deepest gratitude to my professor and supervisor, **Prof. Dr. Maria-Esther Vidal** and **M.Sc. Yashrajsinh Chudasama**, whose expertise, understanding, and patience added considerably to my experience. Their willingness to give their time so generously has been very much appreciated.

I am immensely thankful to my parents, whose love and guidance are with me in whatever I pursue. They are my pillars of strength and have laid the foundations for my values and character. Their sacrifices and unwavering belief in me continue to fuel my desire to move forward.

To my dear wife Raghad, who has been a constant source of love, encouragement, and support. Her understanding and patience during the times I was fully absorbed in my work have made this journey possible. Your belief in me and my work motivates me to strive for excellence.

And to my one-year-old Hisham, who brings endless joy and laughter into our lives. Though too young to understand the effort behind this work, your smiles and energy have been a refreshing and delightful source of encouragement. You remind me every day of the wonder and curiosity with which I started this journey.

I would also like to acknowledge everyone who directly or indirectly contributed to this work. Your support has been invaluable.

Thank you all.

Abdulrahman Arnous
29.12.2023 Hannover
A.Arnous

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rapid development and widespread integration of Machine Learning (ML) and Artificial Intelligence (AI) into various aspects of daily life are indicative of the technological revolution that has occurred in recent years. Once considered a non-core area of computer science, these technologies have now become an essential element of many industries influencing decision-making processes on a global scale [1]. With the expansion of the use of ML, the number of critical decisions that are guided or even made autonomously by ML algorithms is constantly growing. This trend is reflected in a variety of different fields, from medicine to finance and engineering, highlighting the huge impact and the disruptive potential of these technologies [2]. This reliance on automated decision-making systems raises the reliability and traceability of their predictions as a key concern. Even though ML models and AI systems demonstrate impressive capabilities in pattern recognition and prediction accuracy, their interpretability remains one of the biggest challenges [3]. In particular, the so-called "black box models" - systems whose internal decision-making processes are not directly transparent or understandable - pose a significant challenge for users and researchers. The ability to interpret and explain the predictions provided by these models is not only ethically desirable, but also essential for both practical and legal reasons [4]. The current urgent need to interpret and explain ML predictions has led to the development of various interpretation tools [5]. Tools such as LIME [6] and SHAP [7] offer approaches to gain more insight into the decision making patterns of ML models. By breaking down the predictions into contributing factors, they make it possible to capture and understand the significance of certain features for the final result. The interpretability of ML models is of crucial importance, above all in the area of survival time analysis, an essential field of application in medical research and reliability engineering. Survival ML models, predicting the

lifespan or survival of patients under different conditions, offer valuable services for decision-making in medical care and treatment. One such tool is SurvSHAP, which was developed specifically for analyzing survival time data [8]. In the age of data overload, the ability to effectively organize and understand complex information is critical. This is where semantic web technologies come in by enabling a structured and meaningful representation of data. In particular, knowledge graphs, which are rich in structured information and relations, provide a unique opportunity to better contextualize and interpret the decision making processes of ML models [9]. The integration of semantic web technologies and knowledge graphs KGs into the interpretation of survival ML models opens up a new era of model transparency and explainability. The employment of knowledge graphs in the interpretation of survival ML models makes it feasible to visualize and analyze the relationships between different influencing factors and predicted outcomes. This contributes not only to understanding how certain features influence the predictions, but also why these features are significant. By embedding the predictions and their interpretation in a broader semantic context, researchers and users can better recognize and evaluate the underlying patterns and dependencies that shape the predictions. use of semantic web technologies promotes the development of interoperable and extensible systems. By describing data and models in a standardized, semantically rich format, insights and information can be more easily shared and reused between different systems and studies. This is particularly relevant in precision medicine, where the integration and analysis of heterogeneous data sources is essential for personalized treatment approaches. This thesis focuses on the implementation of a new approach to apply the SurvSHAP method and also introduces the InterpretME framework to the field of survival analysis This makes it possible to perform survival data analyses and semantify the results in order to create the InterpretME Knowledge Graph, which describes the process and the results. This thesis is divided into five main chapters: first, the introduction provides an overview of the topic and outlines the structure of the thesis. This is followed by a chapter providing background information on the main technologies and tools relevant to this work. The core of the work is the implementation chapter, this is where a new approach of applying the SurvSHAP process is introduced and discussed in addition to introducing InterpretME tool to the world of survival analysis. This chapter describes precisely each step of the implementation, introduces the developed components as well as the research questions that guided this thesis. The final two chapters focus on the evaluation: here, the test phases and the results of the implemented methods are presented and analyzed, and the research questions posed previously are answered and discussed. Finally, the conclusion provides a summary of the results achieved and draws final conclusions.

Chapter 2

Background

This chapter introduces some basic concepts and technologies used in the scope of this thesis. While this text provides foundational definitions, it does not claim to be exhaustive. For a thorough and detailed understanding of the concepts, it is recommended to consult the referenced literature.

2.1 Knowledge Graphs and Semantic Data

This section lays the groundwork for semantic data handling, focusing on Knowledge Graphs, RDF, SPARQL, and the RML. Each concept is explored in detail, highlighting its role and significance in the broader context of data interpretation and management.

2.1.1 Knowledge Graphs Definition

A KG is a form of a heterogeneous graph, a directed graph in which each node is assigned a type. A heterogeneous graph is described as a tuple $G = (V, E, L, l)$, where:

- $V \subseteq Con$ represents the total number of nodes,
- $L \subseteq Con$ represents the collection of all edge and node labels (Edge/Node Labels),
- $E \subseteq V \times L \times V$ the complete set of edges that define the connections between the nodes,

- And $l : V \rightarrow L$ is what assigns a specific label to each node.

Within such a graph, the nodes (V) represent entities and the edges (E) represent relationships between these entities. The labels (L) provide additional descriptions or information about the types of nodes and edges. Knowledge graphs use this structure to represent complex information and relationships in a way that is understandable and accessible to both human users and machines. They are particularly useful for applications that require rich semantic queries and data integration [10]. A KG can be viewed as a network of various things related to a particular domain or area. KGs are not limited to abstract concepts and relationships but can also contain instances of things like documents and datasets. Figure 2.1 shows an example of a small KG, if Germany is considered a node, 'capital' could be an edge labeled 'is capital of' connecting it to another node, Berlin, representing the capital city:

Figure 2.1: A simple example of a KG.

In the context of KG, ontologies are also frequently mentioned. Ontologies aim to provide a formal representation of the entities in the graph. They are usually based on a specific taxonomy, but since they can contain several taxonomies, they are defined separately. Due to the fact that KGs and ontologies are both represented in a similar way, i.e., by nodes and edges - and are based on the Resource Description Framework (RDF) triple [11], they are usually similar in how they are visualized. For example, the study of a specific venue such as the Camp Nou stadium in Barcelona serves as an example of an ontology. Through an ontology, events in this stadium are differentiated based on a variable such as time. A football club, such as Football Club Barcelona, plays several matches in this stadium in a season. All of these are football matches that take place at the same location. However, each event is differentiated by date and time [11]. The term "*semanticdata*" refers to a special form of numeric data that provides additional information about the meaning of the data entities besides the raw data value. This extra contextual information can enhance the interoperability between different data sources and systems, leading to a more efficient and accurate usage of data. To implement these improvements in practice, various technologies such as the Protocol & Query Language (SPARQL)

query language and RDF play an important role in the Semantic Web community. These techniques can make semantic relationships and meanings explicit so that they can be used by machine learning and reasoning-based systems.

2.1.2 RDF and SPARQL

RDF was originally developed by the World Wide Web Consortium (W3C) as a standard for describing metadata [12]. It serves as a system for representing resources. In practice, RDF is used on the Internet to formulate logical statements about various objects, i.e., resources. Nowadays, RDF counts as a fundamental element of the semantic web.RDFis similar to classical modelling methods such as Unified Modeling Language (UML) class diagrams or entity-relationship models.RDF includes three concepts: the subject, the predicate and the object. In RDF, the object is built by a further resource or also by a further value. These three units form what is known as a triple in RDF. This triple is also called a 3-tuple. In order to be able to perform queries in the RDF data, various query languages have been developed. One example of those query languages is SPARQL.

SPARQL is the query language for the Semantic Web, is adept at querying RDF data, much like how Structured Query Language (SQL) is used for relational data [13]. There are four primary types of SPARQL queries:. There are four primary types of SPARQL queries:

- SELECT: This query specifies a table of all variables like X, Y, etc., that meet certain conditions. It's akin to selecting rows in a SQL query. The result is a table where each row represents a set of variable bindings satisfying the query's conditions.

- CONSTRUCT: This type of query finds all variables such as X, Y, etc., that satisfy specified conditions. It then places these variables into a predefined template to generate new RDF statements. This process effectively creates a new graph, which can be used for further data manipulation or visualization.

- DESCRIBE: This query is used to find all statements in the dataset that provide information about specified resources, identified either by their names or descriptions. The result is a set of RDF triples that describe the attributes and relationships of the resource(s) in question.

- ASK: This query type is used to check if there are any variables like X, Y, etc., that satisfy certain conditions. It returns a Boolean value – true if the

conditions are met by at least one set of variables in the data, and false otherwise. This is particularly useful for validating the existence or absence of specific patterns or data points within the RDF dataset.

2.1.3 RDF Mapping Language (RML)

One more Term is to be explained in this context which is the RML (RDF Mapping Language) [14]. RML is a language for mapping data into RDF records. While RDF provides a general way to describe information, the actual data you want to work with may be in a variety of formats such as XML, JavaScript Object Notation (JSON), relational databases SQL, or others. RML provides a way to describe how this source data should be transformed into an RDF graph. RML provides a way to specify exactly how existing structured data should be transformed into RDF triples [5]. This is important for integrating different data sources and making them interpretable, especially in contexts such as data lakes, data hubs, or any system that handles different types of structured and semi-structured data in this case the InterpretME framework. The relationship between RDF and RML can be described as follows: RDF provides the model for the data and describes how the resources relate to each other. RML, on the other hand, provides the mechanism for translating existing structured data into this RDF model [14]. Therefore, RML is often used as a pre-processing step to transform data into RDF, which can then be queried with SPARQL and integrated into other RDF-based systems.

2.1.4 SHapes Constraint Language (SHACL)

SHACL, serves as a verification tool that RDF graphs comply with certain criteria. These criteria, defined as shapes and constructs, are represented within an RDF graph. In the terminology of SHACL, the RDF graphs used for this purpose are called "shapes graphs", while the ones evaluated against these shapes graphs are known as "data graphs". Not only do shapes graphs in SHACL validate, but they also provide a descriptive framework for the data graphs that fulfill these specific criteria. Beyond validation, these descriptive frameworks have multiple applications, including constructing user interfaces, generating code, and facilitating data integration [15].

2.2 Machine Learning Interpretability

In recent years, ML has made an enormous progress, pushed by increasingly powerful hardware, huge amounts of data and innovative algorithms. Today, ML has the potential to change almost every aspect of our lives, from medicine to automated logistics and a whole lot more. However, while early ML models were often considered "white boxes" whose ways of functioning were clearly understandable and interpretable, modern ML models, especially Deep Learning models, tend to be "black boxes". The term "black box" is often used to describe models whose internal workings are very complex to fully comprehend [6]. Users may input data into the black box to receive an output (prediction), however the process by which the data is processed and the utilization of this data by the ML model to finally deliver a prediction remains largely unexplained and unknown. The black-box characteristics of such models present a big challenge, especially when it comes to ethical, legal or safety issues [16].

This leads to an urgent need for machine learning interpretability. Interpretability goes far beyond academic discussions and has profound ethical and societal impacts, and it implies the ability to answer the question "what caused such a prediction?". The lack of interpretability of ML models could lead to incorrect medical diagnoses, unjust legal judgments and even financial ruin. Therefore, it is essential to understand the mechanisms behind predictions in order to minimize these risks and reap the full benefits of the technology [6]. A scenario in which an advanced black-box machine learning model is available to analyzeMagnetic Resonance Imaging (MRI) scans and detect early signs of cancer cells. The model claims to be able to detect cancer symptoms before any noticeable signs appear. Despite the model's exceptional accuracy rates, neurologists and radiologists are confused by its decision-making process. This lack of transparency, despite the technology's promising potential, could ultimately prevent the technology from being widely adopted due to concerns about its reliability and the impact on patient care.

ML Interpretability can address this issue by making the model's decision-making processes explainable and reveals the most important features for a prediction recognizable. Interpretable ML models facilitates doctors to improve their understanding of the results and now take appropriate early treatment actions. Thus, interpretability is essential not only for physician's trust in the technology, but also for compliance with ethical and legal standards. Various techniques for interpreting black-box models have evolved in the pursuit of transparency and interpretability in machine learning. Among the most promising approaches are LIME and SHAP. [6]. Both techniques offer unique ways to address the often-mysterious decision-making pro-

cesses of modern machine learning models, each one offers theoretical and practical advantages.

2.3 Local Interpretable Model Agnostic Explanation (LIME)

Figure 2.2: LIME is a post-hoc explainable framework proposed by [6], that generates local explanations for ML models [17].

LIME is a method for interpreting the predictions of sophisticated ML models, where the prediction is difficult to interpret. LIME logo is shown in Figure 2.2. LIME is based on the idea of approximating complex models by simpler local linear models that are particularly close to a given observation [6]. This process involves several sequential phases as follows:

- Sampling: Sampling of data points around the observations that is intended to be explained, these data points are permutations of the original observation.
- Distance Calculation: The distance of all permuted observations to the original observation should be determined for example using Euclidean distance.
- Weighting: Similarity measurement between the real observations and the sampled data points must first take place. This can be achieved by converting the distance values into similarity values, most likely using a kernel function (exponential kernel that converts the distance into a weight between 0 and 1) and then weighting based on the similarity score.

- Model Fitting & Training: Fit a simple model (for example, a linear model) to the permuted data. The similarity weights from the weighting step are used as weights for the observations to train the simple model.
- Interpretation: Interpretation of the simple models that give insight into the complex model now feature weights are more easily to be extracted from the simple model. These features weights serve as explanations for the local behavior of the complex model. For example, a high positive weight for a particular feature may indicate that the feature has a strong positive influence on the prediction of the complex model.
- Although LIME is one of the most popular methods of ML explanation, it has some disadvantages like the fact that it is limited to local approximations which may not be globally applicable [6].

Advantages of LIME [6]:

- Feature flexibility: explanations created using LIME can include different features than the original black box model. This can be particularly advantageous when the original features are difficult to interpret.
- Well-researched: the models that serve as local surrogates are well researched in terms of their training and interpretation.
- Model agnostic: LIME is not tied to a specific model, which facilitates changing the underlying black-box model.
- LIME is suitable for different types of data, including tabular data, images, and text.
- Selectivity and contrast: when using methods such as Lasso or short trees, the resulting explanations are selective and can be presented in high contrast.

Disadvantages of LIME [6]:

- Sampling Challenge: Sampling of new data points is mainly done using estimated feature distributions through normal distributions. This does not consider correlations between features, which can lead to the generation of unrealistic data points.
- Core selection: Choosing the right local kernel is problematic because explanations depend heavily on its parameterization.
- Instability: explanations for two very similar observations may differ significantly.
- Linear regression, the interpretation depends strongly on the intercept.

2.4 Shapley Additive Explanations (SHAP)

Shapley Additive Explanations is a method designed to help explaining the predictions of complex machine learning models, icon shown in Figure 2.3. In fact, the concept is based on the Shapley values from cooperative game theory [18].

Figure 2.3: SHAP is a post-hoc interpretability framework proposed by [7] that offers local explanations for machine learning models [19].

As SHAP is model agnostic, it is capable to offer a consistent approach to quantify the contribution of all features to a given prediction in any machine learning model. Shapley values make it possible to analyze the predictions of the model in such a way that the contribution of an individual feature to the model prediction becomes transparent. As a result, it is possible to obtain a local explanation for each observation. However, by aggregating the contribution of each feature over several observations, the global average contribution of each feature to the model prediction is obtained [20]. A simple illustrative example

$$f(s) = f(s_1, s_2, \ldots, s_n) = \beta_0 + \beta_1 s_1 + \beta_2 s_2 + \cdots + \beta_n s_n \tag{1}$$

This equation represents a linear regression model, where $S \in X$ refers to an observation from the dataset X which has n different characteristics X. First the aim is to explain the prediction for observation s locally [20]. In practice the most important is how a particular feature effects the model prediction:

$$\phi_i(s) = \beta_i s_i - \beta_i E\left[xi\right] \tag{2}$$

The contribution is the difference between the feature effect and the average effect. (fix) may also be rewritten as:

$$\phi_i(s) = f(s_1, s_2, \ldots, s_n) - E\left[f(s_1, s_2, \ldots, s_n)\right] \tag{3}$$

This equation represents the difference between the prediction for an observation and the expected prediction for that observation in the case where the first characteristic is unknown [21]. The Shapley values provide a way to calculate the "fair" contribution of a player in a cooperative game. With that in mind, if machine learning model prediction was envisioned as such a game, Shapley values can be used to determine the contribution of each feature to the prediction. Shapley values of a player $i \in X \subset N$in a cooperative game (f, N) can be defined as follows [20]:

$$\phi_i(f) = \sum_{S \subset N \setminus \{i\}} \frac{|S|!\,(|N| - |S| - 1)!}{|N|!} (f\,(S \cup \{i\}) - f\,(s)) \tag{4}$$

- All subsets that do not contain the player are summed.
- $f(S)$is the prediction of the model when only the features in the set S are applied.
- $|S|$ is the amount's cardinality, i.e., the number of elements in S.
- $|N|$ is the cardinality of the set N, i.e., the total number of features.
- The fraction serves as a weighting factor for every possible subset S of features.

SHAP presents an ML model interpretation method that provides consistent and uniform explanations despite the high implementation complexity and computational effort and can provide both global and local model interpretations [21]. It often provides better accuracy than LIME as, unlike LIME, it examines the entire data set. A real-life example is presented below to help better understand the essence and importance of Shapley values.

Example: Shapley Values

A hypothetical coalition consisting of three actors - Noah, Hisham, and Anna is considered. Each of these actors brings specific capabilities to the coalition that collectively generate a total gain of 11 units. To simplify the dynamics, it is assumed that the coalition is joined sequentially. The goal, based on the marginal contribution to the total gain, is to allocate the reward to each actor.

If Noah, was the initiator of the group and the initial group gain was 5 units, which increased to 9 and 11 units through successive accessions by Hisham and Helena, respectively, this results in an initial payoff distribution of is derived. It is found that if Hisham 's and Helena's abilities are similar or overlapping, a different marginal contribution could be provided depending on the joining order. This implies that the order of joining the group has a substantial impact on the distribution of total output. To address this inequality, a methodological approach could be taken that considers all possible orders of entry. For this case, these would be six possible sequences (NHA, NAH, AHN, ANH, HNA, HAN). For each sequence, the marginal contributions of each actor could be calculated, summed, and then divided by the number of possible sequences. This would determine the average marginal contribution of each actor. In cooperative game theory, this procedure is formalized by the use of Shapley values. However, for larger numbers of actors or characteristics, this procedure is considered impractical because the number of sequences to consider is m! (m factorial) is.

Figure 2.4: Illustrating example for Shapley Values with 3 actors.

SHAP:

The calculation of Shapley values determines the relevance of a feature within an observation and provides local explanations. The main idea of the method is to simply compare the prediction of the ML model for every feature with and without the features presence. However, the sequential ordering of the features can affect the prediction, therefore all orders and combinations of other variables must be considered. To provide a global view of the model behavior this this calculation rule must be applied to all observations [7].
Advantages:

- Explanations range: Provides Local and global model explanations.
- Mathematically supported: Shapley values is based on solid mathematical foundation, based on principles such as efficiency, symmetry, and linearity. LIME for example, is based on the heuristic assumption that the model behavior is locally linear.
- Shapley values is model agnostic meaning it applied with any ML Model
- The deviation between the single prediction and the model mean is distributed evenly over the feature values following the axioms. This is ensured by the efficiency axiom.

Disadvantages:

- Selective explanations cannot be made, Shapley values explanations always involve all features.
- Shapley values can produce unrealistic feature values, especially if the features are correlated.
- Only give one value per feature, rather than a full forecasting model. Therefore, they cannot reflect changes in the forecast due to adjustments to the features.
- Computationally intensive, which makes it a slow method.

2.5 Survival Shapley Additive Exlanations

The term SurvSHAP was introduced in [21], and it focuses on the interpretability of ML Models where time is a crucial element. In fact, both LIME and SHAP are model agnostic methods that can deal with all machine learning models, but both are lack the ability to deal with data sets that are time dependent. SurvSHAP extends the SHAP concept and makes it applicable to survival analyses. The term *survival analyses*, is concerned with predicting the time until a certain event occurs. This event can for example be the heart failure of a patient, e.g., [8], the failure of a device or any other expected event. A unique feature of survival analysis is its ability to handle censored or observational data, for example, a data set which has a patient information may not have the survival time of that patient. This means that in the observed time the event (e.g., patient death) did not accrue. In other words, the survival time is longer than the observe time. From a mathematical perspective [21] an entity n in survival analysis can be represented as a triplet(x_n, y_n, δ_n), where$x_n = [x_n^1, x_n^2, x_n^3, ..., x_n^p] \in R^p$. x_n is considered a collection of entities that are associated with the survival time T_n, δ_n corresponds to the occurrence of the event and y_n represents the observation time, this time can either be the actual survival time T_n when $\delta_n = 1$ or, in case the event does not occur $\delta_n = 0$ the censoring time C_n. T_n is therefore an indeterminate value for censored observations. The role of survival analysis in this mathematical context is the ability to predetermine the survival time T_n for a new entity S with a triplet of (x_s, y_s, δ_s). A function of time is more likely to be predicted instead of a single time event. In this context, two main time functions are of concern [8]:

- Survival function $S(t)$, describes the probability of an entity (e.g., patient) surviving until a certain time without experiencing the event (i.e., death)

$$S(t) = P(T > t) = 1 - P(T \leq t) \tag{5}$$

- Hazard function $h(t)$, describes how high, the risk of an event occurrence, is over a very short time interval, assuming the event in question has not happened yet at that point in time.

$$h(t) = \lim_{t \to 0} \frac{P(t \leq T < t + \Delta t \,|\, T \geq t)}{\Delta t} \tag{6}$$

  Both functions are related as:

$$h(t) = \frac{f(t)}{S(t)} \tag{7}$$

Where $f(t)$ represents the Probability Density Function (PDF) of the studied event, and it can be mathematically described as:

$$f(t) = -\frac{dS(t)}{dt} = h(t) \cdot S(t) \tag{8}$$

This can be transformed to:

$$S(t) = exp(-H(t)) \tag{9}$$

knowing that the cumulative hazard function is given as:

$$H(t) = \int_0^t h(s)\, ds \tag{10}$$

When the survival function is combined with the right kind of data aggregation, it's possible to get precise predictions about individual characteristics. For example, it can predict the amount of time before an important event is likely to happen for each individual case. This process involves taking complex data and breaking it down into clear, detailed insights about specific outcomes [8].

2.6 LIME vs SHAP vs SurvSHAP

To better understand the distinctions among LIME, SHAP, and SurvSHAP, we present a comparative analysis in the table below.

Attribute	LIME	SHAP	SurvSHAP
Basic principle	Local Approximation	Game theory	Game theory in survival analyses
Computational complexity	Moderate	High	Very high due to survival analysis
Model Agnostic	Yes	Yes	Yes
Explanation type	Local	Local & Global	Local & Global
Stability	Unstable	Stable	Stable
Efficiency by complex models	Inefficient	Efficient	Efficient
Time Dependency	No	No	Yes

Table 2.1: Comparison of Model Explanation Methods LIME, SHAP, and SurvSHAP [22].

2.7 InterpretME

This section describes the InterpretME framework, delving into its functionality and primary attributes. It presents the principal pipeline of InterpretME, outlining the input and output parameters, configuration choices, and the description of each stage in the pipeline. To conclude, a motivating example is presented, showcasing the significance and real-world utility of InterpretME.

2.7.1 What is InterpretME?

InterpretME is a framework developed by the members of the Scientific Data Management group at the TIB Leibniz Information Centre for Science and Technology. Figure 2.5 depicts the different layers of InterpretME. InterpretME integrates KGs with machine learning techniques. InterpretME goal is to draw deep and interpretable insights from data and ML models [23]. At its core, InterpretME enables the integration of machine learning and semantic web technologies.

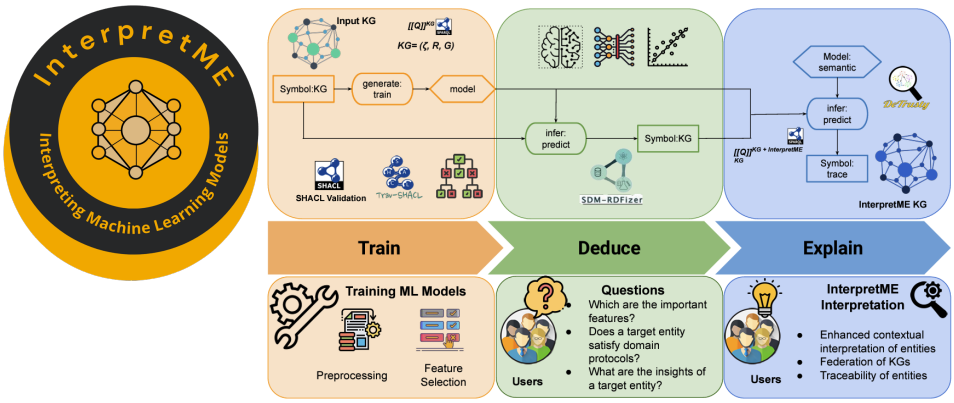


Figure 2.5: InterpretME framework official logo and the different layers of InterpretME [24], [23].

A key feature of InterpretME is its ability to produce decisions that are understandable by both humans and machines. This is especially important in an era where transparency and explainability of algorithms is paramount. InterpretME accepts data in two forms, KGs (via SPARQL endpoints and queries) and standard formats such as $Comma - SeparatedValues(csv)$ or $JSON$.The system is designed to provide detailed visualizations of the main attributes of trained learning models within a KG. This means that it doesn't only analyze data, but also understands the relationships and links between data points in the KG. InterpretME can extract information such as feature details, classes and SHACL constraints from KGs. It classifies the important features into different groups, making it easier to analyze the data and make predictions later. Another notable feature is InterpretME's ability to accurately trace the origin of an attribute or feature in the KG. This provides an additional layer of transparency and understanding for the end user. InterpretME comprises three layers shown in Figure 2.5. The first layer is the training layer, where the input data is taken in, processed, and thoroughly checked. Here, the examined features from the data are identified and used. This layer focuses on training the machine learning model and includes all related tasks, such the hyperparameter tuning. The Deduce layer, tasked with generating predictions using the trained ML model and unveiling the most critical features. This layer is pivotal in translating the model's complex calculations into actionable insights. Upon the generation of predictions, the Explain layer comes in. This layer is characterized by its ability to make predictions interpretable and to trace the decision-making process of the ML model. The output of this layer is the InterpretME Knowledge Graph (KG), which

provides a detailed and structured interpretation of the whole process. The combination of the three layers simplifies data analysis for the user and enables seamless tracking of the predicted entity with all associated attributes in the original KG. In addition, the user gains access to various metrics such as accuracy, recall, and precision, which are presented in combination with LIME interpretations feature. Overall, InterpretME provides a comprehensive solution to the interpretability challenges in machine learning by combining the strengths of KGs and modern machine learning and interpretability methods. It is a step towards more transparent, accountable, and explainable AI.

2.7.2 InterpretME's Inputs & Outputs

To further explain how InterpretME works, it is crucial to understand, what inputs InterpretME expects and is capable of handling, and what kind of output is expect. InterpretME can be used by calling its pipeline function, which is the core of InterpretME. InterpretME's pipeline requires the following inputs to start its process:

- Configuration file path: This input indicates the path pointing to a JSON file which contains several parameters that directly affect the InterpretME pipeline and allow users to add their configuration and preferred set of properties. These parameters can be divided into the following categories:
  - Data location: Path to where the input data is located, if this is a Server URL, this means that the input is a KG, or it is a file path which indicated that the input is a local dataset (i.e., csv file).
  - Data structure and type:
    - Type: type of data (e.g., patient).
    - Index_var: key or index variable.
    - Independent_variable: Features used for predictions.
    - Dependent_variable: Target variable or in other words what is to be predicted.
    - classes: Possible categories or classes of target variables (e.g., 0 = *Dead* and 1 = *Alive*).
  - Modeling and training: Enables the user to choose his preferable ML model, which sampling strategy must be applied, number of important features to be considered by the ML model, or even configuring the hyperparameter auto tuning using AutoML.

- Result paths: Path to where the results of the used interpretation method should be saved.
- Server-Details: In case data needs to be uploaded from an external server (e.g., SPARQL endpoint of the InterpretME KG, Credentials)
- Model parameters: In case the user doesn't want to use the configuration JSON file, this allows the user to specify various parameters for the machine learning model, such as the sampling strategy, the number of cross-validation folds, important features, and the ratio of training to test data.

InterpretME outputs a dictionary containing the results of the ML model interpretation process, the analyses and the made ML predictions. The user is also able to use some of the plotting functions that are provided within the scope of InterpretME. Now that all the inputs and outputs of InterpretME have been outlined, the following will briefly explain how InterpretME works and what are its main functions.

2.7.3 Implementation & Pipeline

InterpretME runs as a pipeline, which is designed to perform several tasks related to data processing, training of machine learning models, interpretation, and visualization of result. In the following is a breakdown of the main functions and flow of the pipeline, where every stage will be divided into tasks and those will be briefly described:

1. Initialization:
    a. Unique identifier: At the beginning of each pipeline run, a unique identifier will be assigned. The identifier serves as a run_ID for the current run, so that the results and data from that run are later easily identified and mapped.
    b. Progress Bar: To provide a user-friendly experience and to keep track of the progress of the different stages of the pipeline, a progress bar is implemented. This bar keeps updating in real time and gives the user visual feedback on how far the process has progressed and which tasks are still pending.
    c. Directory creation: Checks whether certain necessary directories for saving files exist. In case those directories are not present, they will be created automatically. This ensures that all results and files are systematically stored in the defined directories.

d. Statistics Initialization: Statistics collection is activated at the very beginning of the pipeline to gather detailed information about the run. This provides the ability to collect data such as run time, parameters used, and other relevant information that could be useful for analysis and review later.

2. Data Importation:

a. Configuration file: Configuration file: The pipeline begins by reading the JSON configuration file. This file contains specific data details and settings that determine the flow of the pipeline, as this is an expected input for InterpretME which was described in 2.7.2. Based on the information specified in the JSON configuration file, it will be determined whether the data originates from a KG or a local data set. This is a crucial distinction, as the following pipeline stages vary depending on the data source. In case the data source is a KG, this means that InterpretME will grab the KG and extract the necessary data from it using SPARQL query to construct a dataset that will be used in next stages or in case the data source is already a dataset a direct move to the next stage is allowed.

b. User-specific parameters: The user might want to specify additional parameters alongside the configuration file settings when calling the pipeline. InterpretME validates these parameters and utilizes them as needed. For instance, if the user wants to use a different sampling strategy than the default one specified in the configuration file, the pipeline will check if those were provided, if not the default strategy will be used.

3. Data Preprocessing:

a. Data loading: This is where the data is actually loaded, either from a locally stored dataset or from an external SPARQL endpoint. InterpretME ensures that the data is loaded without errors and in its entirety to guarantee seamless performance in the subsequent stages.

b. Data encoding: After data is loaded, encoding is often required, especially if the data involves categorical values. Encoding converts categorical data into a form that can be efficiently processed by machine learning models. This could be achieved, for example, through "one-hot coding" or by assigning numeric values to categories.

c. Data Sampling: InterpretME offers several sampling strategies,e.g., oversampling, undersampling to obtain a representative dataset for machine

learning. This is the stage where data sampling takes place. This process could include, for instance, randomly selecting a portion of the data, performing subsampling or oversampling, etc. Once this process is complete, relevant information about the strategy used and the resulting dataset are stored, this is useful for later analyses.

4. ML Model Building & Classification:

    a. Classifying data: Once the data has been prepared and sampled, it will be classified using the selected machine learning model. The current version supports an ensemble machine learning models like random forest or Gradient Boosting, and the user can specify a specific model, or the default model specified in the configuration file will be applied.

    b. Model training and predictions: The goal of this step is training the machine learning model with the sampled dataset. Once the model training process is complete, InterpretME utilizes the trained model to generate predictions and evaluate the efficiency of the model. The classification function is the core of this step. It selects the appropriate classification strategy based on the number of classes specified by the user. The two main strategies are briefly discussed below:

        i. Binary classification (binary_classification): This method is used when there are only two classes in the dataset to be distinguished. InterpretME uses a decision tree algorithm combined with hyperparameter optimization by AutoML (uses the Optuna library) to select the best model parameters. Those parameters will be applied to the user's model choice weather it is Random Forest, AdaBoost, or Gradient Boosting. Another important part of this process is Feature Importance, where the most important features are selected for classification. This process differs by model:

            · **Random Forest:** The importance of a feature is calculated by the average decrease in impurity (usually measured by the Gini coefficient or entropy) that this feature causes when it is used in the trees of the forest. A higher value indicates greater importance.

            · **AdaBoost:** The feature importance is determined by the number of times a feature is used as a decision threshold in the weighted trees.

- **Gradient Boosting:** Much like Random Forest, the importance of a feature is determined by the average decrease in impurity across all trees.

After the model is trained, feature weights are extracted. The features are then sorted in descending order of importance. The plot_ feature _importance method takes the sorted feature weights and visualizes them. This allows users to identify the most important features in the dataset. In the provided features, the top X features (based on number_important_features in the user's defined JSON configuration file) are selected for further analysis and model training.

ii. Multiclass classification (multiclass): This method is used when there are more than two classes. After selecting the best model, "Feature Importance" is performed to determine the most important features for classification.

The separation of binary and multiclass classification in model development was necessary because of the obvious differences between these two types of classification. Algorithms optimized for binary problems might not perform optimally in a multiclass scenario and vice versa. In addition, the evaluation metrics differ in the two contexts, which means that model evaluation must be handled differently. Even though it is possible to combine the two functions into one, this could make the code cluttered and difficult to maintain. Different considerations of metrics, data imbalance, and optimization strategies for each type would overload the code and make it unnecessary complex.

5. Model Interpretation: Depending on the pipeline input parameter survival, either LIME or SurvSHAP will be used to interpret the model and enable the user to understand why the model made certain decisions.

6. Preparing output for plotting: The results are processed and optimized for subsequent visualization. The output data is expected to be useable directly to create plots, statistics, or other visual representations.

7. Semantifying of results: Transforming data into a machine-readable format such as RDF. This is the part where InterpretME outperforms other ML interoperability frameworks or tools, it is what gives InterpretME its unique nature. This approach makes it possible to store and share the data in a standardized format that can be easily understood and interpreted by both users and machines or applications.

8. Upload to Virtuoso: After the data has been semantically enriched and converted to a machine-readable RDF format, users have the ability to upload it to the Virtuoso database in order to be stored. Virtuoso is a specialized database management system designed for RDF data. By uploading data to Virtuoso, it can be stored efficiently and easily queried through SPARQL endpoints.

9. Performance monitoring: Several metrics and statistics are captured during the pipeline to monitor the performance and efficiency of the process. These include time measurements for various stages of the pipeline, the number of data points processed, and other relevant information. These collected statistics provide valuable insight into how the pipeline is performing and can be used to identify and make potential optimizations or adjustments in the future.

2.8 Motivating Example

There are many options to configure InterpretME as it offers its users multiple features and gives them the freedom to choose from those features. To gain a better understanding what those features offer, a motivation example, Figure 2.6, will be described and discussed in this section.

The motivation behind InterpretME derives from the fact that there is an obvious gap in the need for automated support around merging predictive machine learning modeling systems with KGs. InterpretME focuses on providing orientation by tracking and explaining the predictive models's decisions. The recently developed automated machine learning systems that do have optimized and automated predictive ML modeling processes, fall short when it comes to providing interpretations for its predictions that are both understandable by humans and usable by machines. While tools such as LIME are capable of generating interpretations for specific prediction tasks, they do not provide concrete explanations of the target entity's interpretations and do not consider the characteristics of the target entity within the prediction model, and they don't take into consideration the input data to be derived from KGs.

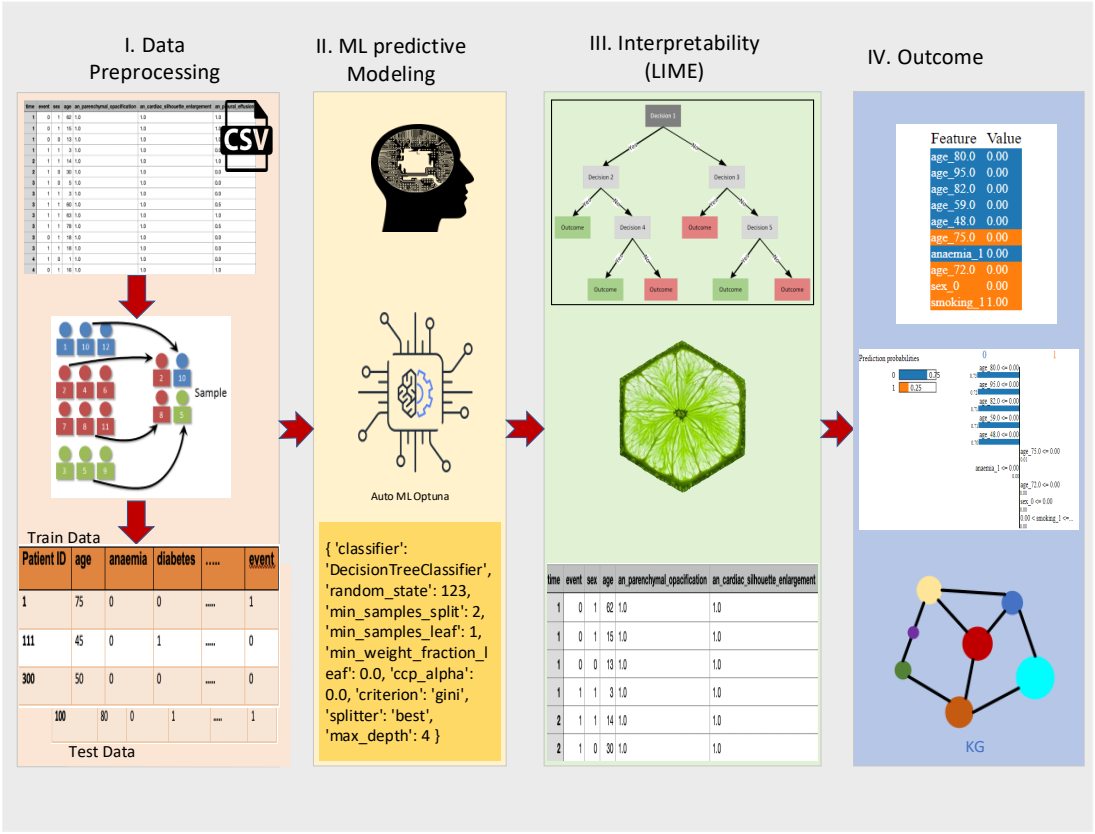


Figure 2.6: Motivation example showing InterpretME's pipeline stages and process.

Figure 2.6 depicts a pipeline in InterpretME. The stage *I. Data Preprocessing* utilizes a dataset taken from a csv file containing radiology reports of various patients in text form, indicating their length of stay in the hospital and an event indicating whether a death occurred during their stay. The dataset includes key features that detail the main characteristics of patients with pulmonary death, such as *patient identifier, gender, age, pleural effusion, pr_firstorder_MeanAbsoluteDeviation*, and other bio-markers detected by radiology. The input data is a csv file, this data must first be prepared and preprocessed through different steps, like data sampling and data splitting which will split the dataset into test and train, as LIMEis unable to deal with time, the time column will be filtered in the scope of the data preprocessing. The predictive task in this case is a binary classification that predicts the death of the patient. Stage *II. ML predictive Modelling* is where AutoML performs the hyperparameter optimization process, and based on its recommendations, models

such as random forests or decision trees are applied with the hyperparameters for the classification task. In stage *III. Interpretability* LIME interpretability tool and decision trees are used to provide localized interpretations for each patient in the test dataset. Decision trees help to identify the significant features that contribute to the model results. Stage *IV. Outcome* is where the LIME results are plotted, which will help the user to understand the contribution of each feature in the predicted decision in addition to transforming the results into a KG. Many questions arise with this motivation example, most importantly:

- Can a specific patient be interpreted by LIME?
- What distinguishing features do patients possess?
- For this classification, what other features are of importance?
- How would the analysis be impacted if time dependency played a crucial and significant role?

2.9 Chapter Summary

In the "Background chapter", the unique challenge posed by time-dependent data (survival data), is identified as a significant limitation to the capabilities of InterpretME. This type of data, which captures events in a temporal context, requires specialized treatment beyond what current tools such as AutoML or LIME can provide. Given this limitation, there is an urgent need to fundamentally renew InterpretME and develop it into a more versatile and widely applicable tool. By specifically integrating capabilities to analyze time-dependent data, especially survival data, InterpretME will be revolutionized. This significant enhancement enables InterpretME to address the complex demands of survival data analysis while adding new levels of interpretability and clarity to predictions. The primary focus is to provide accurate and reliable insights into survival data that are understandable to both professionals and non-experts. This innovation positions InterpretME as a leading tool in the world of advanced data analytics, providing unparalleled support for decision-making and strategic planning. This innovation positions InterpretME as a crucial component in medical research and various other domains, effectively bridging the divide between complex data processing and user-friendly, understandable interpretation. Consequently, InterpretME establishes new benchmarks in data analysis and emerges as an essential instrument for any user seeking to obtain profound insights from their data.

Chapter 3

Implementation

In this chapter, the focus is on the implementation of the survival analysis and the necessary adjustments that were carried out in InterpretME to integrate the survival analysis. The integration of SurvSHAP into the existing code is of particular interest. The implemented functions and mechanisms of SurvSHAP are covered and discussed in detail. These explanations provide a description of the technical aspects of customizing InterpretME to meet the demanding requirements of survival analysis. It also discusses how this integration extends the functionality of InterpretME and significantly enriches its scope. The implementation steps are described with precision and detail to provide a comprehensive understanding of the technical challenges and the applied solution strategies.

3.1 Comparison between regression and survival analysis

Within the framework of this thesis, an overview comparison between regression analysis and survival analysis is to be carried out, which highlights the most important points of the two analysis methods, and these are described in more detail in the following:

3.1.1 Definition and Areas of Application

Regression analysis is a statistical method used to model the relationship between a dependent variable and one or more independent variables. It is used in various areas like economics, social sciences, and ecology, while survival analysis is often

used in fields where time is of the essence like medicine specially in survival studies or in reliability engineering [25]. An illustrative example of survival analysis can be found in medical research, where a dataset might be compiled to investigate patient survival rates. Such datasets typically include features like age, medical conditions (e.g., anaemia, diabetes), lifestyle factors (e.g., smoking status), and patient sex. Additionally, they record the time until an event of interest occurs (often death or relapse) and whether this event occurred during the study period.

In survival analysis, this data is used to model the time until the event, taking into account the various recorded factors. This allows researchers to understand how different variables influence patient outcomes over time, providing invaluable insights into patient prognosis and treatment efficacy.

3.1.2 Input Data Types and Methodology

In regression analysis, continuous, discrete, ordinal, or categorical data are used. Linear regression and logistic regression are the most commonly used forms of regression, in addition to that no consideration to data censoring (the case in which the event has not yet occurred at the end of the study) takes place in the regression analysis. When it comes to survival analysis, time-to-event data is often used. In contrast to the regression analysis, censoring is a fundamental feature in survival analysis where methods such as survival random forest or the Cox proportional hazards model are often used [26].

3.1.3 Focus and Model Assumptions

While regression analysis explores the relationship between variables and evaluates the strength and direction of such relationships, survival analysis is concerned with examining the timing of events and identifying factors that are associated with risks. The assumptions of the models also differ greatly between the two analysis methods. In regression analysis, assumptions such as linearity, normal distribution of residuals, independence, and *homoscedasticity* vary depending on the model type. On the other hand, in survival analysis, the assumptions of the model are that survival times are independent and that the hazard rate may vary over time [26].

3.2 Survival Analysis over InterpretME

Although approaches such as LIME are useful in providing interpretable models for classical prediction tasks, they reach their limits when it comes to time-dependent

events, which are common in survival analysis. While LIME can show the importance of features for a prediction, it does not consider the time element, one that is critical for survival data. This is where SurvSHAP steps in as it was specifically designed to meet the needs of survival analysis. SurvSHAP provides an understanding of the impact of various factors on patient survival over time. Figure 3.1 presents a running example of InterpretME with SurvSHAP integrated, utilizing the same dataset as referenced in section 2.8. However, in this example, the time column is included (survival data).

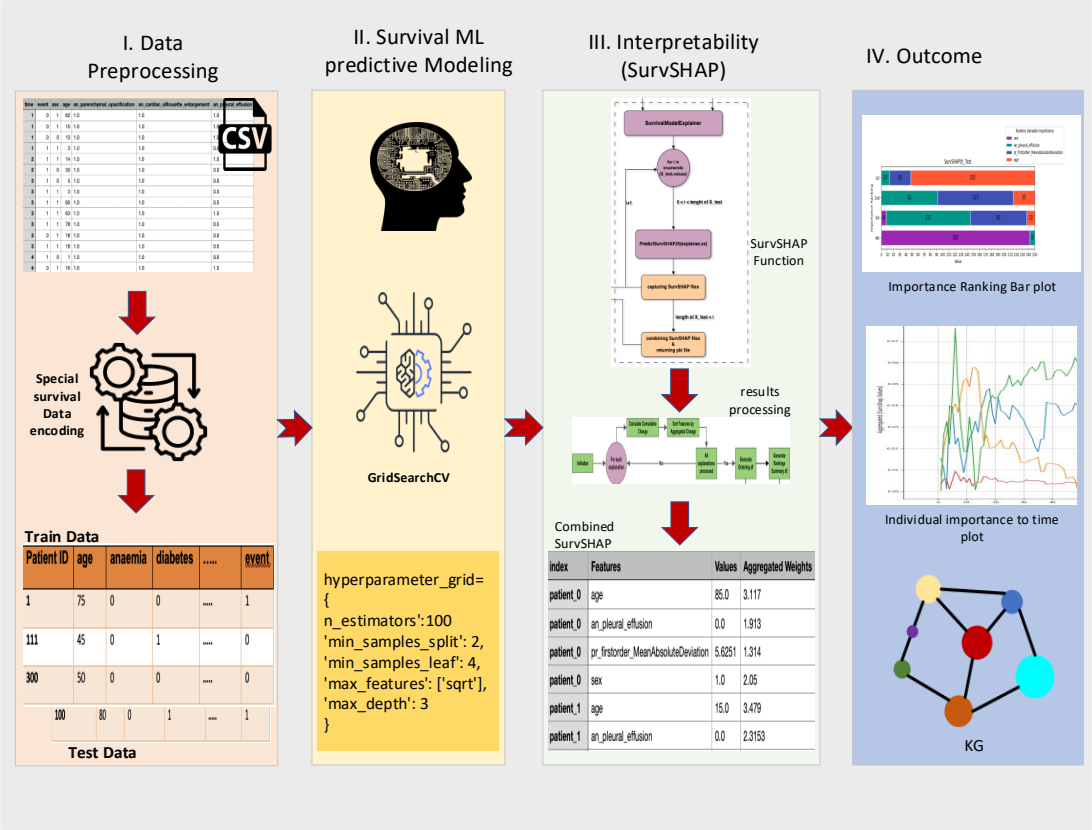


Figure 3.1: Running example of integrating SurvSHAP into InterpretME along with InterpretME's pipeline stages.

In stage *I. Data Preprocessing*, the processes are tailored to accommodate the

unique characteristics of survival data. This involves a distinct method for encoding the input data, details of which are elucidated in the coming sections. After encoding, the dataset is partitioned into training and testing subsets. Stage *II. Survival ML predictive Modeling* focuses on hyperparameter optimization (tuning) to identify the most effective parameters for the survival machine learning model. For this purpose, Grid Search Cross Validation is employed, a systematic approach for parameter tuning to enhance model performance. In stage *III. Interpretability*, SurvSHAP is employed to interpret the predictions made by the trained survival model. Unlike LIME, which offers a static analysis, SurvSHAP incorporates the element of time dependency in its interpretation. For instance, *lung mass* might be a significant predictor at the onset of treatment but may decrease in influence over time, whereas factors like age continue to be crucial throughout. SurvSHAP's dynamic analysis, as opposed to LIME's static feature weighting, provides a more detailed understanding of how predictive factors evolve over time. This depth of analysis is vital for a better comprehension of survival patterns, aiding in the development of more effective treatment strategies. Finally, stage *IV. Outcome* involves the presentation of SurvSHAP results. This visualization aims to address key questions regarding the dynamic influence of various factors over time, offering insights that are critical for informed decision-making in survival analysis, this thesis aims to answer following research questions :

RQ1. How does each feature influence the predictions of the survival time model?

RQ2. How does the individual importance of features of a particular patient develop over time?

RQ3. How does corrupting an input feature affect the interpretability of the survival model?

In summary, SurvSHAP represents a significant enhancement to InterpretME by enabling the analysis and interpretation of time-dependent survival data in a way that was not accessible to previous methods such as LIME. This advance opens new horizons in medical research and practice by bridging the gap between complex survival models and practical, and a more intuitive interpretation.

3.3 SurvSHAP implementation

The subsequent section presents the Python implementation and utilization of the SurvSHAP tool, dedicated to interpreting survival models. Special python packages like SurvSHAP and sksurv were utilized to provide a comprehensive workflow starting from data preparation to model interpretation up to the present of results and plotting those in an understandable and useful way.

3.3.1 Survival machine learning Model

Models such as the Random Forest, Gradient Boost and AdaBoost are very popular machine learning models due to their powerful algorithms and ability to recognize complex patterns in data. However, these models in their standard form are not able to effectively handle time-dependent data found in survival analysis. The reason for this is their primary focus on data sets where each sample is treated as an independent observation, without considering the time dimension that plays a role in survival data. To adequately analyze survival data, more specialized models such as the Random Survival Forest (RSF) or Cox Proportional Hazards models (Cox) are required [27].

These ML models were developed specifically to handle timetoevent data, so they are suitable for processing time-dependent data. The RSF extends the traditional Random Forest approach by taking into account data censoring and estimating a survival function for each observation, allowing for time-dependent risk assessment [28].

The Cox model uses a semiparametric method to estimate the ratio of hazard rates between observations, considering the time to occurrence of an event as a baseline variable [29]. For evaluating survival machine learning models, the most used performance metrics are brier score and the Concordance-Index (C-Index). The brier score is a metric that measures the mean deviation between observed and predicted events over a given period. It is particularly useful for survival time analyses [30].

It considers both the risk of an event and the temporal accuracy of the prediction. A lower brier value indicates a higher accuracy of the model. The C-Index measures how well the model predicts the sequence of event times. Specifically, it compares whether the predicted survival times are in the same order as the observed survival times. The C-Index can handle data censoring adequately and therefore provides a reliable assessment of model accuracy even with censored data [31].

The C-Index is essentially a type of rank correlation measure. It corresponds to the proportion of all possible pairs of observations where the predictions and the actual events match. The C-Index ranges from 0.5 to 1.0. A score of 0.5 demonstrates that the model's predictions are simply no better than chance while a score of 1.0

demonstrates that the model makes perfect predictions. Careful selection and tuning of hyperparameters, which control the complexity of the model and its fit to the data, is required for the configuration of such a machine learning model for survival time analysis. The hyperparameters needed for the RSF and their respective roles are described as follows:

- n_estimators (Number of trees in the forest): This factor determines the number of trees in the forest and has a large impact on the prediction accuracy of the model.Increasing the number of estimators can improve model accuracy, but also leads to a longer training time and possibly overfitting.
- max_depth (maximum tree depth): Limitation of the maximum tree depth that the tree can assume. This hyperparameter can be used to control the complexity of the model and to avoid overfitting. An appropriate depth is crucial for achieving an appropriate balance between bias and variance.
- max_features: Determines the maximum number of features to be considered when searching for the best separation.
- min_samples_leaf: Minimum number of observations required to form a leaf node. It influences how the trees grow and can help to make the model smoother and prevent overfitting from occurring.
- min_samples_split: The minimum number of observations required to further split a node. A higher number prevents the formation of nodes that are too specific for the training data and thus promotes generalizability.
- random_state: Control the randomness of the results by setting a seed for the random generator. This ensures reproducibility over different training cycles.

A designated *hyperparameter_grid* was employed and defined to establish the boundaries for the hyperparameters. This grid outlines the spectrum from which the tuning process seeks to determine the most effective hyperparameter values.

3.3.2 Tuning ML model

Choosing the best hyperparameters is critical to the performance of the model and usually requires careful tuning. A model that is not optimally tuned can either be too simple and fail to capture important data patterns ***Under-fitting*** or too complex and over-fit to specific examples of the training dataset ***Under-fitting***. Accurate hyperparameter tuning helps to find the right balance and deliver a model that is

accurate and adaptable when encountering new data. A common method to fit hyperparameters is to use techniques like grid search [32]. Grid search is a method for optimizing the hyperparameters of a machine learning model. It is based on the brute-force approach [33], in which several possible values are specified for each hyperparameter. These values form a "grid" of parameter combinations. The model is then trained and evaluated with each combination of these hyperparameter values. The evaluation is usually carried out by cross-validation, whereby the data set is divided into several smaller parts. The model is trained on one part of the dataset and validated on another, which is repeated several times so that each part of the dataset is both trained and validated. This method ensures that the evaluation of model performance is not influenced by random data splits. At the end of the grid search process, the combination of hyperparameters that has shown the best performance in the validation process is selected. This optimized parameter combination is then used to retrain the model on the entire available training dataset and finally to evaluate the model performance on a separate test dataset. Using the function Grid Search CV from the sklearn.model_selection package a hyperparameter tuning was achieved and the parameters are used in the RSF ML model [34].

3.3.3 Implementing the survshap_interpretation function

This is the main function that will create SurvSHAPs interpretations, the function is called upon fitting the Random Survival Model assigning it to the variable model. Figure 3.2 shows the flow of the implemented survshap_interpretation function along with its necessary data preprocessing steps. This function takes 6 inputs:

- model: This is the trained survival machine learning model with the tuned hyperparameters using Grid Search CV.
- X_train: The same train data, that the survival machine learning model was trained with.
- X_test: The test data set, consisting of the features for which the SurvSHAP values are to be calculated. These data are the new observations that were not involved in training the model meaning.
- Y_train: The data set with the target variables for training the survival model, usually consisting of two columns: one for the events (e.g. death, failure, etc.), which are displayed as Boolean values, and one for the survival time, which indicates the time until the event.

- Survshap_results: The file path where the results of the SurvSHAP calculations are to be saved as a pickle file in addition to as csv files. The pickle file makes it possible to save the complex outputs of the function and load them again later without having to perform the calculations again, while the csv files make it easier to analyze the results and use them in other process.

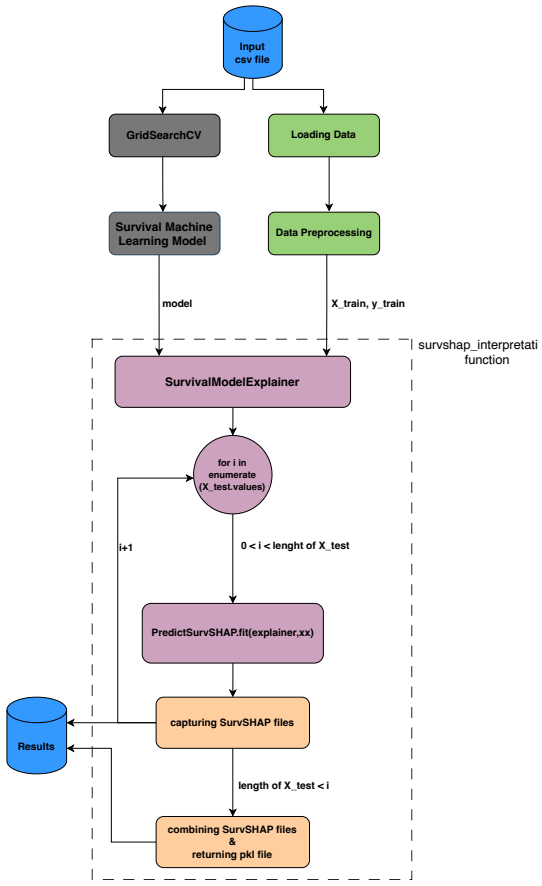


Figure 3.2: The implemented survshap_interpretation function along with its necessary data preprocessing steps.

First action in the function would be calling *SurvivalModelExplainer*. This will create an instance of the *SurvivalModelExplainer* class from the SurvSHAP package. This means that a new *explainer* object is created that has the ability to interpret the used survival model. *SurvivalModelExplainer* utilizes its input parameters (model,

X_train, Y_train) to prepare for the calculation of explanations. The model and training data are saved internally for the upcoming analysis. Once *Explainer* has been created and initialized, it is ready to be used to analyze the contributions of individual features to model prediction. Next step is the core of SurvSHAP, In a for loop, a series of steps is performed to calculate and store the SurvSHAP values for each observation in the test dataset (X_test). For each observation in the test dataset, a DataFrame is initially created to fit the observation data to the structure of the *SurvivalModelExplainer*. Next, a SurvSHAP weights calculation is performed for each of these observations to calculate the SurvSHAP values. The resulted SurvSHAP values for each observation are then saved in two ways, in a initialized list and as an individual csv file in a predefined directory, the path of this directory is the survshap_results parameter. Once all observations have been processed, the individual csv files are combined into one file, this step will be very beneficial and will facilitate the coming mapping process. Finally, the entire SurvSHAP results are stored in a pickle file. The loop ends with the return of these collected SurvSHAP results. The *PredictSurvSHAP* is a class which when initialized, various parameters can be set, those parameters affect the computation of the SurvSHAP values, the most important parameters are:

- Function_type: This parameter defines the type of the function to be evaluated for explanation. Two Options are available for the user to choose from, first one is the survival function (sf), and the other option is the cumulative hazard function (chf). Survival function was chosen for implementation as It's directly interpretable and widely used in medical research applications.

- Calculation_method: This parameter defines the way the SurvSHAP calculations are made. Two Options are here also available, first one is kernel SHAP method and the other one is sampling. Kernel SHAPwas used for in this implementation because Kernel-based methods, like KernelSHAP, are known for providing accurate approximations of SHAP values. They offer a good balance between computational efficiency and the accuracy of the explanations.

- Aggregation_method: This parameter defines what the method will be used to aggregate the SurvSHAP values, the options include, sum_of_squares, integral and mean_abs. integral is the method that was used in this implementation because it aggregates SHAP values across the entire range of the survival function. This provides a comprehensive overview of the impact of features over the entire time spectrum, rather than just focusing on specific points.

Before starting to work with the results, there must be a way to measure the accuracy.

The *calculate_accuracy_from_survshap_interpretations* function was implemented to serve this purpose. It calculates the local accuracy of SurvSHAP explanations for a given set of predictions. This function is useful in the field of machine learning and data analysis, specifically for assessing the reliability and precision of predictive models [8]. In the *calculate_accuracy_from_survshap_interpretations* function, the difference between three values is calculated, these differences are then squared to determine the mean square difference. The three values are:

- Predicted function: This is the value that the model predicts for each instance or time point.

- Baseline function: This is a reference value that serves as a starting point or baseline for the predictions.

- Aggregated SurvSHAP values: These are the SHAP values that represent the contributions of each feature to the deviation of the prediction from the baseline.

Sigma(σ) may now be calculated, Sigma(σ) is a measure of local accuracy and is calculated by dividing the square root of the mean squared difference by the square root of the mean squared prediction [35]. Lower Sigma(σ) values generally indicate higher accuracy. A low sigma value means that the SHAP values explain the results of the model well for a particular case, since it implies less difference between the three values. Now after implementing an accuracy measurement, next step would be preparing the SurvSHAP process output. The output of the survshap_interpretation function must now undergo a process to extract useful information out of its results. The function *calculate_features_order* needed to be implemented, as it aims to evaluate and summarize the importance of features determined by the conducted SurvSHAP analyses. Figure 3.3 shows the flow of the function:

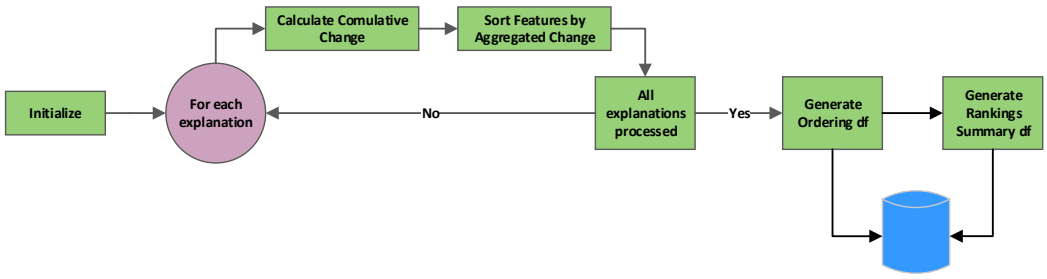


Figure 3.3: The implemented survshap_interpretation function along with its necessary data preprocessing steps

For each SurvSHAP explanation in the explanations input, the cumulative change is calculated over time for each feature. This is performed by taking the absolute SHAPSHAP values and integrating them over time using the *cumtrapz* function. After that the features are sorted in descending order based on their aggregated changes. First after all the explanations have been calculated, the function returns two DataFrames. The first contains the orders of importance of the features for each explanation (Ordering), each column in this DataFrame corresponds to a specific SurvSHAP explanation (analysis) from the explanations input. The values in each column are the names of the features in the order of importance as determined by the SurvSHAP analysis. The top feature in each column is the most important for that specific explanation, followed by the second most important, and so on, and the second output DataFrame is the summary of ranking, where each feature will have value, that summarizes how many times this feature was the most important feature in all explanations, how many times it was the 2nd most important features and so on. After processing the results of SurvSHAP, several plotting functions were implemented to present the SurvSHAP results. The tests that were carried out in this part of the implementation will be discussed in Chapter 5. After implementing the SurvSHAP method, a new challenge arises which is to integrate SurvSHAP in InterpretME.

3.4 Adaptation of InterpretME for Survival Analysis

Since InterpretME was originally developed to handle time-independent input data, it does not have the ability to process survival data. As explained in 3.1, there are significant differences between regression and survival analysis, each with its own specific requirements and functionality. Extensive structural modifications to multiple InterpretME pipeline stages discussed in 2.7.3 were necessary to fully integrate SurvSHAP with InterpretME. Those changes and adjustments will be covered and discussed here: There were no changes to the initial phases such as initialization and data importation. The first modification had to be implemented in the Data preprocessing phase. The reason for this is that standard data sampling methods, such as oversampling and undersampling, cannot be applied to survival data because they do not consider the unique characteristics of survival data. Survival data contain information about the timing of a specific event (e.g., death, failure, or recovery) and the censoring status of each observation [36]. These particularities require the application of special sampling methods that differ from those used for conventional data. The time-dependence of the data is therefore the main reasons why standard sampling methods are inadequate for survival data. This time dependency component is not addressed by techniques such as over-sampling (adding copies of minority class observations) and under-sampling (removing majority class observations) [37]. For instance, generating duplicate events in the data may cause artificial distortions in the time-distribution of events. Furthermore, the elimination of observations from the majority class may result in the loss of important time-to-event information. Censoring is also a critical issue when it comes to sampling survival data. Censoring, as described in 2.5, plays an essential role in survival analysis as it provides important information about the time to occurrence of an event. When traditional sampling methods are applied to survival data, the resulted sampled datasets may be biased because these methods do not adequately incorporate the censored data, fitting ml models with biased data will then lead to inaccurate predictions. Special sampling Methods like *Surv-SMOTE* (Synthetic Minority Over-sampling Technique for Survival Analysis) are therefore used for survival data, this method takes into consideration both the time dependency element and the censoring of data [38]. The implementation of such sampling methods is very complex and time-consuming. For this reason, no special sampling strategy was implemented as part of this thesis. After several attempts to make all necessary adjustments within the InterpretME functions for the survival analysis, it became clear that this effort would be large and would unnecessarily complicate the code. Instead, a better approach was cho-

sen, involving the implementation of two different modes for InterpretME.

To begin with, a new parameter had to be introduced into the InterpretME pipeline: this is the survival parameter. Users of InterpretME must define this parameter whenever they want to call the InterpretME pipeline. They can choose to run InterpretME in non-survival mode or survival mode. The modes can be selected by assigning a value to the survival parameter:

- **survival = 0:** This mode will run InterpretME in a non-survival mode, which means that InterpretME expects its input data to be non-time dependent data and therefore will allow users to apply standard sampling strategies like oversampling and use regression ML models like gradient boost, then the predictions will be interpreted using LIME.

- **survival = 1:** This mode will run InterpretME in the survival mode, which means that InterpretME expects its input data to be time dependent data (survival data) and therefore no standard sampling strategy can be applied, only survival machine learning models like survival random forest can be used and their prediction will be interpreted using SurvSHAP.

Within the scope of this thesis, the survival path for InterpretME and all necessary modifications to the Python source code of InterpretME were implemented. These modifications are discussed in detail below.

First integrating the functions mentioned in 3.3.3 had to take place, a new Python file "*survshap.py*" was added to the InterpretME source code, after modifying all the parameters to adapt them to InterpretME environment, some of InterpretME features were not able to deal with survival data, this applies first of all on the preprocessing of data, the main difference between survival and non-survival data lies in the type of data processing and the preparation of the target variables.

While for non-survival data the focus is on classification, the processing of survival data focuses on analyzing the time to the occurrence of a specific event, considering the specific characteristics and requirements of survival data. For processing non-survival data (if survival == 0), InterpretME preprocessing stage concentrates on preparing data for typical classification tasks. Certain variables are first removed from the list of independent variables. This involves adapting the data to make it usable for machine learning models.

The focus here is on the prediction of class definitions based on the available features. In contrary, when InterpretME is dealing with survival data (survival == 1), it handles the data in a way that meets the unique requirements of survival analysis. Here, the specific columns event and time, which are critical for survival

analysis, are preserved in the process. A survival object would be therefore created, it holds both the information about the occurrence of an event and the time until that event occurs.

As the sampling strategies that are provided in InterpretME don't support survival data, applying them to the survival data will just corrupt the data and lead to wrong predictions. Therefor for now, no sampling strategy is applied when the survival mode is selected. The next change to the InterpretME source code was made to the machine learning model building and classification stage. For non-survival path, standard classification algorithms such as **Random Forest**, **AdaBoost** or **Gradient Boosting** can be applied. The data is evaluated using techniques such as Stratified Shuffle Split and Feature Importance to identify the most important features. Next, hyperparameter optimization is performed to determine the best model.

The accuracy of the model is then evaluated using conventional classification metrics. These include accuracy, i.e., the proportion of correctly classified observations, and a classification report containing more detailed metrics such as precision, recall and F1-score for each class. These steps are characteristic of standard classification problems, which are primarily concerned with predicting a category based on independent variable. The new integrated path for survival analysis, however, shows clear differences.

A special form ML models that suits survival analyses, must be used here, e.g., the Random Survival Forest. The hyperparameter optimization method **Grid Search CV** and training of the model are specifically adapted for survival analysis. The classification report is also built differently from the non-survival mode. Instead of focusing on class accuracy, precision, recall and F1-score, the report focuses on survival analysis-specific metrics. These include the **brier score** and the **Concordance index**. These metrics provide insights into the accuracy and reliability of the model's survival predictions.

After the ML model is trained and can make predictions and classification report is generated, the model interpretation stage of InterpretME can begin. *survshap_interpretation* function has now all the required parameter and is called to interpret the predictions made by the trained survival model. The results of the SurvSHAP interpretation will be saved as pikel file and as csv files in the defined survshap_results path. Under the preparing output for plotting stage, functions like *calculate_feature_orderings*, *create_ranking_summary* and *make_factors* were used to process the SurvSHAP output and prepare the output for plotting. The Next stage of the pipeline that got big changes is the semantifying of results. The semantification module in InterpretME had to be matched with the new implemented survival

path of InterpretME [23]. New RML mappings were implemented to map the informations out of the SurvSHAP process and its results. These mappings transform the data from the results csv file into a structure that is compatible with RDF and make it possible to define relationships between different data points (such as features, values, entities, etc.). The following example shows an implemented RMLs to set up two of the mapping rules:

```
#Define a mapping configuration for SurvSHAPFeatures
<SurvSHAPFeatures>
    rml:logicalSource [
        rml:source "output/SurvSHAP/combined_SurvSHAP.csv";
        rml:referenceFormulation ql:csv; ];
    rr:subjectMap [
        rr:template "http://interpretme.org/entity/{index}_{
           Features}_{Values}_{run_id}_{tool}";
        rr:class intr:SurvSHAPFeatures ];
    rr:predicateObjectMap [
        rr:predicate intr:hasEntity;
        rr:objectMap [ rr:template "http://interpretme.org/
           entity/{index}"; ]
        ];
```

In this RML example, the process begins with the definition of the data source. Subsequently, for each row in the csv file, an RDF subject is generated. These subjects receive unique Uniform Resource Identifiers (URI)s based on a special template that incorporates values from select columns in the csv file. Following this, the *predicate-object map* comes into play. This map establishes diverse relationships between the generated subjects and various objects. Every subject, initially created in the *subjectMap*, is connected to one or more objects. A notable instance of this is the predicate *intr:hasEntity*, which forges a link between a subject and an object, with the object being crafted according to the specified schema. This mapping strategy is similarly employed for other predicates such as *intr:hasSurvSHAPFeature*, *intr:hasSurvSHAPFeatureValue*, and many more, effectively representing specific data facets through corresponding URI templates.

Once all SurvSHAP results are mapped, semantically enriched, and transformed into an RDF-compatible format, along with the necessary data for performance monitoring gathered from other InterpretME stages, users can then upload their results to Virtuoso. Virtuoso not only provides an efficient storage solution but also enables the querying of the data through SPARQL, offering a streamlined approach for users. With the implementation of this last modification, the survival mode in InterpretME

is now fully implemented and ready for operation.

The integration of SurvSHAP within the InterpretME framework overcomes various critical limitations of LIME, that were mentioned in 2.3. This enhancement provides a more robust, stable, and realistic methodology for interpreting machine learning models, especially within the domain of survival analysis. The improvements brought by SurvSHAP over LIME are detailed below [8]:

- **Addressing the Sampling Challenge:** SurvSHAP incorporates sophisticated sampling strategies that consider feature dependencies and interactions. This approach generates data points that are more representative of the true data structure, leading to more reliable and accurate interpretations.
- **Improving Core Selection:** SurvSHAP utilizes survival models that are inherently more robust and less dependent on kernel parameterization. This reduces the sensitivity of explanations to the choice of local kernels, resulting in more consistent interpretations.
- **Improving Stability:** Leveraging the inherent consistency of survival models, SurvSHAP enhances stability in the explanations it provides. This is particularly beneficial for users who require dependable insights from their model interpretations.
- **Beyond Linear Regression:** Accommodating complex, non-linear relationships typical in survival data, SurvSHAP provides more nuanced and accurate interpretations, suited to the intricacies of real-world data.

Several disadvantages of SHAP as well as of SurvSHAP, which were identified in 2.4, were addressed. For instance, the issue of involving all features in explanations has been tackled by implementing a two-step SurvSHAP process. Initially, SurvSHAP runs with all features included, and then, according to the conclusiveness of the results, it conducts a second run focusing only on the most significant features. While this approach increases computational demands, it greatly simplifies interpretation by concentrating on the most significant features. Furthermore, the challenge of correlated features is addressed using advanced methods like Conditional SHAP, as cited by [39]. However, adapting this method for survival analysis remains an area for future development and was not included in this thesis. A persistent and unmitigated issue with SurvSHAP is its computational intensity. The method requires a robust central processing unit to perform all its calculations, significantly impacting the running time. This remains a notable consideration for users and an area for potential future optimization. These mentioned enhancements collectively

demonstrate the value added by integrating SurvSHAP into InterpretME, substantially and from another angle, InterpretME itself brings considerable benefits to the SurvSHAP method, as due to InterpretME's semantic enrichment and KG presentation of results. The semantic layer added by InterpretME allows for more nuanced and context-aware interpretations, while the KG format presents results in a structured, understandable manner for both humans and machines. This integration not only broadens the analytical capabilities of InterpretME but also enhances the accessibility and interpretability of the outcomes, making it a more powerful tool for detailed and insightful data analysis.

3.5 Chapter Summary

In this chapter, the focus was first on comparing regression with survival analysis, outlining their definitions, applications, and methodologies. The chapter discusses the integration of survival analysis in the InterpretME framework, detailing the SurvSHAP implementation and the survival machine learning model. It also covers tuning the ML model and the initial standalone implementation of the SurvSHAP interpretation function. Finally, the adaptation of InterpretME for survival analysis is explored, emphasizing its application and adjustments for survival data and extending InterpretME with the new survival mode.

In the following chapter, the focus is on testing and evaluating the implemented approaches. This includes the investigation of the SurvSHAP function as well as the embedded survival analysis mode in InterpretME. The aim is to thoroughly test both the functionality and efficiency of these innovations in order to confirm their effectiveness and applicability in practice. The research questions motivated and raised by in the scope of this thesis will also be discussed and answered.

Chapter 4

Experimental Evaluation

Upon completion of the methodologies outlined in 3.3 and 3.4, this chapter delves into a comprehensive evaluation of the outcomes. Initially, the performance of the SurvSHAP method is rigorously assessed, focusing on its efficacy and accuracy. Subsequently, an experimental evaluation is conducted to examine the integration of SurvSHAP with InterpretME. This analysis aims to understand the impact and effectiveness of this integration in practical scenarios, providing insights into its applicability and potential benefits in the field of survival analysis.

4.1 Benchmark

In the scope of this chapter two different datasets were utilized for testing, the first dataset is the *tlos_v1* presented in [40], its data is extracted from X-ray images and radiology reports of various patients, indicating their length of stay in the hospital and an event indicating whether a death occurred during their stay. The dataset is huge with over 1200 patients each with about 90 features and therefore only certain features will be picked and analyzed. The second dataset that will be utilized in this section is the *exp3_heart_failure_dataset* presented in [8], its data focuses on patients with heart failure. Heart failure is a chronic condition where the heart is unable to pump blood effectively to meet the body's needs. Each row represents a patient's medical record, and the columns contain various health-related measurements and features. The dataset includes about 300 patients each with 11 features. Table 4.1 summarizes some of the most important characteristics of the used datasets.

Dataset	Description	Number of Features	Number of Patients
tlos_v1	Patients Length of stay in Hospital based on X-rays images	75	1200
Heart failure	Heart failure Patients characteristics	11	300

Table 4.1: Benchmark of the used datasets.

4.2 Evaluating survshap_interpretation

To test and evaluate the implemented function and its functionality, accuracy and impact, the tlos_v1 4.1 will be used in this section and only the following features will be examined: (age, sex, medical_devices, an_pleural_effusion, an_lung_mass). After loading the dataset, the columns patient_id, time, event, and the specified features will be first extracted from the dataset, event and time will be picked out and reformed to align with the survival ml model parameters expected formats.

Next step of data preprocessing would be splitting the data with a test data size of 20% into the following DataFrames: X_train, X_test contain the train and test feature columns, and Y_train, Y_test contain the train and test (time-event) informations, the splits were done with a random state of (123) to ensure reproducibility. Now the hyperparameter_grid dictionary will be defined as in listing 4.1:

```
hyperparameter_grid = {
            'n_estimators': [100, 300],
            'min_samples_split': [2, 5, 10],
            'min_samples_leaf': [1, 2, 4],
            'max_depth': [3, 5, 10],
            'max_features': ['sqrt', 'log2']
            }
```

Listing 4.1: Used Hyperparameter Grid limitations

A Random Survival Forest model is initialized with Grid Search CV for hyperparameter tuning using 3-fold cross-validation. The best model is selected based on the cross-validation results. After the best hyperparameters are extracted from the Grid Search CV process. The model accuracy is evaluated with the cross-validation

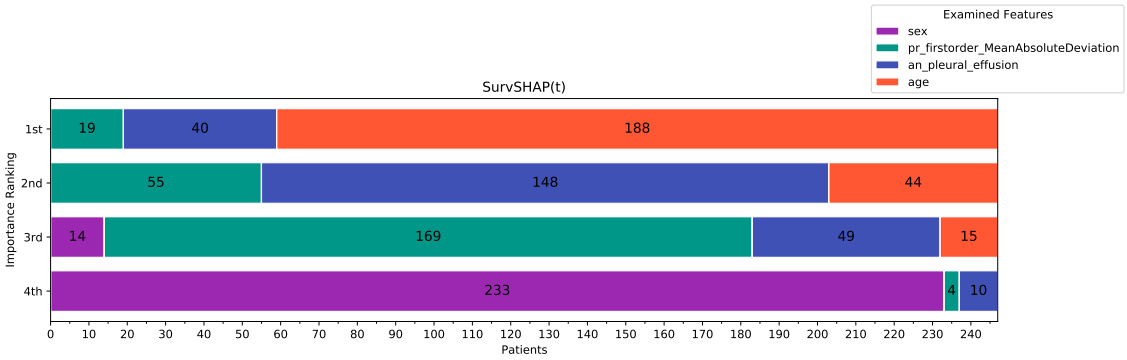


Figure 4.1: The bar plot of the survshap_interpretation results illustrating the ranking importance of the examined features.

mean score. Now the RSF model is trained with the best hyperparameters from the tuning process on the training set.

The SurvSHAP can now be used with the help of the survshap_interpretation function as all required parameters are available. Figure 4.1 shows the first plot of the survshap_interpretation results and it illustrates the ranking importance of the examined features. The plot illustrates the ranking of features based on their importance across patients. Age is highlighted as the predominant feature (1st), as it is recognized to be the most Important Feature (most Important Feature (mIF)) for 188 patients, where 40 patients had an_pleural_effusion as their mIF while 19 patients had pr_firstorder_MeanAbsoluteDeviation. Notably no patients had sex as their mIF. an_pleural_effusion comes in the 2nd place as the majority patients have it as their 2nd mIF, the same thing goes for pr_firstorder_MeanAbsoluteDeviation which comes in the 3rd place and the least important feature is sex.

This plot assists users in understanding the rankings of features according to their importance. Additionally, it shows the proportion of patients that display this importance for each feature. With these results, the first research question (RQ1) is addressed, enabling users to comprehend how each feature influences the predictions of the employed survival model.

Figure 4.2 shows the change in aggregated |SurvSHAP values| over time (days) for all patients. SurvSHAP excels at revealing how the relevance of different patient features evolves over time, a differentiated insight that goes beyond static importance rankings. For instance, while age consistently emerges as the most impactful feature overall, SurvSHAP's analysis uncovers that in the period 60-70 days,

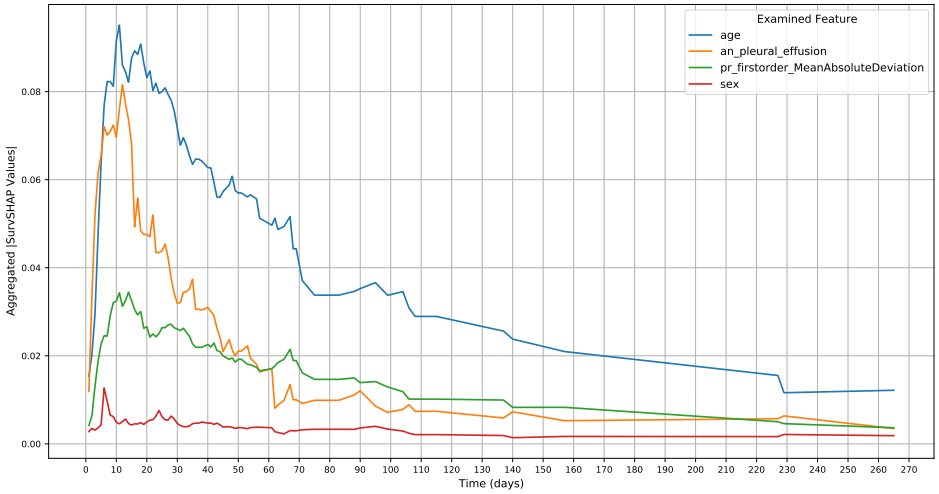


Figure 4.2: The change in aggregated |SurvSHAP values| over time (days) for all patients.

pr_firstorder_MeanAbsoluteDeviation shows a surprisingly higher influence more than an_pleural_effusion, despite being ranked lower in overall importance. This dynamic perspective provided by SurvSHAP is crucial for understanding the temporal effects of various factors on patient outcomes, enabling more informed and time-sensitive healthcare decision. It is also noticeable that changes in the aggregated |SurvSHAP values| are smaller after 70 days is due to the fact, that very small number of patients have survived after 70 days.

Aggregating the SurvSHAP values of all patients provides a concise overview and a general understanding of how feature importance changes over time. However, in practical scenarios, users whether they are researchers, physicians, or others often require more detailed and individual-based information. For this reason, the implemented method facilitates a deeper understanding and examination of specific individuals (patients). This means that users can track and visualize the changes in SurvSHAP values for a particular patient, offering a more personalized and precise insight. Figure 4.3 shows the resulted plot of calling the implemented function with the patient_id = 5 as an input parameter.

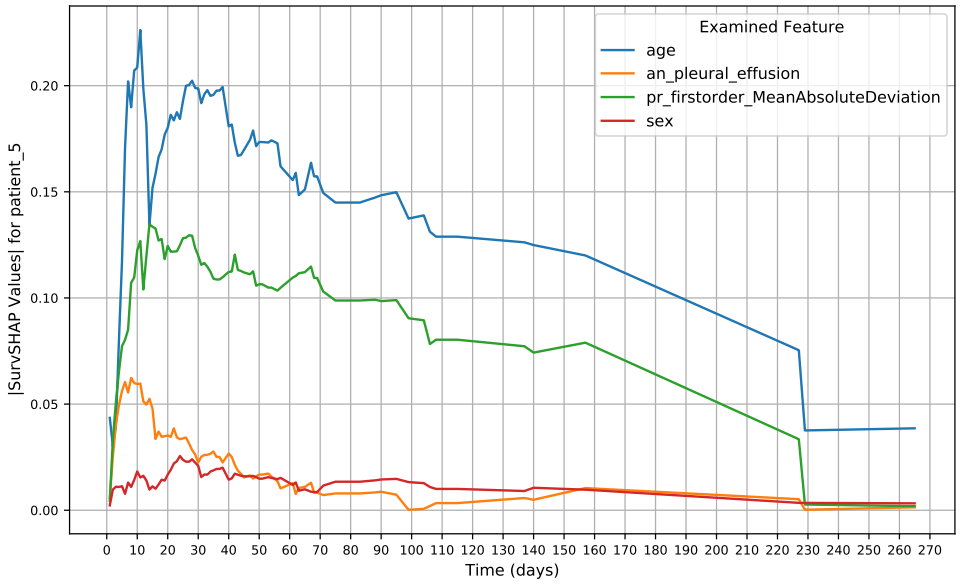


Figure 4.3: The change in |SurvSHAP values| over time (days) for patient_id = 5.

This plot shows big differences from the aggregated SurvSHAP values plot. Even though this patient has age as mIF but when it comes to the 2nd mIF it shows that pr_firstorder_MeanAbsoluteDeviation has higher SurvSHAP values than the an_pleural_effusion feature and therefor, for patient_5 the ranking of features importance deviates from the overall rank. This ability to analyze specific entities benefits users in many ways:

- Personalized analysis: By examining the SurvSHAP values of a specific patient, doctors and researchers can gain a deeper understanding of the individual factors that influence the health condition or disease risk of that patient. As shown in the example of Patient 5, individual analysis can reveal that certain factors (such as pr_firstorder_MeanAbsoluteDeviation) may be more important for an individual patient than generally assumed. This helps in identifying deviations from the average pattern.

- Targeted Treatment Strategies: It helps understanding which factors are most relevant for a particular patient allows for the development of customized treatment and prevention strategies.

- Improved Diagnostic Accuracy: Individual consideration allows for a more accurate diagnosis, as it takes into account the unique characteristics and risk factors of a patient.

The plot results for both the aggregated |SurvSHAP values| and the individual |SurvSHAP values| provide users with valuable insights into the time-dependent changes in feature importance, applicable to both groups of patients and individual cases. With these results, **RQ1** and **RQ2** are satisfactorily answered, demonstrating a comprehensive understanding of how each feature dynamically influences the survival model's predictions over time for both cases, whether for a set of patients or for an individual.

The testing phase for the implemented functions has successfully demonstrated their accuracy and readiness to be integrated into InterpretME. Moving forward, the next phase of testing will focus on evaluating InterpretME in its entirety, with particular emphasis on its performance after the incorporation of SurvSHAP. This comprehensive testing will ensure that the integration is seamless and that the system functions optimally in its enhanced state.

4.3 Survival Analysis over InterpretME

The focus of this section is to test the ability of InterpretME to work with survival analysis, with all of its new implemented additions that InterpretME got. After insuring that InterpretME can perform survival analysis, the input data will be corrupted and the change in the SurvSHAP results will be discussed and examined. For those purposes the heart failure dataset 4.1 will be utilized and only the following features will be examined: (sex, age, anaemia, smoking, high_blood_pressure, diabetes).

As discussed in 2.7.3 InterpretME will be used here by calling its pipeline function, this time in survival mode (survival = 1). The configuration file path is provided and it defines many important configurations, like the examined features Independent_variable and classes (Dead and Alive) and many other essential process configuration like cross_validation_folds (5) and the used ML model (Random survival Forest). The path for survshap_results was also provided to the pipeline function.

The used JSON configuration file is shown below in 4.2 :

```
{
  "path_to_data": "dataset/heart_failure_dataset.csv",
  "Type": "Person",
  "Index_var": "patient_id",
  "Independent_variable": ["patient_id","sex", "age", "anaemia
     ", "smoking", "high_blood_pressure", "diabetes"],
  "Dependent_variable": ["event"],
  "classes": {
    "Dead": "0",
    "Alive": "1"
  },
  "sampling_strategy": "None",
  "number_important_features": 6,
  "cross_validation_folds": 5,
  "test_split": 0.3,
  "model": "Random survival Forest",
  "min_max_depth": 4,
  "max_max_depth": 6
}
```

Listing 4.2: The used JSON Configuration file for testing

After providing the necessary parameters, the InterpretME pipeline can be initiated. The pipeline execution was successful, thoroughly validating all features of InterpretMEs survival mode. The definitions of the implemented classes and the data preprocessing steps were well-prepared for handling survival data. The addition of the new survival machine learning model, specifically the survival random forest, provided robust predictions. Hyperparameter tuning for survival analysis employed the same grid used in 4.2. This tuning process successfully identified the best combination of hyperparameters. Both the Brier score, at 0.235, and the C-Index, at 0.593, were recorded and saved. Although these values indicate a lower model accuracy, this is primarily attributed to the heart failure dataset small size, comprising only 300 data points (patients). A larger dataset is generally necessary for the machine learning model to perform optimally. However, this does not detract from our tests, which are focused on assessing the functionality of all InterpretME features and their ability to yield plausible results.

The SurvSHAP interpretability process executed smoothly and it calculated the SurvSHAP values for all features under examination. These results were illustrated, beginning with Figure 4.4:

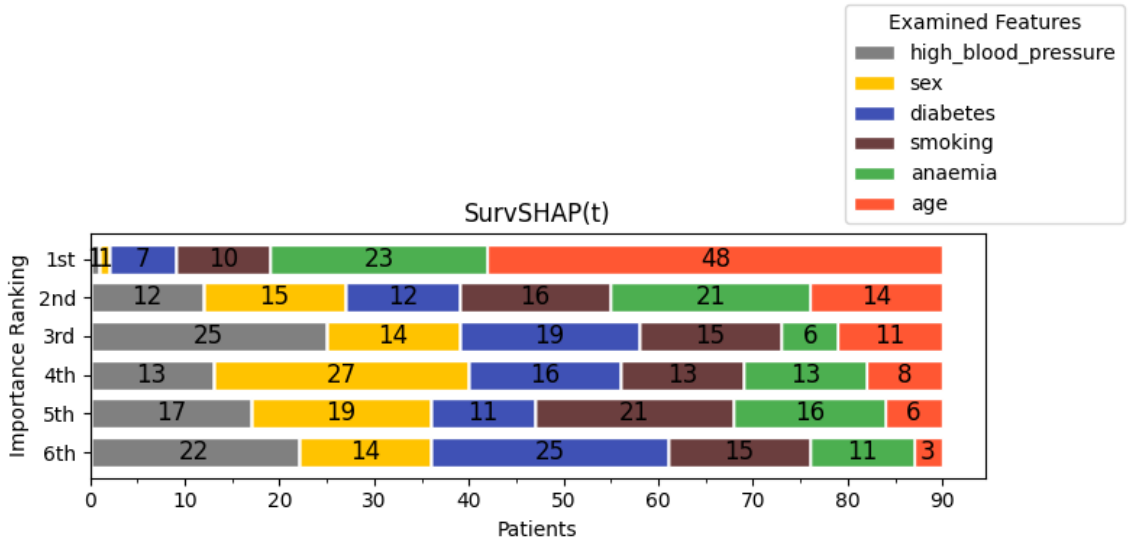


Figure 4.4: The bar plot of InterpretME interpretation results illustrating the ranking importance of the examined features without corruption.

Figure 4.4 presents the ranking of feature importance for the heart failure dataset. Notably, age ranked as the most significant feature, with approximately 53.3% (48 out of 90) of patients indicating age as their mIF. Anaemia followed, identified as the mIF for approximately 25.6% of patients. In second place, anaemia was the 2nd mIF for about 23.3% of patients. High blood pressure was the 3rd mIF for around 27.8% of patients. Surprisingly, diabetes was ranked as the least important feature, with 27.8% of patients having it as their 6th mIF. These findings offer a detailed understanding of the relative importance of features within the patient cohort. Figure 4.5 displays the change in aggregated |SurvSHAP values| over time for the examined feature. Age as the most important feature is dominating with the highest values across the whole observed time while anaemia has the 2nd highest values, however diabetes had interestingly higher values than anaemia for the time period 5-25 days. Even the smoking features shows higher results than anaemia for short period of time. This kind of informations are very important and provide a very usefull insights for the users of InterpretME and they directly answer the raised **RQ1**.

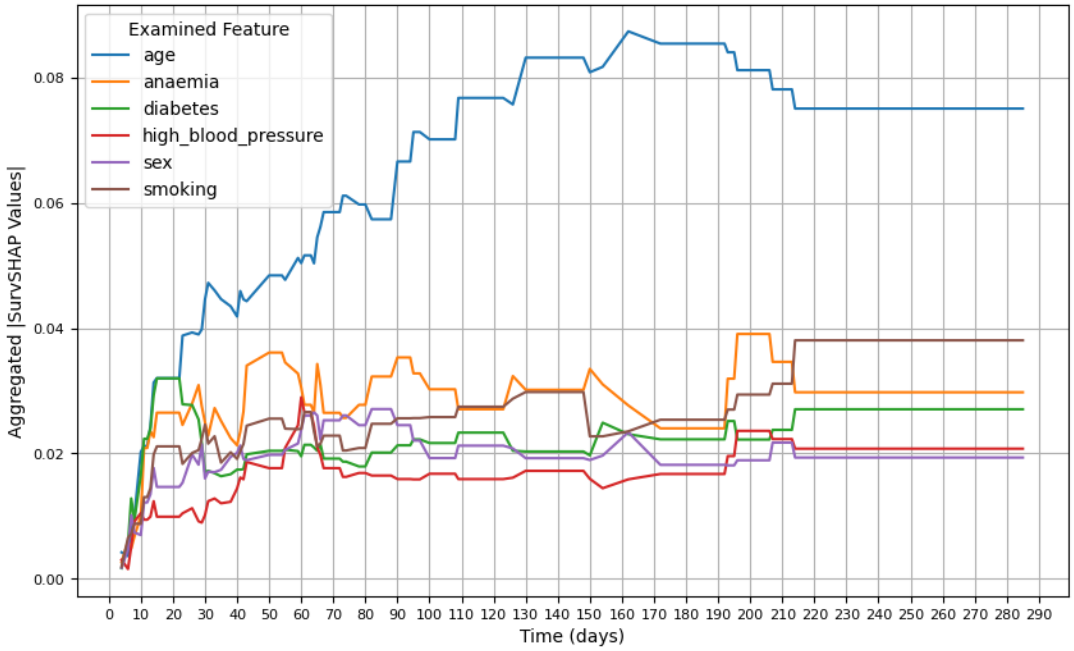


Figure 4.5: The change in aggregated |SurvSHAP values| over time (days) for all patients (Data with no corruption).

All results and configurations of the InterpretME pipeline stages in addition to the SurvSHAP process, were saved for each run. The collected data is mapped using InterpretMEs mappings, including the newly implemented mapping for the SurvSHAP method. At each run, an InterpretME KG is created or updated, this KG encapsulating all data accumulated throughout the pipeline, each datapoint is uniquely marked with an automatically generated run_id. This KG can be used to gain additional aspects and insights on the run. The InterpretME KG can be queried anytime for informations. An illustration of such a query is provided in the following listing 4.3 :

```
select distinct ?entity ?feature  ?run where {
?s  a <http://interpretme.org/vocab/SurvSHAPFeatures> .
?s <http://interpretme.org/vocab/hasEntity> ?entity .
?s <http://interpretme.org/vocab/hasSurvSHAPFeature> ?feature.
?s <http://interpretme.org/vocab/hasRun> ?run .
FILTER (?entity = <http://interpretme.org/entity/patient_151>)
}
```

Listing 4.3: Example of querying InterpretME KG

The query is retrieving distinct sets of informations related to SurvSHAP features from the InterpretME knowledge graph. Specifically, it's focused on data related to a particular entity identified as "patient_151". In more detail, it is querying for:

- entities: The specific entities related to SurvSHAP features, which in this case is constrained to the one identified as "patient_151".
- features: These are the SurvSHAP features associated with "patient_151".
- run: This refers to the unique run identifier, from which the information is queried. The run identifier allows for precise extraction and examination of the data related to a particular analysis.

By filtering the results for patient_151, the query aims to extract a focused subset of the knowledge graph that specifically pertains to this individual, detailing the relevant features and the context (runs) in which they were analyzed. This targeted extraction helps in understanding the specific circumstances and characteristics of "patient_151" as represented in InterpretME's knowledge graph. The results of such a query are presented in Table 4.2. Note: To simplify the presentation of the query result, the table displays truncated URIs, starting after 'http://interpretme.org/entity'.

entity	feature	run
/patient_151	/anaemia_0.0	/1703119817220
/patient_151	/sex_0.0	/1703119817220
/patient_151	/age_72.0	/1703119817220
/patient_151	/high_blood_pressure_1.0	/1703119817220
/patient_151	/smoking_0.0	/1703119817220
/patient_151	/diabetes_0.0	/1703119817220

Table 4.2: Results of querying InterpretME's 4.3 KG with query

These results are showing snapshot of "patient_151" from the InterpretME Knowledge Graph, detailing the examined feature status during a specific analysis run. Each feature is linked to the patient via a unique URI, providing a structured and interconnected way of representing the patient's informations.

This initial test run of InterpretME in survival mode was successful in every aspect and demonstrated a promising functionality that will enable InterpretME users to perform survival analysis and fully exploit the semantic possibilities.

4.4 Feature Corruption effect on the survival analysis of InterpretME

The aim this test is to perform SurvSHAP and InterpretME with three modifications done to the heart failure dataset: First, with the original heart failure dataset without any type of modification; second, the number of dead smokers will be increased by 30% and then by 70%, the corruption alters the dataset by changing the smoking status of a subset of patients classified as dead. Specifically, it converts a proportion of dead non-smokers into dead smokers. Table 4.3 illustrate the the modifications that were carried out to corrupt the smoking feature:

Corruption percentage	Nr. Alive smokers	Nr. Dead smokers	Nr. Dead non-smokers
0%	30	66	137
30% Corruption	30	85	118
50% Corruption	30	99	104
70% Corruption	30	112	91

Table 4.3: Impact of Data Corruption on the Smoking Feature in Heart Failure Patient Records

The purpose of these tests is to explore and observe the effects that the corruption of a feature (in this case smoking) has on the results of the ML model of InterpretME and the reliability of its predictions, i.e. on the results of the SurvSHAP process. The corruption was done by the implemented *corrupt_DataFrame* function models. All tests were performed under identical conditions and configurations, either in terms of the split ratio for training and test data or the boundaries of the hyperparameter grid up to the cross-validation value. The only deviation was the structural corruption of the smoking feature. Table 4.4 shows the brier score and C-Index for each corrupted dataset with the unique run_id.

Dataset	run_id	Brier Score	C-Index
Original	1703119817220	0.2349	0.5929
30% Corruption	1703255486598	0.2063	0.6426
50% Corruption	1703259726169	0.2113	0.6284
70% Corruption	1703255835894	0.2140	0.6182

Table 4.4: Comparison between the InterpretME RSF model accuracy results of the original, 30% and 70% corruption to the smoking feature in the heart failure dataset.

The unusual improvements observed in model performance, as evidenced by the decrease in Brier score and the increase in the C-Index with when the data is corrupted, are contrary to typical expectations. Normally, data corruption leads to a decline in model accuracy. However, at a 30% corruption rate, both the Brier score and the C-Index reached their optimal values, suggesting an unexpected improvement in the model's accuracy and its ability to classify survival times accurately. This anomaly could be attributed to specific characteristics of the dataset, the nature of the corruption introduced, or idiosyncrasies in the model's learning process. Interestingly, while the Brier score showed its best value at 30% corruption, implying that the corruption might have introduced certain patterns or eliminated noise in the smoking feature that the model deemed significant, it began to worsen (indicating a decrease in accuracy) as the corruption rate increased further. A similar trend was observed with the C-Index, which peaked at 30% corruption and then started to decline with higher rates of corruption. Despite anticipating a deterioration due to data corruption, the model demonstrated improved accuracy at a specific corruption level of 30%, pointing to complex interactions between the data corruption and the model's interpretative mechanisms.

These survival ML model results call for more detailed investigation and explanation in future research. However, the focus of this thesis is on the interpretation of the ML model for survival data rather than its performance. Therefore the effect of the corruption on the SurvSHAP results whether it is the importance ranking, the aggregated |SurvSHAP values| or the individual |SurvSHAP values| will be focused and discussed next. Only the results for 30% and 70% corruption will be addressed as theses results provide a full picture on the effect of corruption on the SurvSHAP results. Figure 4.6 shows the importance ranking for the heart failure dataset with 30% corruption. It is clear that there is a significant shift in the importance of features at the 30% corruption level of the smoking feature as smoking rises to the most important mIF for 44 patients making smoking the 1st mIF .

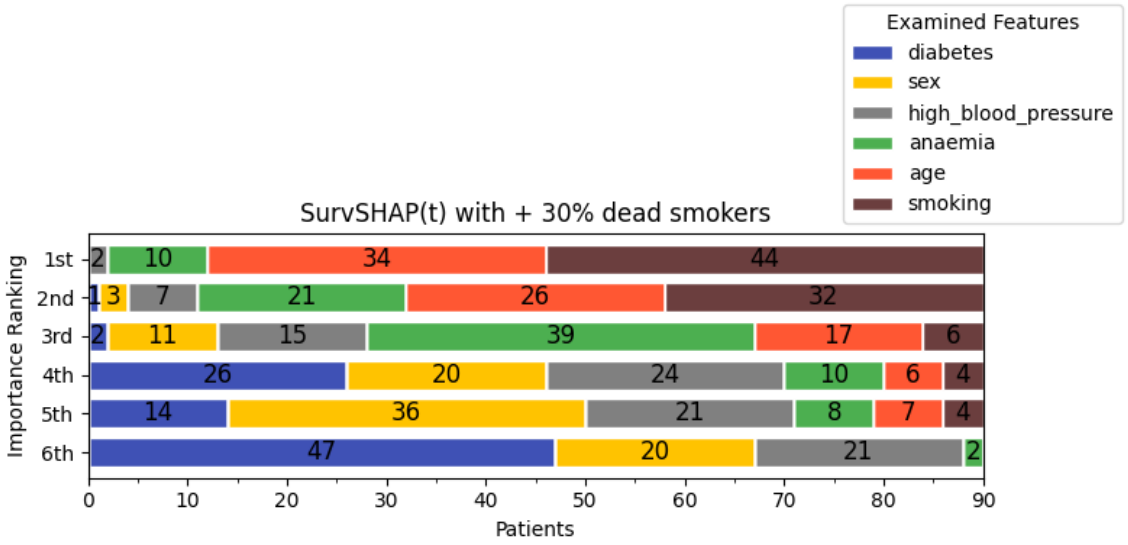


Figure 4.6: The bar plot of InterpretME interpretation results illustrating the ranking importance of the examined features with 30% corruption.

For 32 patients, it ranks as the 2nd mIF. This is a significant increase compared to the ranking of importance of the original dataset shown in Figure 4.4, in which smoking was only ranked first in 10 patients and second in 16 patients. The number of patients for whom smoking is least important decreases from 15 in the original dataset to none in the 30% corrupted dataset. This increase in the importance of smoking is further confirmed and emphasized by the rankings of importance for the 70% corrupted dataset. Figure 4.7 shows importance ranking for the heart failure dataset with 70% corruption. These results confirm the conclusion that the corruption applied to the smoking feature in the heart failure dataset directly effected the importance ranking of features and shifted the ranking toward the the smoking feature. The number of patients that have smoking as their 1st glsmIF increased to 78 leaving only 12 patients having age as their 1st mIF, age was the 1st mIF in the original dataset. only 12 patients have smoking as their 2nd mIF and not a single has smoking as their 3rd mIF and none have it as their 6th mIF.

The observed results provide a compelling indication that the intentional corruption of the smoking feature significantly influences the importance ranking of features within the heart failure dataset which directly answers the **RQ3** . Specifically, with a 30% corruption level, there's a pronounced shift in the feature importance, notably elevating smoking to the most critical feature for a substantial number of patients.

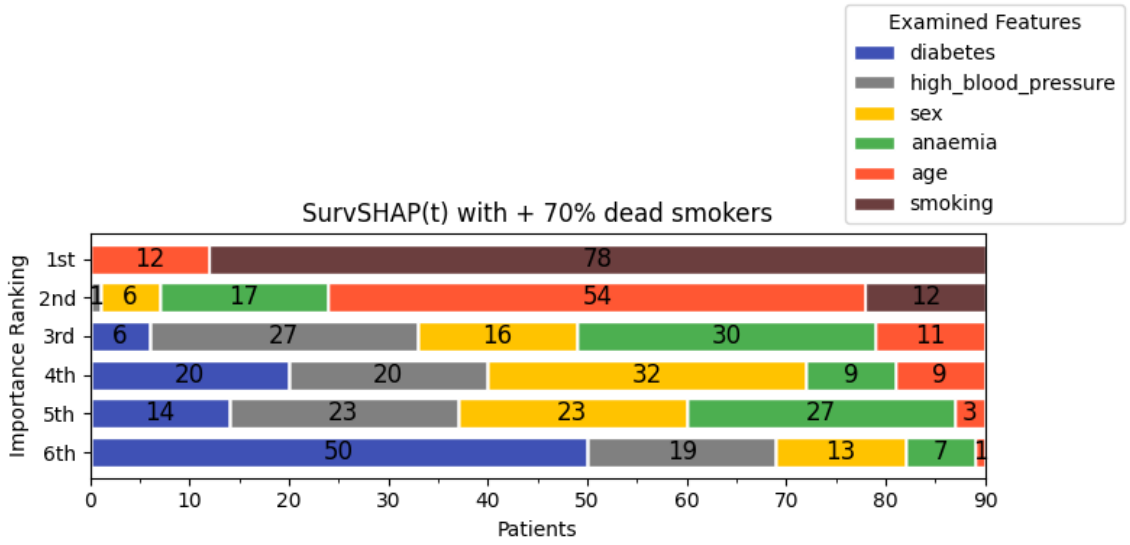


Figure 4.7: The bar plot of InterpretME interpretation results illustrating the ranking importance of the examined features with 70% corruption.

This shift is not just a minor variation but a distinct reordering, moving smoking from a lower-ranked feature in the original dataset to the most significant mIF for a majority of patients in the corrupted dataset. The pronounced increase in patients for whom smoking is the primary or secondary mIF underscores the sensitivity of the model to changes in this particular feature. Additionally, the fact that no patients consider smoking as the least important feature post-corruption starkly contrasts with the original data, highlighting the dramatic impact of the corruption. Furthermore, the distortion remains consistent and even more pronounced at higher levels of corruption, as evidenced by the 70% corrupted dataset. The shift away from age, which was previously the most significant mIF in the original dataset, to smoking in the corrupted dataset, reinforces the notion that the model's interpretation of feature importance is heavily influenced by the manipulation of the smoking data. To observe the changes in |SurvSHAP values| over time, a specific patient (patient_151) was selected as a representative case for detailed analysis on individuals **RQ2**. Figure 4.8 presents three sets of results depicting the evolution of |SurvSHAP values| for patient 151 across various datasets: on top, the original heart failure dataset, followed by datasets corrupted by 30% and 70%, respectively.

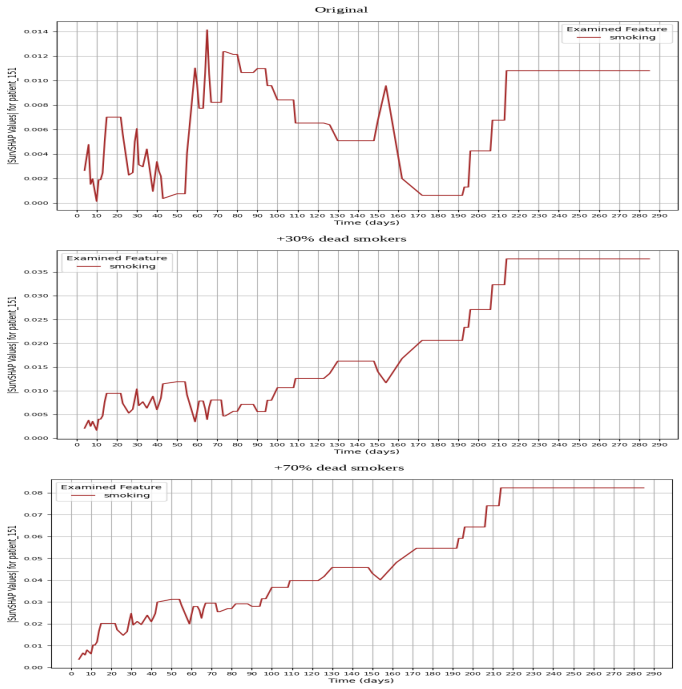


Figure 4.8: The change in |SurvSHAP values| over time (days) for patient_id = 151 for the heart failure dataset without corruption and with 30% and 70% corruption to the smoking feature.

It is proven that the corruption addressed earlier significantly influences the SurvSHAP values of the smoking feature. Notably, as the corruption rate increases, so do the SurvSHAP values. For example, on the $20th$ day, the SurvSHAP value for the original dataset is 0.007, which increases to 0.010 in the 30% corrupted dataset, and further escalates to 0.020 in the 70% corrupted dataset. A similar pattern is observed on the 100th day, where the original dataset SurvSHAP value of 0.008 rises to 0.011 with 30% corruption and peaks at 0.036 with 70% corruption. This trend is consistent for most SurvSHAP values over time, reinforcing the previously drawn conclusions about the impact of data corruption on the model's interpretative outcomes.

4.5 Chapter Summary

In the "Experimental Evaluation" chapter, the testing phase is reported as successful, with InterpretME providing detailed and accurate survival analysis results. The

newly integrated survival mode of InterpretME underwent thorough testing and evaluation within this chapter. Furthermore, the three research questions **RQ1**, **RQ2**, **RQ3** prompted by the implementation of SurvSHAP were effectively addressed, discussed, and resolved, affirming the efficacy and applicability of the approaches.

Chapter 5

Conclusions and Future Work

This thesis embarked on an exploratory journey to enhance the interpretability of machine learning models, particularly focusing on survival analysis within the context of the InterpretME framework. The culmination of this thesis provides valuable insights and contributions to the field of machine learning interpretability and knowledge graph semantics. The successful reproduction of the state-of-the-art approach, SurvSHAP, marks a significant milestone in this thesis. By meticulously implementing SurvSHAP within the InterpretME framework, this thesis not only validates the robustness and versatility of SurvSHAP but also enhances the interpretability features of InterpretME. The integration demonstrates the framework's capability to provide deeper insights into survival models, making it a more comprehensive tool for researchers and practitioners. Through the meticulous tracing of metadata and writing of RML mappings at each step of the survival analysis, this thesis has laid a groundwork for reproducibility and clarity. This systematic approach ensures that future researchers can understand the transformations and interpretations made at each stage, facilitating easier modifications and extensions. The comparative analysis using InterpretME KG with and without corrupted features provides a nuanced understanding of the model's robustness and the interpretability reliability. These experiments are crucial for understanding how feature corruption affects the survival model's interpretability, offering insights into the model's behavior under various scenarios, which is invaluable for real-world applications where data quality cannot always be guaranteed. The preparation of comprehensive code documentation for both the repository and this thesis acts as a valuable resource for future researchers. It covers all the steps, findings, and methodologies employed throughout the thesis, ensuring that the knowledge generated is accessible and understandable.

The work presented in this thesis represents a step towards demystifying the

black-box nature of survival models, making them more accessible and understandable. By bridging the gap between complex machine learning models interpretability and knowledge graph semantics, this thesis contributes to more transparent, reliable, and ethical AI systems. The journey of exploration and discovery in enhancing InterpretME with SurvSHAP and metadata tracing has not only provided a robust tool for today's researchers but also set a foundation for tomorrow's innovations.

5.1 Limitation

The recently integrated survival mode in InterpretME, like any research project, encounters certain limitations. A notable restriction is the absence of a specialized sampling method for survival data within the scope of this thesis. No method was utilized or developed to effectively handle the nuances of survival data sampling.

Moreover, the SurvSHAP method itself, while innovative, is not without its limitations. One such limitation is its provision of a single impact value per feature. Although this offers a straightforward interpretation, it doesn't capture the potential variability with different instances. A broader exploration of SurvSHAP values across similar cases might reveal a more detailed and dynamic understanding of how feature contributions vary, akin to a forecasting model [41]. Another significant challenge is the computational intensity required by SurvSHAP. The extensive calculations have prolonged the testing phase, with some runs of the InterpretME pipeline exceeding four hours. A potential mitigation strategy could involve the use of approximation techniques or the adoption of model-specific optimized versions of SurvSHAP, like TreeSHAP [42] , suitably adapted for survival analysis. Such measures could markedly decrease computation time while preserving the depth of insights.

These limitations are not unusual, particularly in a relatively new and evolving field. They reflect the current state of technology and understanding, serving as starting points for future enhancements and research. As the field matures, so too will the methods and tools, leading to more refined and efficient solutions.

5.2 Future Work

While this thesis has made significant strides in enhancing the interpretability of survival models, the journey does not end here. Future research can explore the following avenues:

- **Extending SurvSHAP:** Investigating further enhancements to SurvSHAP to handle a wider array of survival models and datasets.
- **Robustness to Feature Corruption:** Further studies to understand the limits and capabilities of InterpretME in the presence of increasingly corrupted features and multiple features corruption.
- **Interactive User Interfaces:** Developing user-friendly interfaces for InterpretME, allowing practitioners with limited technical knowledge to leverage its capabilities.
- **Enhancing Computational Efficiency:** The SurvSHAP method, as highlighted in this research, is computationally intensive and time-consuming. Future work should focus on optimizing the process and its computational demands. Enhancing the speed and efficiency of SurvSHAP is crucial for scaling to larger datasets and making it more accessible for real time applications.

5.3 Chapter Summary

In the "Conclusion" chapter, the thesis underscores the significant progress made in enhancing the interpretability of survival models through the InterpretME framework, particularly with the integration of SurvSHAP. It emphasizes InterpretME's unique ability to semantify results and create KGs, providing a structured and insightful understanding of the data. Despite facing challenges such as computational intensity and sampling limitations, the advancements set a solid foundation for future research. The chapter looks forward to opportunities for refining SurvSHAP, improving robustness against feature corruption, developing more intuitive user interfaces, and boosting computational efficiency. Furthermore, it envisions continued innovation in leveraging semantic data and KGs within InterpretME. This concluding chapter serves as a testament to the ongoing evolution and potential in the realm of interpretable machine learning.

Acronyms

AI Artificial Intelligence

C-Index Concordance-Index

csv Comma-Separated Values

JSON JavaScript Object Notation

KG Knowledge Graph

LIME Local Interpretable Model Agnostic Interpretation

mIF most Important Feature

ML Machine Learning

MRI Magnetic Resonance Imaging

RDF Resource Description Framework

RML RDF Mapping Language

RSF Random Survival Forest

SHACL SHapes Constraint Language

SHAP Shapley Additive Explanations

SPARQL Protocol & Query Language

SQL Structured Query Language

SurvSHAP Survival Shapley Additive ExPlanations

UML Unified Modeling Language

URI Uniform Resource Identifiers

W3C World Wide Web Consortium

Bibliography

[1] I.H. Sarker. "AI-Based Modeling: Techniques, Applications and Research Issues Towards Automation, Intelligent and Smart Systems". In: *SN COMPUT. SCI.* 3 (2022). Accessed: 21.12.2023, p. 158. DOI: 10.1007/s42979-022-01043-x. URL: https://doi.org/10.1007/s42979-022-01043-x.

[2] S.A. Alowais, S.S. Alghamdi, and N. Alsuhebany et al. "Revolutionizing healthcare: the role of artificial intelligence in clinical practice". In: *BMC Med Educ* 23 (2023). Accessed: 27.12.2023. DOI: 10.1186/s12909-023-04698-z. URL: https://doi.org/10.1186/s12909-023-04698-z.

[3] Longbing Cao. "AI in Finance: Challenges, Techniques, and Opportunities". In: *ACM Comput. Surv.* 55.3 (2022). ISSN: 0360-0300. DOI: 10.1145/3502289. URL: https://doi.org/10.1145/3502289.

[4] Mary T Dzindolet, Scott A Peterson, Regina A Pomranky, Linda G Pierce, and Hall P Beck. "The role of trust in automation reliance". In: *International journal of human-computer studies* 58.6 (2003), pp. 697–718.

[5] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. "A survey of methods for explaining black box models". In: *ACM computing surveys (CSUR)* 51.5 (2018), pp. 1–42.

[6] M. T. Ribeiro, S. Singh, and C. Guestrin. ""Why should I trust you?" Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 1135–1144.

[7] Scott M. Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems* 30 (2017).

[8] Mateusz Krzyziński, Mikołaj Spytek, Hubert Baniecki, and Przemysław Biecek. "SurvSHAP(t): Time-dependent explanations of machine learning survival models". In: *Knowledge-Based Systems* (2023). ISSN: 0950-7051. DOI: 10.1016/j.knosys.2022.110234. URL: https://www.sciencedirect.com/science/article/pii/S0950705122013302.

[9] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. "Knowledge Graphs". In: *ACM Comput. Surv.* 54.4 (July 2021). ISSN: 0360-0300. DOI: 10.1145/3447772. URL: https://doi.org/10.1145/3447772.

[10] Maria-Esther Vidal. *Data Graphs and Resource Description Model*. Online. Lecture slides for Scientific Data Management and Knowledge Graphs at Institut für Data Science Leibniz University. 2023. URL: https://docs.google.com/presentation/d/1zKW-JOaUe4BfP3nyaCEKKKQpoS4q-kaMailt1I_Y3XI/edit#slide=id.g14b80c162cf_0_240.

[11] M. Klein. "XML, RDF, and Relatives". In: *IEEE Intelligent Systems* 16.2 (2001), pp. 26–28. DOI: 10.1109/5254.920596.

[12] *RDF 1.1 Concepts and Abstract Syntax*. Accessed 10 Dec. 2023. W3.org. 2014. URL: https://www.w3.org/TR/rdf11-concepts/#rdf-documents.

[13] Steve Harris, Garlik, and Andy Seaborne. *SPARQL 1.1 Query Language*. W3C. 2013.

[14] B. De Meester, P. Heyvaert, and T. Delva. *RDF Mapping Language (RML)*. Accessed date: 24.10.2023. 2022. URL: https://rml.io/specs/rml/.

[15] World Wide Web Consortium (W3C). *Shapes Constraint Language (SHACL)*. https://www.w3.org/TR/shacl/. Accessed date: 27.12.2023. 2017.

[16] Davide Castelvecchi. *Can We Open the Black Box of AI?* Nature News. Oct. 2016. URL: https://www.nature.com/news/can-we-open-the-black-box-of-ai-1.20731.

[17] Wallpapers.com. *Lime Pictures Wallpaper*. Accessed on: December 29, 2023. URL: https://wallpapers.com/picture/lime-pictures-yqoee3xzuwpdvrwx/download.

[18] Lloyd S. Shapley. "A Value for N-Person Games". In: (1953).

[19] Debusinha. *Using and Scaling SHAP on Databricks for Turbocharging Your Model Explainability — Part 1*. Accessed on: December 29, 2023. URL: https://medium.com/@debusinha2009/using-and-scaling-shap-on-databricks-for-turbocharging-your-model-explainability-part-1-29099006f6b4.

[20] Y. Meng, N. Yang, Z. Qian, and G. Zhang. "What Makes an Online Review More Helpful: An Interpretation Framework Using XGBoost and SHAP Values". In: *J. Theor. Appl. Electron. Commer. Res.* 16 (2021), pp. 466–490. DOI: 10.3390/jtaer16030029. URL: https://www.mdpi.com/0718-1876/16/3/29.

[21] Y. Wang, Y. Zhao, J. Song, and H. Liu. "What drives patients to choose a physician online? A study based on tree models and SHAP values". In: *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. Mexico City, Mexico, 2022, pp. 1676–1683. DOI: 10.1109/CASE49997.2022.9926467.

[22] Christoph Molnar. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. Visited on: December 29, 2023. URL: https://christophm.github.io/interpretable-ml-book/.

[23] Yashrajsinh Chudasama, Disha Purohit, Philipp D Rohde, Julian Gercke, and Maria-Esther Vidal. "InterpretME: A Tool for Interpretations of Machine Learning Models Over Knowledge Graphs". In: (2023), pp. 8–11. URL: https://www.semantic-web-journal.net/content/interpretme-tool-interpretations-machine-learning-models-over-knowledge-graphs-1.

[24] *InterpretME Logo*. https://github.com/SDM-TIB/InterpretME/blob/develop/images/logo.png. Accessed on: 27.11.2023.

[25] B. George, S. Seals, and I. Aban. "Survival analysis and regression models". In: *J Nucl Cardiol* 21.4 (Aug. 2014). Epub 2014 May 9, pp. 686–94. DOI: 10.1007/s12350-014-9908-2.

[26] TG Clark, MJ Bradburn, SB Love, and DG Altman. "Survival analysis part I: basic concepts and first analyses". In: *Br J Cancer* 89.2 (July 2003), pp. 232–8. DOI: 10.1038/sj.bjc.6601118.

[27] Ping Wang, Yan Li, and Chandan K. Reddy. "Machine Learning for Survival Analysis: A Survey". In: *ACM Computing Surveys (CSUR)* 51.6 (2019), pp. 1–36.

[28] Hemant Ishwaran, Udaya B. Kogalur, Eugene H. Blackstone, and Michael S. Lauer. "Random Survival Forests". In: *Journal of Computational and Graphical Statistics* 17.4 (2008), pp. 841–860.

[29] John Fox and Sanford Weisberg. "Cox Proportional-Hazards Regression for Survival Data". In: *An R and S-PLUS Companion to Applied Regression*. SAGE Publications, 2002, pp. 200–232.

[30] Håvard Kvamme and Ørnulf Borgan. "The Brier Score Under Administrative Censoring: Problems and Solutions". In: *arXiv preprint arXiv:1912.08581* (2019).

[31] Adam R. Brentnall and Jack Cuzick. "Use of the Concordance Index for Predictors of Censored Survival Data". In: *Statistical Methods in Medical Research* 27.8 (2018), pp. 2359–2373.

[32] Lisa Wimmer. "A Hyper-Parameter-Tuned, Confidence-Weighted Collaborative Filtering Model for Implicit Feedback". Bachelor's Thesis. Department of Statistics, Ludwig-Maximilians-Universität München, 2019, pp. 28–30.

[33] Jimmy Lin. "Brute force and indexed approaches to pairwise document similarity comparisons with mapreduce". In: *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 2009, pp. 155–162.

[34] Niklas Klein. "Extending Hyperband with Model-Based Sampling Strategies". In: 2018. URL: https://api.semanticscholar.org/CorpusID:70211351.

[35] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. "From Local Explanations to Global Understanding with Explainable AI for Trees". In: *Nature Machine Intelligence* 2.1 (2020), pp. 56–67.

[36] Peter Hougaard. "Fundamentals of Survival Data". In: *Biometrics* 55.1 (1999), pp. 13–22.

[37] M.S. Shelke, P.R. Deshmukh, and V.K. Shandilya. "A Review on Imbalanced Data Handling Using Undersampling and Oversampling Technique". In: *International Journal of Recent Trends in Engineering Research* 3.4 (2017), pp. 444–449.

[38] Tony Bellotti and Jonathan Crook. "Credit Scoring with Macroeconomic Variables Using Survival Analysis". In: *Journal of the Operational Research Society* 60.12 (2009), pp. 1699–1707.

[39] Kjersti Aas, Martin Jullum, and Anders Løland. "Explaining individual predictions when features are dependent: More accurate approximations to Shapley values". In: *Artificial Intelligence* 298 (2021), p. 103502. ISSN: 0004-3702. DOI: 10.1016/j.artint.2021.103502. URL: https://www.sciencedirect.com/science/article/pii/S0004370221000539.

[40] Hubert Baniecki, Bartlomiej Sobieski, Przemysław Bombiński, Patryk Szatkowski, and Przemysław Biecek. *Hospital Length of Stay Prediction Based on Multi-modal Data towards Trustworthy Human-AI Collaboration in Radiomics*. In review. 2023.

[41] Scott M. Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *NIPS*. 2017. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[42] Scott M. Lundberg, Gabriel G. Erion, and Su-In Lee. "Consistent Individualized Feature Attribution for Tree Ensembles". In: *arXiv preprint arXiv:1802.03888* (2018). URL: https://arxiv.org/abs/1802.03888.