Norwegian University
of Life Sciences

Master's Thesis 2024 30 ECTS
Faculty of Science and Technology

Missing Value Imputation and Survival Analysis for Treatment Outcome Prediction in High-Grade GEP NEN

Erlend Kristiansen Risvik and Nina Rebecca Finsrud Lizana
Data Science

Aknowledgements

We would like to extend our gratitude to the four excellent supervisors Cecilia Marie Futsæther, Stefan Schrunner, Oliver Tomic and Henning Langen Stokmo for assisting us with the thesis. We thank our main supervisor, Futsæther, for steering us through the project and always being available for discussions and meetings. We are grateful to the three co-supervisors, Schrunner, Tomic and Stokmo for participating in the research. Schrunner assisted with knowledge of missing values and statistical insights after moving to Austria. We are thankful for him taking the time to guide us in the project. Tomic, from NMBU, provided expertise in survival analysis and machine learning. This was immensely helpful in shaping the direction of the project. Lastly, Stokmo, from Oslo University Hospital, provided both the dataset and medical domain knowledge. His assistance with the data preparation and insight into our analysis was extremely valuable. All supervisors were always positive and encouraged us to explore new possibilities. Regular meetings with thoughtful discussions significantly impacted our learning, and we appreciate everyone taking the time to participate.

We are grateful to our family and friends for supporting us through the five years of studying. Their encouragement has been invaluable.

Erlend Kristiansen Risvik

Nina Rebecca Finsrud Lizana

Ås, Mai 15, 2024

Abstract

The objective of the thesis was to create a reliable model for outcome prediction in gastrointestinal cancers using data with incomplete variables. This was a two-step process. The first part involved analysis and review of literature on missing values, with a separate experiment conducted to support decisions. The second step involved preprocessing data with assistance from expert knowledge and conducting survival analysis. We found that imputing missing values is always better than discarding information in variables and samples. This is especially important with a small sample size. k-Nearest Neighbor imputation provided accurate single imputations in the experiment and had the most promising impact on the survival models. The two survival models, Coxnet and Component-Wise Gradient Boosting, provided the highest test concordance, with the latter having the lowest integrated Brier score and time-dependent Brier score We argue that the choice of model depends on the application, as a model may excel in one metric and be less effective in another metric. If the intention is accurately predicting the order of events, the model that maximizes concordance should be used. Conversely, if accurate modelling of the survival times is of interest, then the model maximizing the integrated Brier score should be used. We also found that *Number of Courses*, *Ki-67*, and *NSE* were identified as having the most average importance across the four survival models. Additionally, features *WHO Perf Stat*, *Ki-67*, and *Albumin* were identified as equally important, consistent with the results reported by Jenul et al.(2023).[65]

Table of contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Objective . . . 1
- 1.3 Structure of the Thesis . . . 2
- 1.4 Notation, Terminologies and Abbreviations . . . 2
- 1.5 Generative AI and Other Tools Used in This Thesis . . . 2

2 Theory Part A: Survival Analysis **3**
- 2.1 High-Grade GEP NEN . . . 3
  - 2.1.1 WHO Classification System . . . 3
  - 2.1.2 Treatments and Survival . . . 3
- 2.2 Survival Analysis Fundamentals . . . 4
  - 2.2.1 Censoring . . . 4
  - 2.2.2 Survival Function . . . 6
  - 2.2.3 Hazard Function . . . 7
  - 2.2.4 Relationships Between Hazard and Survival Function . . . 8
- 2.3 Survival Analysis Models . . . 9
  - 2.3.1 Cox Proportional Hazard . . . 9
  - 2.3.2 Penalized Cox Model (Coxnet) . . . 11
  - 2.3.3 Random Survival Forest . . . 13
  - 2.3.4 Gradient Boosting . . . 16
- 2.4 Evaluation of Models . . . 17
  - 2.4.1 Cross Validation . . . 17
  - 2.4.2 Performance Metric: Harrel's C-index . . . 19
  - 2.4.3 Performance Metric: Brier Score . . . 20
- 2.5 Data Preprocessing . . . 22
  - 2.5.1 Encoding Variables . . . 22
  - 2.5.2 Outlier Detection . . . 24
  - 2.5.3 Feature Scaling: Standard Scaling . . . 25
  - 2.5.4 Feature Selection: Permutation Feature Importance . . . 25

3 Theory Part B: Missing values **27**
- 3.1 What Are Missing Values? . . . 27
- 3.2 Missing Value Mechanisms . . . 27
  - 3.2.1 MCAR . . . 27
  - 3.2.2 MAR . . . 27
  - 3.2.3 MNAR . . . 27
  - 3.2.4 Identifying the Mechanisms . . . 28
- 3.3 Missing Value Patterns, Influx and Outflux . . . 28
- 3.4 The Basics of Handling Missing Values . . . 31
  - 3.4.1 Two Strategies for Handling Missing Values . . . 31
- 3.5 Removing Rows, Columns or Both? . . . 31
- 3.6 Imputation Methods . . . 33
  - 3.6.1 Global Imputation . . . 33
  - 3.6.2 Multivariate Imputations . . . 34
  - 3.6.3 Multiple Imputations . . . 38
- 3.7 A Note About Categorical Variables . . . 42
- 3.8 Evaluating Imputation Methods . . . 42
  - 3.8.1 Direct Evaluation of Missing Value Imputation . . . 42
  - 3.8.2 Indirect Evaluation of Missing Value Imputation . . . 43

3.9 Limitations of Modern Imputation Strategies . . . 44

4 Materials and Methods **45**
4.1 Data . . . 45
4.1.1 Colon . . . 45
4.1.2 GEP NEN . . . 49
4.2 Software and hardware . . . 53
4.2.1 Software . . . 53
4.2.2 Hardware . . . 53
4.3 Experimental Setup: Missing Values (Colon) . . . 53
4.3.1 Motivation . . . 53
4.3.2 The Strategies Applied . . . 54
4.3.3 Evaluation . . . 54
4.3.4 Missing Value Generation . . . 55
4.3.5 Pipeline for the Missing Value Experiment . . . 57
4.3.6 Parameters and Hyperparameters Used for Simulation . . . 60
4.4 Experimental Setup: Survival Analysis (GEP NEN) . . . 61
4.4.1 Motivation . . . 61
4.4.2 Preparation of the GEP NEN Dataset . . . 62
4.4.3 Survival Analysis Pipeline . . . 66
4.4.4 Hyperparameters Used for the Survival Models . . . 68
4.4.5 Pipeline Methodology for CCA . . . 69
4.4.6 Multiple Imputation . . . 70
4.4.7 Evaluating the Strategies: A Statistical Approach . . . 71
4.4.8 Visualization of Survival Model Performance and Evaluation . . . 72

5 Results **74**
5.1 Experimental Setup: Missing Values (Colon) . . . 74
5.1.1 Runtimes of Missing Value Experiment . . . 74
5.1.2 The Cox Proportional Hazards Model . . . 74
5.1.3 Missing Completely at Random (MCAR) . . . 75
5.1.4 Missing at Random (MAR) . . . 82
5.2 Experimental Setup: Survival Analysis (GEP NEN) . . . 85
5.2.1 Runtimes of the Survival Analysis . . . 85
5.2.2 Hyperparameter Optimization . . . 85
5.2.3 Evalution of Missing Value Strategies . . . 88
5.2.4 Visualization of Survival Model Performance and Evaluation . . . 90

6 Discussion **100**
6.1 Experimental Setup: Missing Values (Colon) . . . 100
6.1.1 Flaws of the Experimental Setup . . . 100
6.1.2 Accuracy . . . 100
6.1.3 Bias . . . 100
6.1.4 Concordance index . . . 101
6.1.5 Summary . . . 102
6.2 Experimental Setup: Survival Analysis (GEP NEN) . . . 102
6.2.1 Data Registration and Privacy . . . 102
6.2.2 Sample Size . . . 103
6.2.3 The Missing Values . . . 103
6.2.4 Data Preprocessing . . . 104
6.2.5 Performance Between the Different Models . . . 106
6.2.6 Evaluation Using Cross-Validation . . . 107

6.2.7 Features Coefficients for Different Alphas . . . 108
6.2.8 Permutation Features Importance . . . 108

7 Conclusion **110**

8 Future work **111**

Appendix A **123**

A Experimental Setup: Missing Values **123**
A.1 MCAR . . . 123
A.1.1 Bias . . . 123
A.1.2 Concordance . . . 124
A.2 MAR . . . 125
A.2.1 Accuracy . . . 125
A.2.2 Bias . . . 127
A.2.3 Concordance . . . 130

Appendix B **133**

B Survival Curves **133**
B.1 Compare Survival Curves Between Coxnet and CGB with the Highest Test C-index 133
B.2 Compare Survival Curves Between the Coxnet Model (alpha: 1000, L1 ratio: 0.0001) with Coxnet Model (alpha: 3, L1 ratio: 0.01) . . . 134
B.3 Compare Survival Curves Between Coxnet and CGB Model for All Patients. . . . 135

Appendix C **149**

C Coxnet Model Performance Across Hyperparameters **149**
C.1 Coxnet Model Performance For L1 Ratio of 0.2 . . . 149
C.2 Coxnet Model Performance For L1 Ratio of 0.3 . . . 150
C.3 Coxnet Model Performance For L1 Ratio of 0.4 . . . 151
C.4 Coxnet Model Performance For L1 Ratio of 0.5 . . . 152
C.5 Coxnet Model Performance For L1 Ratio of 0.6 . . . 153
C.6 Coxnet Model Performance For L1 Ratio of 0.7 . . . 154
C.7 Coxnet Model Performance For L1 Ratio of 0.8 . . . 155
C.8 Coxnet Model Performance For L1 Ratio of 0.9 . . . 156

Appendix D **156**

D Code **156**

List of Figures

2.1 Reasons for censoring . . . 4
2.2 Illustrated different types of censoring . . . 6
2.3 Survival curve example . . . 7
2.4 Mathematical relationship between survival functions . . . 8
2.5 A overview of the operational mechanics behind the RSF algorithm . . . 13
2.6 K-fold cross-validation overview . . . 17
2.7 Stratified K-fold overview . . . 18
2.8 Repeated Stratified K-fold overview . . . 18
2.9 Illustration of how permutation feature importance works . . . 26
3.1 Influx and outflux for variables A, B and C from Table 3.1 . . . 30
3.2 Illustration of the multiple imputations strategy . . . 38
3.3 Illustration of sparsity and heterogeneity . . . 44
4.1 Distribution of the variables from table 4.1, separated by censoring status . . . 47
4.2 Pearson correlation of the variables from table 4.1 . . . 48
4.3 Pearson correlation of GEP NEN dataset . . . 51
4.4 PCA plot of the patients in the GEP NEN dataset . . . 52
4.5 Distribution of censoring over survival time (days) for GEP NEN dataset . . . 52
4.6 Evaluation pipeline for missing value experiment . . . 55
4.7 Pipeline of the missing value experiment . . . 57
4.8 Pipeline for the preparation of the dataset for the new study . . . 62
4.9 Influx and outflux of the columns in the GEP NEN dataset . . . 64
4.10 Correlation matrix for columns with absolute correlation higher than **0.85** . . . 65
4.11 Survival analysis pipeline methodology . . . 66
4.12 Comparison methodology for evaluating missing value stategies . . . 72
4.13 Survival model visualizations overview . . . 72
5.1 Train accuracy as a function of the proportion of missing values using mode, kNN and MICE imputations (MCAR) . . . 76
5.2 Bias of coefficients of the variables as a function of the proportion of missing values after applying CCA, kNN and MICE on simulated missingness (MCAR) . . . 78
5.3 Train and test concordance as a function of the proportion of missing values after applying CCA on simulated missingness (MCAR) . . . 79
5.4 Train and test concordance as a function of the proportion of missing values after applying kNN imputation on simulated missingness (MCAR) . . . 80
5.5 Train and test concordance as a function of the proportion of missing values after applying MICE imputation on simulated missingness (MCAR) . . . 81
5.6 Train accuracy as a function of the proportion of missing values after applying kNN and MICE imputation on simulated missingness (MAR) . . . 82
5.7 Bias of coefficients of the variables as a function of the proportion of missing values after applying CCA and MICE on simulated missingness (MAR) . . . 83
5.8 Train concordance as a function of the proportion of missing values after applying CCA, kNN and MICE on simulated missingness (MAR) . . . 84
5.9 Coxnet model performance across regularization strength . . . 90
5.10 Permutation feature importance across survival models . . . 91
5.11 Mean time-dependent Brier score for each survival model . . . 92
5.12 Survival curves for Coxnet and CGB models with kNN imputations . . . 94
5.13 Survival curves for Coxnet and CGB models with multiple imputations . . . 96
5.14 Coxnet model coefficients at L1 ratio 0.01 . . . 97
5.15 Permutation feature importance for Coxnet model . . . 98
5.16 Coefficient weights for Coxnet model features . . . 99

A.1 Bias of coefficients of the variables as a function of the proportion of missing values after applying mode imputation on simulated missingness (MCAR) 123
A.2 Train and test concordance as a function of the proportion of missing values after applying mode imputation on simulated missingness (MCAR) 124
A.3 Train accuracy as a function of the proportion of missing values after applying kNN imputation on simulated missingness (MAR) 125
A.4 Train accuracy as a function of the proportion of missing values after applying MICE imputation on simulated missingness (MAR) 126
A.5 Bias of coefficients of the variables as a function of the proportion of missing values after applying CCA on simulated missingness (MAR) 127
A.6 Bias of coefficients of the variables as a function of the proportion of missing values after applying kNN imputation on simulated missingness (MAR) 128
A.7 Bias of coefficients of the variables as a function of the proportion of missing values after applying MICE imputation on simulated missingness (MAR) 129
A.8 Train concordance as a function of the proportion of missing values after applying CCA on simulated missingness (MAR) 130
A.9 Train concordance as a function of the proportion of missing values after applying kNN imputation on simulated missingness (MAR) 131
A.10 Train concordance as a function of the proportion of missing values after applying MICE imputation on simulated missingness (MAR) 132
B.1 Survival curves between the highest C-index model for Coxnet and CGB 133
B.2 Survival curves for Coxnet model with the highest C-index model and the model that were selected, with kNN imputations 134
B.3 Survival curves for all patients for Coxnet and CGB models with kNN imputations 135

List of Tables

2.1 WHO's classification and grading criteria for NENs of the gastrointestinal tract and hepatopancreatobiliary organs . . . 3
2.2 Target encoding example . . . 23
3.1 Data with columns A, B and C . . . 29
3.2 The four missing value patterns for variables A, B and C from Table 3.1 . . . 29
3.3 Example highlighting the problem of applying CCA before regression analysis . . . 32
3.4 Dummy data for illustration of kNN imputation . . . 36
3.5 Euclidean distances (dist) from tuple t_3 to the tuples listed in Table 3.4 . . . 36
3.6 Implemented imputation functions in the *mice* package in R . . . 41
4.1 Description of the variables in the colon dataset . . . 46
4.2 Description of numerical variables in the GEP NEN dataset . . . 49
4.3 Description of ordinal variables in the GEP NEN dataset . . . 50
4.4 Description of binary variables in the GEP NEN dataset . . . 50
4.5 Description of non-binary nominal features in the GEP NEN dataset . . . 51
4.6 Strategies for handling missing values used in the experimental setup . . . 54
4.7 Imputation functions used in MICE . . . 58
4.8 Expected number of training rows to keep for different p_{miss} under MCAR . . . 60
4.9 Percentage of NA for each Column in GEP NEN dataset . . . 64
4.10 ercentage of NA for each patient . . . 66
4.11 Hyperparameter values used in the Coxnet model in the GEP NEN dataset . . . 68
4.12 Hyperparameter values used in the CoxPH model in the GEP NEN dataset . . . 68
4.13 Hyperparameter values used in the RSF model in the GEP NEN dataset . . . 69
4.14 Hyperparameter values used in the CGB model in the GEP NEN dataset . . . 69
4.15 Columns removed from CCA dataset . . . 70
4.16 Imputation functions used in MICE depending on the variable datatypes . . . 71
4.17 t-test comparison of missing value strategies . . . 72
5.1 Runtimes of the strategies applied to the simulated missingness in the colon dataset 74
5.2 Cox PH model coefficients fitted on the colon dataset before simulating missingness 74
5.3 Gridsearch runtime for survival analysis models applied to GEP NEN dataset . . . 85
5.4 Top-performing Coxnet models . . . 86
5.5 Top-performing Cox PH models . . . 86
5.6 Top-performing RSF models . . . 87
5.7 Top-performing CGB models . . . 88
5.8 Evaluation of kNN imputation against CCA across different survival models . . . 88
5.9 Evaluating model-based imputation against CCA/kNN imputation . . . 89
5.10 Comparing MI with CCA/kNN imputation . . . 89
C.1 Coxnet models for different *alpha* values with a fixed *L1 Ratio* of 0.2 . . . 149
C.2 Coxnet models for different *alpha* values with a fixed *L1 Ratio* of 0.3 . . . 150
C.3 Coxnet models for different *alpha* values with a fixed *L1 Ratio* of 0.4 . . . 151
C.4 Coxnet models for different *alpha* values with a fixed *L1 Ratio* of 0.5 . . . 152
C.5 Coxnet models for different *alpha* values with a fixed *L1 Ratio* of 0.6 . . . 153
C.6 Coxnet models for different *alpha* values with a fixed *L1 Ratio* of 0.7 . . . 154
C.7 Coxnet models for different *alpha* values with a fixed *L1 Ratio* of 0.8 . . . 155
C.8 Coxnet models for different *alpha* values with a fixed *L1 Ratio* of 0.9 . . . 156

Abbrevations

Description of abbreviations used. Many of these are trivial and universally understood, but for consistency, every abbreviation used is provided.

Abbreviation	**Descriptions**
CCA	Complete Case Analysis
CDF	Cumulative Distribution Function
CGB	Component-Wise Gradient Boosting
CHF	Cumulative Hazard Function
C-index	Concordance Index
Cox PH	Cox Proportional Hazard
GEP	Gastroenteropancreatic
HR	Hazard Ratio
IBS	Integrated Brier Score
kNN	k-Nearest Neighbors
MAR	Missing at Random
MCAR	Missing Completely at Random
MICE	Multiple Imputation by Chained Equations
MiNEN	Mixed Neuroendocrine-Non-Neuroendocrine Neoplasm
MNAR	Missing Not at Random
MAE	Mean Absolute Error
MSE	Mean Square Error
NA	Not Available
NEN	Neuroendocrine Neoplasms
NET	Neuroendocrine Tumors
OLS	Ordinary Least Squares
OOB	Out-of-bag
PCA	Principal Component Analysis
PDF	Probability Density Function
PMM	Predictive-Mean Matching
RSF	Random Survival Forest
RMSE	Root Mean Squared Error
RSKF	Repeated Stratified k-Fold

1 Introduction

1.1 Motivation

Cancer, also known as malignant tumours or neoplasms, is a burden spread widely and rapidly across the world. It is the leading cause of death worldwide; statistics from 2020 show that approximately one in six deaths are cancer-related.[174] Cancer can impact any area of the body and occurs when abnormal cells are rapidly created and grow beyond their usual boundaries. This thesis concerns high-grade gastroenteropancreatic neuroendocrine neoplasms (GEP NEN). This type of cancer has a high mortality rate, and expected long-term survival is poor, even for those with localized disease.[150] Literature on survival analysis of this cancer, compared to other cancers is sparse, compared to other cancers, as it is a rare disease. Most studies are limited to a small number of samples from single institutions.[25,79,80,162]

This highlights the importance of expanding knowledge of this cancer. Using and comparing modern statistics and machine learning, we aim to investigate whether data science approaches can further support the medical domain. For instance, a doctor may have opinions on the expected survival of patients. However, can survival analysis be used to improve the efficiency further? If a doctor is uncertain about a patient's survival, perhaps survival curves produced by machine learning can be used to confirm the alignment between them or support decisions. Data science may give new perspectives and streamline the medical domain. The cooperation between doctors and data scientists may assist in preventing human errors or the other way around by correcting machine errors.

Predicting outcomes in cancer has long been a focal point for clinicians, healthcare workers, and patients.[158] Traditional methods used by the medical domain are heavily weighted towards classical statistics, employing strategies such as the Kaplan-Meier estimator and Cox Proportional Hazards model.[143,153,166] Strategies are often restricted by linearity assumptions with limited capabilities for predictions. Thus, there has been a rising popularity in investigating the potential of using machine learning to improve the accuracy of medical diagnostics.[77,107] Saavedra et al. (2024)[143] found that a boosting approach using component-wise regression as the base learner performed better than the other models, including classical statistical models, for short-term operations of 36 months. Modern machine learning can capture complex and nonlinear patterns. However, they require larger sample sizes as opposed to classical statistical approaches due to the importance of validation.[24] This is especially challenging in the medical domain as there are strict ethical standards regarding privacy and consent. We suggest that effort should be put into preserving the samples that are available, even though variables are incomplete. We are motivated to maximize the use of the available data by overcoming challenges such as large proportions of missing values and small sample sizes.

1.2 Objective

The primary goal of the thesis was to create a reliable model for outcome prediction in GEP NEN using data with incomplete variables. The thesis is divided into two parts, and we consider two research questions.

The first part considered implementing a pipeline for data and feature preprocessing. This included assistance from expert knowledge, handling of missing values, and applying appropriate encodings for categorical variables. After the data and encodings were implemented, the final step was fitting models, model selection and interpretation. The research question for the first part was investigating what models are most suitable for outcome predictions in GEP NENs.

The second part explored the theory and literature of missing values and applied this knowledge to the first part. In addition, a separate experiment was conducted on a different dataset to evaluate different strategies for handling missing values. The research question for the second part was to investigate if it is worth imputing missing values rather than removing all the missingness, either

by the means of omitting variables or samples with missingness.

1.3 Structure of the Thesis

The thesis is divided into two major parts. The first part considers survival analysis of outcome prediction in high-grade GEP NEN, and the other part explores approaches for handling missing values.

The theoretical parts are covered separately, including all necessary materials. Theory is often used interchangeably between the two parts, and we refer to the correct sections where necessary. When it comes to the results and discussion, we present the experiment of missing values before the survival analysis. This makes it easier to refer back to findings when discussing the survival analysis (second part).

1.4 Notation, Terminologies and Abbreviations

We define our data as follows: capital bold letters represent matrices, whereas lowercase bold letters represent vectors. A sample from a vector is lowercase and not bold. Next, consider the feature matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$. Here, n, m are integers where n represents the number of samples and m represents the number of columns. For simplicity, we assume that categorical variables are encoded to a numerical representation to simplify notations. Next, let $\mathbf{Y} \in \mathbb{R}^{n \times 2}$ be the target variable containing the survival times and if the death was observed.

Common terminologies to represent the columns of the matrix $\mathbf{X}$ are independent variables (IV), explanatory variables (EV), covariates, variables, features and attributes. For $\mathbf{y}$, common names are target variable, dependent variable, response variable and outcome variable.

Abbreviations are explained upon first use and can be found in Abbreviations.

1.5 Generative AI and Other Tools Used in This Thesis

With the recent breakthroughs of generative AI and other tools, it is natural to explain what has (and not) been used for the thesis.

OpenAI's ChatGPT 3.5 and 4 are recently published generative AI models with multiple qualities. They can generate text, answer questions based off of its training data, read pictures, analyze datafiles and much more.[108] These models are great tools, however, they have their downsides. They are capable of providing false information and do not hesitate to do so.[2] The two main uses of ChatGPT 4 in this thesis was assisting with LaTeX and coding. For LaTeX, example uses was feeding ChatGPT 4 a reference to receive the appropriate BibTeX entry as reply, or for providing support with creating LaTeX tables. For the coding part, ChatGPT 4 has mostly been present as a technical assistant. For example, providing ideas on how to graph certain features or debug errors. Naturally, all output from ChatGPT was validated before application. The use of generative AI followed the guidelines from NMBU.[105] To emphasize, **no generated information from ChatGPT has been used in the thesis; we always refer to literature.** Github copilot is a software specifically designed to assist with coding and has been a major tool used in the experimental setup for missing values. It has not been used for the survival analysis part because of sensitive information related to the GEP NEN dataset.[95] In addition to generative AI, we have used Grammarly as a writing assistant. It is a convenient built-in Chrome extension that provides grammatical corrections and suggestions.[40]

2 Theory Part A: Survival Analysis

2.1 High-Grade GEP NEN

GEP NEN represents a group of rare tumours emanating from neuroendocrine cells. NENs can develop in various organs and are most commonly found in the gastrointestinal tract, pancreas, lungs, thymus and various endocrine organs.[178]

2.1.1 WHO Classification System

The 2019 WHO classification system categorizes NENs into well-differentiated neuroendocrine tumors (NETs), poorly differentiated neuroendocrine carcinomas (NECs), and mixed neuroendocrine-non-neuroendocrine neoplasm (MiNENs), among other hyperplastic and preneoplastic lesions.[101,135] These are distinguished based on the Ki-67 proliferation index, and are shown in Table 2.1. The Ki-67 is a widely used marker in the assessment of the malignant potential of NENs.[96]

Table 2.1: WHO's classification and grading criteria for NENs of the gastrointestinal tract and hepatopancreatobiliary organs.[101]

Terminology	**Differentiation**	**Grade**	**Ki-67 index**
NET, G1	Well differentiated	Low	<3%
NET, G2		Intermediate	3-20%
NET, G3		High	>20%
NEC, small-cell type	Poorly differentiated	High†	>20%
NEC, large-cell type		High	>20%
MiNEN	Well or poorly differentiated	Variable	Variable

† While NECs that are poorly differentiated does not have a formal grading, they are considered high-grade by definition.

NETs are typically slow-growing, and while they can arise from various tissues, they are predominantly located within the gastrointestinal tract, accounting for 62-67% of all occurences. Pancreatic NENs are notable for their aggressive nature, with a significant proportion presenting as malignant at the time of diagnosis.[13,157,178]

2.1.2 Treatments and Survival

NENs contribute to around 0.5 % of the total cancer cases.[157,178] Xu et al. (2021) found that the age-adjusted annual incidence of GEP NENs are approximately 5.45 per 100 000 in 2015.[176] Detecting NENs primaries and metastases poses challenges due to their tendency to manifest as diminutive lesions and varying anatomical sites.[13,157,178]

Tumours are treated with chemotherapy, and the regimens for NENs vary based on tumour aggressiveness and histological differentiation. Systemic chemotherapy, especially combination therapies, is typically reserved for rapidly progressing or poorly differentiated tumours like NECs. Conversely, well-differentiated gastrointestinal NETs often show less responsiveness to such treatments.[177] The expected survival for patients with high-grade GEP NENs are poor, even after chemotherapy treatments. H. Sorbye et al. (2014) found that patients with GEP NEC who did not undergo chemotherapy had a median survival duration of one month, with a 95% confidence interval ranging from 0.3 to 1.8 months. In contrast, those who were treated with chemotherapy had a median survival of eleven months, with a 95% confidence interval between 9.4 and 12.6 months. The two-year survival rate was 14%, while the three-year survival rate stood at 9.5%.[151]

2.2 Survival Analysis Fundamentals

Survival analysis generally encompasses a set of statistical techniques used to analyze data in which the primary focus is on the time it takes for an event to happen, also known as *time until an event* occurs.[75,124] This type of data is also called lifetime, failure time, or survival data. The event of interest includes both adverse and positive occurrences. Examples of adverse events include mortality, disease recurrence, the onset of adverse reactions, or the emergence of a new medical condition. On the positive side, events could include recovery milestones such as discharge from the hospital or achieving remission, while also involving reaching educational goals or significant achievements in personal development.

In the context of survival analysis, we use the term "survival time" to describe the time variable, representing the duration for which an individual has remained under observation during a specific follow-up period.[75] We refer to the occurrence we are studying as the *event*. This term encompasses various outcomes, including death or disease onset. This term allows for a broader scope of analysis without implying negativity.

2.2.1 Censoring

One of the main characteristics and fundamental concepts in survival analysis is the event indicator. The target values used for training a survival model include two pieces of information: the survival duration and an event indicator, signalling whether or not an event of interest has occurred. A significant challenge in this field is dealing with censored observations, which take place when there is only partial knowledge of an individual's survival time.[75,124,154] Censoring highlights the complexity of survival analysis by pointing to the limitations in obtaining complete data on survival times. Censored patients did not experience the event of interest within the specified study period.[34]

There are generally three reasons why censoring occurs, as illustrated in Figure 2.1

1. A patient departs from the study due to reasons such as death (if death is not the event of interest) or other factors like adverse drug reactions or competing risks.
2. A patient does not experience the event before the study ends.
3. A patient exiting the study.

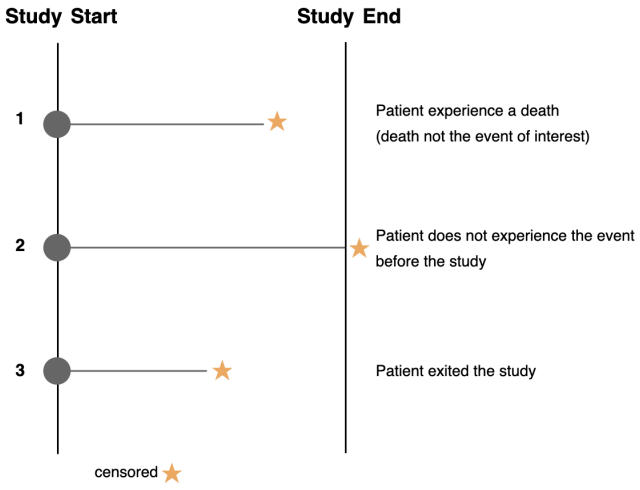


Figure 2.1: Three examples of why censoring occurs. Patient 1 is censored at death due to unrelated causes where the survival time is from study start to death. Patient 2 is censored as it did not experience the event of interest throughout the observation period. Patient 3 is censored due to early exit from the study, and the survival time is calculated from the study's initiation to the point of last contact. The figure is inspired by Leboo et al. (2023)[83].

For instance, in the scenario where the study's event of interest is discharged from the hospital after successful treatment, *patient 1* in Figure 2.1 is censored due to their death. This death, caused by factors unrelated to hospital treatment, prevents the observation of the event of interest, namely, hospital discharge. For *patient 2*, the study ended while the patient was still in remission, indicating that the event of interest, hospital discharge following successful treatment, had not occurred. Consequently, this patient is considered censored at the study's end.[75,83] *Patient C* withdrew from the study, possibly due to adverse effects, personal reasons, or health changes. *Patient C* is censored as the event of interest, hospital discharge following successful treatment, could not be observed. All of these examples are **right-censoring**.

There are three types of censoring: right censoring, left censoring, and interval censoring, as shown in Figure 2.2.

Right Censoring

In right censoring, the complete duration of survival is not fully known.[75,83,144,154] This means the latter part of the survival timeline is not observed and is therefore censored, affecting the rightmost portion of the data we have. Illustrated in Figure 2.2, *patient A* and *patient B* are identified as right-censored since $t_A, t_B > x_2$. This categorization is due to the event of interest not occurring within the observation period, irrespective of whether the event occurs after the study ends. Right-censoring occurs when the time to an event, denoted as t_i, exceeds the study's end time, represented by x_2. Here, i represents individual patients, $i = 1, .., n$:

$$t_i > x_2$$

Left Censoring

Left censoring in survival analysis occurs when the event of interest happens before the start of the study, indicating that the individual's survival time t_i is less than the earliest observable time x_1, which is when the study begins.[75,83,144,154] This scenario arises very rarely. In Figure 2.2, *patient C* is identified as left censored since $t_C < x_1$. This categorization is due to the event occurring before the study commenced. Consider a stroke-related clinical study where some patients experienced a stroke before the start of the study. If it is known that these strokes occurred before the study began but the exact timing is unknown, such patients are classified as left-censored. Left-censoring is defined when the patient´s survival time t_i is less than the value observed when the study starts x_1:

$$t_i < x_1$$

Interval Censoring

Interval censoring in survival analysis occurs when the precise survival time t_i of an individual is not definitively known, but it is known that it occurs during a time interval, from x_2 to x_3.[75,83,144,154] This type of censoring, common in clinical trials or longitudinal studies with periodic follow-up sessions, determines that a patient's event occurs within the interval between two follow-up visits, not at an exact, identifiable moment. In Figure 2.2, *pasient D* is interval-censored since $x_3 < t_D < x_4$. Interval censoring indicates that the interval in which the event occurred is known (dotted line), but the precise time of the event, symbolized by the clock, remains uncertain. Interval-censoring can be described as:

$$x_3 < t_i < x_4$$

Patient E experienced the event of interest during the study period, thus it is not considered censored.

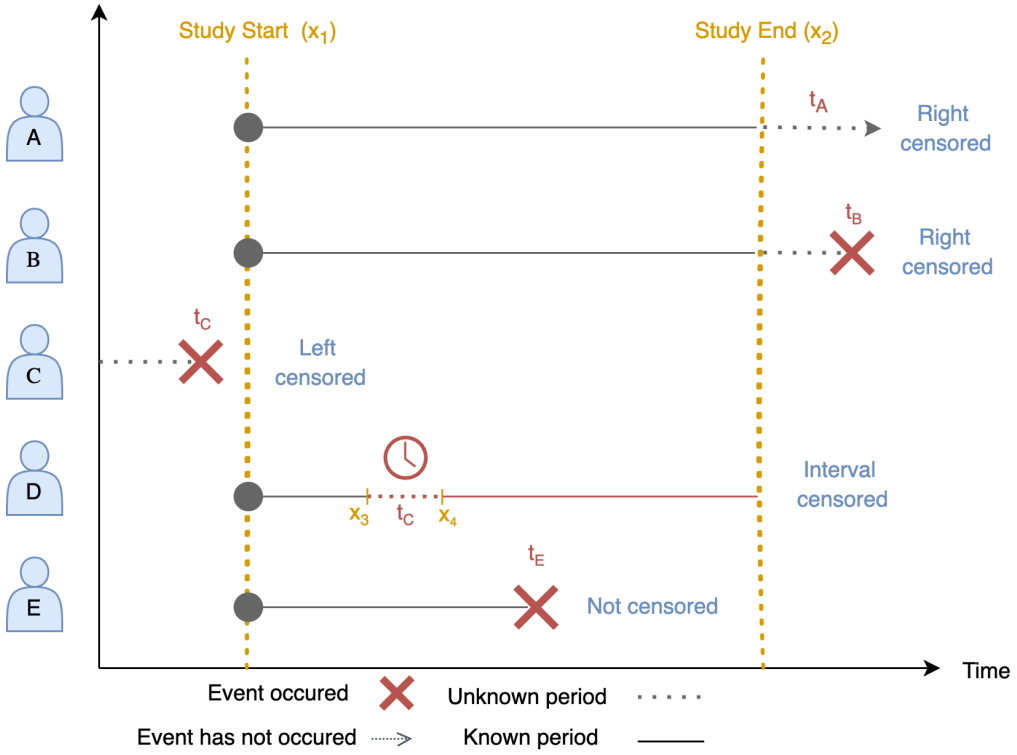


Figure 2.2: The figure illustrates the timeline for each patient from study start (x_1) to end (x_2), highlighting the point at which the event is either known, unknown, censored, or the event occurred. Patients A and B are still in the study at the end (x_2) and are right-censored. Patient C entered the study (x_1) after the event and is therefore left censored. Patient D is interval censored with the event occurring between two observation points (x_3 and x_4). Lastly, patient E is not censored since the event occurred within the study period. The figure is inspired by Leboo et al. (2023)[83].

This thesis will consider right-censored data, meaning it covers situations where the event of interest hasn't happened by the study's end.[75,124,154]

2.2.2 Survival Function

In survival analysis, the emphasis is not on predicting the exact timing of an event, but rather on forecasting a function, specifically the survival or hazard function.[54,155,165] The survival function $S(t)$ returns the probability of survival $P(T > t)$ past time t, and can be expressed as follows:

$$S(t) = P(T > t) \tag{2.1}$$

T represents a random variable that signifies an individual's survival time, quantifying the duration until a particular event of interest.[75,155] It captures the uncertainty and variability in the survival times across individuals. Given that T is defined in terms of time, it can only take non-negative values, meaning T can be any number that is zero or greater. On the other hand, t refers to a specific point in time within the study period. It is used to evaluate the survival function, $S(t)$, indicating the probability that an individual´s survival time T exceeds this specific time t. Essentially, t is the moment up to which survival is being assessed.

For instance, when assessing the probability of a patient surviving beyond 2 years following cancer treatment, we set t to 2.[75,155] This allows us to investigate the value of the survival curve at $T = 2$, represents the probability that an individual survives longer than 2 years, $t = 2$. This probability is expressed as $S(2) = P(T > 2)$, and as illustrated in Figure 2.3, $S(2)$ is determined to be 0.8. The survival function is a monotonically decreasing function of t, indicating that the probability of survival either decreases or remains the same over time. As the time variable moves from 0 towards infinity, the probability of survival decreases from 1 to 0.

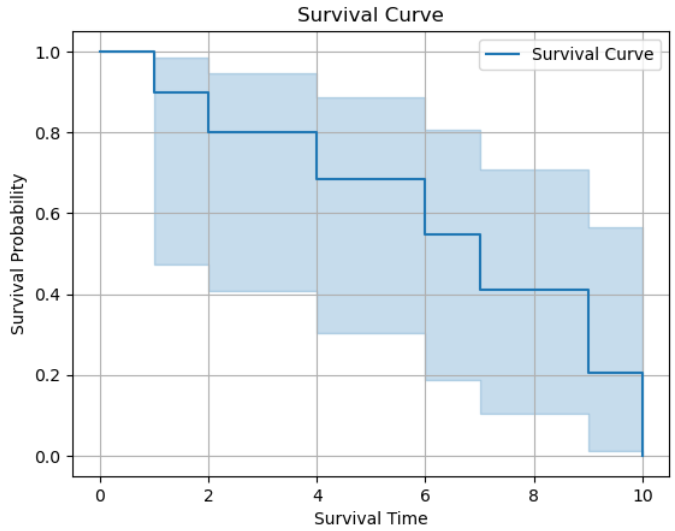


Figure 2.3: Example of a survival curve showing survival probabilities over time. The shaded steps indicate changes in survival due to events. To estimate the probability that an individual survives beyond 2 years, consider $S(2) = P(T > 2)$, which equals 0.8.

2.2.3 Hazard Function

The hazard function $h(t)$ is an important quantitative term in survival analysis. It represents the immediate risk of an event happening per unit of time, assuming the individual has remained alive up to that point. $h(t)$ "gives the instantaneous potential at time t for getting an event, given survival up to time t".[75] The hazard function primarily concentrates on the occurrence of failure, or the event happening. Therefore, it can be viewed as providing information that complements what is offered by the survival function, essentially showing the reverse perspective. The hazard function is mathematically represented as follows:

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t | T \geq t)}{\Delta t} \tag{2.2}$$

Eq. (2.2) defines the hazard function, $h(t)$, as the limit of the probability that the event of interest occurs in a small interval $[t, t + \Delta t]$, given that the individual has survived until t, divided by the length of the interval Δt, as Δt approaches 0.[75,155] The hazard function is also called *conditional failure rate* or *instantaneous failure rate*.[165]

Since we obtain a probability per unit of time, the hazard function is a rate.[75] The hazard function will never be negative and has no upper bound, so it ranges between 0 and infinity. If the hazards are high, that indicates a greater immediate risk of the event occurring, suggesting a shorter expected duration until the event takes place. For example, in a clinical trial measuring patient survival after treatment, the hazard function represents the immediate risk of a patient dying at any moment, assuming they have survived up until that point.

2.2.4 Relationships Between Hazard and Survival Function

The survival and hazard functions are closely related, allowing the derivation of one from the knowledge of the other. The connection between $S(t)$ and $h(t)$ can be equivalently demonstrated through two equations presented in Eqs. (2.3) and (2.4):[75]

$$S(t) = \exp\left[-\int_0^t h(u)\,du\right] \tag{2.3}$$

$$h(t) = -\left[\frac{dS(t)/dt}{S(t)}\right] \tag{2.4}$$

Eq. (2.3) shows that the survival function $S(t)$, which provides the probability of an event not occurring until after a specific time t, can be calculated as the cumulative hazard function (CHF). This CHF, $H(t)$, is calculated using the integral $\int_0^t h(u)$.[75] It reflects the total risk accumulated up to time t, which, when negatively exponentiated, yields the survival probability. This indicates how survival decreases as accumulated risk increases over time.

Figure 2.4 illustrates the relationships between the various functions. $H(t)$ and its relationships with $h(t)$ and $S(t)$ has been detailed earlier.[75,155]The probability density function (PDF), denoted by $f(t)$, provides the probability per unit time of the event at time t. The PDF is derived from $h(t)$ multiplied by $S(t)$. The cumulative distribution function CDF, denoted by $F(t)$, represents the probability that the event has occurred by or before the duration t. It is the integral of the PDF from 0 to t, $\int_0^t f(s)ds$. $S(t)$ is the probability that the event has not occurred by time t. From the relationships shown in Figure 2.4, $S(t)$ can be expressed as $1 - F(t)$, which is 1 minus CDF. It is straightforward to convert back and forth between all the different functions.

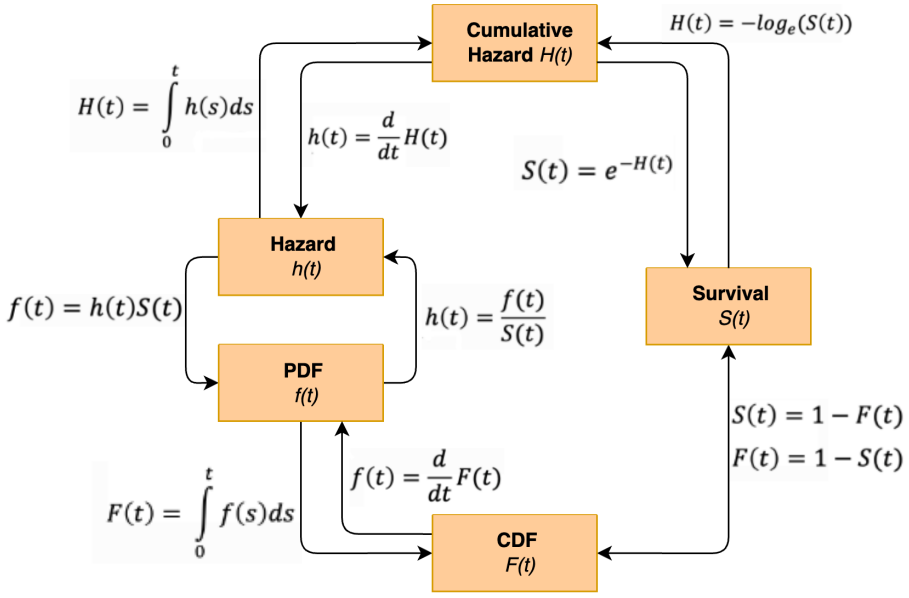


Figure 2.4: A diagram of the mathematical relationships between the Survival Function, Hazard Function, Cumulative Hazard Function (CHF), Probability Density Function (PDF), and Cumulative Distribution Function (CDF). The diagram is inspired by[155].

2.3 Survival Analysis Models

2.3.1 Cox Proportional Hazard

While the Cox proportional hazards (Cox PH) is not strictly a machine learning algorithm, it is a crucial statistical tool used in survival analysis to evaluate the impact of variables on the timing of events. The Cox PH model stands as the foremost and most popular mathematical method for calculating survival curves, while simultaneously accounting for multiple explanatory variables.[75] It is used to investigate the effect of several variables on the time to an event of interest. The model is particularly valued for its capability to handle censored data, a common challenge in survival analysis, although it is worth noting that other survival analysis methods also manage censored data effectively. The model defines the hazard function $h(t|\mathbf{X})$ as the product of a time-dependent baseline hazard function, $h_0(t)$, and the exponential of a linear combination of time-independent covariates $\mathbf{X}$. Recall that m represents the number of covariates:

$$h(t|\boldsymbol{X}) = h_0(t) \exp\left(\sum_{i=1}^{m} \beta_i X_i\right) \tag{2.5}$$

As seen, the baseline hazard function $h_0(t)$ represents the hazard rate for an individual with all covariate values set to zero.[54,75] It is a function of time t but does not involve the covariates directly, allowing flexibility in how the hazard changes over time. $\exp\left(\sum_{i=1}^{m} \beta_i X_i\right)$ is the exponential e of the linear sum of $\beta_i X_i$, where the sum is over the m explanatory $\mathbf{X}$ variables. Here β_i is the coefficient that quantifies the change in the hazard (risk) of the event occurring for a one-unit increase in the covariate, also called the hazard ratio (HR).[75] For instance, if a study reports an HR of 1.03 for age, that means that each additional year of age increases the event´s risk by 3%.[30] This exponential part adjusts the hazard based on the covariate values and is always non-negative. This is because when all the $\mathbf{X}$´s are equal to zero, the equation reduces to the baseline hazard function that is never negative. Therefore, the hazard function will always be positive.[75] Given that $\mathbf{X}^1$ is denoted the set of covariates for one individual and $\mathbf{X}^2$ denotes the set of covariates for another individual, and given that we have maximum likelihood estimates denoted by β_i, the hazard ratio (HR) can be expressed as:

$$HR = \frac{h(t, \boldsymbol{X}^1)}{h(t, \boldsymbol{X}^2)} = \frac{h_0(t) \exp\left(\sum_{i=1}^{m} \beta_i {X_i}^1\right)}{h_0(t) \exp\left(\sum_{i=1}^{m} \beta_i {X_i}^2\right)} \tag{2.6}$$

In Eq. (2.6), the baseline hazards cancel out, allowing the expression to be simplified as:

$$HR = \exp\left(\sum_{i=1}^{m} \beta_i (X_i^1 - X_i^2)\right) \tag{2.7}$$

One of the Cox PH key features is the assumption of proportional hazards, namely that the hazard ratio between two individuals with different covariate values remains constant over time.[54,75] Additionally, the model's semiparametric nature, characterized by its unspecified baseline hazard function $h_0(t)$, allows for great flexibility and robustness in analysis without the need to explicitly define the shape of the hazard over time. This unique combination of covariates makes the Cox PH model a versatile and widely used tool in survival analysis, applicable across various fields, especially in medical research where understanding the impact of covariates on survival time is crucial.

The Cox PH model, while widely used, does face limitations, particularly in handling high-dimensional datasets. Cox PH model struggles because it is not naturally equipped to handle situations where the number of variables is very large compared to the number of observations, which can lead to difficulties in model estimation and stability.[14,26,46] Another challenge with Cox PH models is their use of group sparse regularization, a technique that reduces the number of variables to help deal with high-dimensional data. Unfortunately, this can lead to excessive

smoothing or shrinkage, which can negatively impact the model's predictive performance because it may overly simplify the model and miss important predictors.

When using the Cox PH model, it is important to fit the model with the correct parameters to accurately interpret the impact of covariates on survival times. This fitting process depends on the method of maximum likelihood estimation (MLE), which, in the context of the Cox PH model, is called a "partial" likelihood.[74,75,90] That is because the likelihood for the Cox model does not consider probabilities for all samples. Partial likelihood is also unique because it is based on the order of events rather than their distribution, focusing just on the probabilities for subjects who experience events while excluding those who are censored. The partial likelihood approach allows for the estimation of regression coefficients without the need to fully specify the baseline hazard function $h_0(t)$.

Denoting n as the number of samples and δ as the event (1=observed, 0=unobserved), then D is the index set of observed events $D = \{i|\delta = 1\} \subseteq \{1, .., n\}$.[12,15,75,54] Next, $\mathbf{x}_i$ is the covariate vector for the i-th individual who experienced an event, and $\boldsymbol{\beta}$ is the vector of coefficients for each covariate that are to be estimated. Finally, we use the notation that $R(t_i) = \{j|t_j \geq t\}$, for any $t \geq 0$, as the risk set at time t. The risk set includes all samples who have not yet experienced the event or been censored up to and including time t_i, making them at risk of the event occurring at that time. This risk set decreases in size as the observation time increases. x_j is the covariate vector for the j-th individual, which depends on the risk set. With these notations, denoting $L_p(\boldsymbol{\beta})$ as the partial likelihood is defined as follows:

$$L_p(\boldsymbol{\beta}) = \prod_{i=1}^{D} \frac{\exp(\boldsymbol{\beta}^T \boldsymbol{x}_i)}{\sum_{j \in R(t_i)} \exp(\boldsymbol{\beta}^T \boldsymbol{x}_j)} \tag{2.8}$$

The partial likelihood, $L_p(\boldsymbol{\beta})$, in Eq. (2.8), for the Cox PH model, represents a product of conditional probabilities for observed events, given their occurrence, and also factoring the probabilities of censoring for those samples that have not yet experienced the event. $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are not to be confused with a specific variable in $\boldsymbol{X}$, it is the rows of sample i and j. For the Cox model, after forming the partial likelihood function for a specific model, the next step involves maximizing its natural logarithm, $log(L_p(\boldsymbol{\beta}))$, to achieve the maximization of the overall likelihood function. This is done by taking the partial derivatives of the log of $L_p(\boldsymbol{\beta})$ with respect to each parameter in the model, illustrated in Eq. (2.9):[54,75]

$$\begin{gathered} \frac{\partial \log(L_p(\boldsymbol{\beta}))}{\partial \boldsymbol{\beta}_i} = 0 \\ i = 1, ..., m \end{gathered} \tag{2.9}$$

Then $log(L_p(\boldsymbol{\beta}))$ is defined as:[54,74]

$$\log(L_p(\boldsymbol{\beta})) = \sum_{i=1}^{D} \left[\boldsymbol{\beta}^T \boldsymbol{x}_i - \log \left(\sum_{j \in R(t_i)} \exp(\boldsymbol{\beta}^T \boldsymbol{x}_j) \right) \right] \tag{2.10}$$

2.3.2 Penalized Cox Model (Coxnet)

The Cox PH model, known for making hazard ratios understandable, struggles with high dimensional data across many features due to feature correlation causing singular matrices, which impedes accurate coefficient estimation.[117,147,184] Coxnet, enhancing the Cox model, integrates regularization methods like LASSO (L1 Regularization), Ridge (L2 Regularization), or their combination (Elastic Net Regularization), offering a refined approach to manage complex and high-dimensional datasets by mitigating overfitting and enhancing model stability and performance.[117]

Ridge (L2 Regularization)

Introducing an L2 penalty to the Cox PH model helps mitigate issues with singularity by encouraging coefficient shrinkage towards zero.[117,147,184] This L2-penalty term will be added to the likelihood function in Eq. (2.8), and the modified objective function becomes:

$$\arg\max_{\boldsymbol{\beta}} \left\{ \log L_p(\boldsymbol{\beta}) - \frac{\alpha}{2} \sum_{j=1}^{m} \beta_j^2 \right\} \tag{2.11}$$

In Eq. (2.11), the symbol m denotes the number of features included in the model and $\boldsymbol{\beta}$ represents the vector of coefficients for the m features, $\beta_1, ..., \beta_m$. The function $\log L(\boldsymbol{\beta})$ represents the log-likelihood, while $\alpha \geq 0$ is the regularization parameter that controls the magnitude of the L2 penalty and, therefore, also controls the amount of shrinkage.[9,117,147] The L2 penalty term $\frac{\alpha}{2} \sum_{j=1}^{m} \beta_j^2$ sums the squares of the coefficient values, which contributes to the overall cost function by increasing the penalty as the magnitude of the coefficients increases. This penalization encourages the optimization algorithm to select smaller or more regularized coefficient values, effectively shrinking them towards zero to reduce model complexity and prevent overfitting. The goal is to find coefficient values that optimize the penalized log-likelihood, balancing between model performance and complexity. The penalty ensures that coefficient weights never reach zero, thus preventing the solution from becoming sparse. This regularization process is crucial for preventing overfitting, which is a common concern in any modeling scenario, regardless of whether the number of features (m) is greater than or less than the number of samples (n). However, it becomes particularly vital when m exceeds n ($m > n$) or when m is much greater than n ($m \gg n$), as overfitting can occur more easily under these conditions. However, ridge regression is particularly effective in managing correlated features. In cases where there is high correlation between features, ridge regression tends to distribute coefficient weights among them in a way that mitigates multicollinearity issues.[147] We obtain the standard unpenalized Cox model by setting the log parameter α to zero.[117]

LASSO (L1 Regularization)

The L1-Regualization, also called the LASSO penalty, stands for Least Absolute Shrinkage And Selector Operator.[9,117,147,184] The primary distinction between Ridge and LASSO lies in how they handle coefficients; while Ridge shrinks coefficients towards zero, LASSO has the capacity to set many coefficients to exactly zero, depending on the strength of the penalty, effectively excluding them from the model. While this method is frequently advantageous, it can lead to issues. For instance, when two features exhibit high correlation, LASSO may favour one and completely disregard the other.[147] While this method often improves model performance by reducing complexity, it can create challenges in contexts where understanding the influence of each feature is crucial. As a result, researchers may overlook critical insights associated with the excluded features. The L1-penalty can be added to the likelihood function of Eq. (2.8), and the modified objective function is:[117]

$$\arg\max_{\boldsymbol{\beta}} \left\{ \log L_p(\boldsymbol{\beta}) \ - \alpha \sum_{j=1}^{m} |\beta_j| \right\} \tag{2.12}$$

In the optimization problem defined in Eq. (2.12), the term $\arg\max_{\boldsymbol{\beta}}$ denote the values of the coefficient $\boldsymbol{\beta}$ that maximizes the objective function.[147] The function $\log L(\boldsymbol{\beta})$ represents the log-likelihood, which assesses the fit of the model with parameters $\boldsymbol{\beta}$ to the observed data. The subtracted term, $-\alpha \sum_{j=1}^{m} |\beta_j|$, introduces the L1 regularization, also known as the LASSO penalty. Here, α is the regularization strength, a positive scalar that controls the strength of the penalty applied to the coefficients. A high value of α leads to stronger regularization, encouraging greater sparsity by setting more coefficients to exactly zero. Conversely, decreasing α results in less penalization, allowing more features to remain in the model but potentially increasing the risk of overfitting if α is too low. The sum $\sum_{j=1}^{m} |\beta_j|$ aggregates the absolute values of the regression coefficients, promoting sparsity. This sparsity is crucial to LASSO's feature selection and regularization ability, enhancing the model's prediction performance and interpretability, especially in high-dimensional data contexts.[117]

Elastic Net

The Elastic penalty is a regularization technique that combines the strength of both LASSO (L1) and Ridge (L2) penalties.[9,147,117,184] It is designed to feature select like LASSO while effectively managing the influence of highly correlated features, similar to Ridge. By combining these two approaches, the Elastic Net overcomes the limitations of each, encouraging both sparsity and stability in the coefficients, which is particularly beneficial in models with numerous features. Incorporating the Elastic Net penalty into the optimization of the likelihood function, we redefine the problem as follows:[117]

$$\arg\max_{\boldsymbol{\beta}} \left\{ \log L_p(\boldsymbol{\beta}) - \alpha \left(r \sum_{j=1}^{m} |\beta_j| + \frac{1-r}{2} \sum_{j=1}^{m} \beta_j^2 \right) \right\} \tag{2.13}$$

Eq. (2.13) merges L1 regularization from Eq. (2.12) and L2 regularization from (2.11), with the parameter r blending the regularization balance between these two.[147,117] The model transitions from Ridge to LASSO behavior as r varies from 0 to 1. If r is near 0, the Elastic Net leans towards a Ridge-like regularization, favoring coefficient shrinkage while managing multicollinearity. When r is near 1, the behavior aligns more with LASSO, favoring sparsity and potentially setting some coefficients to zero. This adaptability allows the Elastic Net to combine feature selection and regularization by tuning the model based on the value of r.

2.3.3 Random Survival Forest

Random Survival Forest (RSF) is a nonparametric and advanced ensemble learning method that leverages base learners, such as survival trees, to analyze right-censored survival data.[60,98] By incorporating randomization in selecting data samples and features for tree construction, RSF enhances predictive performance from these base learners. This approach significantly improves traditional ensemble techniques, making RSF a powerful tool for survival data analysis.

RSF operates as a composite estimator, as shown in Figure 2.5, constructing s survival trees from different subsets of the dataset to bolster predictive performance and reduce overfitting.[61,60] The number of survival trees s the model will grow is determined by the hyperparameter $n\_estimators$. These samples in each survival tree are drawn with replacement through bootstrap sampling.[61] As shown in Figure 2.5, the survival trees are grown by randomly selecting a subset of g features at each node. The optimal split is determined using the log-rank splitting rule, focusing on the feature that maximizes survival differences across parent nodes while ensuring each child node has no fewer than a specified number of samples. This is determined by the hyperparameter $min\_samples\_leaf$. To finalize the model, we estimate an ensemble cumulative hazard function (Ensemble CHF) by first computing the cumulative hazard function (CHF) as discussed in Section 2.2.4, within every survival tree. Then, we take the average of these CHF survival trees up to the number of n survival trees. As shown in the Figure 2.5, to further refine the model´s performance, we assess the ensemble´s prediction error using out-of-bag (OOB) data, data not used during the survival tree´s training. This prediction error is critical to the model´s ability to make precise and broadly applicable predictions. Through this comprehensive process, the model is fine-tuned for optimal performance and generalizability.

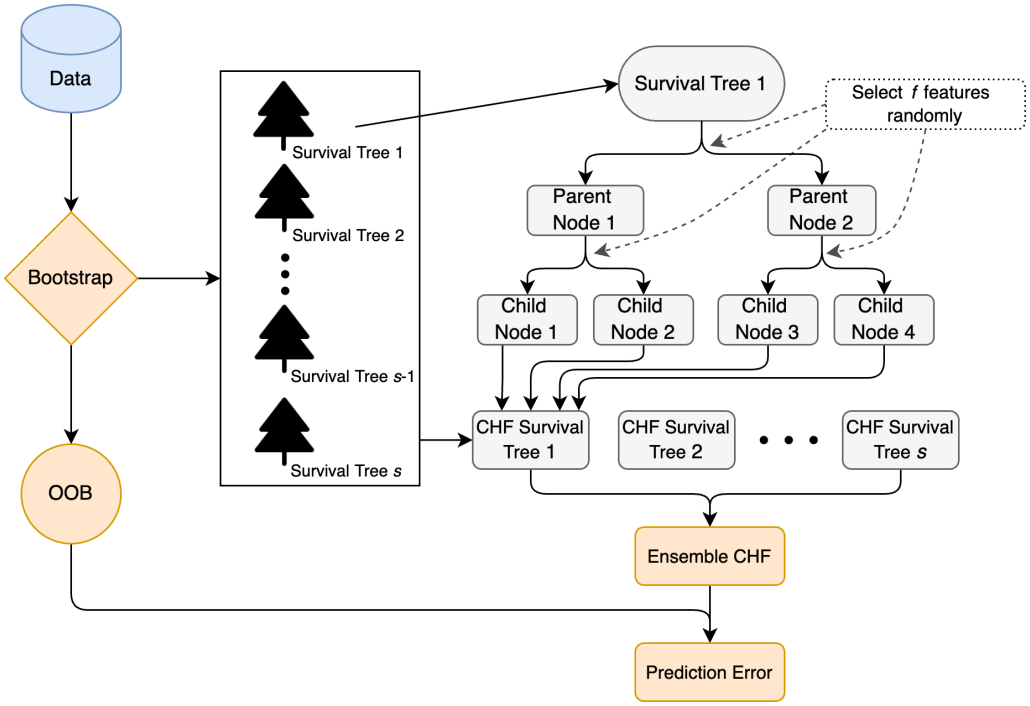


Figure 2.5: Overview of the operational mechanics behind the RSF algorithm, depicting the process from data input through bootstrap sampling to the aggregation of cumulative hazard functions (CHF) from multiple survival trees into an ensemble CHF. The figure illustrates an example with two parent nodes at a depth of two, though in practice, each survival tree can have a larger depth. CHF refers to the cumulative hazard function, which estimates the aggregate risk of an event occurring over time within each tree. The figure is inspired by Chen et al. (2019)[21].

Whenever a decision needs to be made at a parent node based on a specific feature x, we evaluate the potential splits of the form $x \leq c$ and $x > c$, where c represents a threshold value for the feature x.[61] This threshold c is chosen to best separate the data into two child nodes, one left and one

right child node, each with its own subset of survival data. The split will maximize the survival differences across these child nodes. As illustrated in Figure 2.5, this process yields two subsets of survival data located at child nodes 1 and 2 for parent node 1. Similarly, for parent node 2, additional subsets of survival data are illustrated, corresponding to child nodes 3 and 4.

Let $t_1 < t_2 < ... < t_k$ be the distinct event times in the parent node p, and let an individual i with an event in child node j be defined as $i = 1, \ldots, n$ and $j = 1, 2$. The risk set for the parent node is defined as $R(t_{(f)})$, and for the left and right child nodes, the risk sets, $R_1(t_{(f)})$ and $R_2(t_{(f)})$ are defined as:[61]

$$R_1(t_{(f)}) = \#\{t_i \geq t_{(f)}, x_i \leq c\} = n_1 \tag{2.14}$$

$$R_2(t_{(f)}) = \#\{t_i \geq t_{(f)}, x_i > c\} = n_2 \tag{2.15}$$

$$R(t_{(f)}) = R_1(t_{(f)}) + R_2(t_{(f)}) = n_1 + n_2 = n \tag{2.16}$$

where $t_{(f)}$ represents the time at which the number of samples in the child node, either alive or having an event, is determined. Further is the value of x_i for individual i.

The quality of a split within the survival tree is evaluated using the log-rank splitting rule.[61] Eqs. (2.14), (2.15), and (2.16) are applied to calculate the log-rank split, which is pivotal for determining the most informative split. The expression $d_j(t_{(f)})$ denotes the number of individuals experiencing an event at time $t_{(f)}$ within child node j. The k represents the total number of individuals who have experienced an event in the parent node at time $t_{(f)}$. The computation of the log-rank statistic, $L(x, c)$, is defined as:

$$L(x,c) = \frac{\sum_{f=1}^{k} \left(d_1(t_{(f)}) - R_1(t_{(f)}) \frac{d(t_{(f)})}{R(t_{(f)})} \right)}{\sqrt{\sum_{f=1}^{k} \frac{R_1(t_{(f)})}{R(t_{(f)})} \left(1 - \frac{R_1(t_{(f)})}{R(t_{(f)})}\right) \left(\frac{R(t_{(f)}) - d(t_{(f)})}{R(t_{(f)}) - 1}\right) d(t_{(f)})}} \tag{2.17}$$

From Eq. (2.17), the magnitude of $|L(x, c)|$ quantifies the degree of separation achieved by the node. A higher value of $|L(x, c)|$ indicates a greater difference between the two nodes, signifying a more effective split.[61] So to find the best split at a node, find the feature x^* and split value c^* such that $|L(x^*, c^*)| \geq |L(x, c)| \forall \{x, c\}$. After *n_estimators* survival trees in the RSF have been constructed, as shown in Figure 2.5, the model computes the CHF for each survival tree. The CHF for each survival tree is determined by adding up the hazard contributions from each terminal node (the nodes at the bottom of each survival tree), which are child nodes in Figure 2.5. The CHF for a specific terminal node j, denoted by $\hat{H}_j(t)$, is calculated by summing up the ratio of a number of individuals with an event $d_j(t_{(f)})$ to the risk set $R_j(t_{(f)})$ at the ordered failure times for child node $t_{j,(f)}$:

$$\hat{H}_j(t) = \sum_{t_{j,(f)} \leq t} \frac{d_j(t_{(f)})}{R_j(t_{(f)})} \tag{2.18}$$

Each survival tree provides a sequence of estimates $\hat{H}_j(t)$ equal to the number of terminal nodes within the respective survival tree. This implies that each survival tree within the ensemble contributes its own set of CHF for each terminal node.

To compute the CHF for an individual i with features $\mathbf{x}_i$, $\hat{H}(t|\mathbf{x}_i)$, we send $\mathbf{x}_i$ down the survival tree and toward the terminal node. The CHF at the terminal node j for individual i is:[61]

$$\hat{H}(t|\boldsymbol{x}_i) = \hat{H}_j(t), \text{ if } \mathbf{x}_i \in j. \tag{2.19}$$

$\hat{H}(t|\boldsymbol{x}_i)$ in Eq. (2.19) is computed for all individuals i in one given survival tree. Since this estimate represents the cumulative hazard in one terminal node, averaging these CHF across all s survival tree will give the ensemble cumulative hazard estimate, as shown as ensemble CHF in the Figure 2.5. The ensemble CHF, $\hat{H}_e(t|\boldsymbol{x}_i)$, over all s survival tree is:[61]

$$\hat{H}_e(t|\boldsymbol{x}_i) = \frac{1}{n}\sum_{b=1}^{n}\hat{H}_b(t|\boldsymbol{x}_i). \tag{2.20}$$

Where $\hat{H}_b(t|\boldsymbol{x}_i)$ denotes the CHF from Eq. (2.19) for tree $b = 1, ..., s$. To determine the out-of-bag (OOB) ensemble cumulative hazard estimate $\hat{H}^*_e(t|\mathbf{x}_i)$ for an individual i, as indicated by "OOB" in Figure (2.5), the following approach is utilized:[61]

$$\hat{H}^*_e(t|\boldsymbol{x}_i) = \frac{\sum_{b=1}^{n} I_{i,b}\hat{H}_b(t|\boldsymbol{x}_i)}{\sum_{b=1}^{n} I_{i,b}} \tag{2.21}$$

In Eq. (2.21), the calculation of the estimator is specifically performed for bootstrap samples corresponding to individual i, provided that i is an Out-Of-Bag (OOB) sample in the bootstrap sample. The indicator variable $I_{i,b}$ equals one if individual i is OOB sample for survival tree b, and equals zero otherwise as shown in Eq. (2.22):[61]

$$I_{i,b} = \begin{cases} 1 & \text{if individual } i \text{ is an OOB sample for tree } b, \\ 0 & \text{otherwise.} \end{cases} \tag{2.22}$$

Calculating the OOB ensemble in Eq. 2.21 offers a measure of the model´s performance based on samples not included in the training set for each tree. This method ensures that the performance metric reflects the model's ability to generalize to new unseen data, as the OOB set varies for each tree within the ensemble.

The prediction error shown as the last step in Figure 2.5 can be calculated by measuring differences between the ensemble of OOB CHF and the ensemble CHF.[61] This comparison can be quantified using different metrics, such as the C-index and the Brier score, to determine prediction errors

We can also find the ensemble survival function in Eq. (2.23), $\hat{S}_e(t|x_i)$, by calculating it from ensemble cumulative hazard function (CHF) from Eq. (2.20):[98]

$$\hat{S}_e(t|\boldsymbol{x}_i) = \exp\left\{-\frac{1}{\mathrm{n}}\sum_{b=1}^{\mathrm{n}}\hat{H}_e(t|\boldsymbol{x}_i)\right\}. \tag{2.23}$$

2.3.4 Gradient Boosting

Gradient boosting is an ensemble technique, like Random Survival Forest discussed in Section 2.3.3 that combines prediction of multiple simple models, known as base learners, to produce a more robust overall prediction.[56,116] In contrast to RSF, which constructs numerous survival trees independently and averages their predictions, gradient boosting strategically improves predictions iteratively, focusing on optimizing the loss function for one base learner at a time in a method known as greedy stagewise.[35] Each additional base learner is specifically designed to correct the residuals errors from the combined predictions of the previous base learners, thus "boosting" the model´s predictive performance.[35,56] The cumulative prediction of the gradient boosting model is formulated by iteratively adding the contribution of each base learner $k \in \{1, \ldots, K\}$. The overall prediction function $f(x)$ for the gradient boosting model is the sum of weighted base learner function $\beta_k g(\boldsymbol{x}; \theta_k)$:

$$f(x) = \sum_{k=1}^{K} \beta_k g(\boldsymbol{x}; \theta_k) \tag{2.24}$$

In Eq. (2.24), $\beta_k \in \mathbb{R}$ represents the weight of the k-th base learner in the ensemble. The term $g(x; \theta_k)$ refers to the *base learner* itself, which influenced by a set of parameters $\boldsymbol{\theta}$ which are tuned during the learning process to minimize the model´s loss function on the training data.[116,56] These parameters vary across the individual base learner k, tailoring each one to target different aspects of the error in the prediction task. As new base learners are added in the Eq. (2.24), their weights β_k are adjusted to minimize the model's overall prediction error. This process continues iteratively, with each new base learner being added to complement the existing ensemble and improve the model's prediction.

Different gradient boosting models can be employed, tailored to specific loss function and base learner. In our study, we use Component-Wise Gradient Boosting, which uses component-wise least squares as its base learner.[56,116] "Component-wise" means that each base learner fits a regression model to predict the target variable based on a single feature at a time. This model will have a linear model as the final model. Scikit-survival's implementation defaults to optimizing the partial likelihood loss of the Cox Proportional Hazards model (Section 2.3.1) for Component-Wise Gradient Boosting. Additionally, in the Scikit-survival implementation, three important hyperparameters are *n_estimator*, *learning_rate*, and *subsample*.[56,116,134] The *n_estimator* similar to its use in RSF (Section 2.3.3), specifies the number of boosting stages, which corresponds to the number of survival trees. The *learning_rate* determines how much each survival tree influences the final model. Lower *learning_rate* requires a higher number of estimators (*n_estimator*) to model convergence but improve the model´s robustness to overfitting. In contrast, a higher *learning_rate* can speed up the training, but it requires careful adjustment of the number of trees to prevent underfitting. Lastly, the *subsample* hyperparameter defines the proportion of samples to be used for fitting the individual base learners. *subsample* with lower than 1 results in Stochastic Gradient Boosting (SGB).[36] This sets each base learner to train on a random subset of data rather than the full dataset, introducing randomness. This process also reduces the model variance and increases bias.

2.4 Evaluation of Models

2.4.1 Cross Validation

Measuring a predictive model's performance on unseen data is crucial. Cross-validation is a key technique that splits the dataset into separate parts, known as folds, for training and testing, ensuring the model performs well not just on training data but also on new unseen data.[64] By repeatedly training and testing the model across these different folds, cross-validation gives a complete understanding of the model´s expected performance, helping to avoid overfitting and select the most suitable model.

K-fold Cross-Validation

K-fold Cross-Validation, often referred to as *k-fold CV*, is an approach that randomly divides the dataset into k folds of approximately equal size.[64,137] As shown in Figure 2.6 the k is set to 5. The model is then trained on $k-1$ folds, denoted as the orange area in Figure 2.6 while using the remaining fold as the test set, denoted as blue in Figure 2.6 to evaluate performance. The process is iterated k times, with each k used as the test set exactly once. The results from each fold are averaged to produce a single performance metric. This approach enables efficient data utilization, ensuring that each observation is used for training and testing, providing a comprehensive assessment of the model´s performance.

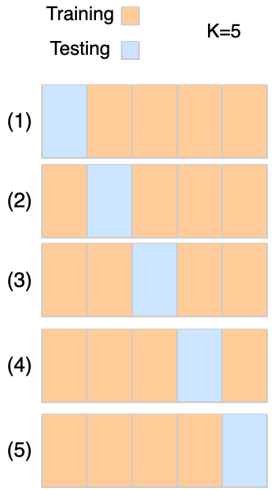


Figure 2.6: Overview of K-fold cross-validation process using $k = 5$. This diagram illustrates how the dataset is divided into five equal parts (folds), each serving as the test set once. For each iteration, one fold is used for testing (shown in blue), and the remaining four folds are used for training (orange). This process is repeated five times, with each fold used as the test set exactly once, ensuring that every data point is used for both training and testing exactly once

Stratified K-fold Cross-Validation

Stratified K-fold, as shown in Figure 2.7 is a variation of K-fold shown in Figure 2.6 ensures that the distribution of each target class is consistent across all train and test sets.[76,119] This method is useful for classification problems with imbalanced distributions of the target classes, as it maintains the relative class frequencies, thereby reducing evaluation bias.

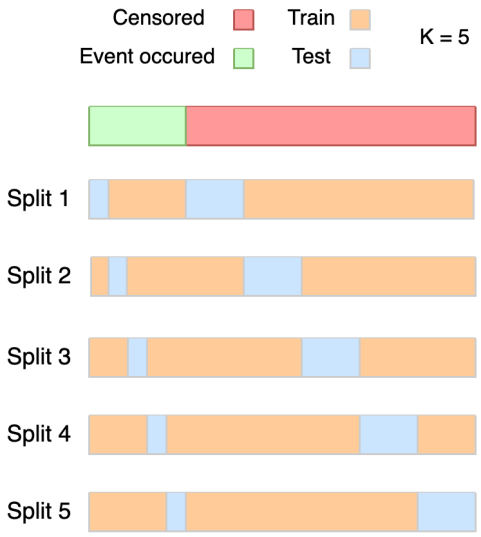


Figure 2.7: This figure demonstrates the Stratified K-fold cross-validation process using $k = 5$ folds. Each split represents how the dataset is divided into training (orange) and testing (blue) segments. This method ensures that the proportion of each class (censored and event occurred) is consistent across all train and test sets, reflecting the overall distribution in the complete dataset.

Repeated Stratified K-fold Cross-Validation

Repeated Stratified K-fold cross-validation, denoted as RepeatedStratifiedKfold, as shown in Figure 2.8, extends this concept by repeating the Stratified K-fold process n times, resulting in multiple sets of k-folds.[76,164] In this thesis, we set k to 5, meaning the data is split into five folds and repeated the process five times, resulting in a total of 25 splits.

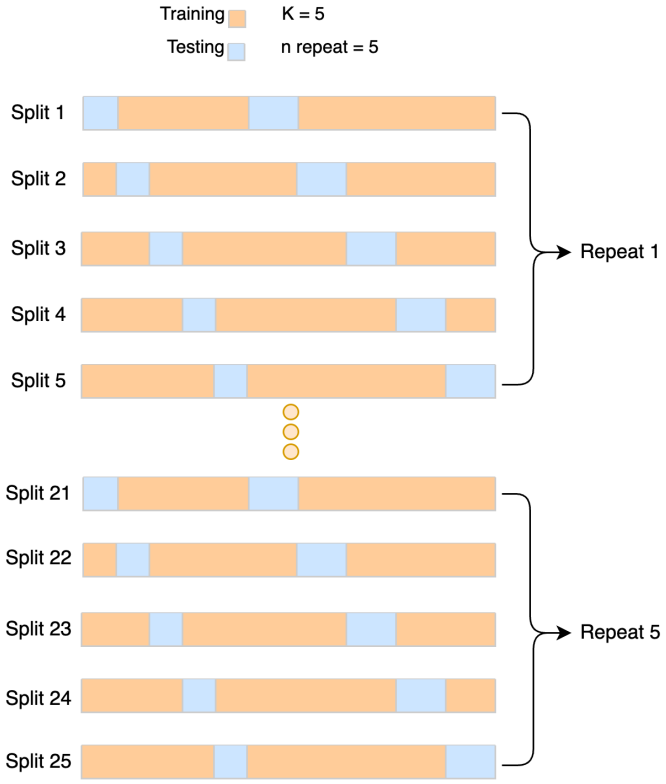


Figure 2.8: Repeated Stratified K-fold Overview. This figure illustrates the Repeated Stratified K-fold cross-validation process with $k = 5$ folds and 5 repetitions, resulting in a total of 25 splits. Each split shows how the dataset is divided into training (orange) and testing (blue) segments, ensuring that every class is proportionally represented in each fold.

2.4.2 Performance Metric: Harrel's C-index

The most frequently used evaluation metric in survival analysis is Harrell´s concordance index, commonly referred to as the C-index.[47,114,155] It evaluates the predictive performance of models by measuring the rank correlation between predicted risks and actual event times and assessing the performance of the time ranking predictions. The C-index assesses whether the model can correctly rank individuals according to their risk of an event. Importantly, the C-index supports right censoring.This aspect is crucial for survival analysis, as it allows for the accurate evaluation of the survival models.[3]

The C-index, closely related to Kendall's τ, calculates the proportion of all evaluable pairs of individauls for which the predicted and observed outcomes are in agreement.[114] If the predictions align with the actual order of events, meaning higher risk scores are associated with earlier events, the pair is considered concordant; if not, it is discordant. Besides these, a third category is tied pairs, which occurs when the predicted risks for a pair of patients are identical.[47,49,114,141] In such cases, the C-index calculation considers tied pairs by contributing a value of 0.5 to the count of concordant pairs instead of 1. In the calculation of concordance, tied pairs are excluded because they do not improve the ranking quality of the samples. The metric ranges from 0 to 1, with 1 indicating perfect concordance, 0 indicating perfect anti-concordance, and 0.5 representing the expected outcome of random predictions.

The C-index extends the concept of the Area Under the Receiver Operating Characteristic (ROC) Curve, also known as AUC, by incorporating adjustments for censoring, thereby quantifying the performance of a binary classifier.[64,121]

To calculate the estimated risk score $\hat{\eta}_i$ by using the estimated model coefficients $\hat{\beta}$ in the Cox model, we use the Eq. (2.25).[64]

$$\hat{\eta}_i = \hat{\beta}_1 x_{i1} + \ldots + \hat{\beta}_m x_{im} \qquad (2.25)$$
$$i = 1, ..., n$$

In Eq. (2.25), the risk score, $\hat{\eta}_i$, is calculated for each sample i, based on a linear combination of the m features.[64]

The C-index (Eq. (2.26)) calculates the proportion of sample pairs, i and i', where the predictions and outcomes are in agreement. This means that the model predicts a higher risk score for the i'-th sample compared to the i-th sample. This distinction is crucial for calculating the C-index, defined as the ratio of concordant pairs, $\hat{\eta}_{i'} > \hat{\eta}_i$ and $t_i > t_{i'}$ to the total possible pairs. This ratio serves as a metric for ranking individual survival times based on their risk assessments.[64,3]

$$C = \frac{\sum_{i,i':t_i>t_{i'}} I(\hat{\eta}_{i'} > \hat{\eta}_i) d_{i'}}{\sum_{i,i':t_i>t_{i'}} d_{i'}} \quad , d_i \in \{0, 1\} \qquad (2.26)$$

From the Eq. (2.26), it is expected that the survival time for the i'-th sample, $t_{i'}$ would be shorter than that for the i-th sample, t_i.[48,49,64,121] Further, d_i represents the event indicator for the i-th sample. It is a binary indicator, where $d_{i'} = 1$ indicates that the event of interest, such as death or failure, has occurred for that individual Conversely, if $d_{i'} = 0$, it indicates that the event has not been observed for the i-th individual within the study period; such cases are referred to as censored observations.

The indicator variable $I(\hat{\eta}_{i'} > \hat{\eta}_i)$ equals one if $\hat{\eta}_{i'} > \hat{\eta}_i$, and equals zero otherwise as shown in Eq. (2.27):[64]

$$I(\hat{\eta}_{i'} > \hat{\eta}_i) = \begin{cases} 1 & \text{if } \hat{\eta}_{i'} > \hat{\eta}_i \\ 0 & \text{otherwise} \end{cases} \qquad (2.27)$$

Limitations

Harrell´s concordance index is affected by the amount of censoring in the data. With high levels of censoring, Harrell´s concordance index may be overly optimistic, potentially leading to an overestimation of model performance.[48,114] When there are numerous censored data points, Uno's C-index is an alternative metric to Harrell's C-index. According to Hartman et al. (2023)[52], the C-index heavily depends on the underlying risk differences of the comparable pairs available in the dataset. Additionally, the C-index is unsuitable for evaluating models that predict specific-duration events because it assesses the order of event times rather than whether the events happen by a scheduled time.[10] Therefore, it may rate a misspecified model higher than it actually is.

2.4.3 Performance Metric: Brier Score

The Brier score is another popular metric used in survival analysis to evaluate the accuracy of probabilistic predictions, particularly in the context of forecasting binary outcomes.[38,120,140] It is calculated as the mean squared difference between predicted probabilities $\hat{p}_i$ and the actual outcomes y_i, where $\hat{p}_i$ is the predicted probabiltity of a class or event for the i-th sample, and y_i is the binary actual outcome, 0 or 1, for the i-the sample. The equation for the Brier score (BS) is given by:

$$BS = \frac{1}{n}\sum_{i=1}^{n}(\hat{p}_i - y_i)^2 \tag{2.28}$$

Predictive models demonstrating greater performance by generating probabilities $\hat{p}_i$ that closely match the true events y_i, will result in a lower Brier Score (BS). In contrast, if a model's probabilities $\hat{p}_i$ are further from the true events y_i, the BS will be higher. The Brier score ranges from 0 to 1, where 0 is the best possible value.[38,47,120]

Notably, Eq. (2.28), illustrates the simplicity of calculating the Brier score when all events of interest are fully observed, that is, in the absence of censoring. In this context, the time-dependent Brier score, $BS(t)$, operates under the same fundamental principles as the original Brier score but is extended to evaluate feature at a specific time point t for all samples $1, ..., n$. The time-dependent Brier score, $BS(t)$, in the absence of right-censoring can be calculated as:[120]

$$BS(t) = \frac{1}{n}\sum_{i=1}^{n}(1_{t_i \geq t} - \hat{S}(t, x_i))^2 \tag{2.29}$$

From Eq. (2.29), $\hat{S}(t, x_i)$ is the estimated survival probability at time t for the i-th observation with covariates x_i. The $1_{t_i \geq t}$ is an indicator function that equals 1 if the events time t_i for the i-th observation is greater than or equal to the time point t, and 0 otherwise, given by:

$$1_{t_i \geq t} = \begin{cases} 1 & \text{if } t_i \geq t \\ 0 & \text{otherwise} \end{cases} \tag{2.30}$$

If the patient is not alive at time t, Eq. 2.29 results in the survival is considered to be zero, resulting in the term $(0 - \hat{S}(t, x_i))^2$. If the patient is alive at time t, Eq. 2.29 turns into the survival is considered to be one, leading to the term $(1 - \hat{S}(t, x_i))^2$.

However, in survival analysis and other contexts such as clinical trials or reliability engineering where right-censoring is present, we have to extend the time-dependent Brier score to account for this. The time-dependent Brier score for censored data at time t, denoted as $BS^C(t)$, is a function that includes the predicted survival function for individual i, $\hat{S}(t, x_i)$, the event times t_i, the censoring indicators δ_i and the estimated probabilities of censoring $\hat{G}(t)$. The equation computes a weighted average of the squared differences between observed survival statuses and predicted survival probabilities, where the weights $1/\hat{G}$ are the inverse probabilities of censoring.[38,120,140]

$$BS^c(t) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{(0-\hat{S}(t,x_i))^2 \cdot 1_{t_i \le t,\delta_i=1}}{\hat{G}(t_i)} + \frac{(1-\hat{S}(t,x_i))^2 \cdot 1_{t_i>t}}{\hat{G}(t)}\right] \tag{2.31}$$

The denominators $\hat{G}(t)$ and $\hat{G}(t_i)$ adjust these squared errors for censoring, where $\hat{G}(t)$ is used to normalize errors for the entire sample up to time t, and $\hat{G}(t_i)$ specifically adjusts errors for individuals censored at their respective times t_i, ensuring the calculations reflect the varied censoring times across the sample. In the left term in the Eq. (2.31), if an individual is censored before time t, $1_{t_i \le t,\delta_i=1}$, their contribution to the Brier score is weighted by the probability of being uncensored until their censoring time t_i. For the right term in Eq. (2.31), if the individual is not censored by time t, $1_{t_i>t}$, the weighting is done with the probability of being uncensored at time t. The second term of Eq. (2.31) considers only uncensored individuals, while the first term accounts for all individuals whose event time t_i exceeds the specific time point t.[38]

The indicator function, $1_{t_i \le t,\delta_i=1}$, described in Eq. (2.32), takes the value 1 if two conditions are met simultaneously, namely that the event time t_i for individual i is less than or equal to a specific time t, and the censoring indicator δ_i equals 1, indicating that the event of interest has occurred. Otherwise, if neither condition is satisfied, the indicator function takes the value 0.[38]

$$1_{t_i \le t,\delta_i=1} = \begin{cases} 1 & \text{if } t_i \le t \text{ and } \delta_i = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.32}$$

The indicator function, $1_{t_i>t}$, described in Eq. (2.33), equals 1 if the event time t_i for an individual i is greater than a specific time t. If the event time t_i is not greater than t, the function takes the value 0. This function is typically used to indicate whether an event has not occurred by time t and is given by:[38]

$$1_{t_i>t} = \begin{cases} 1 & \text{if } t_i > t \\ 0 & \text{otherwise} \end{cases} \tag{2.33}$$

Integrated Brier Score

The integrated Brier score (IBS) is an extension of the Brier score to measure a model's performance over a range of time, providing an overall picture of its predictive performance throughout the entire period observed.[38,47,120,142] t_{max} is the maximum event time, and the integrated time-dependent Brier score, $IBS(t_{\max})$, over the interval $[t_1; t_{max}]$ is defined as:

$$IBS(t_{\max}) = \frac{1}{t_{\max} - t_1}\int_{t_1}^{t_{\max}} BS^C(t)\, dt \tag{2.34}$$

By using the integrated Brier score, we can effectively compare the predictive performance of models across the entire duration for which survival data is available, including adjustments for censoring up to the time of censoring.[38]

2.5 Data Preprocessing

2.5.1 Encoding Variables

In predictive modelling, the encoding of categorical covariates is a crucial step in preprocessing data. It involves converting categorical data into a numerical format that can be handled by machine learning algorithms. The need for encoding arises because humans can interpret text and labels, but most machine learning models require numerical input to perform calculations.[126]

Categorical data typically fall into one of two categories: **nominal** or **ordinal**.[53] Nominal data consists of discrete categories without any inherent order, such as different colours. Ordinal data, however, have a clear, ordered relationship between values, like rankings or levels. This often requires a manual definition of the mapping, as it is not possible to inherently determine the correct order of the labels.

It is traditional to employ one-hot encoding for nominal variables, which generates a set of new dummy variables for each unique value in the nominal variable column.[138] While this method introduces multicollinearity due to the creation of highly correlated features, making matrices computationally challenging to invert and potentially leading to unstable estimates, there are ways to handle such issues.[126] Simply removing one feature column from the one-hot encoded array can significantly reduce the correlation among variables, mitigating the issue of multicollinearity.

To encode ordinal variables, creating an integer representation of the categories is conventional.[168] This is because it is possible to rank the levels of the categorical variable, and appropriate ranking should be obtained from domain knowledge. This strategy assumes that the progression or distance between the category levels is the same. Otherwise, an evenly-spaced integer encoding will be biased.

Target Encoding

When a nominal variable has a large number of categories, one-hot encoding can drastically increase the dataset´s dimensionality. In contrast, target encoding offers a modern solution that efficiently handles these variables without increasing the dataset's variable dimensions.[109] In its simplest implementation, it works by replacing the categorical levels with the mean of a target variable $\mathbf{y}$ of their respective levels. This approach assigns each category a value based on the average outcome of the target variable, thereby embedding the variables with information directly relevant to the model's predictions, which can improve performance.[94,106]

To clarify this approach, we have to differentiate between a numerical and categorical target variable. For example, it is not possible by taking the mean of categories. We discuss three implementations: numerical, categorical binary, and categorical multiclass target variables.

We start with a simple illustration, assuming a train and test set where the target is numerical.[94,109] Let $l \in \{c_1, \ldots, c_k\}$ be an arbitrary level of all k levels of a category for feature $\mathbf{x}_j, j \in \{1, \ldots, m\}$. Then, target encoding simplifies this feature by replacing each category level $\{c_1, \ldots, c_k\}$ with the means of the training target $\{\bar{y}_1^{train}, \ldots, \bar{y}_k^{train}\}$ corresponding to their respective category levels. Each category level is now represented by its mean target value in the new, target-encoded feature, which is used for all instances, such as in the test set. Specifically, the average of the target variable $\bar{x}_l$ for category l from the train data is computed by summing the target values y_i^{train} for all samples where the category is l, followed by division by the number within this category level, N_l.[94,109]

$$\bar{x}_l = \frac{\sum_{i:x_i^{train}=l} y_i^{train}}{N_l} \tag{2.35}$$

We present a simple example for clarity. Referring to Table 2.2, we apply the target encoder in Eq. (2.35) to calculate the encoded values for three categories. Consider the category "cat" within the "Animal" variable as an example: we have five observations of "cat"; thus, $N_{cat} = 5$. The sum of

the target values for "cat" is $\sum_{i:x_i^{train}=cat} y_i^{train} = 1+0+1+0+0 = 2$. Dividing this sum by the count gives us $\frac{2}{5} = 0.4$, matching the "Target encoded Animal" column value for "cat" in Table 2.2.

Table 2.2: Example of target encoding for categorical variables. This table illustrates how categorical animal types (cat, dog, hamster) are converted into numerical values based on the mean of the target variable (1 or 0) for each category.

Animal	**Target**	**Target encoded Animal**
cat	1	0.40
hamster	0	0.50
cat	0	0.40
cat	1	0.40
dog	1	0.67
hamster	1	0.50
cat	0	0.40
dog	1	0.67
cat	0	0.40
dog	0	0.67

In this example (Table 2.2), none of the averages in the groups had the same value, however, if they did, it would indeed be encoded with the same value using target encoding. If, for example, "dog" has an average target value of 0.4, similar to "cat", both categories would receive the same encoded value of 0.4. This means in the encoded space, the two categories, "cat" and "dog", would be equal.

The problem with the presented target encoding in Eq. (2.35) may lead to overfitting, particularly for rare categories and levels, which can cause models to perform well on training data but poorly on unseen test data.[179] Regularized target encoding mitigates this with its smoothing parameter by shrinking the encoded values towards the global mean.[106,109] Additionally, integrating target encoding within a cross-validation framework can help validate the robustness of these encodings across different data folds, improving model performance on test data. This is the standard implementation in software.[94]

Note, for the following equations, we follow the notation of the referenced implementation by Scikit-learn. They use Y to refer to the target variable. Let us first consider the case of a binary target variable. The formula for target encoding is given in Eq. (2.36).[94,106,111]

$$S_i = \lambda_i \frac{n_{iY}}{n_i} + (1-\lambda_i)\frac{n_Y}{n} \tag{2.36}$$

Here, the symbol n_{iY} counts how many times the target variable Y is 1 within category i. n_i is the total number of times category i appears in the data, and n_Y sums up all observations where the target variable is 1 across all categories. n is the total number of observations, and λ_i is the shrinkage factor for category. The shrinkage factor adjusts the influence of category-specific averages, preventing overfitting, particularly in categories with few observations. The shrinkage factor λ_i is defined in Eq. (2.37).[94,106,111]

$$\lambda_i = \frac{n_i}{m + n_i} \tag{2.37}$$

Here, m is the smoothing parameter that adjusts the shrinkage, blending the category mean with the global mean based on the number of observations n_i within category i.

Scikit-learn's TargetEncoder class has a smoothing parameter *smooth* that can be controlled.[112] A high value for the smoothing parameter will put more weight on the global mean, but if one set `smooth = "auto"`, the smoothing parameter will be calculated as a Bayes estimate using Eq. (2.38).[94,139]

$$m = \frac{\sigma_i^2}{\tau^2} \tag{2.38}$$

Here is σ_i^2 the variance of target Y with category i, and τ^2 is the global variance of Y.

In situations involving multiclass targets, the target encoding formula is adapted to Eq. (2.39).[94,139]

$$S_{ij} = \lambda_i \frac{n_{iY_j}}{n_i} + (1 - \lambda_i) \frac{n_{Y_j}}{n} \tag{2.39}$$

This extends the binary classification approach by considering the occurrence of each class j within category i. Here, n_{iY_j} is the count of class j observation within category i, and n_{Y_j} is the total count of class j across the dataset.

The target encoding equation for continuous targets is similar to the method used in binary classification. It is presented in Eq. (2.40).[94]

$$S_i = \lambda_i \frac{\sum_{k \in L_i} Y_k}{n_i} + (1 - \lambda_i) \frac{\sum_{k=1}^{n} Y_k}{n} \tag{2.40}$$

Eq. (2.40), L_i represents the set of observations corresponding to category i, and n_i is the count of observations within the category. Y_k represents the target value for the k-th observation in the dataset, iterating through all relevant data points, either within a subset of data belonging to a particular category i, denoted by L_i, or across the whole dataset for the global average. n signifies the total number of observations across all categories in the dataset.[94]

Conventionally, missing values are treated as their own distinct category, meaning they are neither ignored nor filled with other values such as the mean or median from the dataset. They are grouped and encoded based on the average of the target variable for instances where the feature is missing.[94,106] In our thesis, we did not treat missing values as a separate category as we would not be able to impute these. We applied target encoding only to nominal variables with more than two categories and initially ignored missing values. Later, we used different imputation methods to fill in these missing values.

2.5.2 Outlier Detection

An outlier is an observation that significantly differs from the other data points in the sample where it appears.[44,89] Considering that most of the variables in our datasets discussed in Section 4.1 are categorical, outliers may not influence models much. This is because categories do not have "extreme" values. One could argue, however, that non-sensible combinations of categorical levels could exist for a sample, such as giving a treatment even though they are healthy. This is out of the scope of the thesis as it would require a separate special analysis of the categorical variables.

Principal Component Analysis

There are many popular strategies for detecting outliers; however, we will limit the search for outliers to applying principal component analysis (PCA) with manual inspections of the score plot.[1]

PCA is a widely applied strategy for analysis of high-dimensional data with the purpose of reducing the dimensions of the original dataset. Though it is not designed for the purpose of detecting missing values, it presents desirable properties by reducing the dataset dimensions. It works by projecting the samples onto a new subspace, where the transformation is given by linear combinations of the variables that maximize the explained variance of the data. These linear combinations are called principal components. The first component always explains the most variation, and each subsequent component captures progressively less of the remaining variation under the constraint that it is orthogonal to all previous components.[66,41]

To be more precise, consider the covariate matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$, and let q be the number of principal components computed. The goal is to find the weights $\{\boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(q)}\}$ used to transform

$\mathbf{X}$ into scores $\mathbf{T} \in \mathbb{R}^{n \times q}$, where $q < m$. The transformation is given by the linear combination of the weights and the entries $\mathbf{T} = \mathbf{XW}$ where $\mathbf{W} = [\boldsymbol{w}^{(1)}, \dots \boldsymbol{w}^{(q)}] \in \mathbb{R}^{m \times q}$. The weight vectors are often referred to as loadings, and the projected samples as the scores. Scaling the data so that the contributions to the explained variance are fair between the covariates is generally advised. Feature scaling is discussed in the next subsection.

In terms of outlier detection, applying PCA to the data will provide components and scores that can be easily visualized in comparison to plotting all the variables separately. In particular, by visualizing the scores of the first two principal components, samples far off the origin may be an outlier. The benefit of this strategy is that it considers contributions from all the variables and allows for more understanding through investigations of the loadings.

2.5.3 Feature Scaling: Standard Scaling

Feature scaling is a method for standardizing the range of data features. It is important because it brings all the variables to the same scale, allowing algorithms that rely on the distance between samples, such as K-Nearest Neighbors, to perform more effectively.[126] Additionally, feature scaling benefits other algorithms, such as logistic regression, by providing numerical stability.

Standard scaling is a specific approach to feature scaling and is often more beneficial for many machine learning algorithms, especially for linear models.[126] By standardizing, we adjust the i^{th} sample´s feature value, $x^{(i)}$, so that the transformed features x, across all samples has a mean, μ_x, of zero and a standard deviation, σ_x, of one. This process rescales the data to have properties of a standard normal distribution, thereby simplifying the weight learning process due to its uniformity. The standardization Eq. (2.41) is given as:[126]

$$x_{\text{std}}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x} \tag{2.41}$$

Standardization also preserves outlier information without letting them overly influence the algorithm.[126]

2.5.4 Feature Selection: Permutation Feature Importance

Permutation feature importance is a model inspection technique that evaluates individual features' importance in the model by observing their random shuffling's impact on model performance as shown in Figure 2.9. This shuffling breaks the association between the feature and the target and, therefore, gets an insight into the feature´s prediction power and importance.[11] Building on this foundation, permutation feature importance was first introduced by Breiman (2001)[11] for Random Forests as a means to understand the interaction of features contributing to predictive performance. This method has since been applied to various modelling approaches, including ensemble methods like RSF.

As discussed in Section 2.3.3, RSF utilize feature importance within the survival model framework to select relevant features.[115,126] This selection is based on the measurement of impurity at each child node resulting from a feature´s split thereby the importance of each feature is determined by the reduction in impurity due to a split.

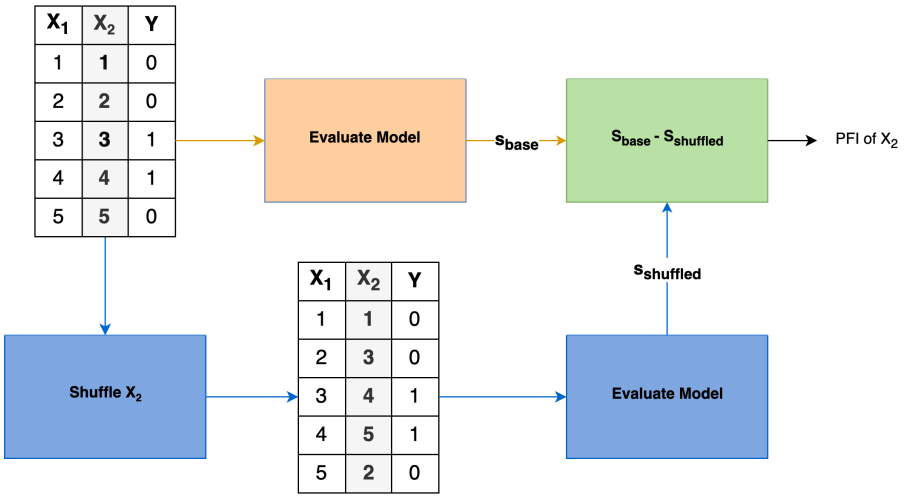


Figure 2.9: Diagram demonstrating the process of permutation feature importance. The original dataset with features X_1 and X_2 is used to evaluate a model to establish a baseline performance metric (S_{base}). The feature X_2 is then shuffled, disrupting its relationship with the target variable Y, and the model is re-evaluated to determine the impact on performance ($S_{base} - S_{shuffled}$). The difference in performance metrics quantifies the importance of X_2 in predicting Y.

Now, let us examine the mathematical basis of permutation feature importance within the context of a supervised learning model.[113,136] As shown in Figure 2.9, the dateset for the baseline is represented as $\mathbf{X}$ with corresponding targets $\mathbf{Y}$. We measure the model´s baseline performance score, denoted by s_{base}, using the model f. During the permutation process, as shown in the blue part of Figure 2.9, the samples of one feature j are shuffled while the samples of all other features remain unchanged. To get a robust estimate of the importance, the process is repeated K times. Thus, for each permutation iteration $k \in \{1, \ldots, K\}$ in the dataset $\mathbf{X}$, we calculate the performance score $s_{k,j}$ of the model f on the newly shuffled dataset where variable j is shuffled. The permutation importance, PFI_j, for the j-th feature is then determined as follows:

$$PFI_j = s_{base} - s_{shuffled} = s_{base} - \frac{1}{K} \sum_{k=1}^{K} s_{k,j} \tag{2.42}$$

The importance of feature j, denoted as Eq. PFI_j in (2.42), is determined as the difference between the baseline performance score s and the average performance scores from the K shuffled datasets $s_{shuffled}$. This calculation reflects how much the model's performance is affected by the random shuffling of each feature's samples. The larger the difference is, the more important the feature j is considered to be.

3 Theory Part B: Missing values

3.1 What Are Missing Values?

By definition, missing values are the values of data that remain unrecorded or absent for a specific variable in the observed subject.[70] Missing data are common in research and, if investigated improperly or neglected, can introduce incorrect statistical conclusions.[39] For example, removing samples with missingness reduces the information available in the data and can make machine-learning models prone to overfitting.

Missing values can occur for multiple reasons. Human error, such as when converting data from analogue to digital, machine error, such as equipment malfunctioning, and survey respondents avoiding questions are some examples. It is often also not possible to determine the source of why data is missing.[57,156] Generally, the treatment of missing values receives relatively little attention compared to the level of engagement that papers demonstrate in creating accurate and reliable statistical or machine learning models.[17,22]

In the subsequent subsections, we will discuss the underlying mechanisms of missing values and alternatives for handling them. In addition, we will explain with simple examples why it is important to be thorough when handling them.

3.2 Missing Value Mechanisms

The underlying mechanisms that source missing values were formally established by Donald Rubin in 1976 as a theoretical framework widely used today.[128] He assigned missing values into three categories: missing completely at random (MCAR), missing at random (MAR) and missing not at random (MNAR). For the following subsections concerning the missing value mechanisms, 'observed variables' refers to the variables in $\mathbf{X}$, whereas the 'unobserved variables' are those not present in $\mathbf{X}$ (not to be confused with missingness).

3.2.1 MCAR

MCAR is the simplest mechanism to deal with and the least likely to introduce bias. As this type of missing value is, given by its name, completely random, the probability of observing missing values of this type does not depend on the observed and unobserved variables. This is desirable because, other than the loss of information, the consequences related to the missing data can be ignored. Under MCAR, one would not expect the distribution of the variable to change as there is no systematic reason for the missingness. Examples of MCAR are data lost due to equipment malfunctions or doctors forgetting to fill in a value in a form.

3.2.2 MAR

MAR, on the other hand, is a broader mechanism than MCAR. This type of missing value is only dependent on the observed variables and is independent of the unobserved ones. It is helpful to think of MAR as a conditional missingness. Compared to the complete randomness in MCAR, MAR, on the other hand, has some systematic relationship between the missing values in a variable and the other variables. To give an example, consider two observed variables *age* and *blood pressure* collected by a doctor. It is likely that blood pressure is missing if the patient is young because there is no need to test for this. Thus, potential missingness in *blood pressure* depends on the observed variable *age*, but not the blood pressure itself. Generally, modern imputation strategies assume MAR.[18]

3.2.3 MNAR

Missing values are MNAR if it is neither MCAR nor MAR. This type of missing value depends only on the missing variable itself. MNAR is the most difficult missing value to account for, and

handling this can be impossible because it depends on unobserved data.[31,86,129] To give an example, suppose a student collects information on exam grades. It is likely that students with good grades report them, but it is less likely for those who fail. Thus, potential missingness in this variable is most likely to be a lower grade, but it can also be a top grade.

3.2.4 Identifying the Mechanisms

MAR and MCAR can be modelled very well and are the standard implementations in software.[170] To describe MNAR, models would have to introduce a latent variable to describe the real missingness, which would complicate algorithms and potentially introduce bias.

It is impossible to distinguish between MAR and MNAR only using the observed data; it is advised to seek expert or domain knowledge concerning the missingness.[152] If this is not possible, the alternative suggestion is to compare subjects with missing values with the subjects with complete data. If the samples with missing and complete data show variations, it is conceivable that they might also vary in the unobserved variables. This can strengthen the plausibility of the missingness being MAR, but considering it employs univariate comparisons, it does not provide a proof.[7]

Interestingly, in a literature review of missing values between 2006 and 2017, Lin and Tsai (2019) discovered that many articles and papers overlook the importance of all three missing value mechanisms.[85] To be specific, the majority of the papers assumed MCAR, and only 15 out of 111 considered all three.

The key takeaway is that the underlying mechanics of MCAR and MAR can be ignored because they, in a sense, remove the necessity of taking into account the distribution of the missing values when imputing.[99] More care would be required for MNAR, and thus, it is important to identify the mechanisms in data containing missingness before dealing with them.

3.3 Missing Value Patterns, Influx and Outflux

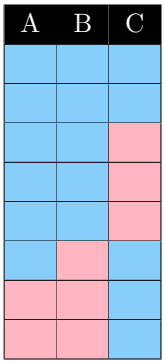
An important consideration for missing values is the pattern of missing values, as by using the pattern information, it is possible to remove predictors based on proportion statistics called influx and outflux, as defined by Buuren (2018).[18] All information presented in this subsection is under chapter *4.1 Missing data pattern*.

Let $\mathbf{x}_j$ and $\mathbf{x}_k$ be two column pairs of the data $\mathbf{X}$. They have the following patterns:

1. rr if both $\mathbf{x}_j$ and $\mathbf{x}_k$ are observed.
2. rm if $\mathbf{x}_j$ is observed and $\mathbf{x}_k$ is missing.
3. mr if $\mathbf{x}_j$ is missing and $\mathbf{x}_k$ is observed.
4. mm if both $\mathbf{x}_j$ and $\mathbf{x}_k$ are missing.

Thus, one can produce a symmetric matrix for each of the patterns for all columns in the data matrix $\mathbf{X}$. For illustration, we use the same example as in Buuren (2018).[18] Let A, B and C be the columns of the data given in Table 3.1.

Table 3.1: Data with columns A, B and C. Blue are observed and red are missing.



The four defined patterns of Table 3.1 are described in Table 3.2. What is shown is the count of each pattern for the variable pairs. For rr and mm, the main diagonals are the counts of observed and missing samples in a single variable, respectively. The main diagonals in rm and mr will always be 0 because a sample cannot be observed and missing at the same time.

Table 3.2: The four missing value patterns for variables A, B and C from Table 3.1.

rr

	A	B	C
A	6	5	3
B	5	5	2
C	3	2	5

rm

	A	B	C
A	0	1	3
B	0	0	3
C	2	3	0

mr

	A	B	C
A	0	0	2
B	1	0	3
C	3	3	0

mm

	A	B	C
A	2	2	0
B	2	3	0
C	0	0	3

For example, the pair (A,A) has 6 observed samples (rr). Only five observations exist where both A and B are observed (rr). There is one observation where A is observed, but B is missing (rm). Note that rm and mr are the transpose of each other.

We will not be showing this type of table for our data because there are too many columns to present. Instead, we introduce influx and outflux, which are two simpler statistics to report. Influx and outflux extend the information from columns $\mathbf{x}_j$ and $\mathbf{x}_k$ to all columns and thus return scalar statistics. Let r be a random variable denoting whether the sample is observed (1 = observed, 0 = missing). Then, for column index $j \in \{1, \ldots, m\}$ with the set of samples $\{1, \ldots, n\}$, the influx is defined as

$$I_j = \frac{\sum_{i}^{n} \sum_{k}^{m} (1 - r_{ij}) r_{ik}}{\sum_{i}^{n} \sum_{k}^{m} r_{ik}} \tag{3.1}$$

Influx depends on the proportion of observed values in the variable $\mathbf{x}_j$. The numerator can be read as the total number of variable pairs $(\boldsymbol{x}_j, \boldsymbol{x}_k)$ where $\boldsymbol{x}_j$ is missing and $\boldsymbol{x}_k$ is observed, whereas the denominator is the total number of observed samples. For a completely observed variable, the influx is 0, and for a completely missing variable, it is 1. If two variables have the same proportion of missingness, the one with a higher influx might be easier to impute.

Outflux is defined as

$$O_j = \frac{\sum_i^n \sum_k^m r_{ij}(1 - r_{ik})}{\sum_i^n \sum_k^m 1 - r_{ik}} \tag{3.2}$$

The outflux is a measurement of the usefulness for imputing other variables using $\mathbf{x}_j$. The numerator is the number of variable pairs where $\boldsymbol{x}_j$ is observed and $\boldsymbol{x}_k$ is missing, whereas the denominator is the total number of missing samples. For a completely observed variable, the outflux is 1, and for a completely missing variable, it is 0. [(1)]

The point to raise here is that both a high influx and outflux are desired. The former might be counterintuitive. A variable with a high influx might be easier to impute because there are many available columns that can be used to impute this variable. However, it also signifies that the proportion of missingness in this variable is high. A high outflux is desired because it means that the variable has a high potential to impute the other variables. Note that it is only a potential because the variable could be completely useless for imputation even if the outflux is high. Consider the case of imputing $\boldsymbol{x}_j$ using information from the variables $\mathbf{X}_{\neq j}$. Suppose some of the covariates in $\mathbf{X}_{\neq j}$ are binary and imbalanced. If the imputation step happens inside of a k-fold cross-validation, the imbalanced variables will likely end up constant because the rare cases end up in different folds. Thus, they have no imputational power and should be omitted. Note, however, that reliable features capable of yielding good imputations are limited if the outflux is low.

Influx and outflux for Table 3.1 is shown in Figure 3.1. C is better connected to the data, which is simple to see in the Table 3.1. This is because both A and B have missingness in the same samples (2 in particular), and C has no missingness related to A and B. Thus, more information is available in C.

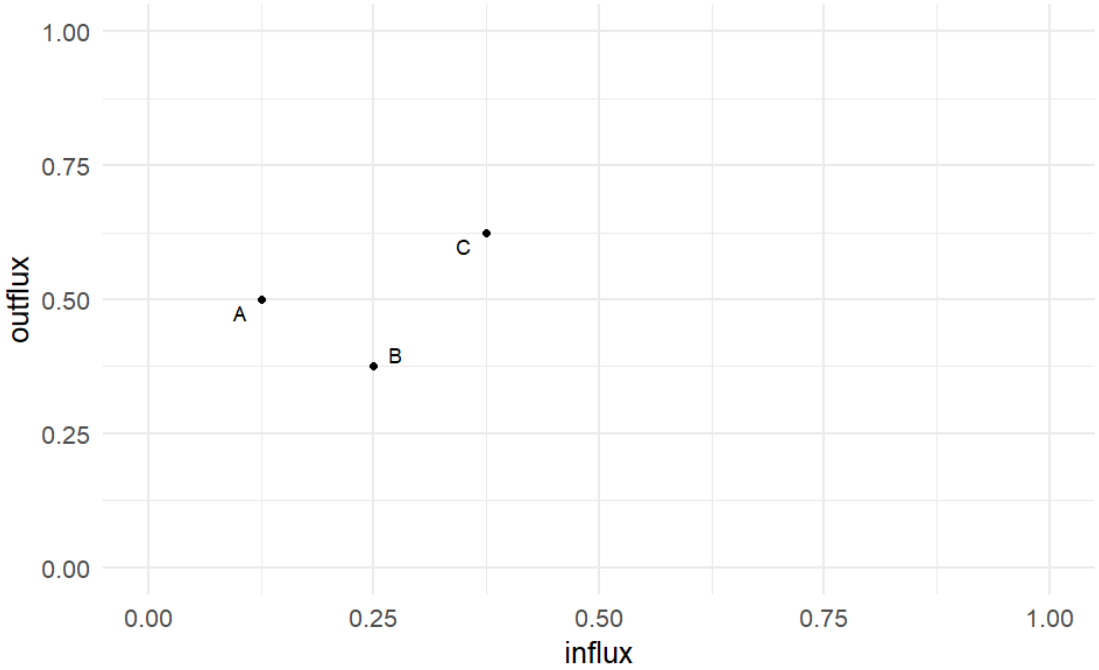


Figure 3.1: Influx and outflux for variables A, B and C from table 3.1 calculated using eqs. (3.1) and (3.2). The influx relates to how well a variable's missingness is related to the observed data in the other variables. The outflux relates to how well the observed samples in a variable are connected to the missing data in the other variables.

[(1)] The formulas for influx and outflux by Buuren (2018)[18] were slightly adjusted to better fit to our investigated scenario. This was discussed with the supervisors.

3.4 The Basics of Handling Missing Values

We extend the notations of $\mathbf{X} \in \mathbb{R}^{n \times m}$ from Section 1.4 to include missing values which are not in the set of real numbers. If we have an incomplete covariate matrix, we write $\mathbf{X} \in (\mathbb{R} \cup \{NA\})^{n \times m}$. Suppose that we pick out some column in $\mathbf{X}$ denoted as $\boldsymbol{x}_j$ that contains missing values; then we consider the complete and missing parts as $\boldsymbol{x}_j^{(c)}$ and $\boldsymbol{x}_j^{(m)}$, respectively. Thus, if we combine the two, assuming the indices are preserved, then $\boldsymbol{x}_j = \boldsymbol{x}_j^{(m)} \| \boldsymbol{x}_c^{(c)}$. When missingness in incomplete variables are replaced with real values, they are imputed. This involves drawing an estimate often inferred from a variable, or multiple variables.[29] We denote imputed variables with a hat notation. For example

$$\mathcal{I} : (\mathbb{R} \cup \{NA\})^{n \times m} \to \mathbb{R}^{n \times m}$$
$$\mathcal{I}(\mathbf{X}) = \begin{cases} \hat{\mathbf{X}} & \text{if } \mathbf{X} = NA \\ \mathbf{X} & \text{if } \mathbf{X} \neq NA \end{cases}$$

would be an imputation function that imputes the missing variables. A simple example is replacing the missing values with the mean of the corresponding variable, calculated from the complete variables.

3.4.1 Two Strategies for Handling Missing Values

Consider the covariate matrix $\mathbf{X} \in (\mathbb{R} \cup \{NA\})^{n \times m}$ that contain missing values. There are two ways of handling missing data. The first is deleting samples (rows) or variables (columns), and the second is imputing.

The former method is generally the simplest and is frequently utilised by researchers.[4] This approach is austere because it requires no deduction on the missing values and works as a good baseline strategy for comparing against more sophisticated approaches. The latter approach of imputing involves replacing the missingness with values generally inferred from the data. We say generally, because the imputation function does not necessarily require to be parameterized by the data. An example of such a function is replacing the missing values with some fixed constant, such as 0, or a factor, like 'unknown', for continuous and discrete data, respectively. However, using the complete parts of the data to infer the missing values is more common.[29] Missing data is rarely acknowledged in published research, and the progression has been very slow to be adopted by non-specialist researchers.[59] However, poor handling of missing data can lead to biased or inefficient parameter estimates.[170] Various papers and literature reviews demonstrate the importance of imputation, and propose that imputing variables with missing values can reduce bias and increase precision.[7,73] This naturally requires that the data is thoroughly investigated and methods are carefully applied. Theory concerning the removal of missingness and imputations are discussed in detail in Section 3.5 and 3.6, respectively.

3.5 Removing Rows, Columns or Both?

In this subsection, we will consider three options for handling missing data that do not involve imputations. These are either deleting the missing samples (rows), variables (columns) or both.

One traditional way to handle missing values is complete case analysis (CCA). This involves removing the samples that contain missing values, such that what is left is only the complete cases.[42,167] The benefit of this strategy is that it is very simple, and if the complete samples are representative of the population, very little information is lost by the removal. Newman (2014) proposes that CCA is unbiased under the MCAR mechanism.[104] This is because there is no systematic reason for the missingness. However, we suggest that this should be interpreted with vigilance. If deleting entries results in losing levels in categorical variables, those omitted

will never have any contribution to machine learning models. Multiple studies and reviews have highlighted the disadvantages of dropping samples.[86,132,133] For instance, Ambler (2007) found in their simulation study that CCA leads to a substantial bias and poor predictions in a risk model with binary outcome.[5] Furthermore, with incomplete variables, there is a risk of losing precision and power when estimating statistical parameters. To be specific, the loss of sample size contributes to more uncertainty and, in some cases, can give misleading or biased parameter estimates.[103] In the specific case of regression analysis, the model is prone to overfitting if there are many independent variables present when estimating statistical parameters using standard techniques (i.e. OLS) after removing samples with missing values.[50]

To illustrate this, consider a simple example where there are two independent variables $\mathbf{x}_1$ and $\mathbf{x}_2$ that model some target variable $\mathbf{y}$ using a linear regression model. These variables have no missing values. Furthermore, say, arbitrarily, that both $\mathbf{x}_1$ and $\mathbf{x}_2$ have significant regression coefficients β_1 and β_2, respectively. Next, introduce a third variable $\mathbf{x}_3$ that contains many missing values. If regression of $\mathbf{y}$ is done using all three variables as well as the CCA strategy, it may be very likely that either $\mathbf{x}_1$ or $\mathbf{x}_2$ have an insignificant coefficient. The lack of effect is not tied to the predictive power of $\mathbf{x}_3$, but instead to the sample size reduction. See Table 3.3 for a simple illustration.

Table 3.3: Example highlighting the problem of applying CCA before regression analysis. All values are arbitrary. *coef* is the coefficients and p is the p-value. The asterisk represents the significant codes. For case 1, the complete data is used for fitting because $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ have no missing values. Case 2, $\boldsymbol{x}_3$ has 960 missing entries, thus CCA leads to fitting with only 40 samples.

(a) Case 1

	$y \sim \boldsymbol{x}_1 + \boldsymbol{x}_2$ n=1000	
	coef	p
$\boldsymbol{x}_1$	5	< 0.001 (***)
$\boldsymbol{x}_2$	10	< 0.001 (***)

(b) Case 2

	$y \sim \boldsymbol{x}_1 + \boldsymbol{x}_2 + \boldsymbol{x}_3$ n=40	
	coef	p
$\boldsymbol{x}_1$	0.01	0.3
$\boldsymbol{x}_2$	0.9	0.01 (*)
$\boldsymbol{x}_3$	1	0.05 (*)

Next, we consider the option of removing the columns with missing values rather than the samples. This becomes a viable option with high dimensional data because when the feature dimension increases, it becomes infeasible to remove samples.[99] Consider our defined matrix $\mathbf{X}$ that has the dimensions $n \times m$ and let p denote the probability of an entry missing independently of the others (MCAR). Using the example from Zhu et al. (2019), when $p = 0.01$ and $m = 5$, applying the CCA strategy would result in keeping around 95% of the entries. However, if we use the same probability but increase the feature dimensions such that $m = 300$, we expect to keep only 5% of the original samples.[183]

If the variable(s) removed had very little effect on the target, it may not be a substantial loss by removal. Generally, unless inference on all variables is important to consider, removing variables is a better choice than samples.[171] This is especially the case if removing samples would make the complete cases non-representative of their original population.[171,152] The apparent disadvantage of dropping a variable is that it will not be available for statistical models to infer information from.

It is also important to look at the missingness mechanisms before omitting any variables. If the variable with missing data is MCAR or MAR, preserving the column can be reasonable. However, for MNAR, it is different, as the probability of observing this variable depends on the unobserved parts of the variable itself, it is reasonable to remove it. Some software, such as *mice* discussed in Section 3.6.3, assume MAR so preserving these variables are beneficial.[170]

Moreover, considering all the points made above, we propose that applying both strategies is the most sensible approach. This proposition can be deduced as follows: consider the covariate matrix $\mathbf{X}$ with the dimensions $n \times m$. Let $j \in \{1, \ldots, m\}$ be a column index of a variable where

all samples except for one are missing, and $i \in \{1, \ldots, n\}$ be a sample index with many missing values. In this instance, removing the column with index j would be required, otherwise CCA would result in a single row vector of dimension $1 \times m$. Moreover, assuming that the remaining samples $\{1, \ldots, n\} \setminus i$ are representative, removing sample i would make sense. Therefore, in short, if removing the missing values is desired, it would require removing variables that are not of interest or contain too many missing values, and samples that do not affect the remaining ones. For large datasets, this is generally not an issue, but care is required for smaller ones.

In conclusion, we have highlighted one of the simple approaches of handling missing values, namely by removing them. With simple examples, we show that it is important to carefully consider the proportions of missing values and also their missingness mechanism. This approach should be considered as a baseline.

3.6 Imputation Methods

This section considers some common imputation strategies, It is divided into three parts, with increasing complexity. The first and second parts concern global imputations and multivariate imputations, respectively, where only one variable has missing values. The third part concerns multiple imputations, where missing values occur in multiple variables.

3.6.1 Global Imputation

Consider $\mathbf{X} \in (\mathbb{R} \cup \{NA\})^{n \times m}$ where only one variable $j \in \{1, \ldots, m\}$ is incomplete. If the sample size is small, it is a motivation to impute the missing values with some sensible real numbers. As illustrated in the previous subsections and with support from literature, blindly removing samples or variables can lead to bias in conclusions or loss of information, respectively. Thus, it is important to be familiar with imputation techniques. This subsection considers some simple global imputers for handling missing values. Global imputations mean that all the missing values in a variable are imputed by the same value. We consider imputations of discrete and numerical variables separately.

Categorical Imputation

A global imputer for categorical data is mode imputation.[28] It might be tempting to replace missingness with an indicator value such as 'unknown'. However, we will show that this is not necessarily a good strategy.

The term 'mode' in statistics represents the most common value from a set of values.[45] Thus, mode imputation involves replacing all missing values in a variable with the most common occurrence of the categories in that variable. If the most common value is not unique, it is possible to randomly sample from the ties. This is the easiest strategy to handle missing values of discrete data; it is computationally efficient and easy to implement. However, as the proportion of missing values increases, this strategy becomes proportionally more unsuitable. Firstly, as the missingness increases, the variable will approach a constant category. This changes the distribution of the variable, and artificially reduces its variance. A drawback to this is loss of predictive performance, but also potentially biased conclusions. A. B. Soom et al. (2022)[149] studied the effects of mode imputation on data with simulated MCAR missingness, and showed that, for their particular dataset, mode imputation yielded a higher discrepancy between the original and imputed data. In contrast, predictive mean matching, a strategy we will discuss later, was much more accurate.

Next, we highlight why it is generally not good practice to set missingness to a new category. The apparent problem with this is that it generalizes every missing entry to the same category, but in reality, they can be missing for various reasons. The variable could also be biased in the sense that the 'unknown' indicator (may have) contradicting effects with the other variables.[42,67] Consider, for example, two variables *sick* and *tumour location*. If there are two patients, one sick and one healthy, but both are missing the tumour location, it makes little sense to assume that their location is the same using the indicator replacement.

The mentioned global imputers do not require any form of encodings or numerical representation of the category levels. This is an important consideration for imputation strategies that require numerical representations in intermediate steps, such as k-Nearest Neighbor imputations when calculating distances and computing averages, discussed in Section 3.6.2. We address this in Section 3.7 after discussing multivariate imputers.

Numerical Imputation

Next, we consider global imputations of missing numerical values. Popular techniques are mean and median imputation.[29,63]

The mean, in our particular case, arithmetic mean, is the sum of a collection of numbers divided by the count of numbers in that collection.[62] It signifies the central point of a set of numbers. Thus, mean value imputation is the strategy of replacing the missing values in a variable with the average of the complete values in that variable. This is the most popular approach to handling continuous variables with missing values. However, it has its downsides. The mean is not a robust statistic, which directly implies that if the distribution of the observed (and unobserved) variables is not Gaussian, it will either over- or underestimate the true value. In addition, mean value imputation artificially removes variance in the data. As the number of missing values approaches zero the standard deviation will approach zero.[7] There is also a minor problem where mean imputation does not guarantee integers, but if it is required, rounding the imputation is a simple workaround.

The median is the value separating a set of numbers' lower and higher half. This is a robust statistic of the central tendency because it does not matter if the distribution is Gaussian or not. Median imputation involves replacing the missingness in a variable with the median of the complete values of said variable. It is suitable if the variable is skewed or has extreme values present that do not make sense to remove.[146]

We propose that imputing missing values with zero can be reasonable under certain circumstances. As this does not use any information from the observed samples, it makes the most sense if the real value is known to be 0 or have no contribution, but is missing for some reason. Consider, for example, two variables *smoking* and *smoking amount*. If the sample is not smoking, then the amount smoked could be missing, and zero would be the appropriate value to assign. However, if zero is not within the typical range of the distribution or does not fit the nature of the data, this will not be suitable. For example, for variables *age* or *salary*, it would be a poor choice to impute with zero.

Random Imputation

All global imputers have the inherent problem of reducing the variance of the imputed variable. Thus, variables with large proportions of missing data may become less distinctive after imputing. A proposed solution to this is random imputations.[69]

Random imputations involve sampling the imputations from a uniform distribution. For numerical variables, this could be sampling from a uniform distribution that respect the domain of the variable. For example, imputing a variable *age* by randomly sampling values between 18-90. For categorical variables, it could be randomly sampling from all possible values the variable take. This strategy relaxes the global imputations by preventing the reduction in variance.

3.6.2 Multivariate Imputations

For this section, we assume that missingness in $\mathbf{X}$ is only present in one variable. The apparent downside to global imputation is that they do not utilize the information in the other variables. This is not a problem for MCAR if the variable's distribution does not change, but for MAR, one has to be more cautious. We start by presenting k-Nearest Neighbors as a standalone imputer, and after that, the four imputation functions used in multiple imputations (discussed in Section 3.6.3).

K-Nearest Neighbor Imputation

The k-Nearest Neighbor imputer (kNN) is the most widely applied multivariate imputer.[85] It was shown in a study comparing imputation methods for categorical data that the kNN imputer outperformed all other imputation strategies that it was compared against.[93] This makes it an interesting choice for our datasets, as the majority of the variables are categorical. The algorithm is a neighbour-based approach that fetches the k most similar neighbours to the missing entry. Each value of the variable from the k-nearest neighbors is averaged and used as imputation. By default, the distance metric used in our software (discussed in Section 4.2) is given by

$$dist(\boldsymbol{x}, \boldsymbol{y}) = \text{sqrt}(w\text{*sq. distance from present coordinates}) \tag{3.3}$$

and

$$w = \frac{\#features}{\#\text{non missing features}}$$

is a weighting factor.[148] To illustrate with a simple example, suppose $\boldsymbol{s}_1 = (3, NaN, NaN, 6)$ and $\boldsymbol{s}_2 = (1, NaN, 4, 5)$ are two samples of the variables $x_1, .., x_4$. Then only the first and fourth variables with values 3 & 1 and 6 & 5 contribute to the distance, and the weighting w would be $\frac{4}{2}$ because two variables are omitted. This would give the distance $dist(\boldsymbol{s}_1, \boldsymbol{s}_2) = \sqrt{2((3-1)^2 + (6-5)^2)}$.

To explain the kNN algorithm more precisely, we use the notation of tuples. The notation is inspired by Zhang et al. (2019).[180] Let $r = \{t_1, \ldots, t_n\}$ be all the tuples of the data and $\mathcal{R} = \{x_1, .., x_m\}$ the schema for the variables. Then $t_i[x_j]$ is the value of the tuple $t_i \in r$ on variable $x_j \in \mathcal{R}$. In addition, we say that $t_{miss} \in \mathcal{R}$ is a tuple over $\mathcal{R}$ with a missing value on variable x_{miss}. Then $\mathcal{F} = \mathcal{R} \backslash \{x_{miss}\}$ is the schema of complete variables, under the assumption that missingness is only present in the one variable. The algorithm for imputing a single tuple can be summarized in algorithm 1.

Algorithm 1 k-Nearest Neighbor Imputation

Let $NN(t_{miss}, \mathcal{F}, k)$ denote the k nearest neighbor tuples of the tuple t_{miss} on attributes $\mathcal{F}$ from r. These are the tuples with the smallest distance to t_{miss}. For a tuple $t_i \in r$, the distance to the tuple with missing value t_{miss} is given by

$$d(t_{miss}, t_i) = \sqrt{\sum_{x \in \mathcal{F}} (t_{miss}[x] - t_i[x])^2}$$

The imputation is then summarised in two steps:

1: $T \leftarrow NN(t_{miss}, \mathcal{F}, k)$
2: $\hat{t}_{miss}[x_{miss}] \leftarrow \frac{1}{k} \sum_{t_j \in T} t_j[x_{miss}]$

Step 1 of the algorithm involves finding the k set of tuples that lie closest to the tuple with missingness t_{miss} denoted as T, and step 2 is the imputation step. Here the imputation is the arithmetic mean of the missing feature x_{miss} in the observed tuples T.[163,6] The imputation is not limited to the average as it can be any statistic such as median or weighted mean. One could extend it further and apply linear interpolation to the local neighbourhood. As kNN relies on computing distances, it is important to scale the data before imputing.

We tacitly assume that only one variable contains missingness. When there are missing values in more than one variable, the appropriate distance to use is Eq. (3.3). In this case, the distances only have contributions where both values in two tuples are observed. Thus, all samples will contribute to the distance calculation, but not necessarily all variables. So, if many of the samples being compared have many missing values, the distance metric may not be very representative. In other words, the variable being imputed should have a high outflux to get neighbors that are

representative and not primarily based on the weighting w.

Let us consider a simple example of three variables $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$. Here, we want to illustrate the imputation of one of the values in $\mathbf{x}_3$ where we have NA present in two of the three columns in the data. See Table 3.4.

Tuple ID	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$
t_1	1	0.0	1.1
t_2	1	1.0	1.2
t_3	1	1.5	NA
t_4	1	1.0	1.5
t_5	0	NA	NA
t_6	1	2.0	0.6
t_7	0	0.0	0.0
t_8	0	1.0	1.0

Table 3.4: Dummy data for illustration of kNN imputation.

The distances using Eq. (3.3) are listed in Table 3.5. Most of them are straightforward because only the distances in $\mathbf{x}_1$ and $\mathbf{x}_2$ are calculated, and then weighted by $\frac{3}{2}$. The minor deviation is the distance between t_3 and t_5, because only one complete feature is observed for t_5, thus the weight and distance calculation changes. So, for our example, the four nearest neighbours to t_3 are $T = \{t_2, t_4, t_6, t_8\}$. The imputation is then the arithmetic mean of $\mathbf{x}_3$ of the neighbors T, i.e. $\hat{t}_3[x_3] = \frac{1.2+1.5+0.6+1}{4} = 1.075$

Table 3.5: Euclidean distances (dist) from tuple t_3 to the tuples listed in table 3.4. Distances are calculated using Eq. (3.3).

Tuple ID	w	**Formula**	**Dist**
t_1	$\frac{3}{2}$	$\sqrt{\frac{3}{2}(1-1)^2 + (1.5-0)^2}$	1.84
t_2	$\frac{3}{2}$	$\sqrt{\frac{3}{2}(1-1)^2 + (1.5-1)^2}$	0.61
t_3	$\frac{3}{2}$	$\sqrt{\frac{3}{2}(1-1)^2 + (1.5-1.5)^2}$	0.0
t_4	$\frac{3}{2}$	$\sqrt{\frac{3}{2}(1-1)^2 + (1.5-1)^2}$	0.61
t_5	$\frac{3}{1}$	$\sqrt{\frac{3}{1}(1-0)^2}$	1.73
t_6	$\frac{3}{2}$	$\sqrt{\frac{3}{2}(1-1)^2 + (1.5-2)^2}$	0.61
t_7	$\frac{3}{2}$	$\sqrt{\frac{3}{2}(1-0)^2 + (1.5-0)^2}$	2.21
t_8	$\frac{3}{2}$	$\sqrt{\frac{3}{2}(1-0)^2 + (1.5-1)^2}$	1.37

This example shows that for the distance between t_5 and t_3, only $\mathbf{x}_1$ contributes to the distance, and it is therefore heavily weighted. This relates back to the idea that it is preferable to have high outflux variables to get a more accurate contribution to the distances.

Predictive-Mean Matching Imputaton

Predictive-mean matching (PMM) is an efficient strategy that works well on all data.[170] It is attractive because it assures that the imputations respect the domain of observed variables. For example, samples that are required to be integers or strictly positive, or the fact that counts cannot be negative. Given any predictive model, often a linear main effects model, a pool of the observed variables that have predictions close to the missing sample is created. The imputations are drawn randomly from the donor pool of observed samples. Thus, imputations will not generate new values but instead be sampled from the observed ones.

Logistic Regression Imputaton

Logistic regression is a traditional function used to model a binary outcome variable.[82] Given the linear model

$$\mathbf{y} = \beta_0 + \beta_1 \boldsymbol{x}_1 + \ldots + \beta_m \boldsymbol{x}_m + \boldsymbol{\epsilon} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where $\mathbf{X} \in \mathbb{R}^{n \times m+1}$ and the sigmoid

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$
$$\mathbb{R} \to (0, 1)$$

The logistic regression model is given by

$$p(\boldsymbol{y} = 1|\mathbf{X}) = \frac{1}{1 + \exp(-\mathbf{X}\boldsymbol{\beta})}$$

The interpretation of the logistic model is that, given some data $\boldsymbol{x}_i$ (column vector), and a binary outcome y_i, $i \in \{1, \ldots, n\}$ with values $\{1, 0\}$, the logistic function gives the probability $p(y_i = 1|x_{i,1}, \ldots, x_{i,m})$ of a sample's event being 1 given the information in the variables.[23] For predictions, a threshold is set for assigning class membership. For example, probabilities above 0.5 are assigned to class 1, and 0 otherwise.

Multinomial Logistic Regression Imputaton

The multinomial logit model, also known as polytomous regression, is an extension of logistic regression that considers more than two outcomes in the target variable.[78] Let k denote the number of unique categories in the target variable $\mathbf{y}$, and let the first category be the reference. The extension from binary logistic regression is that instead of treating the outcome as a probability of belonging to class 1, the multinomial logistic model compares the probability of membership in each category level $j \in \{2, \ldots k\}$ to the probability of membership in the reference category.[32,172] This results in $k - 1$ binary logistic regression models being fitted.

The multinomial logistic model is given by

$$p(\mathbf{y} = j|\mathbf{X}) = \begin{cases} \frac{\exp(\mathbf{X}\boldsymbol{\beta}_j)}{1 + \sum_{i=2}^{k} \exp(\mathbf{X}\boldsymbol{\beta}_i)} & j = 2, 3 \ldots, k \\ \frac{1}{1 + \sum_{i=2}^{k} \exp(\mathbf{X}\boldsymbol{\beta}_j)} & j = 1 \end{cases}$$

$\boldsymbol{\beta}_j$ is not to be confused with a scalar coefficient. It is the vector of coefficients estimated for class j when it is compared against the reference. For predictions, the class membership is assigned to the class with the highest probability.

Ordered Logit Imputaton

The ordered logit model is necessary when the order of the category matters. For example, 'poor', 'fair', 'good'. The idea behind the ordered logit model is looking at the cumulative probability of the outcome. For a target $\mathbf{y}$ that contain an order of categories $c_1, c_2, \ldots, c_k$, such that $c_1 < c_2, \ldots, < c_k$, the model for one sample looks at $p(y_i \leq c_j|\mathbf{X})$. For example, with the categories 'poor', 'fair' and 'good', looking at 'fair' implies the probability $p(y_i =$ 'fair' or 'good'$|\mathbf{X})$. Naturally, the sum over these cumulative probabilities is 1.[43]. The formula for the cumulative probability is given by

$$p(y_i \leq c_j|\boldsymbol{x}_i) = \frac{1}{1 + \exp(-a_j + \boldsymbol{\beta}\boldsymbol{x}_i)}$$

Here, a_j is defined as the threshold parameter, and is a constant that determines if the sample is

more likely to belong to the current rather than previous category. The thresholds are additional parameters estimated. The reference category, as mentioned in the multinomial logistic model, gets assigned this value to 0. For predictions, there is a probability of falling within the range of each threshold. To assign a class membership, it is assigned to the class with the highest probability.

3.6.3 Multiple Imputations

There are two reasons for seeking more sophisticated imputation approaches: either removing the samples or variables is not viable due to low sample size, or one may want to see if more complicated strategies can outperform the already mentioned approaches in such a way that it is reasonable to choose the new, more intricate approach. It is also a suitable approach if there are missing values in more than one variable.

As previously mentioned in Section 3.6.1, global imputations are unrealistic in the sense that there is little to no variability in the imputations. We distinguish between two approaches: single imputation vs multiple imputation. All methods discussed so far are single imputations because a missing value gets assigned a single point estimate when imputing. There are, however, disadvantages to this approach. Tests and confidence intervals can often be distorted by wrongful precision.[87] Intuitively, it makes sense because it is difficult to infer certainty from point estimates with no information about the variations. Therefore, an alternative approach is applying methods of multiple imputation.[18] This means that each missing value gets assigned multiple unique imputations instead of a single point estimate. This results in distinct datasets containing the imputed values, and analysis can be applied to each of these. Pooling the multiple analysis results allows for understanding and assessing the uncertainty of the missing values.[7] More specifically, because multiple datasets are generated, it is possible to look at the distribution of the imputations of a variable. If the variability of the imputations is low, it can be an indication that the imputation is fitting. In addition, if some model is fit on the imputed datasets and used for prediction, it is possible to get an empirical distribution over the predictions.

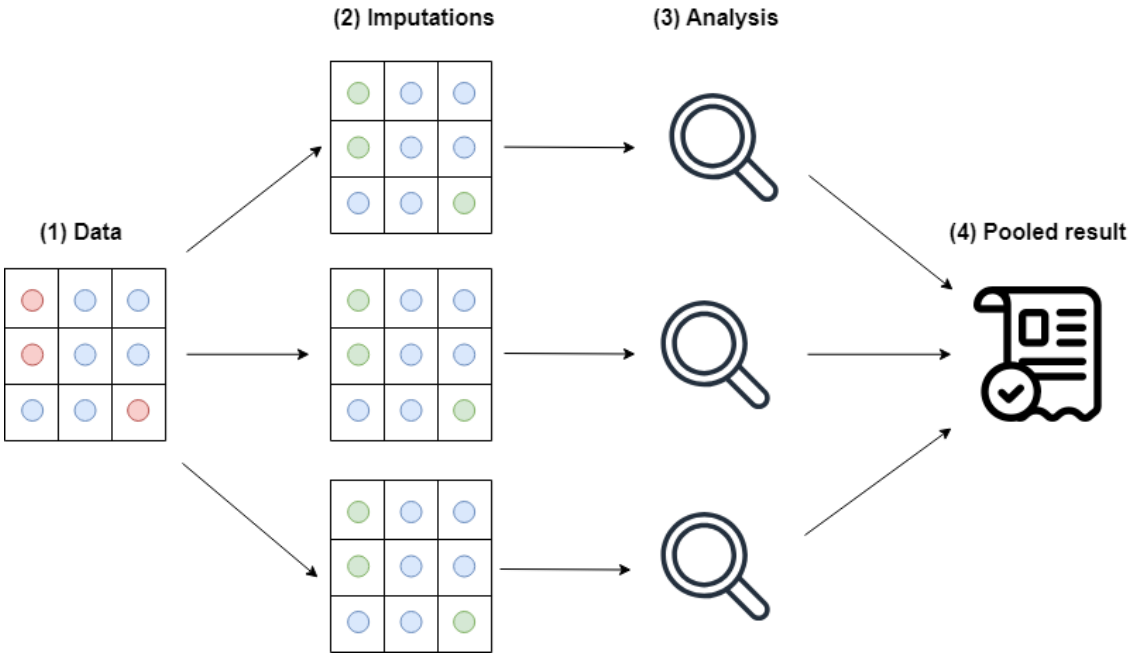


Figure 3.2: Illustration of the multiple imputations strategy. (1) Data with missing values (red) and complete values (blue) are parsed to the imputer (2). Analysis (3) is done on each of the datasets generated from (2). The multiple results from the analysis step can be pooled (4) to a single point estimate using Rubin's rules.

The general approach for multiple imputations are shown in Figure 3.2. For the thesis, we let G be the number of imputed datasets to generate. This means we get G uniquely imputed datasets in step (2), where each of these is analysed separately in step (3). The estimates of interest derived from the G datasets can be pooled into a single estimate using Rubin's rules in step (4).[129] These are rules based on asymptotic theory derived from the Bayesian framework. Examples of estimates are the expectation of a prediction or statistical parameters of a linear model. The benefit is that the imputation provides an empirical distribution of the estimates with information on the variance.[170]

Rubin's rules can be summarized as follows: Let G be the number of imputed datasets generated, and $\hat{\boldsymbol{\theta}} \in \mathbb{R}^G$ be the vector where each element in the vector corresponds to the estimand of interest of each G imputed datasets. The overall, or pooled, estimate $\bar{\theta}$ is the mean of the vector entries:

$$\bar{\theta} = \frac{1}{G} \sum_{j=1}^{G} \hat{\theta}_j. \tag{3.4}$$

According to Rubin, the variance of the estimate has a slightly more intricate calculation to address both the within-imputation variation and between-imputation variation. Denote W as the within-imputation variance and B the between imputation variance, then the variation of the estimates $\hat{\boldsymbol{\theta}}$ is:

$$\text{Var}(\hat{\boldsymbol{\theta}}) = W + \left(1 + \frac{1}{m}\right) B. \tag{3.5}$$

where

$$W = \frac{1}{G} \sum_{j=1}^{G} W_j$$
$$B = \frac{1}{G-1} \sum_{j=1}^{G} (\hat{\theta}_j - \bar{\theta})^2$$

The most popular strategy for multiple imputation is multiple imputation by chained equations.

Multiple Imputation by Chained Equations

Multiple imputation by chained equations (MICE), also known as fully conditional specification or sequential regression multiple imputation, is a popular multiple imputation technique.[7] It works by specifying a multivariate imputation model for each variable individually, using a series of conditional densities tailored for each variable with missing data. The missing values are replaced by drawing imputations from each conditional density. The benefit of this strategy is that it imputes each variable conditional to the others by effectively using the information from the variables.

MICE assumes that data respects the MAR mechanism.[8] This is important because it offers some systematic relationship between the variables to use for imputations. Consider the simple example with two variables *age* and *blood pressure*. Blood pressure measurements are often not necessary for young patients and may not be routinely taken. However, when blood pressure data is available for some young individuals, MICE can utilize this information to impute the remaining missing ones.

A general description of the algorithm can be summarized in algorithm 2.

Algorithm 2 MICE algorithm[7,20,18]

Let $\mathbf{X} \in (\mathbb{R} \cup \{NA\})^{n \times m}$ be the matrix with a mix of complete and missing values, and $D_m = \{j : \exists i, X_{ij} = NA,\ j = 1, 2, \ldots, m\}$ be the set of column indices with at least one missing value in $\mathbf{X}$. Furthermore, denote $\boldsymbol{x}_j^{(m)}$ the vector of missing values and $\boldsymbol{x}_j^{(c)}$ the vector of non missing values in $\boldsymbol{x}_j$ for $j \in D_m$, respectively. This means that $\boldsymbol{x}_j = \boldsymbol{x}_j^{(m)} \| \boldsymbol{x}_c^{(c)}$.

The algorithm requires two parameters: the number of imputed datasets to generate G and a specified imputation model $\mathcal{I}$ for each variable.

1: **Initialization**
2: **for all** $j \in D_m$ **do**
3: **for all** $i \in \{1, \ldots, \dim(\boldsymbol{x}_j^{(m)})\}$ **do**
4: $x_{ij} \leftarrow \text{random}(\boldsymbol{x}_j^{(c)})$
5: **end for**
6: **end for**
7: **for all** $j \in D_m$ **do**
8: Set $\boldsymbol{x}_j^{(m)} \leftarrow NA$
9: $m_j(\boldsymbol{x}_j^{(c)}) = fit(\boldsymbol{x}_j^{(c)} \sim \mathcal{I}\,(\boldsymbol{X}_{\neq j}))$.
10: $\boldsymbol{x}_j^{(m)} \leftarrow pred(m_j, \boldsymbol{X}_{\neq j})$
11: **end for**
12: Steps 7-11 are repeated multiple times (suggested 5-20) to iteratively update the imputations, where the final imputations are retained.
13: Repeat steps 1-12 G times to get G imputed data sets.

Output: $\hat{\mathbf{X}} \in \mathbb{R}^{n \times m \times G}$

Step (4) of the MICE strategy is the initialization of MICE; any missing values are temporarily filled with placeholder values that are random samples drawn from the existing observations in their respective variables. In steps (8)-(10), the algorithm sequentially goes through each variable to prepare the imputations. For each column that is being imputed, the placeholder values are set back to missing, and the specified imputation model is fit on the complete data using the other variables as predictors. Note that because all other variables have placeholder values where they are initially missing, only the missingness in the current variable determines how many samples are used in the fit. The final imputations are the predictions from the last iteration in step (12).

The strategy of specifying imputation functions is why it is named 'chained equations'; it is advised to specify the imputation functions for each single variable to address them properly. For instance, it makes sense to use logistic regression for binary variables and predictive-mean matching for integers.[20] A list of implemented imputation models in R's *mice* package is listed in Table 3.6. The four default strategies are predictive-mean matching (PMM), logistic regression (logreg), polytomous regression (polyreg) and the ordered logit model (polr).

Table 3.6: Implemented imputation functions in the *mice* package in R.[20] Default Y denotes that if no imputation functions are specified, it will default to these strategies.

Method	**Description**	**Scale type**	**Default**
pmm	Predictive mean matching	numeric	Y
norm	Bayesian linear regression	numeric	
norm.nob	Linear regression, non-Bayesian	numeric	
mean	Unconditional mean imputation	numeric	
2L.norm	Two-level linear model	numeric	
logreg	Logistic regression	factor, 2 levels	Y
polyreg	Multinomial logit model	factor, >2 levels	Y
polr	Ordered logit model	ordered, >2 levels	Y
lda	Linear discriminant analysis	factor	
sample	Random sample from the observed data	any	

It is important to consider what variables should be used in the imputations. Including more variables provides more information to use in the imputations, and it makes the MAR assumption more plausible.[131] This will likely not cause issues for small datasets. However, larger datasets can run into problems such as multicollinearity and long computational time. A useful feature in the *mice* software in R is the ability to specify what variables should be used to model every incomplete variable.[18]

The MICE algorithm uses information from incomplete samples, which makes it more efficient than CCA. If the data is MAR, it will also correct the bias.[84]

3.7 A Note About Categorical Variables

Most algorithms require a numerical representation of categorical variables before imputing. Consider two variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ that contain the set of values {Yes, No} and {Small, Medium, Big, Huge}, respectively. Using kNN imputation would not work because the distance metric in Eq. (3.3) requires real numbers. A solution is mapping the categories to integers. This works reasonably well if the variables are ordered or binary, but caution is required for nominal ones.

We first consider the case of ordinal variables. Suppose we map $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ to an integer representation $\boldsymbol{x}_1 = \{0, 1\}$ and $\boldsymbol{x}_2 = \{0, 1, 2, 3\}$. Now, suppose we have some arbitrary missingness in $\boldsymbol{x}_1$ or $\boldsymbol{x}_2$, and use kNN to find the 4 nearest neighbors. Let the four tuples $T = \{(0, 1), (0, 1), (0, 3), (0, 3)\}$ (the first value is $\boldsymbol{x}_1$ and the second is $\boldsymbol{x}_2$) be the nearest neighbors. Then it can fairly be said that if $\boldsymbol{x}_1$ was to be imputed, 0 is the correct imputation. If $\boldsymbol{x}_2$ was to be imputed, we would impute using $\frac{1+1+3+3}{4} = 2$. As $\boldsymbol{x}_2$ is ordered, it makes some sense to say that the average of medium and huge is big.

A problem arises when $\boldsymbol{x}_2$ is nominal. Suppose we have the exact same example, but instead of $\{Small, Medium, Big, Huge\}$, we have $\{France, England, Norway, Australia\}$. Then the average of England and Australia would be Norway, which may not make much sense. In addition, if an imputation returns a float, it is required to round it to an integer to be able to map the imputation back to the categories.

We propose a solution to this by employing the target encoding discussed in Section 2.5.1. Instead of mapping category levels to integers, each category is mapped to the average of the target variable. Generally, target encoding is mostly used with nominal variables with many levels, such as zip codes. However, it resolves the problem mentioned above of mapping nominal variables. Target encoding the nominal variables allows for both computing sensible distances and employing imputations that are reasonable. The apparent downside is that it is difficult, if not impossible, to map the encoding back to categories. This is because the averages are computed using cross-validation, and it is not guaranteed that the averages of each factor are the same. If they overlap with other category levels, it is an extra level of complication. Thus, this has to be done as a last step before the variables are passed on to a model.

3.8 Evaluating Imputation Methods

When replacing missing values, one key point is assessing how suitable the imputations are. As suggested by Lin and Tsai (2019)[85] in their literature review on missing value imputation, one should consider three approaches: direct evaluation, indirect evaluation and computational time. The latter should be considered because it provides context to the efficiency of the imputations, and it is especially important for larger datasets. Investigating all three evaluation strategies allows for a better understanding of the performance of the imputation, and also potential suggestions for developing better techniques.

To limit the thesis to a reasonable scope, we do not emphasise computational efficiency as we are not developing any new imputation strategies. We will, however, mention the runtimes of the experiments. Our main focus is evaluating the imputation methods using direct and indirect evaluations. Details on both strategies follow in the two subsequent subsections.

3.8.1 Direct Evaluation of Missing Value Imputation

Direct evaluation of an imputation involves measuring the distance between the ground truth value and the imputed value.[85] Naturally, the disadvantage of direct evaluation is that the missing value must be known to have something to compare with the imputed value.

There are many evaluation metrics to choose from. For numerical data, mean absolute error (MAE), root mean square error (RMSE) and mean square error (MSE) are the most common, and for categorical data, proportion of correct predictions (PCP) is a valid option.[85] The choice

of metric depends on the application. For example, the disadvantage to MSE is that it deviates from the original scale of the data because it is squared. Another example is in comparisons of imputation performance across different datasets of different sizes, then blindly applying one of the mentioned metrics can be misleading. Consider the scenario with two datasets, one containing information on sturgeons and one on whales; when imputing a variable such as weight, it would make sense to normalize the metric in order to make imputations between these comparable.

A brief point to raise is Buuren (2018)[18] suggesting that RMSE is not the best metric to use for assessing the quality of imputations. The minimum RMSE is acquired when the missing value is predicted by a linear model where the weights are acquired by the ordinary least squares (OLS) estimate. This will, however, always replace the missingness with the same value repeatedly because it replaces the missing value by the most likely value defined by the model. The downside to this is that it completely omits the uncertainty of the missingness, and should therefore not be used as a metric to separate good and bad imputations. While they do not propose alternative metrics, it may encourage the strategy of using indirect evaluations.

For the following definition, recall that $\hat{x}_{ij}$ is the imputed value, and x_{ij} is the ground truth value for the indices (i, j). n denotes the number of samples, and m the variables (Section 1.4). The missing value experiment in Section 4.3 only considered categorical variables. Thus, we limited the direct evaluation to the accuracy classification metric to assess the quality of the imputations.[55] As an imputation is either correct or incorrect, we use the definition

$$accuracy = \frac{1}{|T|} \sum_{(i,j)\in T} 1_{\hat{x}_{ij}=x_{ij}} \tag{3.6}$$

Here T denote the set of coordinates of the missing values $T = \{(i, j) : 1 \leq i \leq n, 1 \leq j \leq m, x_{ij} \text{ missing}\}$. The denominator is the magnitude of the set, i.e. the total number of samples that are missing. The lowest score is 0, and the highest is 1; values close to 1 are desired.

3.8.2 Indirect Evaluation of Missing Value Imputation

Indirect evaluation of an imputation involves estimating some effect of the variable(s) after imputing. We define two categories of indirect evaluations: the first is where no information on the real value is required, and the second evaluation is where the ground truth value has to be known.

An example of the first one is the R^2 of a regression model.[37,102] This is because the underlying real value does not need to be known in order to estimate this parameter. This has limitations, for example, when these effects cannot be estimated after imputing. Consider a model that requires calculating the inverse of a matrix. In some situations, the imputation model can induce a singular matrix - thus, the problem has infinite (or no) solutions. There are solutions to this problem, such as computing an alternative inverse or penalizing the loss. However, it would require changing the problem, which in itself is a sign that the imputation, or problem, is ill-posed.

The second category is dependent on the missing values themselves. For example, comparing the imputed variables' impact on fitting a model where the ground truth model is known from the complete data. Lee and Huber (2021)[84] simulated MCAR, MAR and MNAR of different proportions, where the original data was modified in such a way that they had complete data. Using the complete data, they estimated the 'real' parameter of interest, namely the mean of a variable. Next, they measured the absolute bias for each missing value mechanism and different proportions after either removing missing rows or using multiple imputation.

It is possible for the indirect approach to shadow the effect of an imputation. Consider a strategy where the real model $\mathbf{y} = \beta_0 + \beta_1\mathbf{x}_1 + \boldsymbol{\epsilon}$ is known. Here we just present a regular linear model with no relation to imputation. Next, suppose it is of interest to know the deviation of β_1 from an estimated $\hat{\beta}_1$ given by $\mathbf{y} = \hat{\beta}_0 + \hat{\beta}_1\mathbf{x}_1 + \boldsymbol{\epsilon}$, where the hat $\beta' s$ are model parameters estimated from imputations of the original dataset. If the true $\mathbf{x}_1$ has no significant effect on the target $\mathbf{y}$, then the assigned weight to β_0 will lie somewhere close to zero. Thus, assessing the difference of

β_1 and $\hat{\beta}_1$ will most likely not be very large, and poor imputations may not show any bias of $\hat{\beta}_1$.

3.9 Limitations of Modern Imputation Strategies

Imputation strategies are still a domain that requires attention, and we will discuss a few main concerns about current limitations. Two issues related to numerical data are the problems of sparsity and heterogeneity. In addition, we discuss the problems of valid imputations.

Sparsity refers to the fact that a sample may not share similar values for imputation.[180] This issue is more pronounced with smaller datasets. A consequence is that imputation strategies which directly use the values of the other entries to extrapolate will have poor performance. For example, kNN imputation uses the average of the closest neighbors as the imputed value. However, the average will not be a good estimate if no tuples share similar values. Heterogenity relates to the fact that data can originate from different sources, and variables can describe different things. Thus, finding a single imputation model that fits all the data may be unsuitable.[130] Figure 3.3 illustrates both sparsity and heterogeneity. It is clear that linear interpolation is unsuitable due to non-linearity, and kNN suffers from sparsity.

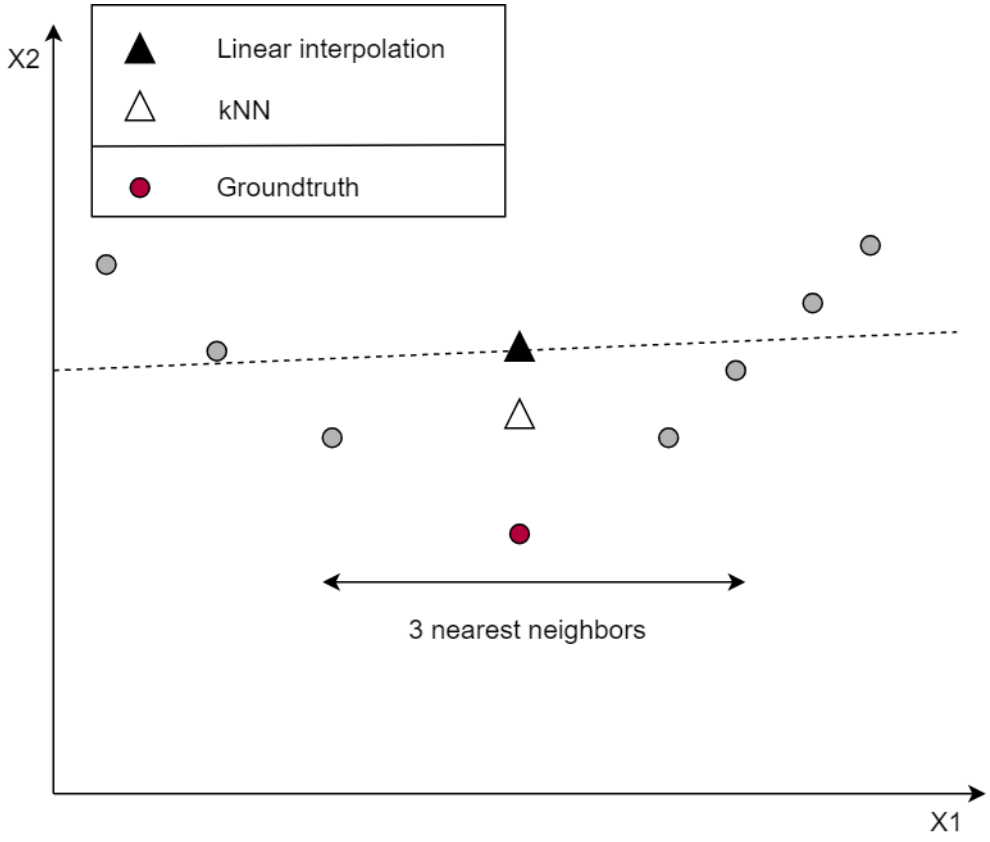


Figure 3.3: Illustration of sparsity and heterogeneity. kNN assumes 3 neighbors for imputation. While linear interpolation is not a strategy we have discussed or implemented, we present it as a brief example to illustrate a potential flaw. The figure is inspired by Zhang et al. (2019).[180]

Apart from the technicalities of finding a good fit for the data, a different problem is that imputations are not guaranteed to be valid. Nothing stops an imputation of pregnancy to suggest that a male is pregnant, which makes no sense. This is difficult to combat because it relates to more than respecting the domain of a variable. For instance, PMM in multiple imputations assures that the suggested imputation will never be outside of the domain. If a variable contains only positive values, then the donor pool for PMM will always contain positive values. The problem is that it does not guarantee that it respects the relations with the other variables. This is related to the brief point raised in Section 2.5.2 of detecting outliers of odd combinations of factor levels.

4 Materials and Methods

This section covers the two experimental setups. It is divided into two major parts: (1) the experimental setup for handling missing values and (2) survival analysis of GEP NEN. The end goal is explainable and accurate survival modelling of patients with high-grade GEP NEN. The motivation for conducting the missing value experiment is to address two main concerns from the literature: the scarcity of missing value experiments when working with survival models and the main issues found in litterature reviews raised in the theory part.

4.1 Data

We considered two datasets in our study. For the missing value experiment, we used the open source colorectal dataset found in R[81] and for the survival study, we used the GEP NEN dataset.[153]

The GEP NEN dataset contains patients diagnosed with gastrointestinal or pancreatic cancer treated with one or multiple rounds of chemotherapy.[153] This was the one of scientific interest, however it posed various challenges. The most prominent one was that it was a combination of two clinical studies with a different number of variables, which led to a high number of missing values and a mismatch in features. It also had many missing entries that were unrelated to what study they belonged to. Therefore, it was important to handle the missingness with care and utilize as much information in the variables as possible. This is why we used the colon dataset as the main study object for the missing value experiment. It was complete and thus could be used for realistic simulations of missingness by dropping values to mimic an incomplete dataset.

4.1.1 Colon

The colon dataset originates from one of the first successful trials of chemotherapy for colon cancer.[81] The data is open source and can be found in R by using $data(cancer, package = "survival")$ from the survival package. A description of the variables and the encodings are given in Table 4.1. The two treatments given were Levamiosole and Levamiosole+5-FU. The Observation group was the control group. The time to event variable *time* was the target variable of interest.[160]

The dataset contains 16 variables and 1858 samples. However, all patients were listed twice, with one record for recurrence and one for death. After removing the recurrent rows, the sample dimension was halved. The two variables *nodes* and *differ* contain 36 and 46 missing values, respectively, which lead to the final sample dimension of 888 when removing the samples with missingness. For the variables, *age* was removed because it violates the proportional hazard assumption. *nodes4* is a binary variable indicating if the number of lymph nodes with detectable cancer in the *nodes* variable is greater than 4. We removed *nodes4* due to the strong correlation between them. Finally, removing the non-informative columns *id* (patient ID), *study* (constant), *etype* (constant after removing recurrent rows) gave the final dataset dimensions 888×11. A plot of the distribution of the variables, separated by censoring status, is given in Figure 4.1. See Figure 4.2 for the correlations. *nodes* and *nodes4* are highest correlated as the latter is derived from the first.

There are two main motivations for choosing this dataset as opposed to simulating an arbitrary one. The first being that it is closely related to the GEP NEN dataset from a medical perspective; the colon and gastro is anatomically similar. The second is that it poses similar challenges as the GEP NEN dataset presented in Section 4.1.2, namely that the majority of the variables are categorical, with heavily imbalanced class distributions in some variables, as seen in figure 4.1. For instance, *perfor* has the distribution of 3% Yes and 97% No. For the four levelled variable *extent*, the levels make up the following proportions: Submocosa 2.1%, Muscule 11.5%, Serosa 82.2% and Contiguous_structures 4.2%.

Table 4.1: Description of the variables in the colon dataset.[161] Variables with an asterisk (*) were omitted in the experiment. The variable *nodes*, marked with (†) was binned to a categorical representation (discussed in Section 4.3).

Variable	**Type**	**Description**	**Encoding**
id (*)	Numerical, integer	Patient ID	-
study (*)	Numerical, integer	1 for all patients	-
rx	Factor, 3 levels	Treatment type	Target
sex	Factor, 2 levels	Gender	One-hot
age (*)	Numerical, integer	Age in years	-
obstruct	Factor, 2 levels	Obstruction of colon by tumour	One-hot
perfor	Factor, 2 levels	Perforation of colon	One-hot
adhere	Factor, 2 levels	Adherence of nearby organs	One-hot
differ	Factor, 3 levels	Differentiation of tumour	Ordinal
extent	Factor, 4 levels	Extent of local spread	Target
surg	Factor, 2 levels	Long or short time from surgery to registration	One-hot
nodes (†)	Numerical, integer	number of lymph nodes with detectable cancer	Ordinal
node4 (*)	Factor, 2 levels	more than 4 positive lymph nodes	-
status	Factor, 2 levels	Censoring status	Boolean
time	Numerical, integer	Time to event or censoring	-
type (*)	Factor, 2 levels	Event type (recurrence or death)	-

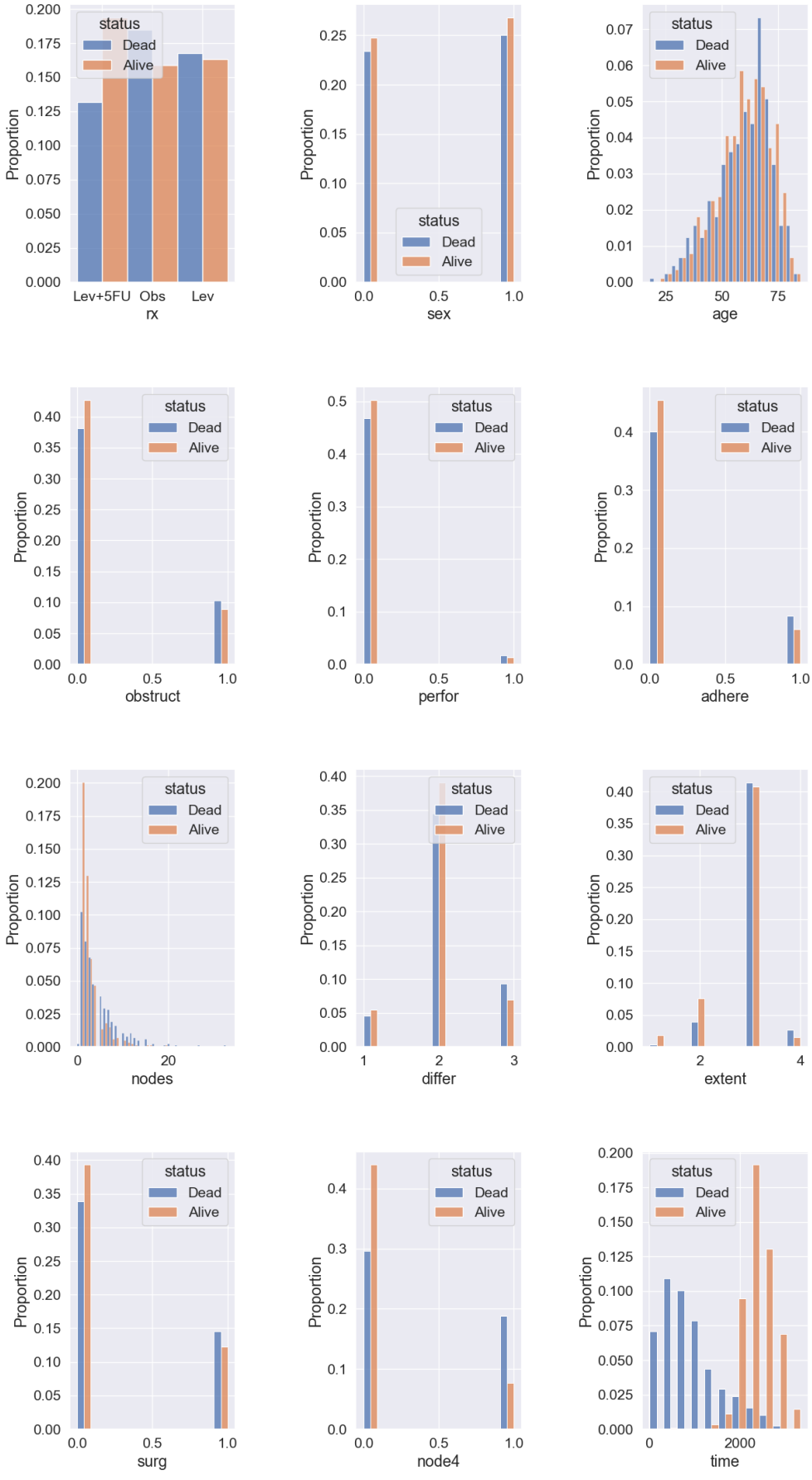


Figure 4.1: Distribution of the variables from Table 4.1, separated by censoring status.

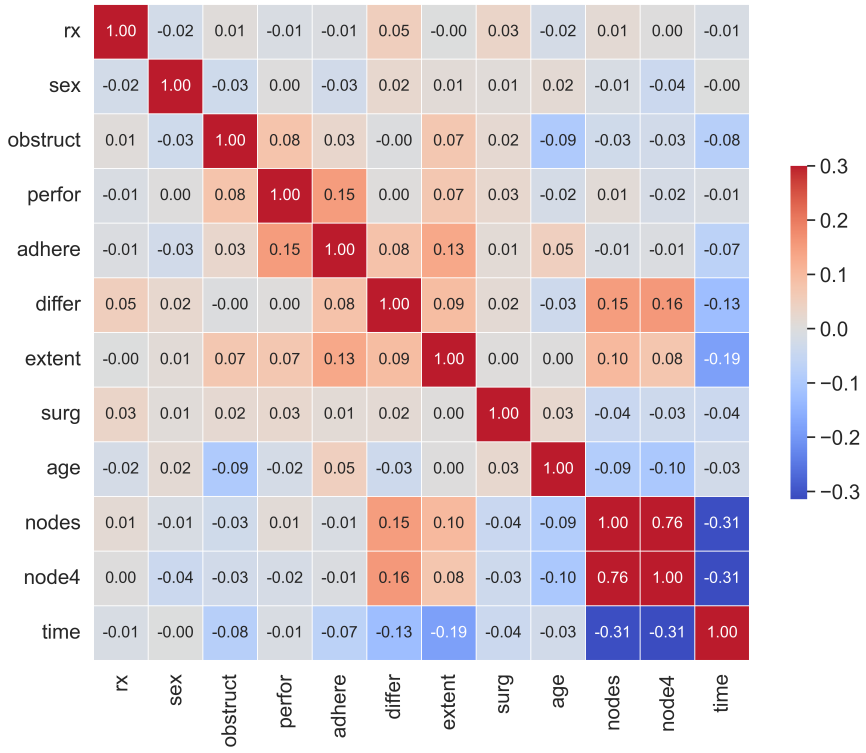


Figure 4.2: Correlation of the variables from Table 4.1. The figure shows correlations before *nodes* was binned.

4.1.2 GEP NEN

This study involved 192 patients treated at the Department of Oncology, Oslo University Hospital from January 2000 to July 2018.[153] These patients were part of two multi-institutional Nordic NEC studies overseen by the Nordic Neuroendocrine Tumour Group. All patients received treatment in the form of surgery, chemotherapy, or both.[65] The effectiveness of the chemotherapy was evaluated using CT/MR imaging, applying the Response Evaluation Criteria in Solid Tumors (RECIST).[65,151] This evaluation primarily measured changes in tumour size from one imaging session to the next. The target variable measured was overall survival (OS), expressed in days. OS was calculated from the date of diagnosis to the date of death or last observation. Patients alive at the last observation date were censored.[153]

The dataset includes 192 patients, 79 from the original study and 113 from the subsequent study. This latter study, an extension of the original, aimed to examine additional variables, necessitating a new case report form (CRF). For analytical clarity, we designated these as the "old" and "new" studies. From these studies, we generated three main datasets for detailed exploration and analysis: one containing only patients from the old study, another exclusively for patients from the new study, and a third, a combined dataset that includes only the features common in both studies. Our evaluations determined that the dataset from the new study was the most optimal. The dataset from the old study was found to be less informative, and the combined dataset was constrained by a limited number of variables. Consequently, following preparations for the new study, the dataset included 99 patients and 50 variables. Tables 4.2, 4.3, 4.4, and 4.5 contain the variables included in the new study. The tables show the datatypes, a short description of the variable, and the missingness mechanisms discussed in Section 3.2.1. The latter were found from expert knowledge. Categorical variables were divided into three types based on how they were gonna be encoded: binary, ordinal, and nominal. Binary variables were processed using one-hot encoding, while nominal variables with more than two categories were target encoded (Section 2.5.1).

Table 4.2: This table provides an overview of various numerical variables used in the dataset for the new study, including their data types and brief descriptions of what they are. It also details the missingness mechanisms, along with the percentage of missing values (NA%) for each variable. Note that all float variables were integers, just of float datatype.

Variable	**Type**	**Description**	**Mechanism**	**NA (%)**
BMI	Numerical (float)	Body Mass Index	MCAR	3
Ki-67	Numerical (float)	Ki-67 expression	-	0
Absolute Neutrophil Count	Numerical (float)	Neutrophil count per μL blood	MCAR	2
Albumin	Numerical (float)	Blood albumin level	MCAR	1
CRP	Numerical (float)	Blood C-Reactive Protein level	MCAR	1
Number of Courses	Numerical (float)	Number of treatment courses	MAR	15
Time from diag to mets (days)	Numerical (float)	Time from diagnosis to metastasis in days	-	0
Age at Diagnosis	Numerical (int)	Patient´s age at diagnosis time	-	0
OS (days)	Numerical (float)	Overall Survival (days)	-	0

Table 4.3: Description of ordinal variables. This table summarizes the ordinal variables used in the dataset for the new study, categorized by type, description, and the missingness mechanisms. It also presents the percentage of missing data for each variable.
Note that the descriptions of Chromogranin A and A2 were the same, as they initially had the same variable names but different categorical levels.

Variable	**Type**	**Description**	**Mechanism**	**NA (%)**
WHO Perf Stat	Factor, 4	Status of the patient's well-being	MCAR	5
T-stage	Factor, 5	Tumour size/extent	MCAR	3
N-stage	Factor, 4	Lymph node spread/metastasis	MCAR	3
Chromogranin A	Factor, 4	Neuroendocrine tumour markers	MCAR	7
Chromogranin A2	Factor, 4	Neuroendocrine tumour markers	MCAR	26
Synaptophysin	Factor, 4	Synaptophysin level in blood	MCAR	7
LDH	Factor, 3	Lactate dehydrogenase level in blood	MCAR	4
NSE	Factor, 3	Neuron-specific enolase level	MCAR	7
CD-56	Factor, 3	Neuroendocrine tumour marker	MCAR	63
Differentiation	Factor, 3	Cell maturity level	MCAR	40
Octreoscan	Factor, 3	A radionuclide imaging scan	MAR	54
ALP	Factor, 3	Liver enzyme levels	MCAR	5

Table 4.4: Description of binary variables. This table lists the binary variables used in the dataset for the new study, all classified as type factor 2 (binary), description, the missingness mechanisms, and the percentage of missing data (NA%).

Variable	**Type**	**Description**	**Mechanism**	**NA (%)**
Sex	Factor, 2	Gender (M/F)	-	0
Co-morbidity Severity	Factor, 2	Comorbidity-severity level	MCAR	28
Co-morbidity	Factor, 2	Presence of co-morbid conditions	-	0
Hist Exam Metastasis	Factor, 2	Diagnosis confirmed from metastasis	-	0
Hist Exam Primary Tumur	Factor, 2	Diagnosis confirmed from primary tumour	-	0
Living Alone	Factor, 2	Living situation	-	0
Prior Other Cancer	Factor, 2	Previous cancer history	-	0
Primary Tumour Resected	Factor, 2	Surgical removal of primary tumour	-	0
Stage grouped	Factor, 2	Cancer stage classification	-	0
Haemoglobin	Factor, 2	Haemoglobin concentration	-	0
Platelets	Factor, 2	Platelets count in blood	MCAR	1
WBC	Factor, 2	White blood cell count	-	0
Dev of Bone Mets	Factor, 2	Development of bone metastases	MCAR	7
Loc Adv Resectable Disease	Factor, 2	Locally advanced disease resectability	-	0
Mets(Other)	Factor, 2	Metastases to other sites	-	0
Mets(Lung)	Factor, 2	Lung metastases	-	0
Mets(Liver)	Factor, 2	Liver metastases	-	0
Mets(LN)	Factor, 2	Lymph node metastases	-	0
Mets(LN Retro)	Factor, 2	Retroperitoneal lymph node metastases	-	0
Mets(LN Regional)	Factor, 2	Regional lymph node metastases	-	0
Mets(LN Distant)	Factor, 2	Distant lymph node metastases	-	0
Mets(Bone)	Factor, 2	Bone metastases	-	0
Status	Factor, 2	Event/censored	-	0

Table 4.5: Description of non-binary nominal features. This table details the variables in the study that have more than two categories, each defined as a type of factor with a specified number of categories (e.g., a factor of 3 has 3 categories). It provides an overview of each variable's description, the missingness mechanism, and the percentage of missing data (NA%).

Variable	**Type**	**Description**	**Mechanism**	**NA (%)**
Tumour Morphology	Factor, 3	Tumour cell type	MCAR	27
Primary Tumour	Factor, 9	Primary Tumour Location	-	0
Smoking	Factor, 4	Smoking status	MCAR	10
Best Response (RECIST)	Factor, 6	Response to treatment	MCAR	24
Treatment Stopped	Factor, 5	Reason for cessation of treatment	MAR	15
Chemotherapy Type	Factor, 4	Chemotherapy regimen used	MAR	14

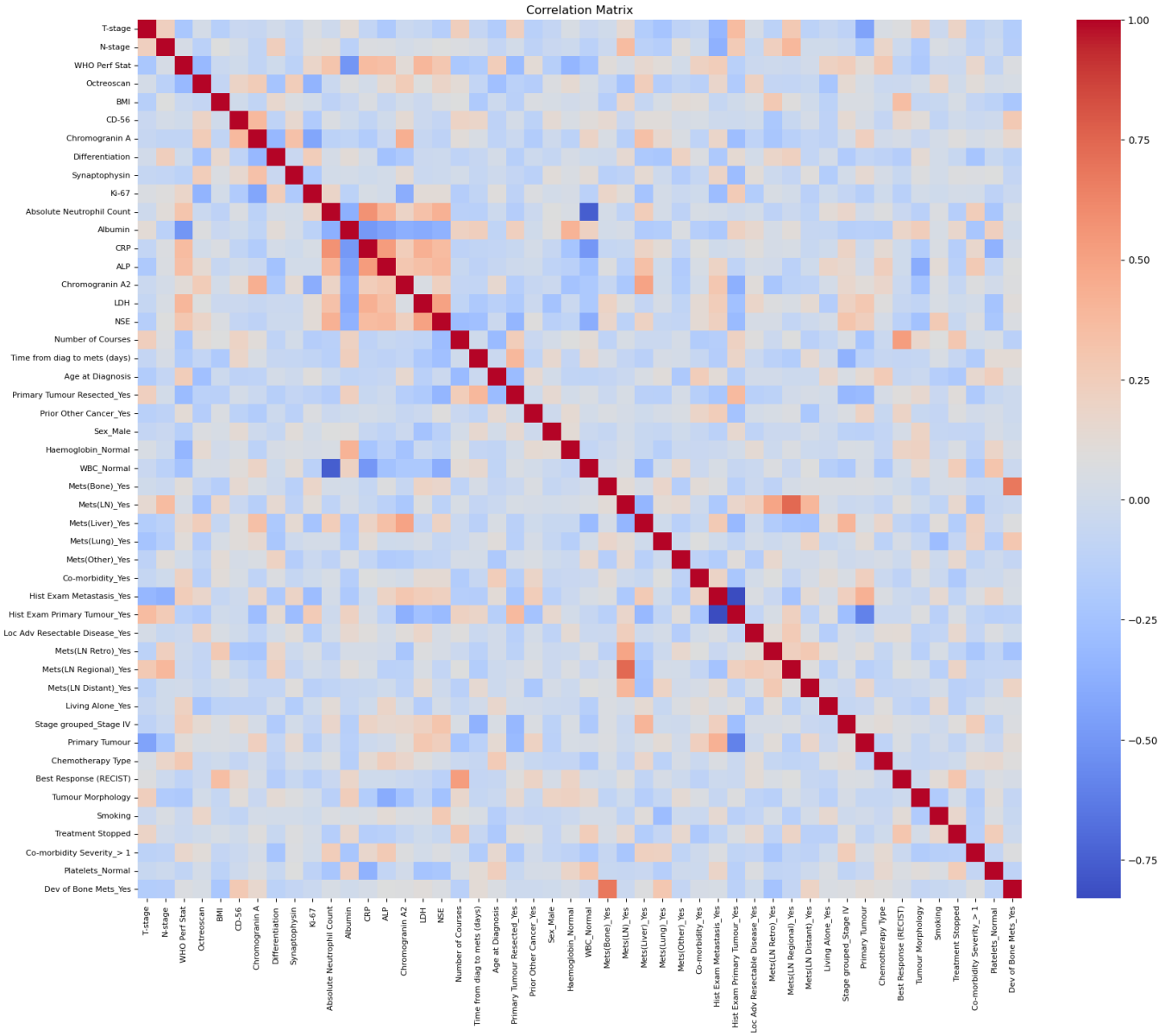


Figure 4.3: Pearson correlation of the GEP NEN dataset illustrates the Pearson correlation among all variables. The color spectrum from blue to red represents correlation values ranging from -1.00 to +1.00, indicating strong negative to strong positive correlations.

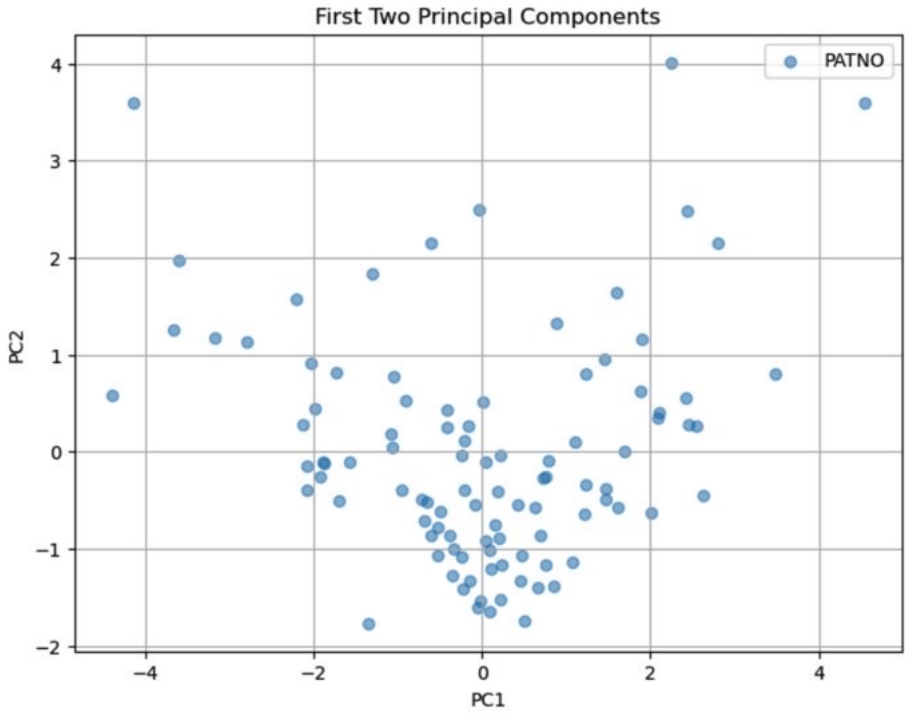


Figure 4.4: PCA plot of the patients (PATNO) in the GEP NEN dataset using only the numerical variables. The plot visualizes the distribution of patients along the first two principal components (PC1 and PC2). Each point represents a patient and helps identify potential outliers among them. Discussed in Section 2.5.2.

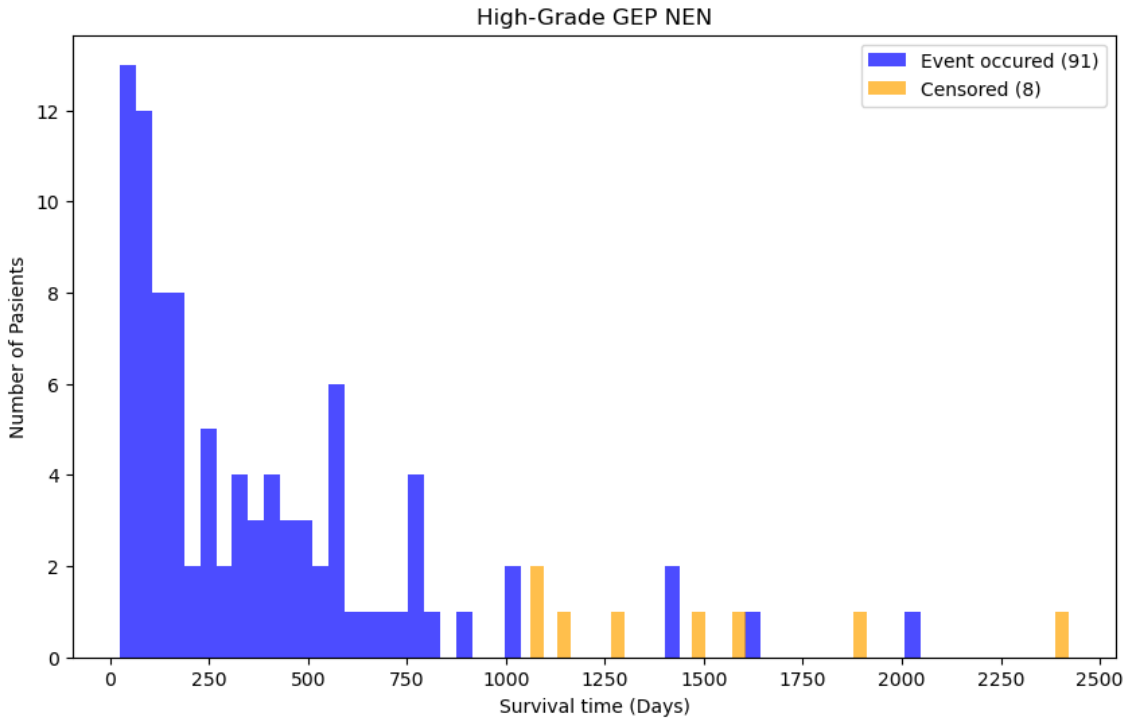


Figure 4.5: Distribution of censoring over survival time (days) for GEP NEN dataset. 8 patients were censored, and 91 patients experienced an event.

4.2 Software and hardware

4.2.1 Software

The main software used for the thesis was Python (3.11.7)[122] and R (4.3.1)[159]. Python was primarily based on notebooks, but where convenient such as utility files, .py files were used. R was ran in R-studio in notebooks.

The main packages used in Python was *scikit-learn* (1.3.2)[111], *Pandas* (2.1.4)[92] and *NumPy* (1.26.4)[51] for preprocessing and handling data. *Matplotlib* (3.8.0)[58] and *SeaBorn*[169] (0.13.2) were the main graphical tools. For survival analysis, *lifelines* (0.27.8)[27] and *scikit-survival* (0.22.2)[116] were used. *PyTorch* (2.2.1)[110] was used for parts in the missing value experiment described in Section 4.3.

For the experimental setup in Section 4.3, a local clone of a GitHub repository was used for simulating missing values. This is discussed in detail in Section 4.3.4 of generating missing values under the experimental setup.

R was primarily used for the *mice* (3.16.0)[20] package for multiple imputation. In addition to imputations, the package contains built-in functionality for calculating influx and outflux. The missing value experiment in Section 4.3 was primarily conducted in Python, however, we used the *mice* package in R for multiple imputations. As it was necessary to move simulated data with missingness from Python to R, the package *reticulate* (1.35.0)[68] was used for this.

4.2.2 Hardware

The code was run on two separate machines, denoted machines (1) and (2). (1) The missing value experiment in Section 4.3 was on a Windows 11 laptop with CPU AMD Ryzen 9, and dedicated GPU AMD Radeon RX 6800HS (3.3 GHz) and 32 GB RAM. (2) The survival models on a Macbook Air M2-chip with an 8-core CPU and 16 GB RAM.

4.3 Experimental Setup: Missing Values (Colon)

4.3.1 Motivation

We briefly introduce the important topics we considered in the missing value experiment and the motivation for addressing them. We will mention (a) missingness with survival models, (b) missingness mechanisms, (c) robustness of repeated simulations, (d) both direct and indirect evaluation and (e) validation.

Point (a) relates to the concern of papers that demonstrate statistical or machine learning models where they have data with missingness, but they perform quick imputations with little consideration as a preprocessing step.[173] Pairing this with papers that discuss survival analysis, the list grows short.

Furthermore, point (b) considers the problem that many disregard the missingness mechanisms, as found in the literature reviews in Section 3.2.1. This is a crucial step for imputing, especially because of imputation model assumptions. For our experiment, we considered MCAR and MAR, because we found it plausable that MNAR is not present in the GEP NEN dataset.

Next, (c) is not mentioned in the literature reviews we cite but is a concern anyway. The idea is that performing a single simulation of missingness to then impute and evaluate is not sufficient. It can very likely happen that the parts of the data containing simulated missingness are not representative, thus leading to biased imputations and conclusions. For this reason, we will repeatedly simulate missingness by varying the samples and variables used in simulating missingness and finding the expected values and variations of the metric of interest over the repeated simulations.

Point (d) is related to the suggestion of multiple evaluations. Measuring the quality of imputations by means of direct evaluations (such as minimizing RMSE) may not be sufficient. It was

inspired by the literature review of Lin and Tsai (2019)[85] in Section 3.8 where they propose that future literature should both directly and indirectly evaluate the imputations.

Finally, point (e) is present to be consistent with modern machine learning approaches. More specifically, splitting data into train and test is standard. The goal is then only to use information inferred from the train part to handle the test part. So, for example, applying mode imputation on categorical features would imply that the missingness in the test set is replaced by the mode in the training set.

The experimental setup is described in the following order: the strategies applied in the experiment (4.3.2), the evaluations (4.3.3), how missing values are generated (4.3.4) and finally the pipeline (4.3.5).

4.3.2 The Strategies Applied

The purpose of the experimental setup was to see the effect of various strategies applied when handling missing values under MCAR and MAR, for increasing proportions of missing values. See Table 4.6 for an overview of the strategies applied, along with the missingness mechanisms. CCA was discussed in Section 3.5, and the imputation strategies in Section 3.6.

The experiment was limited to categorical variables only. We would require separate metrics for evaluating the numerical and categorical imputations if both were included. This could be applying RMSE to the numerical variables and accuracy to the categorical. However, under MAR, the sampling of which variables are chosen to contain missingness would yield a large variation in the metrics as RMSE is unbounded. Thus, under MAR, interpreting the RMSE would not make any sense. The only numerical variable in the colon dataset was *nodes*. As this variable was skewed, we found that binning it into the four groups $[1], (1-3], (3-7]$ and $(7, 33]$ gave the most balanced distribution. For this reason, we did not consider any numerical global imputers.

The motivation for the strategies was as follows: for the global, mode imputation was one of the few options available to compare against the ground truth values. For example, by replacing a missing category with an indicator variable such as 'unknown', it would not be possible to compare it with the ground truth. Next, for the multivariate, kNN is a popular strategy for multivariate imputations, and literature suggests it performs well. For the multiple imputations, MICE is widely praised in literature for providing state-of-the art imputations. CCA was applied to compare imputations with no imputations.

Table 4.6: Strategies for handling missing values used in the experimental setup of missing values on the colon dataset.

Scope	**Strategy**	**Mechanism**
Global, categorical	Mode	MCAR, MAR
Multivarite	kNN	MCAR, MAR
Multiple	MICE	MCAR, MAR
-	CCA	MCAR, MAR

4.3.3 Evaluation

We considered three ways to evaluate the strategies applied to the missing data. (1) accuracy (direct evaluation), (2) bias of coefficients of a Cox PH model (indirect evaluation) and (3) Harell's concordance index (indirect evaluation). The accuracy from Eq. (3.6) was suitable as we restricted the colon cancer dataset to categorical variables only.

The bias (2) can be described as follows: with the complete dataset with no missingness simulated, we fitted a Cox PH model from Eq. (2.5) and extracted the coefficients. The idea was to use coefficients fitted on the complete data with no missingness as a proxy for the 'ground truth' coefficients. Next, for each missing value strategy applied, a new Cox PH model was fitted, and the bias of these newly estimated coefficients were measured.

Let $i \in \{1, \ldots, M\}$ represent the index of one of the simulated datasets, and $j \in \{1, \ldots, m\}$ be the index of a variable in $\mathbf{X}$. Then $\hat{\beta}_j^{(i)}$ is the estimated parameter for the j-th imputed variable in the i-th dataset, and β_j is the proxy ground truth parameter from the Cox PH model fitted on the complete dataset. We define the bias as:

$$Bias(\hat{\beta}_j^{(i)}) = \frac{\hat{\beta}_j^{(i)} - \beta_j}{|\beta_j|} \tag{4.1}$$

For the equation, the numerator represents the deviation from the proxy ground truth parameter, and the denominator is present for standardization. The absolute value is to keep the direction of the bias consistent. Note that if the proxy ground truth model weights are very small, the bias becomes very large as it is sensitive to small changes.

Finally, (3) refers to Harell's concordance index defined in Section 2.4.2. For each imputation strategy, the concordance index of the same fitted Cox PH model from above was measured.

It may be helpful to visualize the concept of the evaluations. See Figure 4.6 for an illustration. The evaluations can be thought of as three different levels. Data with missing values was passed from step (1) to (2). In the second step, missing values were handled by either imputation or CCA. With imputations, the imputer could be evaluated directly (such as RMSE or accuracy) as a level one evaluation. The imputed data was then passed to a predictive model in step (3), in our case, Cox PH. For level two, it was possible to evaluate the imputations on a model based level, such as the influence on statistical parameters associated with the model. Finally, step (4) related the imputations to the output of the model, such as predictions. For level three, we could assess the influence imputations had on the predictions. In our experiment, accuracy was level one, bias was level two, and concordance was level three.

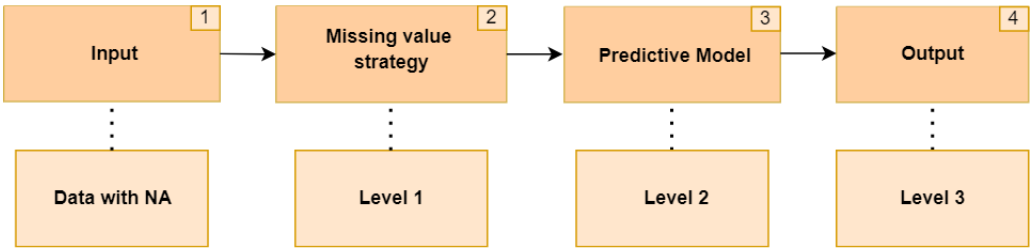


Figure 4.6: Evaluation pipeline for missing value experiment. The quality of the imputations was assessed on three levels. Level one was a direct evaluation of the imputation, level two was the evaluation of the model parameters and level 3 was the evaluation of the predictions.

4.3.4 Missing Value Generation

We will introduce how we artificially generated missing values for the colon cancer dataset. For the mechanisms, MCAR was straightforward. However, MAR was more intricate.

Simulating missingness with MCAR and MAR mechanisms was based on Muzellec et al. (2020).[99] Their utility function is available on GitHub.[100] The repository was cloned, and a few necessary modifications to the utility function file were made.

Firstmost, they did not have the functionality of MCAR. This was implemented where missingness was determined by the realization of a Bernoulli random variable. For each variable $1, \ldots, m$, there was a p probability of a sample being dropped. After simulating, it was expected for each variable to have approximately the same proportion of missing values. We denote the proportion of missing values as p_{miss}.

The MAR mechanism works as follows: a fixed proportion of randomly sampled columns were set aside and would not contain any missing values. Next, logistic models with random weights were fitted on the remaining variables using the non-missing variables as features. An intercept was added using a line search to ensure the desired proportion of missingness. The proportion of

missing values for MAR is also denoted as p_{miss}, and the proportion of retained columns is denoted p_{obs}.

Since the utilities file had no seed functionality, we added this. Two functions *set_sample_seed(seed)* and *set_column_seed(seed)* were defined to be able to control for the randomness in the columns and samples, respectively. The randomness of the samples refers to the missingness in the rows, and the randomness in columns is of interest when looking at MAR because of the possibility of reserving a proportion of columns that will not contain missingness. To be more specific about the latter, by setting a column seed, a (random) set of columns with proportion p_{obs} would not be simulated with missingness. By then changing the column seed, a different set of columns with the same proportion would be chosen.

The experiment was repeated many times for the same p_{miss} and p_{obs}, and therefore introduced two options for the MAR mechanism: either fix the column seed such that for each iteration, the same columns are retained, or vary the column seed such that new, possibly different columns with no missingness are introduced. The first choice is prone to an issue, namely, what columns are chosen (and what not). Because we have m number of variables in the dataset, we would be forced to set p_{obs} to 0 to have a fair exclusion of all the variables. However, this will not make sense as we require at least one variable to model the others. Thus, the smallest possible value of p_{obs} we can set is $\frac{1}{m}$. Then, we would expect to keep exactly one variable. However, this will introduce some bias in the sense that for the whole experiment, one column will never contain any missingness. The alternative choice is objectively better and is the one we applied. For each iteration, randomly vary the column seed so that potentially new variables are sampled. This will not introduce any bias in the column selections and also have the largest possible variance - the variation from selecting the columns, and the variation due to simulated missingness.

4.3.5 Pipeline for the Missing Value Experiment

The experiment was primarily segmented into two parts: MCAR and MAR. The major difference between MCAR and MAR, other than the missingness mechanism, was the ability for MAR to control for the proportion of columns to retain, i.e. columns that would not be simulated with missingness and instead be used for modelling the missingness in the remaining variables.

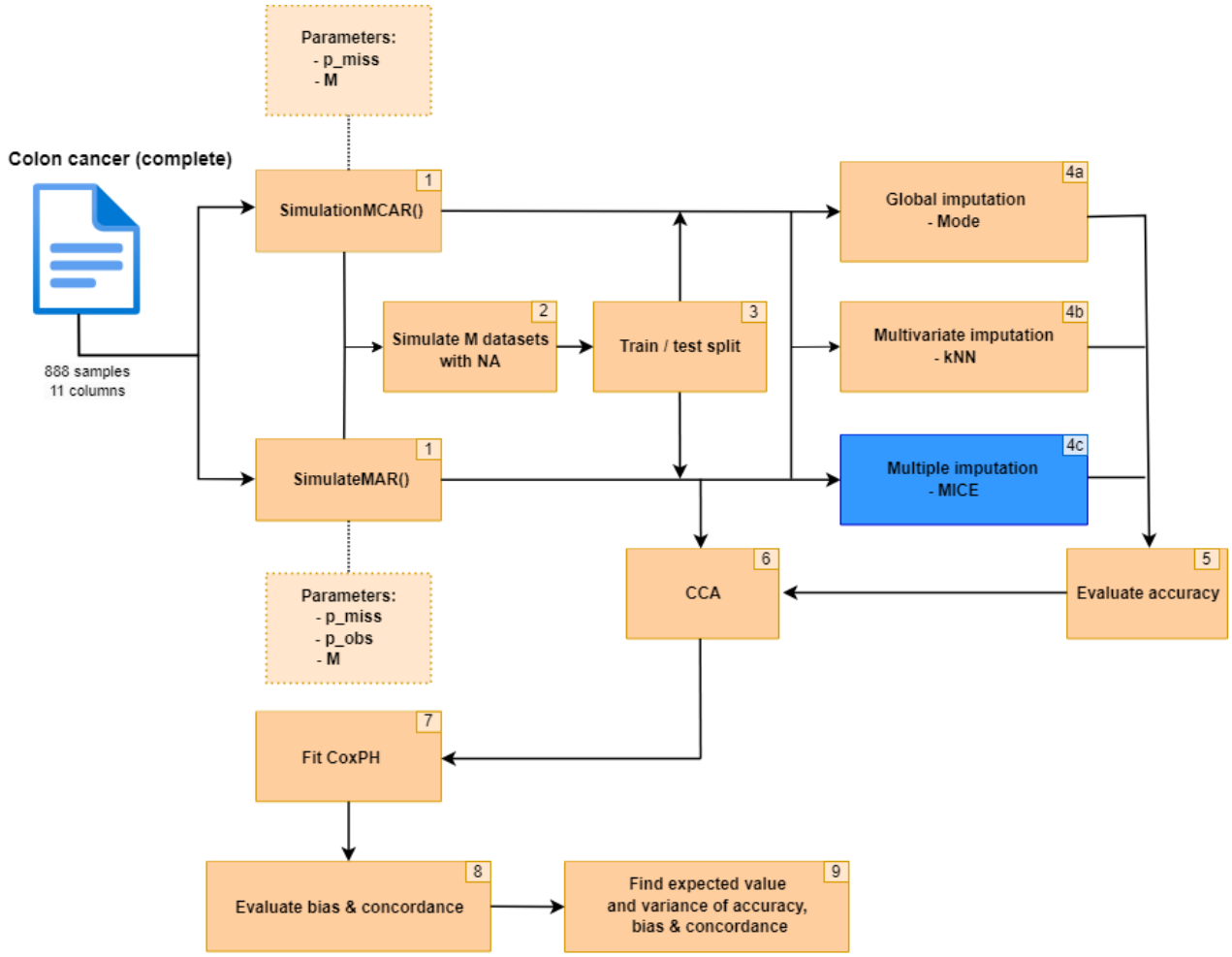


Figure 4.7: Illustration of the experimental setup of the missing values pipeline. The complete data is passed through steps (1)-(9), and step (1) determines whether MCAR or MAR missingness is being simulated.

The pipeline in Figure 4.7 was based primarily on two parent classes in Python: one for single imputation and one for multiple. Both have two subclasses, one for MCAR and one for MAR. The classes contain all the necessary methods for fitting and extracting information. For clarity, some intermediate steps such as temporarily converting categories to integers or Torch tensors to NumPy arrays are not shown.

The pipeline starts at (1) by taking in the colon cancer dataset, which contained no missing values and only categorical variables, by initializing either the MCAR or MAR subclass. The pipeline was designed in such a way that any dataset with only categorical variables can be passed in. The initialization also required the name of the time and status variables, as missingness was not simulated for these.

It is not present in the figure, but during initialization, a Cox PH model was fitted on the complete data where the weights and concordance index were stored. The weights were used as the proxy ground truth for calculating the bias defined in Eq. (4.1).

Next, (2) concerns the actual simulation of missingness. Here, all variables but *time* and *status* were passed to the *utilies.py* file M times to simulate missingness. The parameters passed to the

simulation were M (number of datasets with missingness to generate), p_miss (the proportion of missing values) and p_obs (only for MAR, the proportion of variables to retain). This yielded three tensors of dimensions $n \times m \times M$. The first tensor was the initial data (just repeated M times), the second was the same, but with missing values, and the third was a mask of booleans indicating if a value was missing or not. Each simulated missingness $1, \ldots, M$ were unique because the seeds vary as mentioned in Section 4.3.4.

After simulating the missingness, the tensors in (3) were split into train and test using 70% for train and the remaining 30% for test. Thus, we got six tensors, the three mentioned previously but for both train and test. We also kept track of the *time* and *status* corresponding to train and test that was omitted before simulating.

Next is the imputation steps (4a-c). All categorical variables were mapped to integers before imputing. As discussed in Section 3.7, this was not an ideal strategy for nominal variables and can give misleading imputations. However, there was no simple alternative. As we were interested in comparing the imputation with the corresponding ground truth value, we could not target encode the nominal categories as we would be unable to return to the original categories. Furthermore, all imputations were performed on the train set first and after on the test set using information from the fit on train. So, for example, considering mode imputation, the test set was imputed using the mode of the training set. The strategies applied are found in Table 4.6. For MICE, the tensors from (3) were saved locally and run in R-studio. For each of the simulated datasets $1, \ldots M$, MICE generated G imputed datasets and returned them to Python with dimensions $n \times m \times G \times M$ to proceed with the pipeline.

Table 4.7: Imputation functions in MICE. See table 4.7 for a description of these.

Variable	**Imputation model**
rx	polyreg
sex	logreg
obstruct	logreg
perfor	logreg
adhere	logreg
differ	polr
extent	polyreg
surg	logreg
nodes (binned)	polr

After all the imputations were complete, step (4) calculated the accuracy between the imputed and ground truth data tensors defined in Eq (3.6).

The next step in the pipeline is the comparison between imputation and removing samples. Step (6) did not impute anything but instead removed all samples that were missing. Note that because we wanted to measure the bias of coefficients for the Cox PH model, we did not remove any variables. We looked only at the effect of removing samples.

After applying all the strategies for handling the missing values (4a-c, 6), step (7) fits a Cox PH model on each of them. This was done using *CoxPHSurvivalAnalysis* from the scikit-survival package. An important consideration before model fitting was the appropriate encodings. The encodings were automatically decided during initialisation: categories with binary levels were one-hot encoded where the first level was dropped, ordinal categories were assumed to already be integers, and nominal categories were target encoded. The target encoding used the survival times as the target. In addition, the data was scaled after encoding. This was linked to the target encoding as it introduced variables of magnitude 10^3. The next consideration was matrix inversion problems. The pipeline was designed to flow automatically and, therefore, was required to handle this. The simplest solution we employed was attempting to fit a model without regularization but gradually increasing the regularization strength if the fit failed due to inversion problems.

With models fitted for all strategies, the next step (8) involves calculating Harrel's concordance

index and the coefficients' bias. For accuracy and concordance, we got estimates for both train and test. This was not the case for bias, as it is an estimate based on the coefficients tied to the fitted model.

Step (9) combines all estimates into a single point estimate, including the variations. For all the instances with mimicked missingness $1, \dots M$, we found the expected value as the arithmetic mean and the standard deviation as the population standard deviation. For multiple imputations, it was a little different. We found the arithmetic mean and standard deviation for each of the $1, \dots, G$ datasets (for each $1, \dots M$). To pool them into a single estimate for each of the M simulations, we found the expected value of both the mean and the standard deviation of the G datasets.

To make the pooling more clear, denote $\boldsymbol{\theta}$ as the parameter of interest (accuracy, bias or concordance). This is a vector of size M for single imputations and a matrix with dimensions $M \times G$ for multiple. The bias is an exception, as every variable has an associated bias. For simplicity, assume only one variable is being pooled. The strategy for pooling the bias is simply applied to each variable iteratively.

Pooling Single Imputation

For single imputations, the parameters were pooled across the M imputed datasets. Let μ denote the mean and σ the population standard deviation, they are given as

$$\mu(\boldsymbol{\theta}) = \frac{1}{M} \sum_{i=1}^{M} \boldsymbol{\theta}_i$$

$$\sigma(\boldsymbol{\theta}) = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (\boldsymbol{\theta}_i - \mu(\boldsymbol{\theta}))^2}$$

Pooling Multiple Imputation

For multiple imputations, the within mean $\mu^{(w)}$ and population standard deviation $\sigma^{(w)}$ of dataset index $i \in \{1, \dots, M\}$ were found in the G imputed datasets as follows

$$\mu^{(w)}(\boldsymbol{\theta}_i) = \frac{1}{G} \sum_{j=1}^{G} \boldsymbol{\theta}_{ij}$$

$$\sigma^{(w)}(\boldsymbol{\theta}_i) = \sqrt{\frac{1}{G} \sum_{j=1}^{G} (\boldsymbol{\theta}_{ij} - \mu^{(w)}(\boldsymbol{\theta}_i))^2}$$

then the outer mean $\mu^{(o)}$ and population standard deviation $\sigma^{(o)}$ was given by the arithmetic mean of these two estimates across the M datasets. These outer statistics were the ones reported.

4.3.6 Parameters and Hyperparameters Used for Simulation

We will briefly examine the parameter and hyperparameter selections for the simulation. For the parameters, we was concerned with p_{miss}, p_{obs} and M and G. For the hyperparameters, we only mention $n\_neighbors$.

First, consider p_{miss}, the parameter that decides the proportion of missing values to simulate. For all imputation strategies, we simulated missingness for the values $\{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$. For CCA, we only used $\{0.05, 0.10, 0.15, 0.20, 0.25\}$. The reason for limiting the missingness in CCA was because of the expected rows to keep under the MCAR mechanism. This was mentioned in Section 3.5, and we will illustrate now with numbers from our dataset. Recall that the original dimension of the colon cancer dataset was 888×9 (after removing event and duration). Next, after performing a 70/30 train/test split, we expected to keep 622 rows for the train. For an overview of the expected number of rows to keep after deleting entries with missing values under each proportion of missing values, refer to Table 4.8. In short, everything after 30% became infeasible to use.

Table 4.8: Expected number of training rows to keep for different p_{miss} under MCAR

Probability (p_{miss})	Formula	Num. rows kept
0.05	$(1-0.05)^9 \approx 63\%$	392 rows
0.1	$(1-0.1)^9 \approx 39\%$	240 rows
0.2	$(1-0.2)^9 \approx 13\%$	83 rows
0.3	$(1-0.3)^9 \approx 4\%$	25 rows
0.4	$(1-0.4)^9 \approx 1\%$	6 rows
0.5	$(1-0.5)^9 \approx 0.002\%$	1 row

It was not as easy to calculate the expected number for MAR because the variables were not independent. For consistency, we used the same proportions for both mechanisms. Next, we considered p_{obs}, the proportion of variables that would not contain missing values. Because we had 9 variables to simulate, a reasonable set of proportions was $\{\frac{8}{9}, \frac{7}{9}, \frac{6}{9}, \frac{5}{9}, \frac{4}{9}, \frac{3}{9}\}$. This guaranteed that the numerator was the number of variables to keep. It also gave a nice grid of 3×2 or 2×3 subplots for visualization, and was a fair proportion of columns to retain.

The choice of how many simulations to run, M, was arbitrary, and as M increased, the estimates become more precise. However, the simulation was computationally expensive, so we found that 50 was a suitable number of missing datasets to generate. Buuren (2018) suggests that the number of imputed datasets, G, to impute with MICE, should be as high as possible, though it requires more computational power and storage.[18] Anywhere between 5-20 was sufficient, but the range between 20-100 could be beneficial for higher-quality imputations. We set G to 50.

Finally, the choice of $n\_neighbors$ in kNN was 10. There is no fixed integer that works best. However, the default implementation for kNN imputer is 10 and 5 for R and Sklearn, respectively.[16,127]

4.4 Experimental Setup: Survival Analysis (GEP NEN)

4.4.1 Motivation

We introduce the important topics we considered in the survival analysis experiment. It was of interest to explore how data science can help to make better decisions for clinical patients. In particular, we wanted to address the challenges with the GEP NEN dataset, which had issues such as small sample size and large proportions of missing values, complicating our analysis. This involved thorough preprocessing to establish a good foundation for further analysis with assistance from expert knowledge. Our preprocessing pipeline addressed missing values and applied appropriate encodings for categorical variables. This was crucial for selecting the most effective models to predict outcomes in GEP NENs, aiming to make medical predictions more accurate and improve clinical decision-making.

The experimental setup for the survival analysis contained two main parts. The first involved developing a survival analysis pipeline that preprocessed the GEP NEN dataset and identified the most effective models for predicting outcomes. The second part evaluated various strategies (CCA, kNN, and MICE) for handling missing values to determine the most suitable method to apply in a survival model to the GEP NEN dataset.

The experimental setup for the survival analysis is described as follows: preparation of the GEP NENs dataset (Section 4.4.2, survival analysis pipeline (Section 4.4.3), hyperparameters used for the survival models (Section 4.4.4), pipeline methodology for CCA (Section 4.4.5), multiple imputations for survival analysis (Section 4.4.6), evaluating the imputations strategies used (Section 4.4.7) and interpretation of the survival models (Section 5.2.4).

4.4.2 Preparation of the GEP NEN Dataset

The preparation of the data contained several steps, illustrated in Figure 4.8. Duplicated data were removed and was not included in the new study's original dataset.

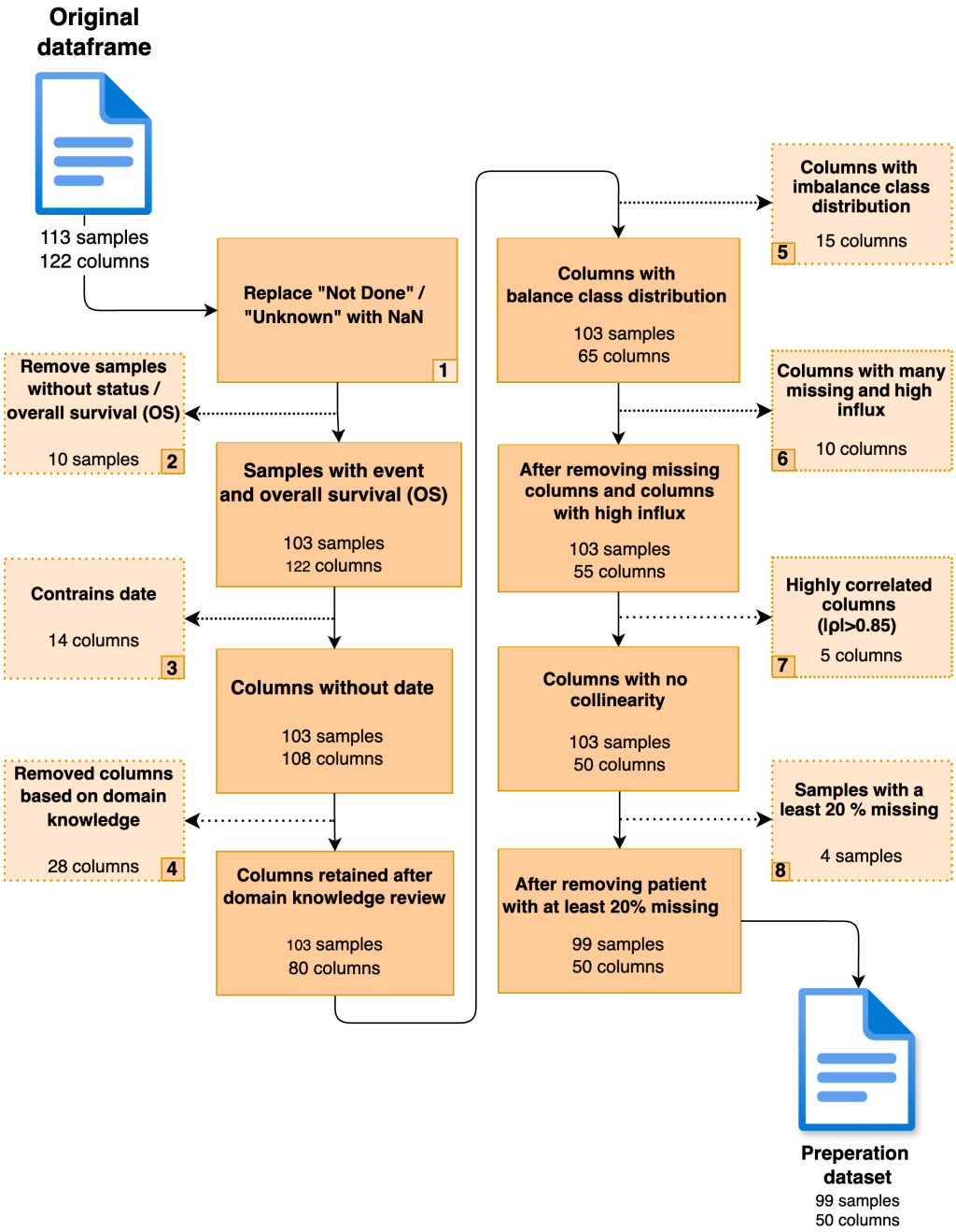


Figure 4.8: Pipeline for the preparation of the dataset for the new study.

(1) Replace 'Not Done'/'Unknown' with NA

The term 'Not Done' indicated that the outcome or status of the method or experiment was unclear or not established, which means that even though the activity may have been conducted, the results or details about it are not available or have not been disclosed. The difference between 'Not Done' and 'Unknown' lies in the stage of action and the availability of information. 'Not Done' explicitly indicated that the activity had not been initiated hence there were no results or data to report. 'Unknown' implied that the activity might have happened, but the results or details were missing or unclear, leaving a gap in our understanding. We decided to transform all values containing 'Not Done' or 'Unknown' into missing values represented by NA.

(2) Samples Without Status/Overall Survival

For survival analysis, it was important to have data for the target values, *status* and survival time, *OS (days)*. Nine patients were removed, as patients did not belong to any of the studies we examined. One patient had no information about the survival time and was removed. Thus, a total of 10 patients were removed in step 2.

(3) Columns with Dates

Since the overall survival (OS) calculation had been completed, there was no longer a need to use calendar dates, so they were removed from the analysis. This removal not only simplified the dataset by avoiding complications with date formats but also enhanced the privacy of the individuals by eliminating potentially identifiable information. A total of 14 columns were removed in step 3.

(4) Columns Based on Domain Knowledge

Based on expert knowledge, several variables were omitted as they would not be useful for the analysis. In total 28 columns were removed in step 4.

(5) Columns with Imbalanced Class Distributions

When conducting a survival analysis model with cross-validation, it was important to ensure that all categories within each feature were represented in both the training and testing folds. This was particularly challenging with imbalanced features, where certain categories might not be included in all the folds. For binary variables, imbalanced features could cause singular matrices when estimating parameters, and removing these made the model more stable. To solve this problem, we implemented a thresholding approach to exclude features that did not have all their categories represented in the test folds. The threshold was determined by iteratively testing different values until the minimum threshold was found that allowed the survival analysis model to run without matrix inversion problems. To quantify the imbalance across features, we calculated the imbalance ratio. This was done by dividing the occurrence count of the least frequent category by that of the most frequent category within each feature. The formula for the imbalance ratio was as follows:

$$\text{Imbalance Ratio} = \frac{\text{Count of Least Frequent Category}}{\text{Count of Most Frequent Category}}$$

The threshold was set at $\mathbf{threshold = 13}$. Features that had higher values than the imbalance ratio threshold of 13 were removed. In total 15 columns were removed as of imbalance class distribution in step 5.

(6) Columns with Large Proportions of Missingness, Low Influx and Outflux

Handling missing values was a critical preparation step to ensure the quality and reliability of survival analysis. Table 4.9 shows the top 20 columns with the highest percentage of missing values in descending order. Retaining variables with a high proportion of missing values could introduce bias, reducing the analysis' performance (Section 3). Moreover, it might weaken the model's predictive performance by affecting its ability to establish meaningful relationships between features. In preparation step 6, we also used methods of influx and outflux (Section 3.3) to assist with what columns should be removed.

Table 4.9: Percentage of NA for each Column. All features above the line are removed.

Column Name	NA (%)
Albumin Not Done	99.0
Toxicity Yes	99.0
CRP Not Done	99.0
ANC Not Done	99.0
Exploration Only, No Resection	91.0
Date of Brain Mets	90.0
Pack–years	85.0
OctreoScan	80.0
Resection	78.0
5–HIAA 24h Urine	72.0
CD–56	64.0
SRI	60.0
Octreoscan	55.0
Differentiation	41.0
Tumour Morphology	29.0
Chromogranin A2	28.0
Best Response (RECIST)	27.0
Co-morbidity Severity	27.0
Number of Courses	18.0
Treatment Stopped	18.0

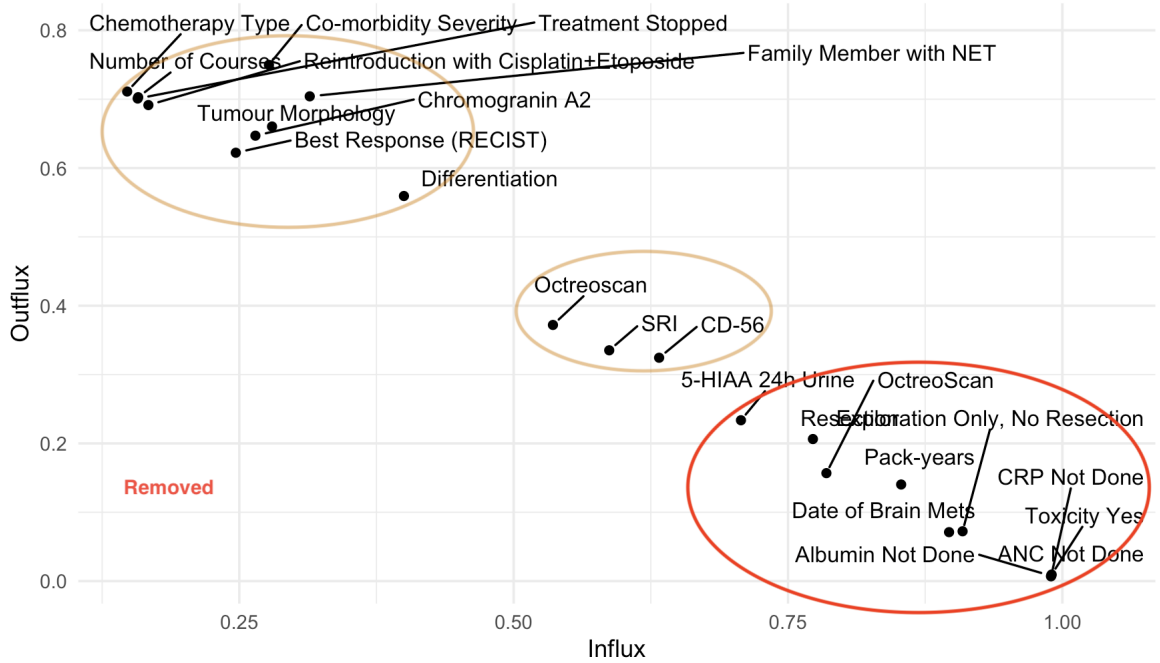


Figure 4.9: Influx and outflux of the variables in the GEP NEN dataset before removal variables marked in red.

Figure 4.9 shows the values of influx and outflux for features with lower outflux than 0.8. As discussed in Section 3.3, a high outflux was preferable because it indicates that a feature had a high potential to impute missing values in other variables. A high influx was also preferable, as it suggests that the feature may be easier to impute. The variables located higher up in Figure 4.9 was more complete and therefore potentially better to use for imputations.

If we removed all features with outflux lower than **0.30** and influx higher than **0.65** as detailed

in Table 4.9, these features would correspond to those removed under a threshold of **65%** missing values, as shown in Table 4.9. In conclusion, in step 7 in Figure 4.8, we removed 10 features due to many missing and high values for the influx and low outflux.

(7) Highly Correlated Columns ($|\rho| > 0.85$)

Highly correlated columns in a dataset signify that some columns share a strong linear relationship, often leading to redundancy. This can complicate analysis and predictive modelling by introducing multicollinearity, which may skew results.[145] The correlation matrix shown in Figure 4.10 was a powerful tool for identifying the strength and direction of linear relationships between the columns, showing those with an absolute correlation exceeding **0.85**. When deciding which columns to remove based on these correlations, we chose to keep the column with fewer missing values. As a result, the columns *M-stage*, *TNM-staging_NEC*, *Time from diag to mets (months)*, *SRI*, and *Age at Death* were removed. Five columns were removed during step 7, as shown in Figure 4.8.

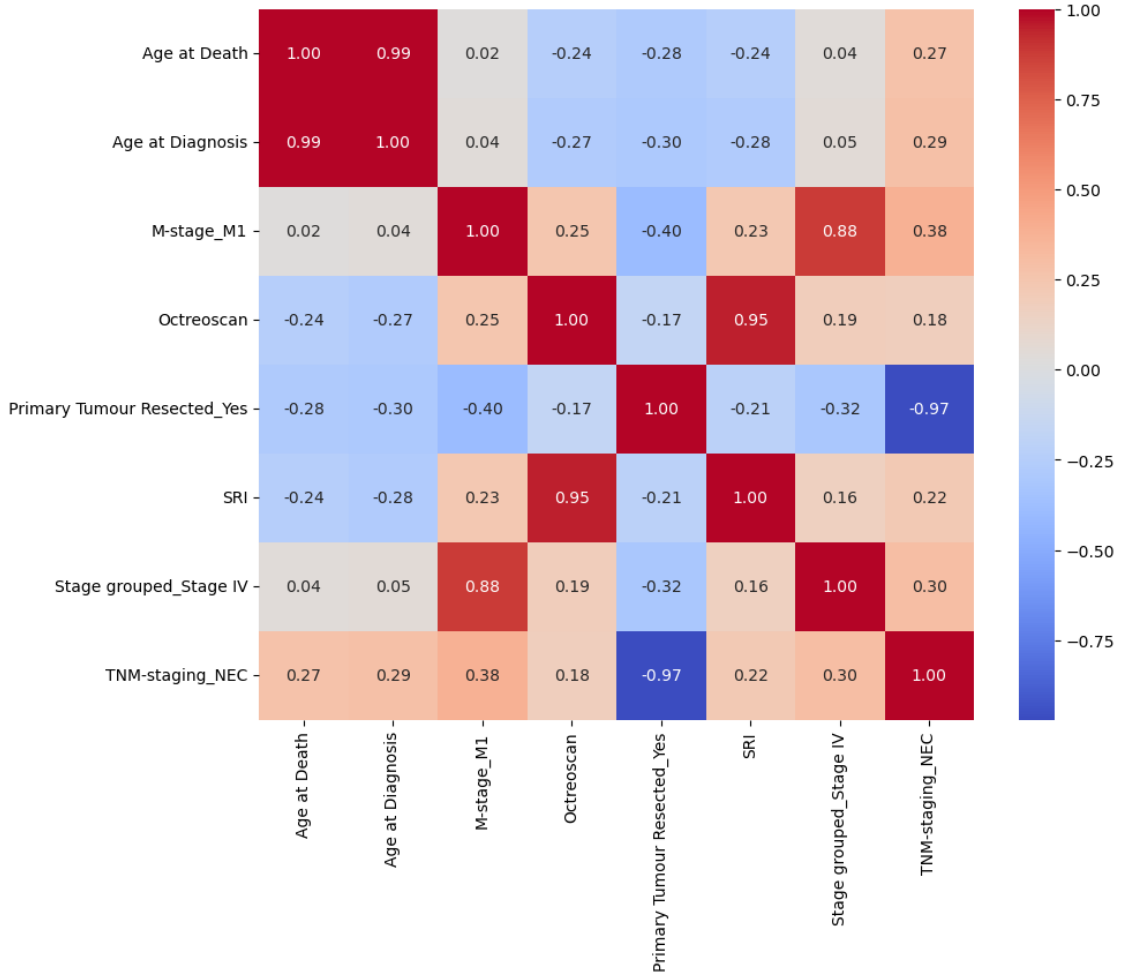


Figure 4.10: Correlation matrix for columns with absolute correlation higher than **0.85**. Keep the column with fewer missing values for each correlation pair.

(8) Removing Samples with Many Missing Values

In addition to removing variables with high proportions missing values, it was important to remove samples with large proportions of missingness. The samples with missingness exceeding **20 %** were removed. Table 4.10 details the samples with the highest percentages of missing values; all samples above the line were removed. In total, four samples were omitted.

Table 4.10: Number of missing values (NA (%)) for each patient (PATNO).

PATNO	NA (%)
9063	70.0
9012	68.0
9020	36.0
9010	36.0
9098	20.0
9120	18.0
9039	18.0
9060	16.0
9033	16.0
9056	16.0

The original dataframe consisted of 113 samples and 122 columns. After preparation, the dataset was reduced to 99 samples and 50 features.

4.4.3 Survival Analysis Pipeline

Figure 4.11 illustrates the pipeline methodology used across various survival models, including Coxnet, Cox PH, RSF, and CGB.

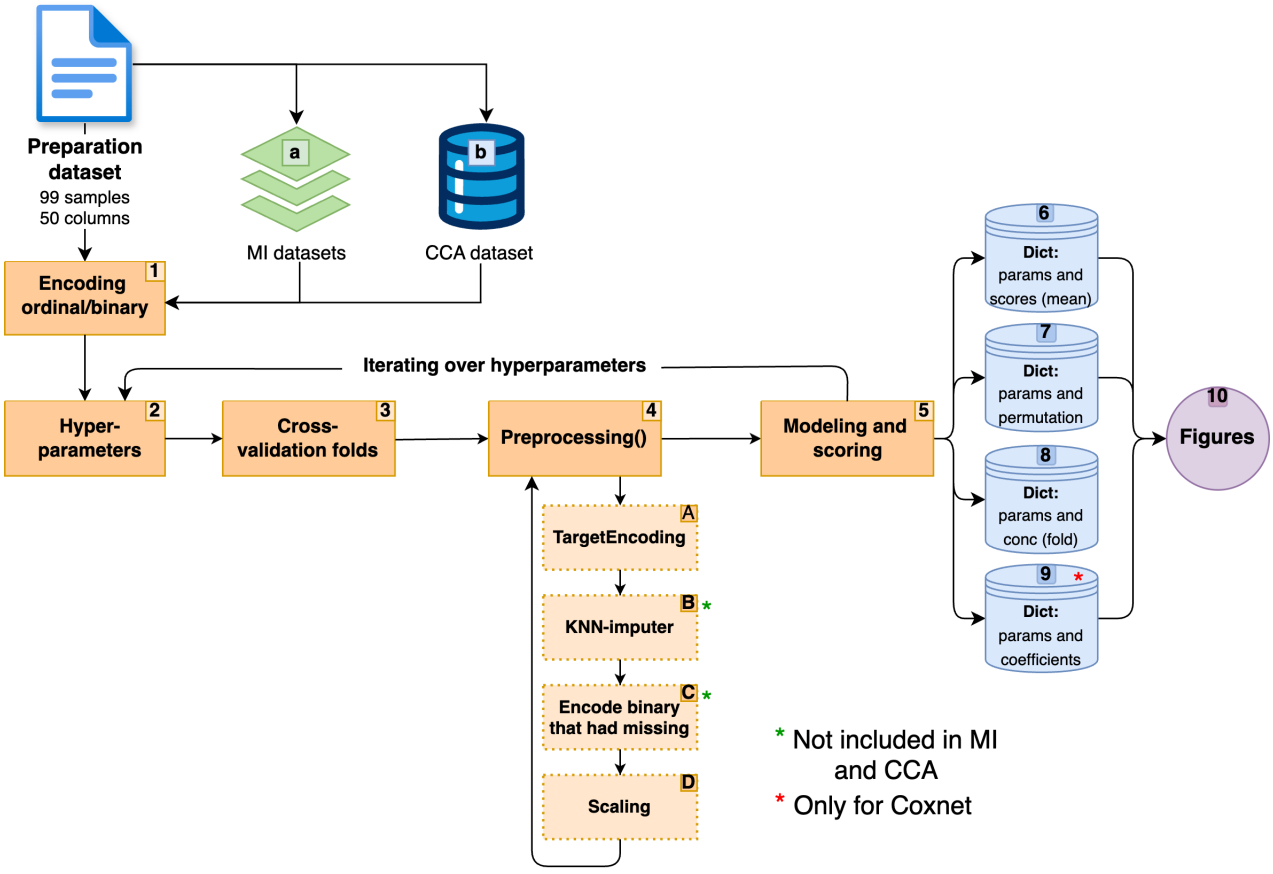


Figure 4.11: Survival analysis pipeline methodology. Used the prepared dataset to generate MI datasets and the CCA dataset. The pipeline involves encoding categories, kNN-imputation, and hyperparameter tuning. After these steps, all scores and coefficient values are organized into four dictionaries. These four outcomes are used to plot figures.

After the preparation, the resulting dataset consisted of 99 samples and 50 columns. The prepared dataset was used to generate MI datasets (a) and the CCA dataset (b). The survival

analysis pipeline can be utilized for all three datasets. However, steps 4B and 4C were not applied to the MI datasets (a) and the CCA dataset (b) as they did not include any missing values.

The first step (1) involved encoding the ordinal and binary categorical variables. All binary categorical variables without missing values were encoded using the Python function `get_dummies` by the Pandas library to one-hot encode (Section 2.5.1). For binary categorical variables with missing values and ordinal variables, we manually encoded them to integers temporarily using the `map` function in Pandas. Thus, after step (1), all binary and ordinal variables had been encoded.

In step (2), the pipeline entered the hyperparameters tuning. Various hyperparameters were defined and iterated to optimize the model´s performance. The step was creating cross-validation folds (3), where the dataset was partitioned into multiple folds. We used the RSKF (Section 2.4.1), with 5 splits and 5 repetitions, resulting in 25 folds. In every fold, the dataset was divided into train and test. These divisions were utilized throughout the remainder of the pipeline.

When calculating the Brier score, it was essential to ensure the survival times for test data estimation (slicing the time) did not exceed the maximum survival times observed in the train set. If any test times were longer than max training time, they had to be truncated. Otherwise, it was not possible to correctly determine individual survival probabilities.

Step (4) used the function called `Preprocessing()` within each fold of the cross-validation and conducted a series of four steps (4A-D). The first step (A) used a target encoder with the function `MultiTargetEncoder`, based on *sklearn*'s target encoder, to encode nominal variables with more than two categories. The default target encoder did not ignore the missing values but handled them by replacing them with another category (Section 2.5.1). Therefore, employing our `MultiTargetEncoder` ensured that missing values were ignored during the encoding process. Using the target encoder, our target variable, survival time, which is a continuous variable, was used as the target value when calculating the encoding for the nominal variables.

After step (A), where all variables had been encoded, the next step involved imputing missing values (B) with kNN imputation (Section 3.6.2). Prior to imputing, variables were required to be scaled (Section 2.5.3). In the next step (C), binary variables that originally contained missing values prior to step (2) were rounded to the nearest integer, resulting in either 0 or 1. These integers were mapped back to their original categories before being one-hot encoded. This was done as they had now been imputed. In the final step (D), the entire dataset, both the train and test folds, was scaled using standard scaling (Section 2.5.3).

After completing step (4), we obtained a fully preprocessed dataset. In the modeling and scoring step (5), the machine learning model was trained using preprocessed data, and its performance was evaluated using the survival models scoring metrics Brier Score, IBS and Harrell's C-index (Section 2.4.3 and 2.4.2). The outcomes of the survival models, including their respective parameters, were organized into four dictionaries (right side, Figure 4.11). In the dictionary (6), the average scores for Brier score, IBS and Harrell's C-index across all folds were compiled for each set of hyperparameters. The dictionary (7) encompassed all permutations (Section 2.5.4) of variables for each fold alongside their corresponding hyperparameters. The dictionary (8) aggregated Harrell's C-index scores for each fold under each hyperparameter. The final dictionary (9) included all coefficients for each variable across each fold alongside their corresponding hyperparameters.

These outcomes of the survival models (6-9) were used to create Figures (10) that are further described in Section 4.4.8 in Figure 4.13. All four survival models used dictionaries (6-8) in Figure 4.11, and only the Coxent model used the dictionary (9).

4.4.4 Hyperparameters Used for the Survival Models

We will explain the hyperparameter used in each survival model.

Coxnet

For the Coxnet algorithm (Section 2.3.2), the hyperparameters optimized were *alpha* and *L1 ratio* as shown in Table 4.11. *Alpha* controls the strength of the regularization. In contrast, the *L1 ratio* controls the balance between L1 and L2 regularization. The *alpha* values we used ranged from 0 to 1000, with 19 different values within this interval, totaling 21 *alpha* values. This range encompassed both low and high regularization strengths, ensuring exploration of the model's behavior across various scenarios. As for the *L1 ratio* was set to 0.0001, 0.001 and in addition from 0.1 to 0.9 in increment of 0.1, totaling 12 *L1 ratios*. This selection was intended to explore the influence of both sparse and dense models on the performance.

Table 4.11: Hyperparameter values used in the Coxnet model in the GEP NEN dataset.

Parameter	**Values**
Alpha	0, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 3, 5, 10, 20, 50, 70, 100, 200, 500, 700, 1000
L1 Ratio	0.0001, 0.001, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9

Cox Proportional Hazards

In the Cox PH algorithm, *alpha* served as the only hyperparameter as shown in Table 4.12, with the identical set of *alpha* values used as in the Coxnet algorithm.

Table 4.12: Hyperparameter values used in the CoxPH model in the GEP NEN dataset.

Parameter	**Values**
Alpha	0, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 3, 5, 10, 20, 50, 70, 100, 200, 500, 700, 1000

Random Survival Forest

In the RSF algorithm (Section 2.3.3), four hyperparameters were tuned as shown in Table 4.13: *n_estimators*, *max_depth*, *min_samples_split*, and *min_samples_leaf*. The *n_estimators* parameter, which determines the number of survival trees, was tuned from 10 to 50 in increments of 10. The *max_depth* of the survival trees influenced the complexity of the model. A deeper tree can model more complex patterns but might lead to overfitting. Tuning from 2 to 5 allowed us to explore a range from simpler to moderately complex models, aiming to find the optimal model for generalization.

min_samples_split is the minimum number of samples required to split an internal node. Higher values of *min_samples_split* prevent the model from learning specific patterns, thus reducing overfitting. Values of 2, 6, and 8 were chosen for *min_samples_split*. We also tuned the *min_samples_leaf* parameter, which determines the minimum number of samples a leaf node must have, ranging from 1 to 5. This hyperparameter aims to prevent the model from capturing noise in the train data, thus improving its predictive stability.

Table 4.13: Hyperparameter values used in the RSF model in the GEP NEN dataset.

Parameter	Values
n_estimators	10, 20, 30, 40, 50
max_depth	2, 3, 4, 5
min_samples_split	2, 6, 8
min_samples_leaf	1, 2, 3, 4, 5

Component-Wise Gradient Boosting

In the Component-wise gradient boosting algorithm (Section 2.3.4), three hyperparameters were tuned as shown in Table 4.14: *n_estimator*, *learning_rate*, *subsample*. The *n_estimator* was tuned with the same identical set of *n_estimator* values used as in the RSF model. The *learning_rate* and *subsample* were tuned from 0.1 to 1 in increments of 0.1. Tuning the *learning_rate* affects how a slower or faster learning rate affects the model's performance. For the *subsample*, we want to tune the whole range where it occurs to achieve the optimal model. Additionally, this lets us explore the effect of stochastic gradient boosting (*subsample*$<$ 1.0) by varying the degree of randomness in the training process.

Table 4.14: Hyperparameter values used in the CGB model in the GEP NEN dataset.

Parameter	Values
n_estimator	10, 20, 30, 40, 50
learning_rate	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0
subsample	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0

4.4.5 Pipeline Methodology for CCA

CCA was a dataset without using any imputation method. Instead, it leaves out columns and rows with missing values from the preparation dataset as illustrated in Figure 4.11. Initially, the number of columns with the highest proportion of missing values was removed, followed by the removal of samples containing missingness. To create the highest-performed CCA dataset, we introduce a hyperparameter that establishes a threshold for the percentage of missing values a column can have before removal. For instance, setting the threshold to 3 implies that any column with more than 3% missing values will be removed. By integrating this additional hyperparameter, the model explores performance across various CCA datasets, ultimately identifying the best-case CCA dataset. The hyperparameter that produced the highest-performing dataset was consistently set to **3** across all four survival models. The columns removed under this hyperparameter setting were shown in Table 4.15, along with their respective percentages of missing values. The pipeline methodology was the same as for multiple imputation datasets where 4(B-C) was ignored, illustrated in Figure 4.11.

Table 4.15: Columns removed from the CCA dataset based on a hyperparameter threshold of 3% missing values, showing the percentage of data missing for each column.

Variable	**NA (%)**
CD-56	62.62
Octreoscan	53.53
Differentiation	40.40
Co-morbidity Severity	28.28
Tumour Morphology	27.27
Chromogranin A2	26.26
Best Response (RECIST)	24.24
Treatment Stopped	15.15
Number of Courses	15.15
Chemotherapy Type	14.14
Smoking	10.10
NSE	7.07
Chromogranin A	7.07
Synaptophysin	7.07
Dev of Bone Mets	7.07
ALP	5.05
WHO Perf Stat	5.05
LDH	4.04
N-stage	3.03
T-stage	3.03
BMI	3.03

4.4.6 Multiple Imputation

Figure 4.11 shows the pipeline for CCA, kNN and multiple imputations. The difference between them was that for multiple imputations, the imputed datasets in (a) were passed to step (1) instead of the preparation dataset, and steps (4B-C) were skipped as they were already imputed.

First, multiple imputation was done in R using the *mice* package (Section 3.6.3) using the Preparation dataset in Figure 4.11. The pipeline was repeatedly applied to each of the $1, \ldots, G$ imputed datasets. As multiple datasets were generated instead of one, Rubin's rules were used to pool parameters to point estimates. We also generated a model-based imputation using MICE, where only one imputed dataset was generated. This was included to assess the performance of mimicking single imputations with the benefit MICE has by the changed equations.

Multiple imputations were carefully applied, following most of the approaches recommended by the software author and this book.[18] They approached the implementation from a statistical point of view as the goal was to make good imputations. They suggest including all available information, such as the target variable and the entire dataset - not just a subset. They only recently added the functionality of train/test imputations.[19]

This meant two things. Firstly, we did not use MICE's train/test imputation feature. In the first iteration, we tried this. However, the test imputations were poor and gave very low concordance, but it also posed a technical limitation. The target encoding in Figure 4.11 (step 4A) was conducted within each cross-validation fold, and there was no feasible way to impute using only information from each fold in R, and then somehow refer back to those folds in Python. We investigated Python implementations of MICE, but most of them lacked the elements required for our application. An example of one we tried was Autoimpute, a popular Python variant with

numerous imputation strategies.[72]. The problem with this software is that it was very strict with categorical data. For example, if a variable was fitted to a category with three levels, but only two showed up in the test set, it would fail.

We avoided using the variables representing status and survival times for imputation. We were interested in seeing how well imputations perform without introducing too much data leakage to make the multiple imputations comparable to the kNN imputations.

For the specifics of the actual imputations, we can spare some reading by not listing all the imputation functions used for each variable. Instead, refer to Table 4.16 for an overview of the imputation functions used for each variable's datatype. The variables was in detail described in Tables: 4.2 (numerical), 4.3 (categorical, ordinal), 4.4 (categorical, binary) and 4.5 (categorical, nominal). In addition to tailoring the imputation functions to each variable, we also tried imputing using only PMM.

Table 4.16: Imputation functions used in MICE depending on the variable datatypes. See table 3.6 for MICE details.

Variable	**Imputation model**
Numerical	pmm
Factor, ordinal	polr
Factor, nominal	polyreg
Factor, binary	logreg

Note that all the numerical variables were floats but rounded down. Thus, we found that PMM was appropriate. Alternatives would be one of the continuous models, such as linear regression. The latter model has accompanying assumptions about the residuals, like being independent, Gaussian distributed and with constant variance.[118] Verifying this would be difficult.

The next step before doing imputations was a variable selection of predictors. Variables with too many missing values can do more harm than good. The preprocessing step in Section 4.4.2 dealt with this by looking at the proportions of missingness, along with influx and outflux. Next, we investigated the predictor matrix used for imputations. This is a binary $m \times m$ matrix indicating what predictors (columns) should be used for imputing (rows). It was important not to use all variables as predictors as the occurrence of collinearity would be more likely, i.e. one or more variables explain more or less the same information, which can trouble the imputations. The final concern was the degrees of freedom available for the models. This was how many samples were independent of the estimated parameters, and it relates to how much information was left for imputing. Nonparametric imputation functions, such as PMM, did not suffer from this. However, all the others we listed do. In fact, by including all the predictors, we ended up with negative degrees of freedom. We found it sensible to filter the variables such that they required at least 30% absolute correlation with the imputation target to be used.

4.4.7 Evaluating the Strategies: A Statistical Approach

We used a t-test to determine if there was a significant difference between the means of the model performances after applying each missing values strategy (CCA, kNN, MICE). Since we used RSKF, the t-test analyzed a list of C-index scores for the testing set to determine whether there was a significant difference. We verified the normality assumptions, and all the groups compared were reasonably normally distributed. Figure 4.12 illustrates the methodology for retrieving the C-index list for the highest-performing model. The process involved selecting the hyperparameters that yielded the highest C-index in the test phase of the model as part (6) in Figure 4.11. Subsequently, these parameters were used to retrieve the C-index for each fold as part (8) in Figure 4.11 that corresponds to these parameters.

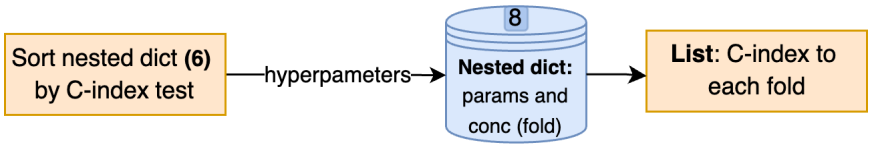


Figure 4.12: Comparison methodology to find the list of C-index used in the t-test.

First, kNN imputation was compared with CCA. This comparison was conducted across various survival analysis models, including Coxnet, Cox PH, RSF, and CGB. Subsequently, we identified the superior model from these based on its performance with either kNN imputation or CCA. This high-performance model was then used to evaluate whether the application of the model-based imputations method (MI with only one imputed dataset) resulted in any significant differences. Lastly, we evaluated whether the application of MI also resulted in any significant difference. An overview of all the comparisons conducted using the t-test is illustrated in Table 4.17.

Table 4.17: t-test comparison between KNN, CCA, Model-based, and MI.

Comparison: t-test	**Survival Model**
kNN vs CCA	Coxnet
	CoxPH
	RSF
	CGB
kNN vs Model-based	Coxnet
kNN vs MI	Coxnet

Since MI generated multiple datasets, obtaining a list of C-indexes was more complex. The process began by executing the pipeline shown in Figure 4.11 for each MI dataset. The next step was to identify the model on each dataset with the highest C-index during the testing folds. The hyperparameter combinations that were most common across all datasets were then selected to rerun the pipeline in Figure 4.11, but now applied to all the datasets. Finally, to correctly collect the list of C-indexes, we calculated the mean of the C-index values from all the datasets.

4.4.8 Visualization of Survival Model Performance and Evaluation

Figure 4.11 shows plots that visualized the operation of different survival models executed in the pipeline. These plots were created for the imputation method that gave the highest performance of the model. All the plots, 10(a-f), are illustrated in Figure 4.13.

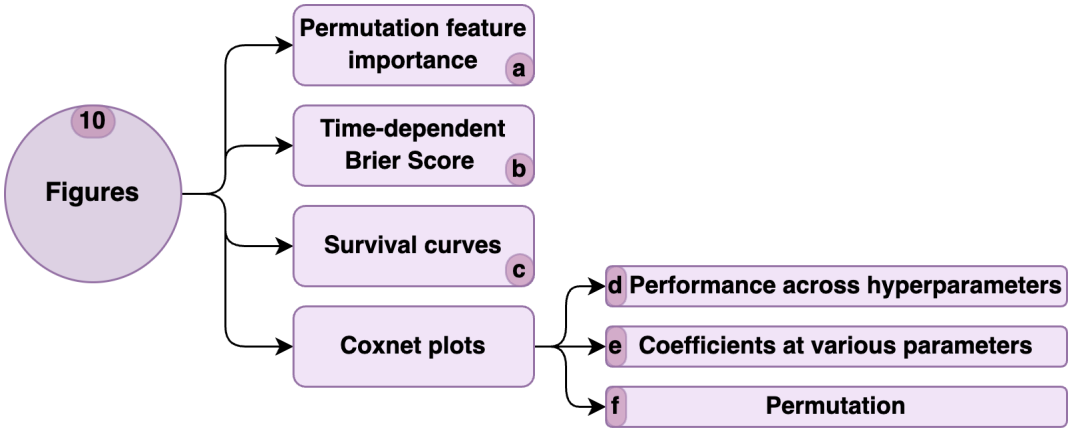


Figure 4.13: Figure overview: figures (a-c) used all survival models, while figures (d-f) exclusively for the Coxnet model.

(10a) Permutation Feature Importance

The permutation feature importance (10a) (Section 2.5.4), as shown in Figure 4.13, was plotted across all four survival models using the dictionary (7) illustrated in Figure 4.11. Since we used RSKF, it was necessary to average the feature importance values for each feature across all folds. These averages were then ranked by their highest values to identify the most important features. The average feature important across all survival models was also calculated.

(10b) Time-Dependent Brier Score

The time-dependent Brier score plot (10b) (2.4.3) in Figure 4.13 used the Brier score at each survival time for each survival model. Since we used RSKF, the stratified k-fold was repeated five times, which implied that the maximum time point in the train folds varied. Therefore, the times from each set of five folds were adjusted by truncating the minimum length observed among them. Then, Brier scores from these adjusted time points were averaged and plotted. The average of these five-folds was plotted along with the standard deviation to show the overall Brier score and variability.

(10c) Survival Curves

Two survival curves were generated for four selected patients, one for single imputations and one for multiple imputations.

The first was a survival curve in the testing fold using a RSKF method with five folds and five repetitions. Consequently, each patient was included in the testing set five times, and each patient´s survival probability was plotted five times. The mean and standard deviation of the survival probabilities were computed across all these five survival probabilities.

The second was a survival curve in the testing fold using stratified k-fold, for 50 imputed datasets with MICE. This implied that each patient was in the test set once for each of the 50 imputed datasets. The mean and standard deviation of the survival probabilities were computed across all 50 survival probabilities.

Both survival curves' observed and expected survival times were also plotted. The expected survival time was computed by discrete integration of the mean survival function over time.

(10d-f) Coxnet Plots

The performance of the Coxnet model across various *alpha* values and *L1 ratios* (10d) was plotted using dictionaries (6) and (9) as shown in Figure 4.11. Detailed model performance metrics were presented in the dictionary (6), while information regarding the coefficients of each model feature was found in the dictionary (9). To determine the features eliminated through feature selection, we evaluated the coefficients: features with a coefficient of zero were identified as removed. This approach allowed us to visualize the impact of different hyperparameters, *alphas* and *L1 ratios*, on both the feature selection process and the performance metrics.

To observe how the coefficient of a feature and performance metrics varies with different regularization parameters (10e), we used the dictionaries (6) and (9) illustrated in Figure 4.11. Permutation feature importance (Section 2.5.4) was computed similarly to (10a) in Figure 4.13, but specifically for the Coxnet model.

5 Results

5.1 Experimental Setup: Missing Values (Colon)

This section covers the results from the missing value experiments on the colon dataset described in Section 4.3. We investigated the effects of various strategies applied to the MCAR and MAR missingness mechanisms for increasing proportions of missing values. The three effects we measured were accuracy, concordance and bias of coefficients of a Cox PH model.

Representative results are shown, due to the large amount of figures. The train and test results were approximately the same if both are not shown. More figures are found in Appendix A.

5.1.1 Runtimes of Missing Value Experiment

The runtimes for each strategy applied to the missing value experiment are given in Figure 5.1. MAR generally had longer runtimes as the extra parameter p_{obs} led to an additional loop to iterate through.

Table 5.1: Runtimes of the strategies applied to the simulated missingness in the colon dataset. This was run on machine (1) (Section 4.2.2).

Strategy	Runtime (MCAR)	Runtime (MAR)
Mode	32s	3m 58s
CCA	16s	2m 38s
kNN	49s	4m 47s
MICE	48m (R) + 5m (Python)	1h 28m (R)+ 28m (Python)

5.1.2 The Cox Proportional Hazards Model

The two evaluations, bias of the coefficients and concordance, were related to a Cox PH model. Table 5.2 shows the proxy ground truth Cox PH model fitted on the colon dataset with no missing values. It was fitted using the same pipeline as for the ones with simulated missing values (scaling, encoding, etc.). The coefficients *coef* were used for calculating the bias in Eq. (4.1). The variables *nodes* and *extent* had coefficients significantly different from 0 at a 5% rejection level. The concordance was 0.66.

Table 5.2: Ground truth Cox PH model fitted on the colon dataset before simulating missingness. Coef shows the proxy ground truth coefficients, with standard deviation (se), upper 95% confidence intervals (coef lower 95%) and lower 95% confidence intervals (coef upper 95%) are also shown. The z-statistic (z) and p-value (p) with $H_0 : coef = 0$ vs $H_A : coef \neq 0$ is provided. The c-index (concordance) is also included.

	coef	se(coef)	coef lower 95%	coef upper 95%	z	p
nodes	0.44	0.05	0.34	0.53	9.06	< 0.005
rx	-0.08	0.05	-0.18	0.02	-1.60	0.11
differ	0.09	0.05	-0.02	0.19	1.66	0.10
extent	-0.16	0.05	-0.27	-0.05	-2.94	< 0.005
sex_M	-0.01	0.05	-0.11	0.09	-0.31	0.76
obstruct_Y	0.09	0.05	-0.00	0.18	1.89	0.06
perfor_Y	-0.03	0.05	-0.12	0.05	-0.54	0.59
adhere_Y	0.09	0.05	-0.00	0.19	1.86	0.06
surg_S	-0.11	0.05	-0.20	-0.02	-2.32	0.02
concordance					0.66	

5.1.3 Missing Completely at Random (MCAR)

Figure 5.1 shows the training accuracy after applying all the imputation strategies under MCAR, for increasing proportions of missing values. kNN and MICE were the least affected by increasing the missingness. kNN and mode imputations had the highest and smallest overall accuracy, respectively. The latter performed slightly worse when the proportion of missingness increased. The performance of MICE was close to mode imputation.

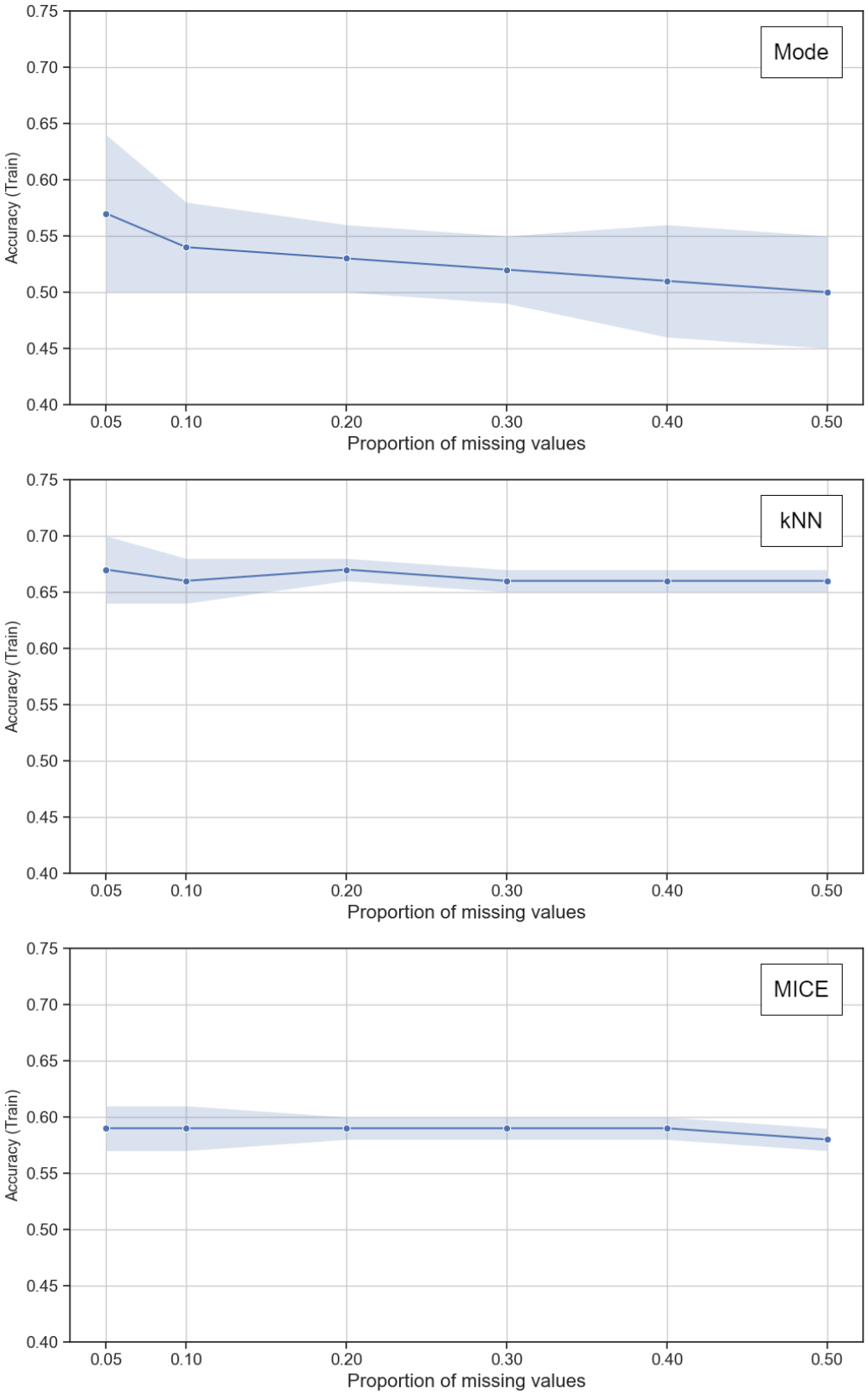


Figure 5.1: Train accuracy as a function of the proportion of missing values using mode, kNN and MICE imputations on simulated missingness (MCAR) in the colon dataset, where missingness was simulated 50 times. The standard deviations (shaded area) and averages (solid line) are shown.

The bias of the Cox PH model coefficients after applying all strategies, except for mode, is shown in Figure 5.2. For mode, see Appendix A.1. CCA had the highest bias in *perfor_Y* for simulated 25% missingness. For the same proportion, the variables *sex_M* and *adhere_Y* followed with a large standard deviation of 24 and 11, respectively. The heatmaps of kNN and MICE may be misrepresentative in comparison to CCA, due to the scale of the colors for CCA. MICE and kNN had approximately the same bias in most variables, with MICE having slightly lower standard deviations. Both corrected the bias of *perfor_Y* as their maximum bias were 1.08 ± 2.14 for kNN, and 1.21 ± 1.93 for MICE.

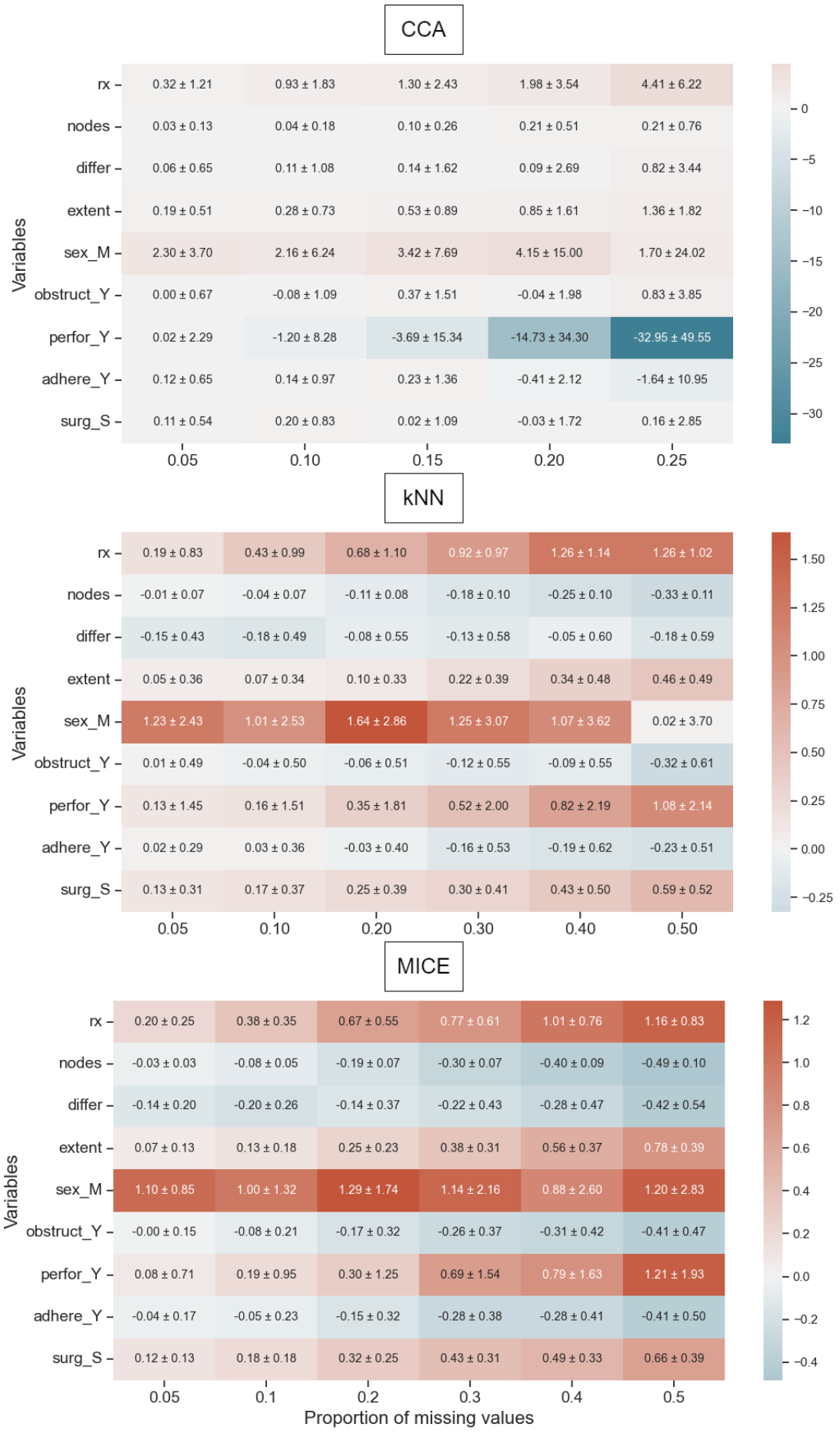


Figure 5.2: Bias of coefficients from a Cox PH model of the variables as a function of the proportion of missing values after applying CCA, kNN and MICE on simulated missingness (MCAR) in the colon dataset, where missingness was simulated 50 times. The whitest colour implies no bias, while blue and red lean towards negative and positive bias, respectively.

The train and test concordance after applying CCA under MCAR, when increasing the proportions of missing values, gave a larger discrepancy between the curves as the proportion of missingness increased, as shown in Figure 5.3. There was an increasingly larger standard deviation in the test concordance.

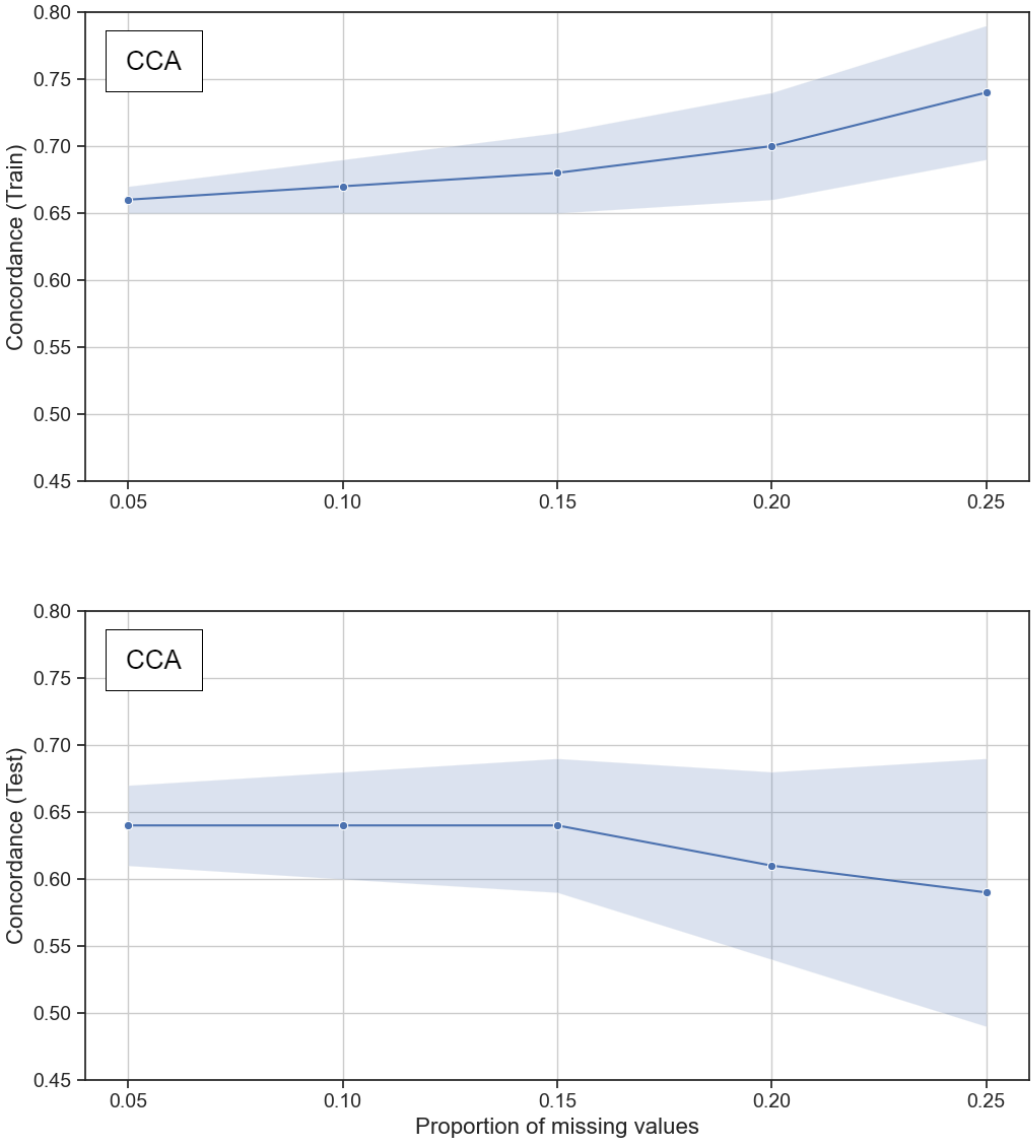


Figure 5.3: Train concordance as a function of the proportion of missing values after applying CCA to simulated missingness (MCAR) in the colon dataset, where missingness was simulated 50 times. The standard deviations (shaded area) and averages (solid line) are shown.

For all the imputation strategies, the concordance decreased as the proportion of missingness increased. See Figure 5.4 for kNN, and 5.5 for MICE. All imputation strategies, for both train and test, had approximately the same descent. MICE had the smallest standard deviation. Mode imputation is found in Appendix A.2.

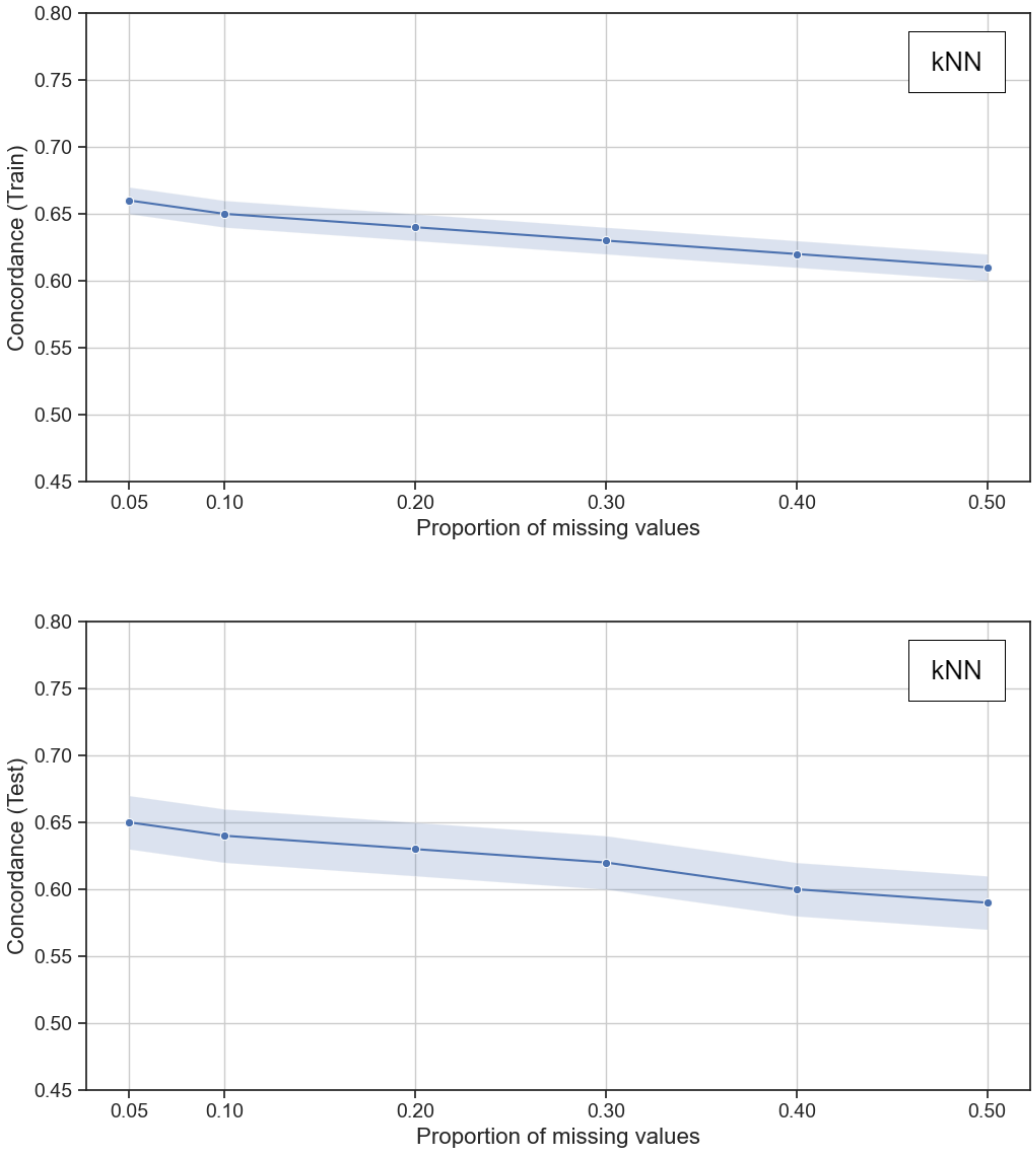


Figure 5.4: Train and test concordance as a function of the proportion of missing values after applying kNN imputation on simulated missingness (MCAR) in the colon dataset, where missingness was simulated 50 times. The standard deviations (shaded area) and averages (solid line) are shown.

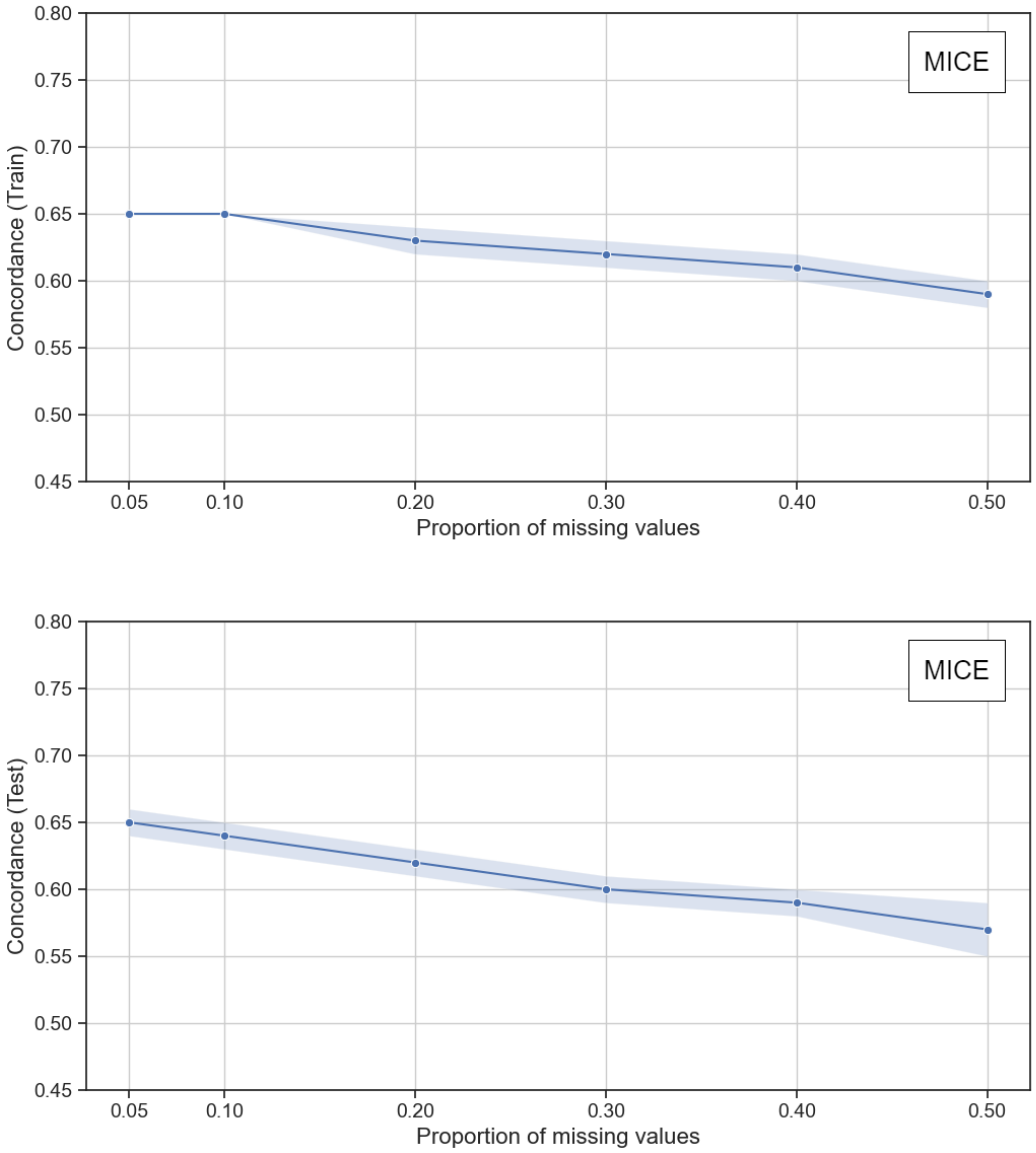


Figure 5.5: Train and test concordance as a function of the proportion of missing values after applying MICE imputation on simulated missingness (MCAR) in the colon dataset, where missingness was simulated 50 times. The standard deviations (shaded area) and averages (solid line) are shown.

5.1.4 Missing at Random (MAR)

For MAR, we investigate strategies that consider the relationships between the variables, such as MICE.

See Figure 5.6 for the accuracy after applying kNN and MICE iputations under MAR, for increasing proportions of missing values. We selected only the figures showing the proportion of observed variables (p_obs) for $\frac{8}{9}$ and $\frac{3}{9}$. This was because it was a general trend that the standard deviation decreased when p_obs decreased. For the complete figures, see Appendix A.3 for kNN and A.4 for MICE. For all models, MICE had the smallest standard deviations. However, if only looking at the expected accuracy, kNN performed the best as it had the highest accuracy.

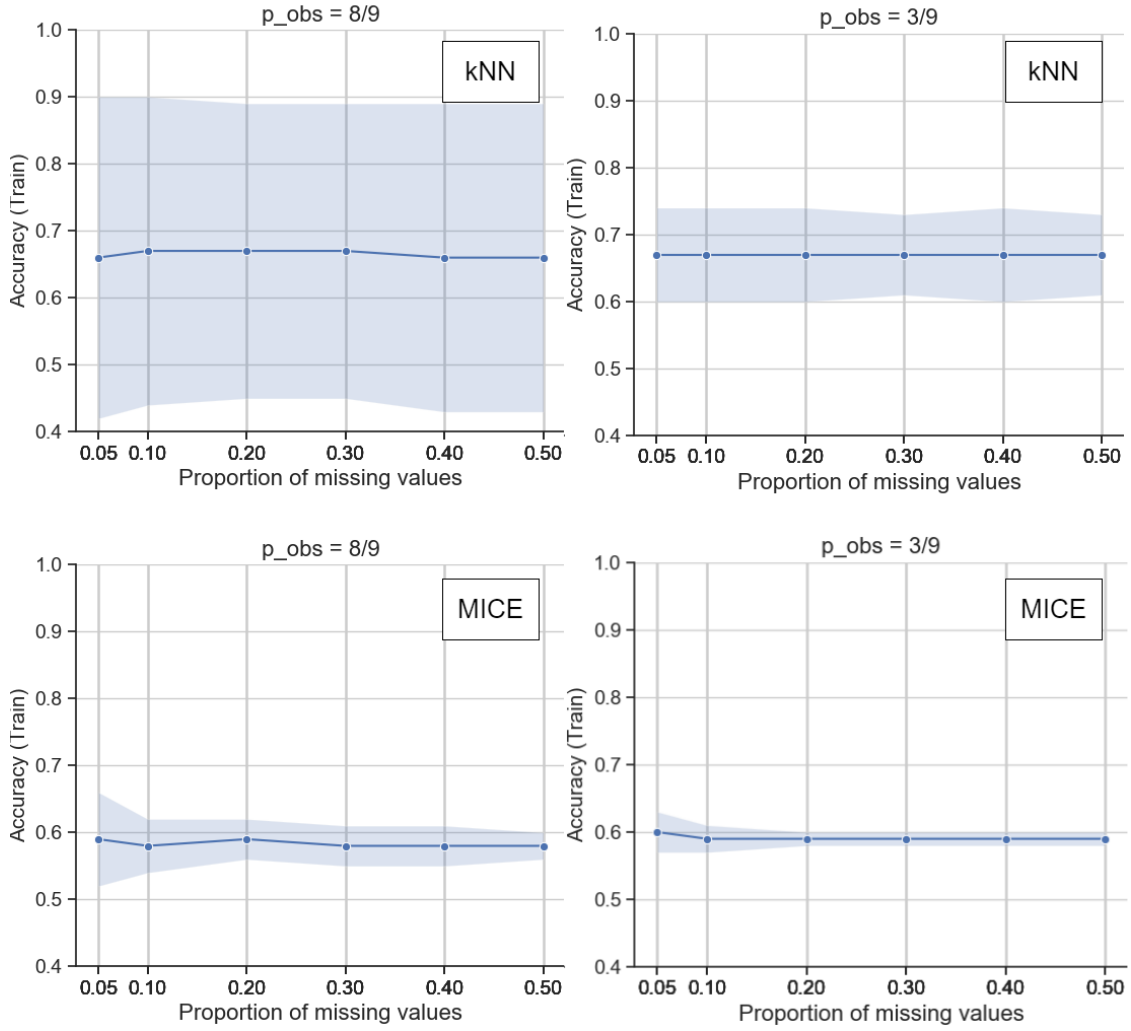


Figure 5.6: Train accuracy as a function of the proportion of missing values after applying kNN and MICE imputation on simulated missingness (MAR) in the colon dataset, where missingness was simulated 50 times. Only figures for p_obs $\frac{8}{9}$ and $\frac{3}{9}$ are included. The standard deviations (shaded area) and averages (solid line) are shown.

Similarily to MCAR, there was bias present in *perfor_Y* for MAR when applying the CCA strategy, as seen in Figure 5.7. It was only present when p_{obs} decreased and the proportion of missingness was high. There was not much bias in the other variables for both CCA and MICE. CCA showed smaller averages in the most variables, however, compared to MICE, it had larger standard deviations. Little bias was present, with averages ranging mostly between -1 and 1 with small standard deviations when excluding CCA. This includes the global imputations and kNN, though they are not shown. The complete figures are found in Appendix A.5 for CCA, A.6 for kNN and A.7 for MICE.

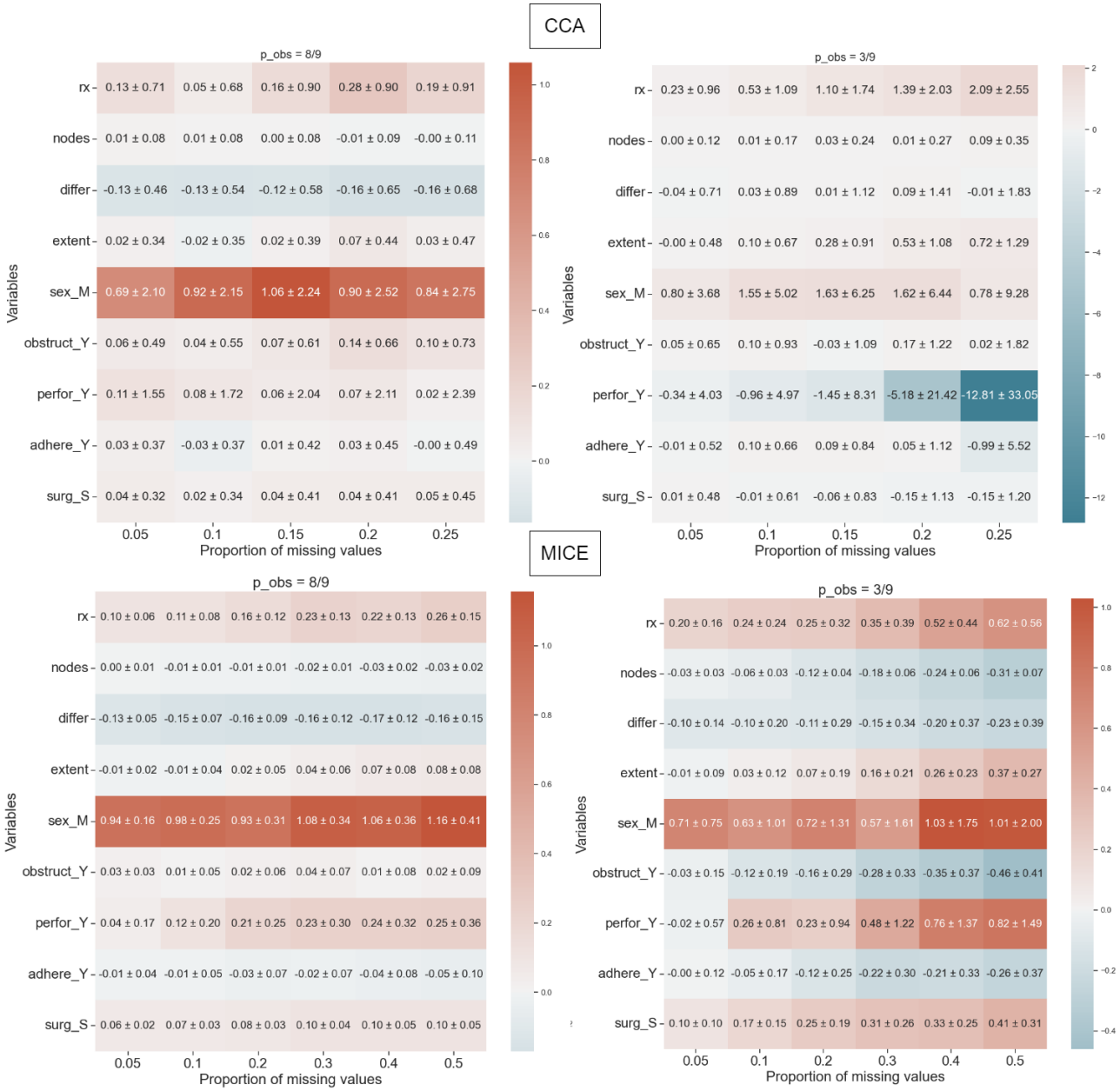


Figure 5.7: Bias of coefficients from a Cox PH model of the variables as a function of the proportion of missing values after applying CCA and MICE on simulated missingness (MAR) in the colon dataset, where missingness was simulated 50 times. Only figures for *p_obs* $\frac{8}{9}$ and $\frac{3}{9}$ are included. The whitest colour implies no bias, while blue and red lean towards negative and positive bias, respectively.

The concordance of all strategies except CCA followed the expected behavior, as the proportion of retained variables (p_{obs}) decreased, or the proportion of missing values (p_{miss}) increased, the concordance went down. This is shown for CCA, kNN and MICE in Figure 5.8. For CCA, it remained approximately constant, with the exception of when p_{obs} got very small, the disparity between the train and test concordance (not shown) grew larger, similarly as for MCAR. kNN and MICE were affected by both p_{obs} and p_{miss}. They performed worse when the proportion of missingness got larger, and especially in combination with few variables being left with no missing values. When most variables were retained, MICE had zero standard deviation. The complete figures are found in Appendix A.8 for CCA, A.9 for kNN and A.10 for MICE.

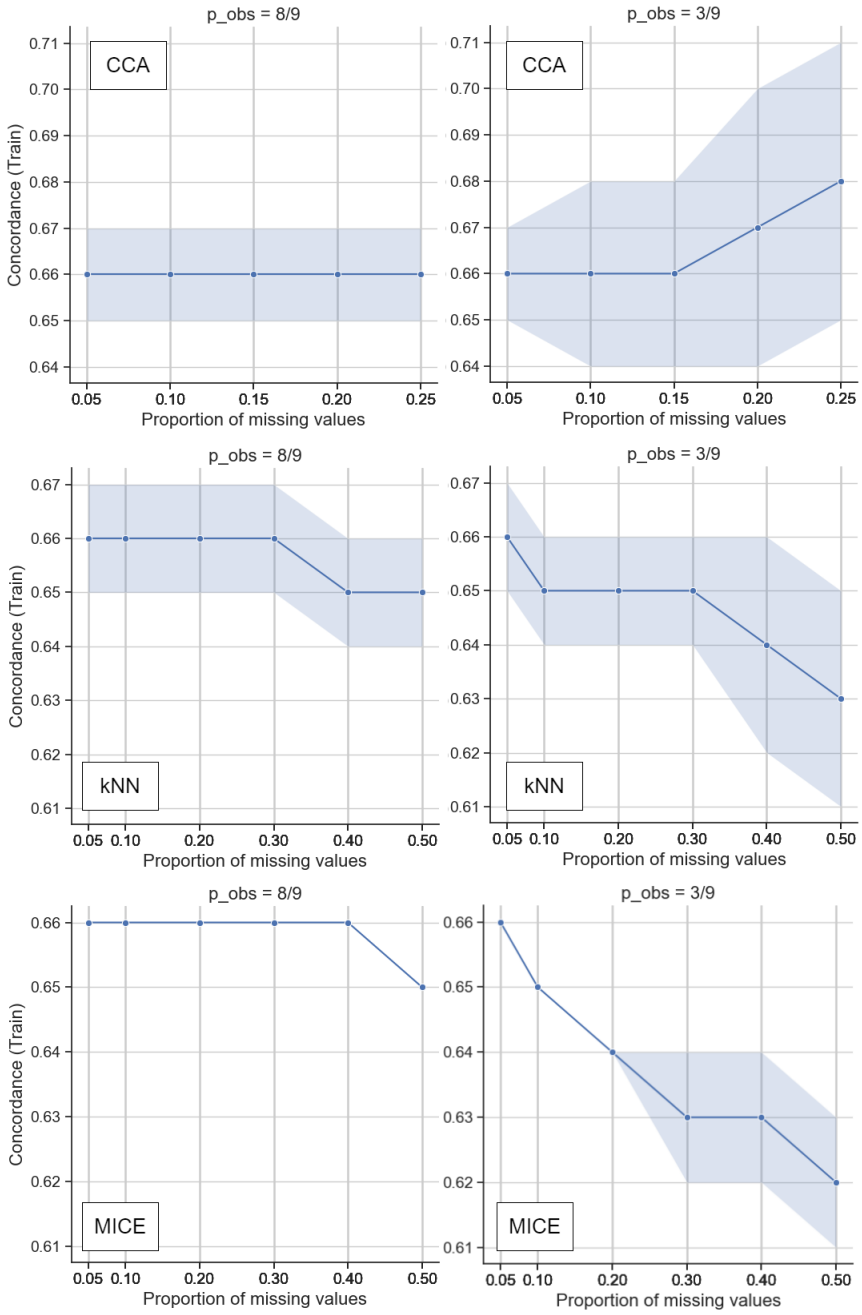


Figure 5.8: Train concordance as a function of the proportion of missing values after applying CCA, kNN and MICE on simulated missingness (MAR) in the colon dataset, where missingness was simulated 50 times. Only figures for *p_obs* $\frac{8}{9}$ and $\frac{3}{9}$ are included. The standard deviations (shaded area) and averages (solid line) are shown.

5.2 Experimental Setup: Survival Analysis (GEP NEN)

This section presents the results from the survival analysis experiments. In the first part, the survival analysis models are presented, and their representative performance will be assessed using both the C-index and the integrated Brier score. We investigate whether there was significant differences between all survival models when using kNN imputation compared to no imputation (CCA). We also compare the performance of kNN imputation and no imputation against both model-based and multiple imputation methods. Recall that model-based imputation refers to MICE where only one imputed dataset is generated. This comparison will focus on the survival model with the highest testing C-index. Multiple imputation used the imputation functions given Table 4.16. Lastly, different visualization plots (Figure 4.13) will be presented.

5.2.1 Runtimes of the Survival Analysis

The runtimes for each survival model applied to GEP NEN dataset are given in Table 5.3. RSF and CGB models had longer runtimes as they are more complex models.

Table 5.3: Gridsearch runtime for survival analysis models applied to GEP NEN dataset, where kNN and CCA were used for all four models. Model-based (MICE with one imputed dataset) and MI methods were used exclusively for the Coxnet model, which achieved the best performance model as measured by the test C-index. This was run on machine (2) (Section 4.2.2).

Model	Runtime (kNN)	Runtime (CCA)	Runtime (Model-based)	Runtime (MI)
Coxnet	50m 48s	35s	1m	31m 27s
Cox PH	3m 47s	9s	-	-
RSF	113m 9s	8m 14s	-	-
CGB	112m 56s	14m 48s	-	-

5.2.2 Hyperparameter Optimization

Four survival models were applied to the GEP NEN dataset following imputations with kNN. This was the primary imputation strategy used when tuning hyperparameters for each model. The choice of using kNN follows from the results of the experimental setup in Section 5.1, where it was shown that kNN performed well under both MAR and MCAR.

Coxnet

For the Coxnet algorithm, the hyperparameters optimized were regularization strengths *alpha* and *L1 ratio* (Section 4.4.4). Table 5.4 shows top 10 performance scores for C-index. *Alpha* represented the regularization strength where higher values indicated stronger regularization. The values vary significantly across models, ranging from 3.0 to 1000.0. *L1 ratios* indicated the proportion of L1 regularization in the elastic net model (Section 2.3.2). All models in the Table 5.4 used low *L1 ratio* of 0.01, 0.001, and 0.0001, suggesting a preference for L2 regularization (Ridge). There were marginal differences across the top 10 hyperparameter combinations for both the train and test sets in terms of the C-index. For IBS in the test dataset, there were differences between the top and worst contenders, where the difference was approximately 3 %.

Given this, the model with *alpha* of 3.0, *L1 ratio* of 0.01, a C-index test of 0.790, a C-index train of 0.827 and an IBS test of 0.123, achieved the best IBS performance without much decrease in the C-index.

Table 5.4: Sorted by their C-index test performance, the top-performing Coxnet models, optimized through tuning of the regularization strength *alpha* and *L1 ratio*, are listed. The table shows the performance scores for the C-index on both the training and test sets, along with the IBS score on the testing set.

	Alpha	L1 ratio	C-index test	C-index train	IBS test
1	1000	0.0001	0.796	0.813	0.161
2	100	0.001	0.795	0.813	0.159
3	700	0.0001	0.795	0.812	0.161
4	70	0.001	0.794	0.813	0.158
5	5	0.01	0.793	0.820	0.132
6	10	0.01	0.792	0.815	0.145
7	500	0.0001	0.792	0.812	0.161
8	50	0.001	0.792	0.813	0.156
9	3	0.01	0.790	0.827	0.123
10	5	0.001	0.790	0.823	0.129

The Appendix C Tables C.1, C.2, C.3, C.4, C.5, C.6, C.7 and C.8 show that *L1 ratios* of 0.2 or higher with *alpha* values of 3 and more results in a C-index performance of 0.5. This indicated that these models was performing no better than random guessing.

Cox Proportional Hazard

In the Cox Proportional Hazard model, the regularization strength *alpha* was the only hyperparameter used to tune the model (Section 4.4.4). Table 5.5 shows that models with higher *alpha* values consistently achieved higher C-index scores on the test set and demonstrated less overfitting compared to those with lower *alphas*. IBS tends to increase with higher *alpha* values, which is especially noticeable for *alpha* values exceeding 50. This trend indicated that performance marginally decreases with stronger regularization (*alpha*). The C-index for the test set showed negligible differences between *alpha* values 500 and 200. Additionally, the similarity between the test and train results suggests minimal overfitting. Given the decrease in the IBS from 13.2% to 11.9% when reduced *alpha* from 500 to 200, the optimal model was with *alpha* 200.

Table 5.5: Sorted by their C-index test performance, the top-performing Cox PH models, optimized through tuning of the regularization strength *alpha*, are listed. The table shows the performance scores for the C-index on both the training and test sets, along with the IBS score on the testing set.

	Alpha	C-index test	C-index train	IBS test
1	500	0.789	0.820	0.132
2	200	0.788	0.832	0.119
3	700	0.787	0.818	0.137
4	1000	0.787	0.816	0.142
5	100	0.785	0.841	0.112
6	70	0.783	0.848	0.110
7	50	0.779	0.854	0.109
8	20	0.766	0.870	0.109
9	10	0.761	0.879	0.113
10	5	0.756	0.884	0.120

Random Survival Forest

In the Random Survival Forest algorithm, four hyperparameters were tuned, the regularization strengths *n_estimators*, *max_depth*, *min_samples_split*, and *min_samples_leaf* (Section 4.4.4). Table 5.6 illustrates that the top 10 C-index scores were relatively close to each other, indicating consistency in performance across all these combinations of hyperparameters. IBS performance also remained relatively stable. Despite a general trend of overfitting observed in all top 10 models, the first row from the top model showed the least overfitting, indicating that it may be the most balanced and potentially the best-performing model for the RSF. Therefore, the optimal hyperparameter combinations for the RSF included 40 estimators, a max depth of 5, a minimum sample split of 8, and a minimum sample leaf of 4.

Table 5.6: Sorted by their C-index test performance, the top-performing RSF models, optimized through tuning of the regularization strength *n_estimator*, *max_depth*, *min_samples_split* and *min_samples_leaf*, was listed. The table shows the performance scores for the C-index on both the training and test sets, along with the IBS score on the testing set.

	n estimator	**max depth**	**min samples split**	**min samples leaf**	**C-index test**	**C-index train**	**IBS test**
0	40	5	8	4	0.777	0.883	0.121
1	40	5	6	4	0.777	0.883	0.122
2	40	5	2	4	0.777	0.883	0.121
3	40	4	2	6	0.776	0.862	0.122
4	40	4	8	6	0.776	0.862	0.121
5	40	4	6	6	0.776	0.862	0.118
6	40	5	2	6	0.775	0.863	0.121
7	40	5	8	6	0.775	0.863	0.125
8	40	5	6	6	0.775	0.863	0.121
9	40	3	8	6	0.775	0.856	0.118

Component-Wise Gradient Boosting

In the Component-wise gradient boosting algorithm, three hyperparameters were tuned, the regularization strengths *n_estimator*, *learning_rate*, *subsample* (Section 4.4.4). In Table 5.7, the training and test sets for the C-index show marginal differences across all models and demonstrate minimal overfitting. This marginal difference was also consistent for the IBS. Since there were small differences between the models in C-index and IBS, the first row from the top was the optimal model with 10 estimators, a learning rate of 0.6, and a subsample rate of 0.6.

Table 5.7: Sorted by their C-index test performance, the top-performing CGB models, optimized through tuning of the regularization strength *n_estimator*, *learning_rate* and *subsample*, are listed. The table shows the performance scores for the C-index on both the training and test sets, along with the IBS score on the testing set.

	n estimator	learning rate	subsample	C-index test	C-index train	IBS test
1	10	0.6	0.6	0.789	0.818	0.111
2	10	0.7	0.6	0.787	0.822	0.110
3	20	0.6	0.7	0.786	0.838	0.112
4	20	0.5	0.2	0.785	0.831	0.111
5	10	0.5	0.3	0.785	0.814	0.115
6	10	1.0	0.6	0.784	0.824	0.110
7	20	0.6	0.8	0.784	0.839	0.112
8	30	0.3	0.6	0.784	0.831	0.113
9	10	0.5	0.6	0.784	0.817	0.113
10	20	0.5	0.9	0.784	0.834	0.111

5.2.3 Evalution of Missing Value Strategies

For the missing values, we used a t-test to determine if there was a significant difference between the performances based on the missing value strategies applied to the data. This was an indirect evaluation (Section 3.8). Since we used RSKF, the t-test will compare the list of C-index scores for the test set to determine whether there was a significant difference in their means. Initially, we evaluated how well kNN imputations was compared with CCA across various the survival analysis models Coxnet, Cox PH, RSF, and CGB. Subsequently, we used the highest-performed survival model to determine whether the model-based imputation resulted in any significant difference. Recall that model-based imputations refer to using MICE with only one dataset being generated. Finally, we estimated whether applying multiple imputations with MICE resulted in any significant differences.

For multiple imputations, PMM gave an equal score to the third digit, so we will not present these as separate results.

Comparing kNN Imputation with CCA Across Different Models

Table 5.8 shows the t-tests that compared the performances of kNN imputation with CCA across the survival models. The differences in mean performance were statistically significant for in models.

Table 5.8: Comparative analysis of kNN imputation and CCA across survival analysis models Coxnet, Cox PH, RSF, and CGB. The table reports the mean and standard deviation of the C-index test performance (RSKF) for each model under both kNN imputation and CCA. It also provides the t-test statistics and corresponding p-values to assess the statistical significance of the differences between the kNN-imputation and CCA.

Model	CCA	kNN imputation	t-test statistic	p-value
Coxnet	0.75 ± 0.049	0.796 ± -0.038	-3.562	0.00084
Cox PH	0.75 ± 0.049	0.789 ± 0.033	-3.3961	0.00025
RSF	0.739 ± 0.063	0.777 ± 0.037	-2.5644	0.01359
CGB	0.752 ± 0.05	0.788 ± 0.045	-2.6268	0.01153

Model-based Comparison

For the most common survival model, Coxnet was chosen for in-depth analysis. Table 5.9 provides t-test comparisons for the Coxnet model using different imputation methods by comparing the model-based imputation with kNN and CCA. The comparisons suggest that model-based imputation provided an improvement over CCA, as indicated by the significant p-value. When comparing model-based imputation to kNN, they are not statistically different.

Table 5.9: Comparisons for the Coxnet model using model-based imputation (MICE with one imputed dataset), kNN imputation and CCA. The table reports the mean and standard deviation of the mean C-index test performance (RSKF) for model-based imputation, kNN imputation, and CCA. It also provides the t-test statistics and corresponding p-values to assess the statistical significance of model-based imputation to kNN imputation and CCA.

	Mean ± SD		
Model-based	0.788 ± 0.046	**t-statistic**	**p-value**
CCA	0.750 ± 0.049	2.7058	0.00940
kNN	0.796 ± 0.038	-0.6541	0.5161

MI Comparison

For the most common survival model, Coxnet was chosen for consistency and comparisons. Table 5.10 provides the t-tests for Coxnet when comparing MI with kNN imputations and CCA. The results show that MI was a better choice over CCA. In contrast, when comparing MI with kNN, their means were not statistically significant, given the relatively high p-value.

Table 5.10: t-test comparisons for evaluating the performance differences between multiple imputation (MI), kNN imputation, and CCA using the C-index test set values. The table reports the mean and standard deviation of the mean C-index test performance (RSKF). For multiple imputations, the performance is averaged across all datasets (mean MI). It also provides the t-test statistics and corresponding p-values to assess the statistical significance of MI to kNN imputation and CCA.

	Mean ± SD		
Mean MI	0.785 ± 0.041	**t-statistic**	**p-value**
CCA	0.750 ± 0.049	2.7014	0.0095
kNN	0.796 ± 0.038	-0.9794	0.3323

5.2.4 Visualization of Survival Model Performance and Evaluation

The visualization plot of the survival model performances were described in Figure 4.13 in Section 4.4.

Analysis of Coxnet Model Performance across Regularization Strength

Figure 5.9 presents the effects of varying the regularization strengths, *alpha* and *L1 ratio*, on the performance metrics, including IBS and the C-index, both on train and test folds, as well as feature selection within the Coxnet model. Figure 5.9 is divided into four plots, each representing a different L1 ratio: 0.0001, 0.001, 0.01, and 0.1. As the *alpha* value increased, indicating stronger regularization, the number of features selected by the model decreased. For higher *L1 ratios*, feature weights quickly reduce to zero at lower *alpha* values, reducing the number of selected features. These results lead to performance equivalent to random guessing as all features got omitted by the regularization.

For an *L1 ratio* of 0.0001, the test C-index improved as overfitting was less persistent with increasing *alpha* value, while the number of features stayed almost constant. At an *L1 ratio* of 0.001, an *alpha* of 200 offered a good balance between minimal overfitting and a high number of features. For an *L1 ratio* of 0.01, the ideal *alpha* values were around 3, 5, or 10. In contrast, the *L1 ratio* of 0.1 generally leaned towards overfitting compared to the others. Specifically, at an *alpha* of 1, the model overfitted more while maintaining a higher number of features, whereas an *alpha* of 3 resulted in less overfitting but also reduced the number of features significantly.

The IBS across all four subplots showed an improvement simulatinously with the C-index. Unlike the C-index, the IBS initially decreased before rising and stabilizing as alpha changes. The IBS changes happened at the same point the C-index started getting better and stopped when the C-index began to drop. The best performance of the IBS was seen in the middle of this range.

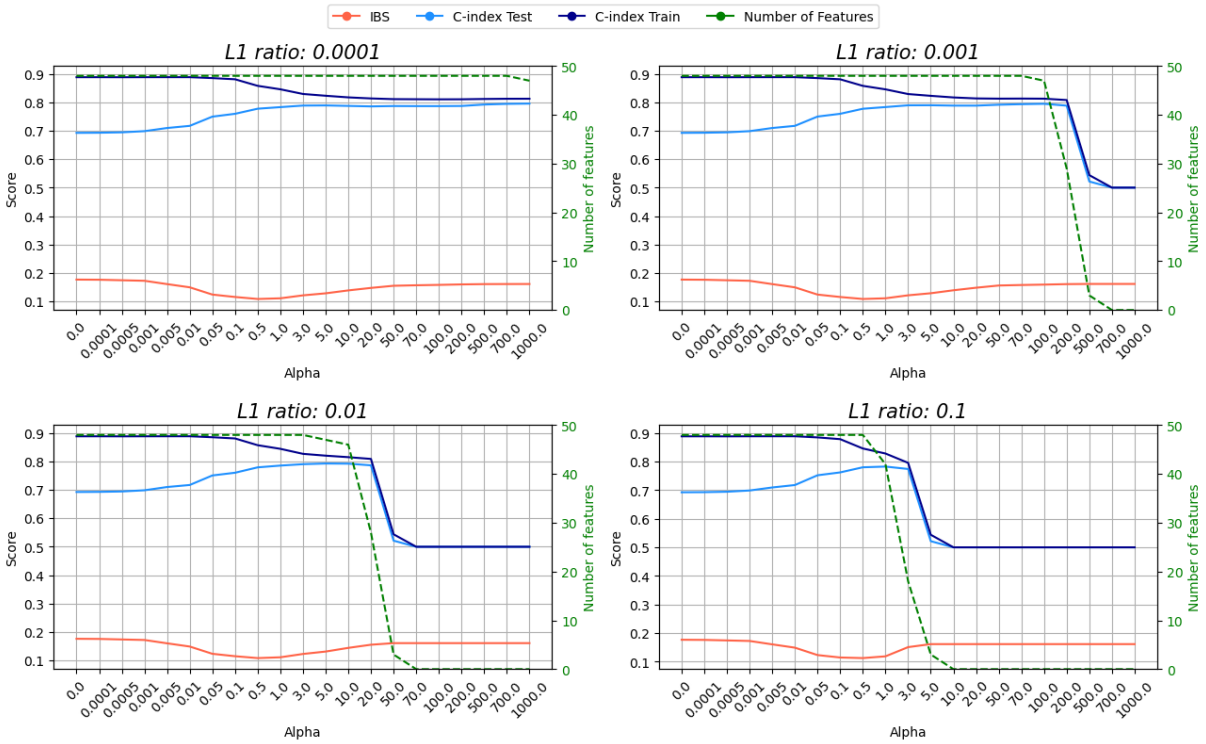


Figure 5.9: Coxnet model performance across regularization strength, *alphas* and *L1 ratios*. These plots show the C-index for the training folds (dark blue line), the C-index for the test folds (light blue line), IBS (red line), and the number of features used (green dashed line), as the *alpha* parameter varies. Each plot has two y-axes: the left y-axis shows metric performance (ranging from 0 to 1), and the right y-axis shows the number of features used (ranging from 0 to 50). The x-axis, common to all plots, represents the *alpha* parameter, ranging from 0.0001 to 1000. Note that the spacing between each point on the x-axis varies due to the different *alpha* values.

Permutation Feature Importance

The top 15 most important features for the four survival models (Coxnet, Cox PH, RSF, and CGB) are compared in Figure 5.10. Alongside the bars, the purple line represents the average importance across the models for each feature. Each bar´s height indicates the importance of that feature in the model, with separate colours for each model. Figure 5.10 shows that some features, like *Number of Courses*, *Ki-67*, and *NSE*, were considered highly important across all survival models. The selected hyperparameters for the survival models were described earlier (Section 5.2.2).

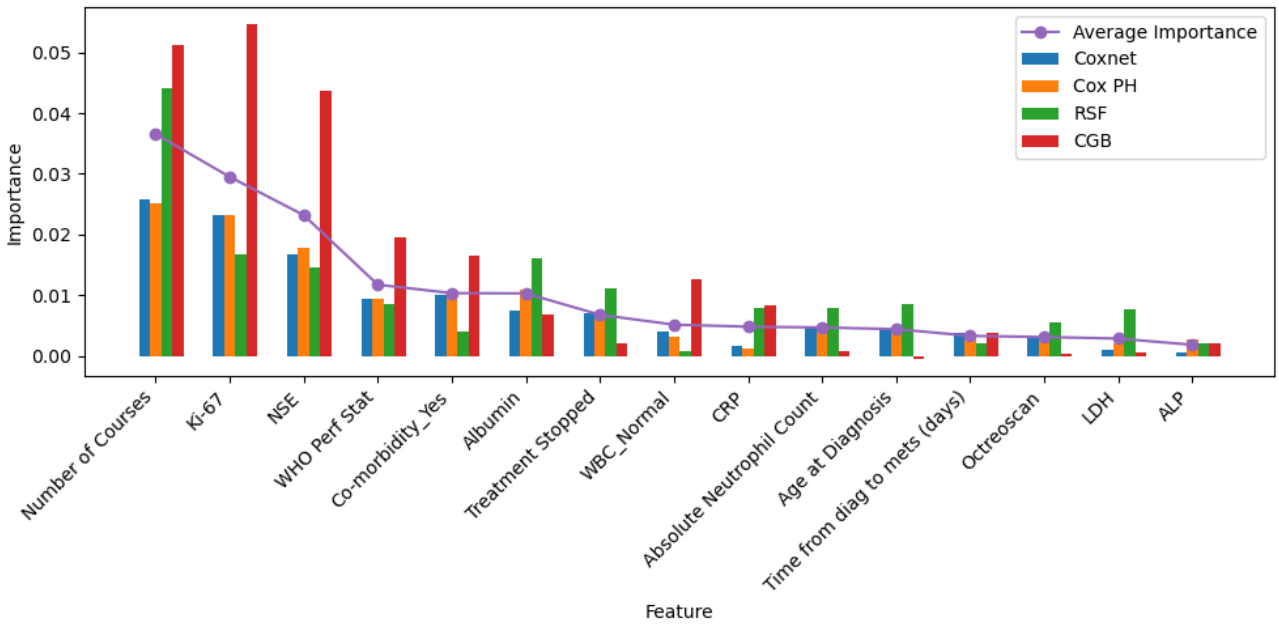


Figure 5.10: Permutation feature importance across the different survival models, where the purple line represents the average importance across the models for each feature. Coxnet (*alpha*: 3, *L1 ratio*: 0.01), Cox PH (*alpha*: 200), RSF (*n_estimator*: 40, *max_depth*: 5, *min_samples_split*: 8, *min_samples_leaf*: 4), CGB (*n_estimator*: 10, *learning_rate*: 0.6, *subsample*: 0.6)

Time-Dependent Brier Score

The time-dependent Brier score in Figure 5.11 compares the performance of four different predictive models over time in terms of their mean time-dependent Brier score. The selected hyperparameters for the survival models were described earlier (Section 5.2.2).

Looking at IBS for each of the survival models, the Coxnet model, yielded an IBS of 12.3%, while the Cox PH model´s resulted in a slightly lower IBS of 11.9%. RSF and CGB models had IBS values of 12.0% and 11.1%, respectively. Notably, CGB had the lowest overall Brier score throughout the entire range of survival times, as detailed in Figure 5.11. Although the Cox PH and RSF models were quite similar in Brier score performance across these times, they fell behind CGB.

Initially, all models start with very low Brier scores, suggesting that the models excels with accurate predictions of early survival, which gradually decrease, peaking around 400 days. After that, the performance started to get improve towards 800 days, with some variations, until it stabilized after 800 days.

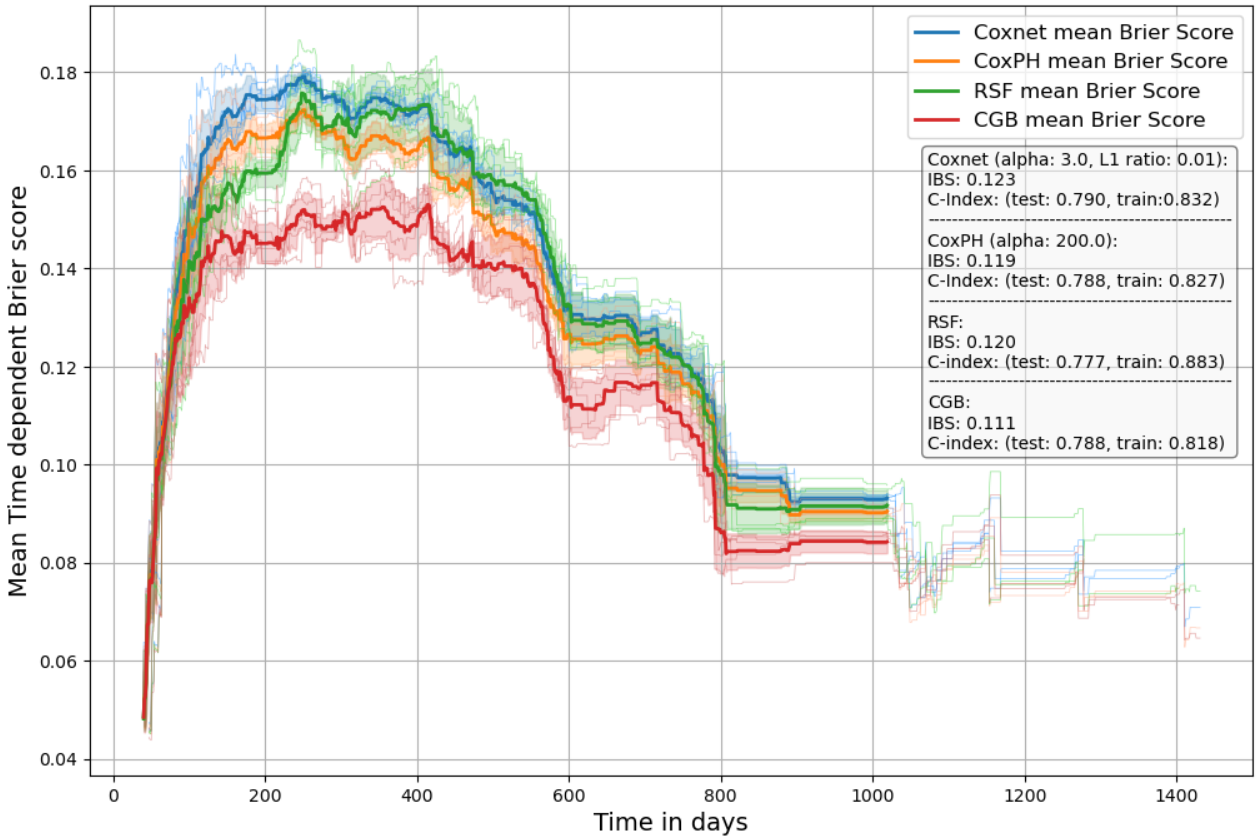


Figure 5.11: Mean time-dependent Brier score for the four survival models. The shaded area is the standard deviation. The table on the right side shows the performance (C-index and IBS) and the hyperparameters for each model. Coxnet (*alpha*: 3, *L1 ratio*: 0.01), Cox PH (*alpha*: 200), RSF (*n_estimator*: 40, *max_depth*: 5, *min_samples_split*: 8, *min_samples_leaf*: 4), CGB (*n_estimator*: 10, *learning_rate*: 0.6, *subsample*: 0.6)

Survival Curves

Survival curves obtained using Coxnet and CGB for four representative patients are shown in Figures 5.12 and 5.13. The survival curves by RSKF imputed with kNN are shown in Figure 5.12, and the survival curves between the suggested imputations with MICE are shown in Figure 5.13. Both show the same patients and were fit with the same hyperparameters, selected in Section 5.2.2, for comparability. The prior shows the variation in the predictions when the covariates used for predicting a patient vary and the imputations are fixed, whereas the latter shows variation in predictions when the covariates are fixed and imputations (potentially) vary.

Both figures show survival probabilities starting at 100% and declining over time. The survival curves have two dashed lines, where the expected survival time offers a benchmark for evaluating the model's performance with the observed survival time.

We first consider Figure 5.12. Both models performed well for patients 9007, closely matching the observed survival times. For patients 9024, 9065, and 9040, both models showed a decrease in predictive performance as the observed survival times exceeded 400 days. For patient 9024, the expected survival time was closer to the observed survival time for Coxnet than for the CGB model, and the CGB model had a larger standard deviation. For patient 9065, both models displayed nearly identical standard deviations, but the expected survival time for CGB was closer to the observed than for Coxnet. The last patient, 9040, had a longer survival time than most in the GEP NEN dataset, and none of the models were close to the observed survival time. Both models showed low variability and nearly similar survival curves.

Next, by considering Figure 5.13, larger variations between the curves was present. CGB showed high standard deviations for patients 9024, 9065 and 9040. The expected survival times with CGB for these patients were still closer to the actual survival times than Coxnet. Similarly to Figure 5.12, both models performed well for patient 9007, as their expected and observed survival times were close. The expected survival time of patient 9065 was predicted closer to the actual survival time for multiple imputations with both Coxnet and CGB. There is much less variation in the survival curves Coxnet predicted than CGB predicted.

As illustrated in Figures 5.12 and 5.13, among the four patients, Coxnet generally provided more conservative survival estimates, displaying similarity in the curves across different patients with minimal variation. In contrast, CGB showed more distinct curves between the patients, and acknowledged uncertainty with higher standard deviations. Both models were challenged by the long survival times.

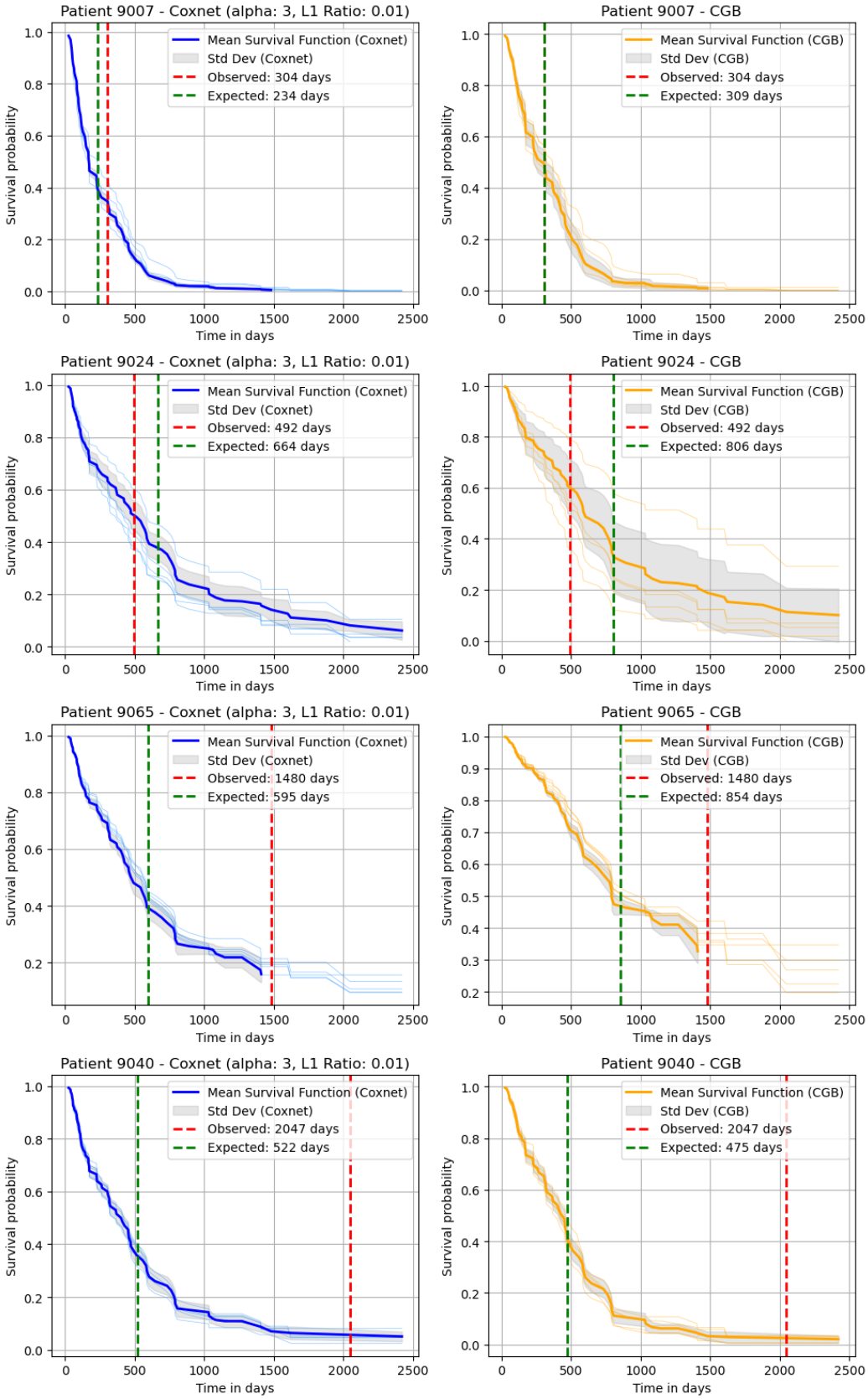


Figure 5.12: Survival curves for the different patients (9007, 90024, 9065, and 9040), for Coxnet and CGB models with kNN imputations. The different curves are given by repeated stratified k-fold. The blue curve is the average estimated survival curve and the shaded area (Std Dev) is the standard deviation. The red line indicates when the event of interest occurred (observed survival time) and the green line marks the model's prediction for when the event was expected to happen (expected survival time). Coxnet (*alpha*: 3, *L1 ratio*: 0.01), CGB (*n_estimator*: 10, *learning_rate*: 0.6, *subsample*: 0.6).

Appendix B.3 extends the findings from Figure 5.12 by providing survival curves for all patients who experienced an event. CGB typically performed better than Coxnet in cases with shorter observed survival times. Figure 5.12 shows that the CGB model performed better than Coxnet when survival exceeded 400 days. However, Appendix B.3 shows other cases with also opposite results. Uniformity in Coxnet's survival curves might not be immediately clear when examining a few patients, as illustrated in Figure 5.12. However, a broader comparison involving all patient survival curves in Appendix B.3 provides clearer insights. Figures B.1 and B.2 show the Coxnet model set to an *alpha* of 1000, which yielded the best testing C-index performance, demonstrates that the survival curves remained even more consistent across the patients.

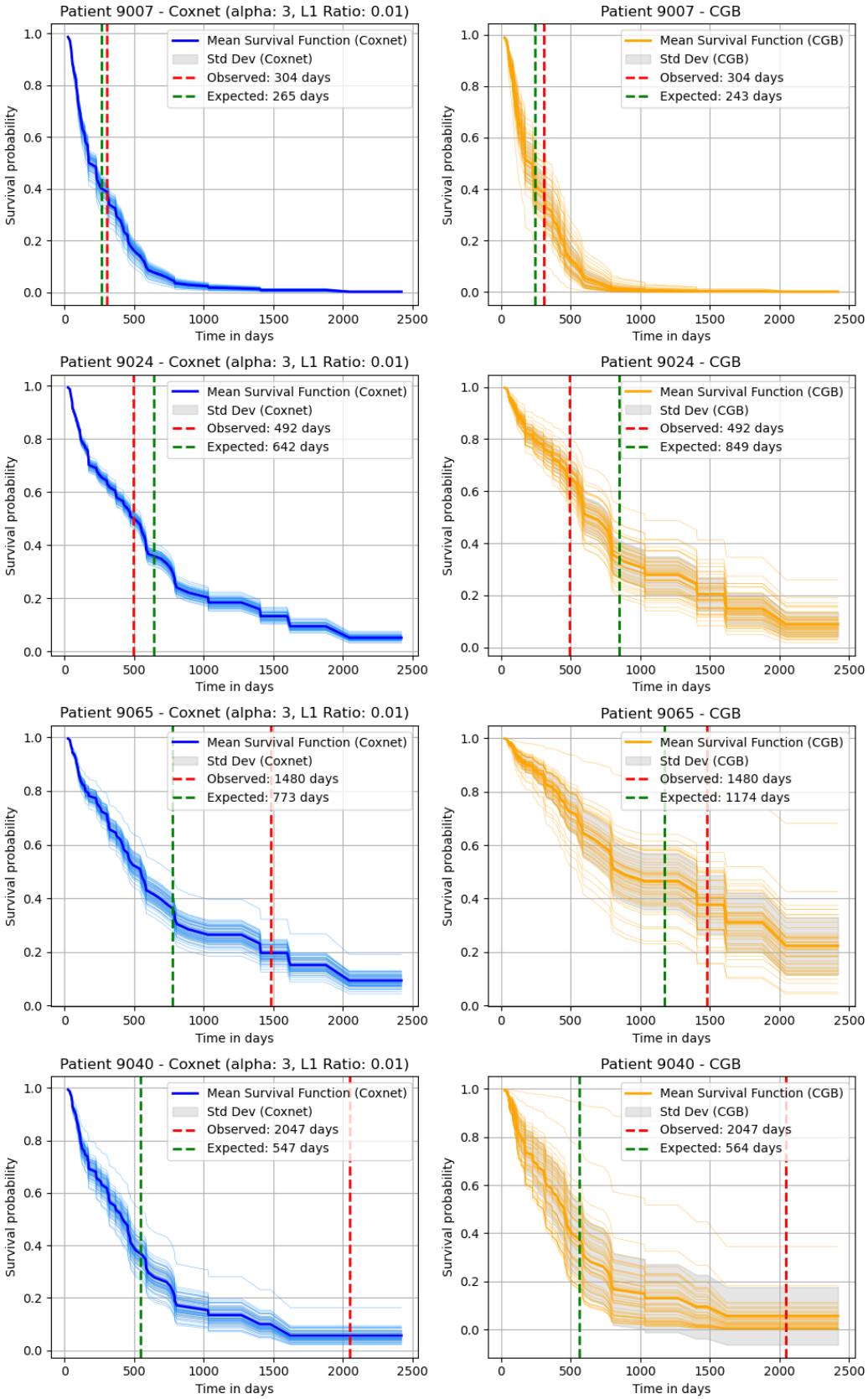


Figure 5.13: Survival curves for the different patients (9007, 90024, 9065, and 9040), for Coxnet and CGB models with multiple imputations using 50 imputed datasets. The blue curve is the average estimated survival curve and the shaded area (Std Dev) is the standard deviation. The red line indicates when the event of interest occurred (observed survival time) and the green line marks the model's prediction for when the event was expected to happen (expected survival time). Coxnet (*alpha*: 3, *L1 ratio*: 0.01), CGB (*n_estimator*: 10, *learning_rate*: 0.6, *subsample*: 0.6).

Analyzing Feature Coefficients with Varying Regularization Strengths

The two subplots for feature *Absolute Neutrophil Count* and *Chemotherapy Type* in Figure 5.14 illustrate the relationship between the regularization parameter *alpha*, the C-index and integrated Brier score for the test fold. Each plot shows how the feature coefficients vary with changes in the *alpha* regularization parameter. As *alpha* increased, we generally observed coefficient values converging to zero. Simultaneously, the C-index demonstrated an optimal range before declining, underscoring the trade-off between model complexity and predictive performance. In contrast, IBS increased, suggesting a drop in how well the model predicts outcomes as we adjust *alpha*. For both features, *Absolute Neutrophil Count* and *Chemotherapy Type*, their coefficients shift sign towards zero. Note that both variables changed the direction of the weight as *alpha* got larger.

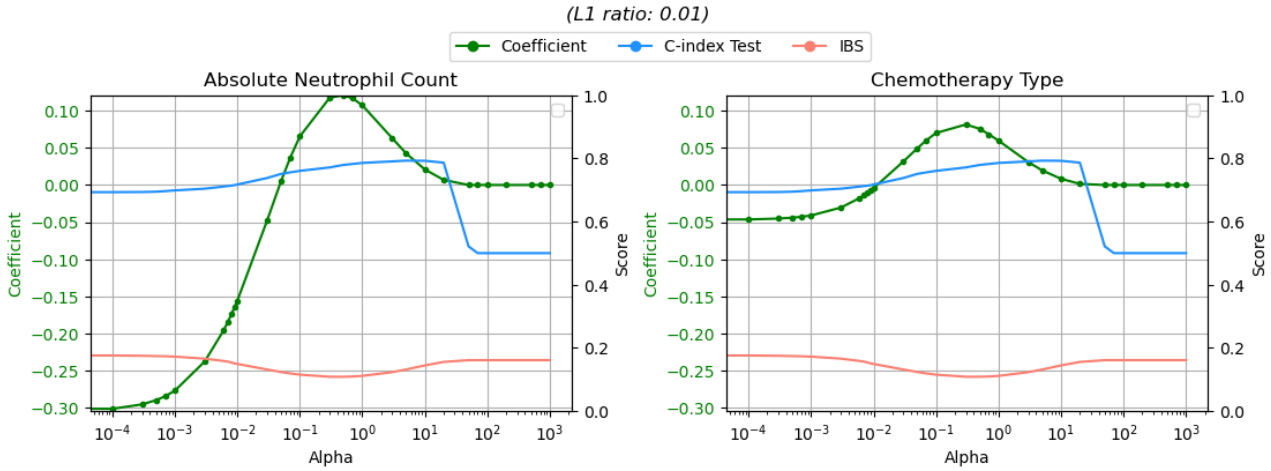


Figure 5.14: C-index (blue), IBS (red), and model coefficients (green) as a function of the regularization parameter alpha for Coxnet model of *L1 ratio* of 0.01. Showing changes across different *alpha* values for the *Absolute Neutrophil Count* and *Chemotherapy Type* features.

Permutation Feature Importance for Coxnet Model

Figure 5.15 illustrates the permutation feature importance for the Coxnet model, with parameters *alpha* set to 3.0 and *L1 ratio* to 0.01. The bar chart ranks features by their impact on model performance when they were permuted. Features that greatly influenced the model's predictive performance appear at the top, with *Number of Courses*, *Ki-67*, and *NSE* being the most important ones.

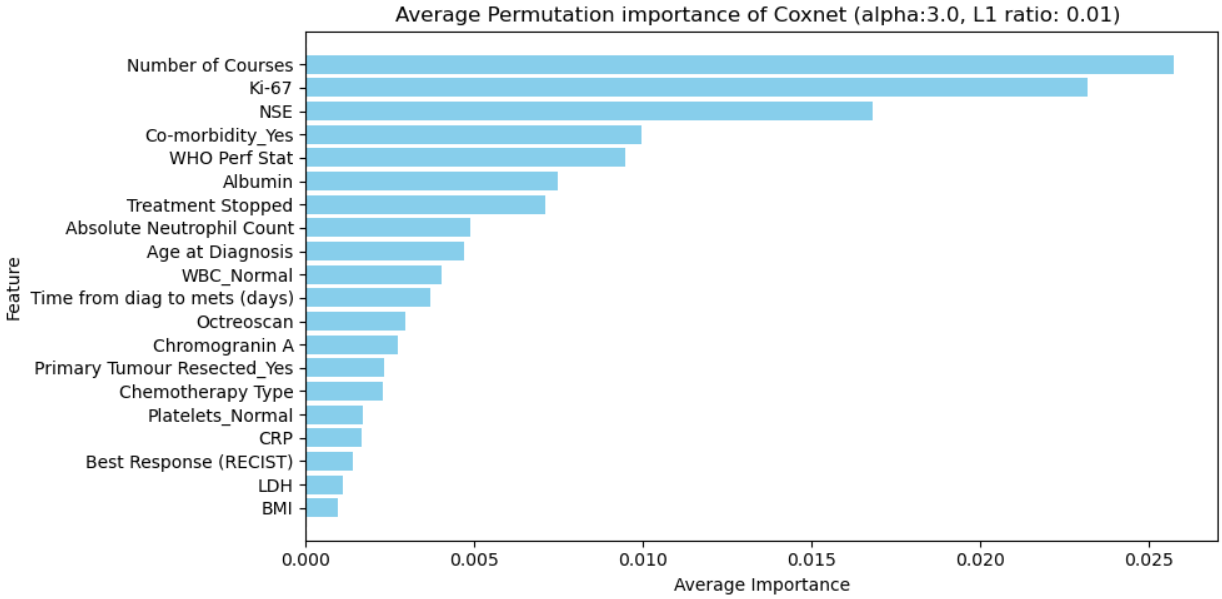


Figure 5.15: Permutation feature importance for the Coxnet model (alpha: 3.0, L1 ratio: 0.01) showing the average importance of each feature.

Coefficients of the Features for Coxnet

Figure 5.16 is a horizontal bar graph representing the coefficient weights assigned to each feature in the Coxnet model, with an *alpha* of 3.0 and *L1 ratio* of 0.01. The length and direction of the bars indicate the magnitude and effect direction (positive or negative) of the features on the model's predictions. Features with longer bars are more influential in the model, with positive coefficients indicating increasing risk and negative coefficients indicating a reduced risk for model predictions. Figure 5.16 shows that *Ki-67* and *NSE* had a positive effect, while *Number of Courses* appeared to have a negative effect.

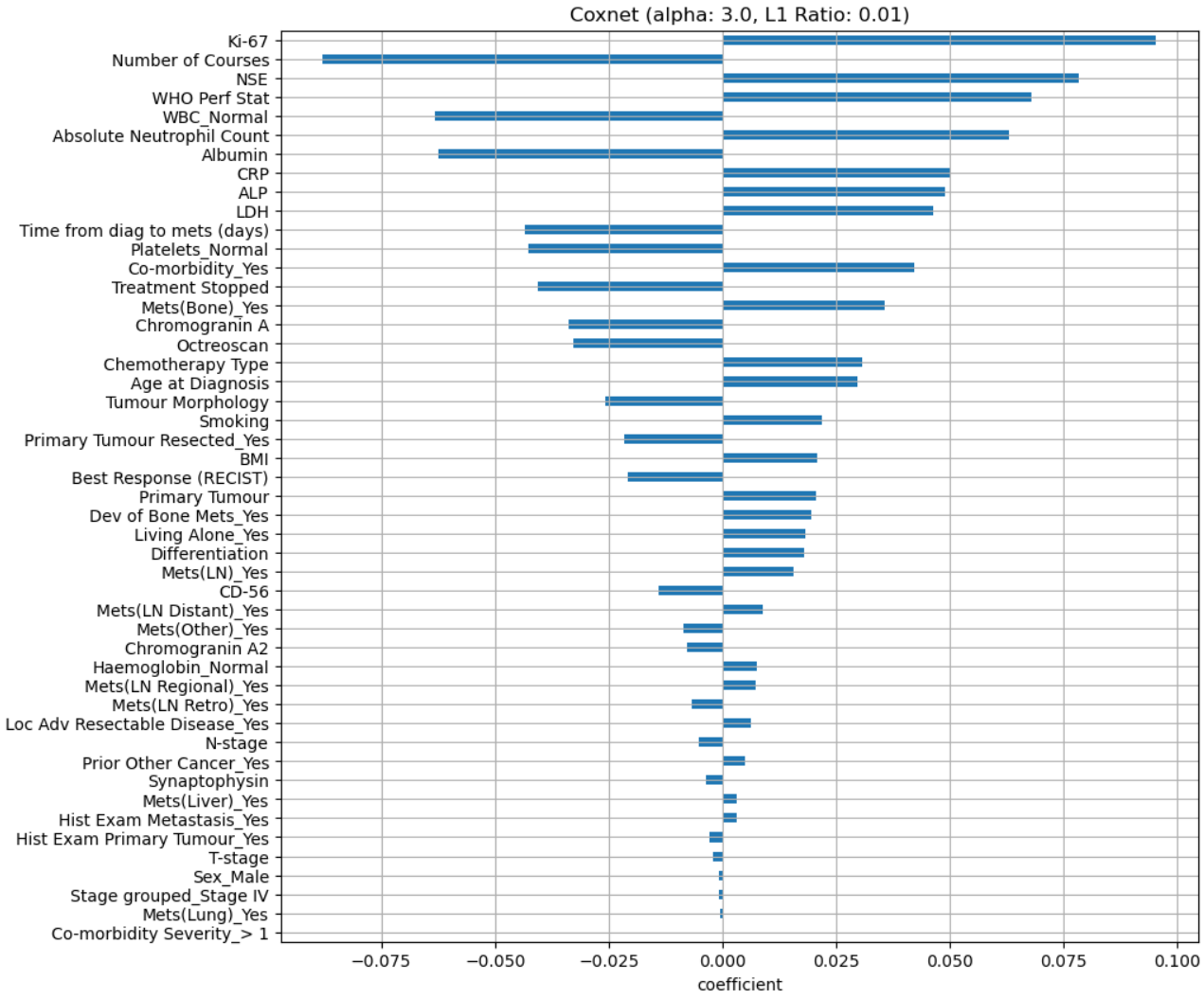


Figure 5.16: The bar chart shows the coefficient weights for all features in the Coxnet model (alpha: 3.0, L1 ratio: 0.01), with the magnitude and direction indicating their respective influence on the model's predictions.

6 Discussion

6.1 Experimental Setup: Missing Values (Colon)

For the missing value experiment, we will discuss the most interesting results presented in Section 5.1. In particular, we will discuss the flaws of the experimental setup, RMSE, the behaviour of the concordance index and the corresponding bias.

6.1.1 Flaws of the Experimental Setup

We want to discuss the experimental setup's flaws as it can give clarity to some of the results. First, we have not looked at combinations of the missingness mechanisms; we have only looked at separate cases. While this was not found in the literature, it is unrealistic to only have MCAR or MAR present and not both. However, by going for mixtures of both, there would be an additional layer of complexity, namely all the pairs of combinations of hyperparameters. Next, the *status* variable was not incorporated in any way when simulating missingness. This was because the status and time-to-event variables were reserved for the Cox PH model, and we did not want to impute the target variable. There were about 40% censored samples in the complete dataset, which means 40% were still alive at the time of the event. Thus, it may not be reasonable to assume that the distribution of the covariates was equal between these two groups. This is shown in Figure 4.1 from Section 4.1.1. In particular, there was a substantial difference in the *nodes* variable between censored and non-censored samples. The point to raise here is that by not incorporating *status*, it may have challenged the imputations. For example, mode imputation assumes a global mode, but that may not be the case if there are differences between the censored and non-censored samples. As for kNN, the same argument holds true, and the neighbors may not be representative. The last flaw is that the ground truth Cox PH model was fitted on the complete dataset. However, when comparing the bias of the coefficients from the imputed datasets, we used the training coefficients. This results in potential bias from two sources: the missing 30% reserved for testing, and the actual imputations. One could claim that it would be better to fit the ground truth model on the complete training data. This would have two potential issues. The first and most obvious one being that less information would be used to estimate the proxy ground truth coefficients, and the second being computational efficiency. The latter is because each of the $1, \ldots, M$ imputed datasets had unique train/test splits, so it would have required fitting the ground truth model to each of these unique train/test splits.

6.1.2 Accuracy

For MCAR, kNN imputation performed the best in terms of attaining the highest accuracy, whereas mode was the worst. MICE's performance was in-between mode and kNN imputations. This is likely explained by the violation of the MAR assumption mentioned in Section 3.6.3. Although, for MAR, while the accuracy of mode imputations was not shown, their rankings were still the same. MICE and kNN were slightly closer together in performance, and the latter still performed the best. All imputations had higher standard deviations, except for MICE, which remained unchanged. One would expect MICE to perform better for the MAR mechanism; however, primarily evaluating the accuracy as a direct evaluation may be insufficient.[85,18]

6.1.3 Bias

For the bias there are a few important key points to raise. Notice how the concordance after applying CCA increased with the proportion of missingness, but at the same time, the bias increased as well. To be specific, for MCAR under 25% missing values, *perfor_Y* and *sex_M* had bias estimated to -33 ± 50 and 2 ± 24, respectively (Figure 5.2). For MAR, the same variables had at their worst -13 ± 33 and 1 ± 9 in the same order (Figure 5.7). Thus, dropping samples seemed to

have a large effect on the bias of the coefficients, even if the training concordance was promising. This illustrates why a single metric for evaluation can be misleading. For the imputations, under both MAR and MCAR, kNN and MICE had the lowest overall bias, with MAR having a slightly lower bias. This was expected because of the additional information in the observed variables for the MAR mechanism. Notice an important detail. The highest bias using MICE of the variables *perfor_Y* and *sex_M* were, for MCAR, 1.2 ± 1.9 and 1.2 ± 2.8, respectively, compared to CCA's much higher standard errors.

Note that CCA was only measured for the proportion of missingness $\{0.05, 0.1, 0.15, 0.20, 0.25\}$ and all imputations $\{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$. This means that for comparison's sake, the worst estimates of *perfor_Y* and *sex_M* from MICE stemmed from twice as large a proportion of missingness, and yet the bias was much lower. Now, there are two potential sources that might cause this: either the automatic regularization applied to guarantee successful fit gave coefficients that are far off in the sense that they are close to 0, or overfitting in the sense that the coefficients were large in magnitude. Recall that our definition of the bias in Eq. (4.1) has a denominator present for scaling. This implies that if the proxy ground truth coefficient is small, the division by near 0 will cause small changes in the estimated coefficients to make the bias blow up in magnitude. For example, with $\beta_j = 0.01$ and $\hat{\beta}_j = 2.01$, we would get $bias(\hat{\beta}_j) = 200$. We defined the bias in such a way that it measures the count of standard deviations from the real value, much like a z statistic. If the source of the bias is indeed from forcing the weights very small, the worst possible scenario would be an estimated coefficient of 0 with the ground truth weight far away from 0, which would give a bias of ± 1. This applies to any of the coefficients because the denominator scales it. The bias will, at most, be one standard deviation away from the proxt ground truth coefficient if the coefficients are shrunk to 0. It is more likely that the high biases were from overfitting. A manual check was done to see if the regularization strength ever increased, which it did not. [(2)] The unusual coefficients were also double-checked to see if they made sense. For $perfor\_Y$, in some of the iterations, three randomly sampled coefficients were estimated to be $-0.34, 0.75$ and 0.13. By referring to Table 5.2 for the groundtruth coefficients, these three resulted in biases of $-33, 76$ and 14, respectively. Thus, not only was it overfitting, but the direction of the bias was inconsistent. It is not surprising that overfitting was the case, as that is generally the implication of reducing the sample size.[33,179]

Notice how the two candidates that projected the largest bias, namely $perfor\_Y$ and sex_M, were the two least significant coefficients from the proxy ground truth coefficients in Table 5.2. As these coefficients were close to 0, the division of these small numbers caused minor changes in the estimated coefficients to cause the bias to blow up. This means that while CCA overestimated the effect of these insignificant coefficients, the quantification of how much they overestimated may be difficult to interpret.

6.1.4 Concordance index

For the concordance index it is worth discussing the interesting behaviour after applying CCA. Recall that the training concordance index increased as the proportion of missingness increased for CCA from Figures 5.3 (MCAR) and 5.8 (MAR). It was clear overfitting, due to both of the test concordances for MCAR and MAR having a decreasing trend in comparison to the training's increasing trend. This is explained by the expected number of samples retained after removing the samples with missing values. Recall that the expected number of rows kept decreased rapidly as the proportion of missingness increased. For MCAR, at 25%, which yielded the highest train concordance, we expected to keep $(1 - 0.25)^9 \cdot 622 \approx 47$ rows, which was very little compared to the number of samples before omitting the rows with missing values. For MAR, we cannot calculate this as the variables are not independent. This highlights the importance of having a

[(2)] It would have been useful to provide the regularization strengths for each of the proportions of missingness, however, due to repeated simulations, it may vary. For that reason it was not included. This is also why including variable selection was infeasible; the variables selected can vary for each iteration.

test set available for evaluation or fitting models with cross-validation. Trusting only the training concordance can be very misleading.

As for the imputations, no data was lost, and the decrease in concordance was primarily related to the quality of the imputations. For MCAR, the decrease was rather rapid, and at the highest proportion of missing values, kNN and MICE had approximately a 5-6 % decrease in concordance. For MAR, both the proportion of retained variables and the proportion of missing values had joint effects. In particular, the prior had to decrease and the former to increase to result in lower concordance. This is no surprise, as the more variables are forced to be imputed, the higher the impact it has on the Cox PH model. Under MAR, when $p\_obs = \frac{3}{9}$ and $p\_miss = 0.5$, kNN and MICE had a loss of 3% and 4%, respectively. MAR being less affected than MCAR is not surprising, as, for MAR, three variables had no missingness compared to MCAR, where all variables had missingness.

6.1.5 Summary

With all the major points discussed, we can see that much of the findings are consistent with the literature and apply to survival analysis. For instance, the popular strategy of removing samples with missingness did indeed contribute to bias and loss of precision. In addition, the paper discussing that kNN performs the best on categorical data is also consistent with our findings.[93] On the contrary, the most unexpected result was the performance of MICE. According to (Buuren, 2018)[18], Austin et al. (2021)[7], Klebanoff and Cole (2008)[73] and Zhou et al. (2001)[182], it was expected to provide state-of-the-art imputations, but in many cases, a simple multivariate imputer such as kNN provided better imputations. Arguments can be made in favour of why MICE did not perform so well. For example by investigating the correlation matrix in Figure 4.2, notice how, after removing the variables *nodes4*, *time* and *status* which were not included in the imputations, *nodes* and *differ* had the highest absolute correlation of 0.15, which was very weak. It is also reasonable to say that there may have been insufficient covariates in the dataset. More covariates give more information for MICE to infer information from. Paired with little correlations, not much information was available to MICE to use for imputations. To complicate matters even more, the variables *perfor*, *adhere* and *extent* were all imbalanced, which challenged the imputations. The variations in all the figures highlight the importance of repeated simulations. In addition, we showed the disparity between the direct and indirect evaluations. As the accuracy (direct) was little affected by increasing the proportions of missingness, both the bias (indirect) and concordance (indirect) were negatively affected by the increase in missingness.

6.2 Experimental Setup: Survival Analysis (GEP NEN)

We separate the survival analysis discussion into two subsections, where we will discuss the most interesting results presented in Section 5.2. The first concerns the approaches for handling missing values and the other for the models and interpretation. For the first part, we consider the best approach that maximizes the test concordance.

6.2.1 Data Registration and Privacy

The medical domain differs from others, with strict rules for privacy and anonymity. This makes storing and handling data more difficult. We will, in addition, briefly discuss the problem of variation in covariates.

Doctors often detail medical data about patients on paper forms. Different doctors might register information on these forms differently, and sometimes mistakes can occur because they manually write down patient information. Thus, there is a higher chance of making a human error. The data from all these forms would have to be manually entered into a computer system like Excel for further analysis. If the forms are not registered correctly, this can affect patient

care and the study outcomes for which they are being used. For instance, incorrect data can lead to a misunderstanding of how effective a treatment is. In the GEP NEN dataset, it was identified that the *albumin* variable had values for two patients that were incorrectly documented as a consequence of converting from analogue to digital. This issue is further discussed in Section 6.2.8, which discusses the potential for data registration errors in patient information.

A limitation of the survival analysis of the GEP NEN dataset was the variation in the predicted survival times. One source of this variation was potential inconsistencies in when the covariates were registered from the time of diagnosis. For example, it was plausible that some patients had taken blood tests on the same day that they were diagnosed, whereas others may have waited a week or two. It was also highly unlikely that imaging (CT/MR) and blood tests were taken on the same day. Other examples are patients diagnosed at external hospitals, who may experience delays before being referred to Oslo University Hospital, where further testing happened. Such delays can span several weeks. These different gaps introduce inconsistencies in the data between patients and act as an additional source of unexplainable variation, causing discrepancies in the predicted survival times, making it challenging to calculate overall survival (OS) accurately. To address this issue, we propose that a solution is to downscale the target variable, making it less fine-grained at the cost of covering this unexplained variance. For example, instead of looking at overall survival in days, one could encode the target variable, *overall survival (OS)*, by months. This approach was implemented by Jenul et al. (2023)[65].

6.2.2 Sample Size

After preprocessing, the GEP NEN dataset ended up with 99 samples with 50 features. This relatively small dataset size may present challenges, such as limiting the statistical power of analysis, potentially leading to less reliable conclusions. Smaller datasets can make it harder to trust the study's results because there are not enough data points to ensure they are accurate.[33] This made it difficult to determine whether the results would be generalizable to all the patients in the dataset. So, it is important to be careful when interpreting the results and consider whether they apply to a larger patient group. Another argument for having more samples is to prevent overfitting. The analysis of missing values presented in Section 6.2.3 provided empirical evidence that overfitting was a consequence of loss of sample size.

6.2.3 The Missing Values

The main interest was to investigate whether imputing rather than removing the missing samples and variables was beneficial. We also discuss single versus multiple imputations.

Most conclusions from the experimental setup of missing values and literature were consistent with findings in the GEP NEN dataset. Tables 5.8, 5.9 and 5.10 show that no imputation was the poorest choice out of all strategies. Applying CCA instead of imputing resulted in approximately a 3-4% loss of test concordance for all survival analysis models. The expected concordance of imputations compared to CCA all showed significant differences using the t-tests. The results from the experimental setup of missing values from Section 5.1 highlighted that overfitting was present due to the reduction of sample size when only samples were removed. This indicates that the poor performance was likely a result of both loss of information from removing columns and overfitting from reducing the samples. Two of the variables that were removed was *Number of Courses* and *NSE*, which showed high feature importance in Figure 5.10. Recall from Section 4.4.5 that the strategy for generating the dataset without imputation was done by tuning the proportion of missingness required for removing columns and then removing the samples. After this, the dataset that yielded the highest test concordance was picked. This was done for each model. Thus, the CCA strategy favored the best combination of variables and samples to remove. This shows that the loss of columns was more beneficial for gaining a higher validation concordance as opposed to the loss of samples. It is also important to note that the original dataset had very

few samples, so it was not unexpected that performance decreased when discarding information. While we are not able to measure the bias of the coefficients because we do not have the ground truth ones, it is reasonable to assume that CCA introduced some bias.

For imputations, kNN performed slightly better than MICE when attempting to use the model-based imputation (only generating one imputed dataset) as seen in Table 5.9 and the multiple in Table 5.10; however, the differences were not significant. One would have expected MICE to benefit more, given all the variables available to use for imputations.[18] However, MICE has an important limitation: the assumption that data follows the MAR mechanism as mentioned in Section 3.6.3.[8] We uncovered in Section 4.1.2 that only 4 of the 25 variables with missingness respected the MAR mechanism, and all others were under MCAR. In addition, MICE benefited from using the complete dataset for imputations, whereas kNN imputation was done within the RSKF. While the hyperparameters and parameters for MICE followed the recommendations of Buuren (2018)[18], we suggest that maybe some alternative approaches could have been made. For instance, PMM was used as the imputation function for all numerical variables. However, there may have been an improvement in performance by using Bayesian linear regression.

We propose two reasons why kNN performed well. The first is that variables with low influx and outflux were omitted in the first place, which, as discussed in Section 3.6.2, contribute to more accurate distances. The second relates to the sparsity problem raised in Section 3.9. This is only an issue for numerical variables because they are susceptible to high cardinality compared to categorical variables.[180] Thus, numerical variables are likely to be scattered and yield non-representative tuple neighbors used in computing the average. Most of the variables in the GEP NEN dataset were categorical, and thus, the sparsity problem was less persistent. The few numerical variables present were much more prone to this problem, especially when considering the small sample size.

We suggest that the choice of imputation method should be decided depending on the application. For the GEP NEN dataset, kNN and MICE performed approximately the same, with kNN being slightly better. The benefit of kNN was that it is simple to use and only returns a single dataset. It was easier to work with a single dataset rather than multiple, especially when considering computational time. However, multiple imputations had the added benefit of quantifying the variation of the imputations; it was possible to look at accompanying distributions, which was not possible with kNN. Consider, for instance, investigating the imputations of samples for a specific variable after applying MICE. The distributions of the imputations of this variable will provide information on the agreement between the imputd datasets. If there is an agreement in the imputations for all the samples, it may not be that harmful for the variable to be exposed to missingness. On the contrary, if there is disagreement, it can be a helpful indicator of preventing missingness in that variable for future data collection. This can also be incorporated into machine learning by providing distributions over predictions of samples. This was shown under *Survival Curves* in Section 5.2.4. Note that because this process is sample-wise, it is unsuitable for large-scale datasets and is most likely not beneficial. We found that this is an effective strategy in small-scale datasets, like the GEP NEN, where specific patients are of interest.

6.2.4 Data Preprocessing

As mentioned in the material part (Section 4.1.2) of this thesis, the GEP NEN dataset consisted of two studies, forming three datasets: a new study, an old study, and a combination of both studies. The combined one had a larger sample size than the new and old but with fewer features. The validation C-index for the survival analysis models was higher for the new study dataset compared to the combined one. The underlying reason for this observation was that the combined dataset only included features shared by both the new and old study, thus excluding important features unique to the new study.

A major work for the thesis was on data preparation (Section 4.4.2). One instance was on how to handle categorical values such as 'Unknown' and 'Not Done'. We decided, in agreement with

expert knowledge, to treat these as missing data to simplify our analysis, reducing the complexity and the number of categories in features containing these values. This method made it easier to apply statistical models, but it risked losing potentially valuable information and introducing biases. Mirkes et al. (2016)[97] point out these concerns, especially in cases with 'Unknown' results, emphasizing the need to carefully consider how to classify such cases as missing data. Another approach we considered was grouping "Not Done" and "Unknown" into a single category. However, this would be unsuitable as each variable provides distinct information, and thus, it made little sense to treat them as the same.

In the preparation phase, due to time constraints, we had to decide how many features to exclude based on their missing values exceeding a certain percentage threshold. As outlined in the methodology section 4.4.2, we evaluated the features by comparing their influx and outflux to this threshold. As illustrated in Figure 4.9, we categorized the features into three groups. We initially retained the two groups with the highest and middle outflux and discarded the group with the lowest. Subsequently, we also removed the middle group for comparison. We observed no differences in performance, whether we removed just the lowest group or both the lowest and middle groups. However, eliminating only the lowest group retained more features for further analysis in our dataset. We initially tried to preserve more variables by allowing more missingness, though this caused MICE to give warnings that some predictors ended up as constant during imputations. This is because when MICE was imputing, only the complete part of the variable of interest was used.[18] Thus, if covariates were imbalanced, it was likely that some category levels would be omitted.

Figure 4.4 shows the PCA plot of the numerical variables of the patients. We measured the models' performance after removing the samples farthest from the cluster, even though they did not strictly appear as outliers. This was done by omitting samples outside of the interquartile range of 1.5. This only affected three samples. While we did not present figures or tables for this, the performance decreased. Thus, we decided to stay with all the samples. It was not aurprise, as none of the samples stood out as obvious outliers.

Encoding and Imputation Challenges

One-hot encoding is a common method for encoding nominal features. However, it becomes impractical with variables of many levels. It also comes with the problem of providing an additional column containing missing values (NAs) for all categories containing missing values that would have to be removed. However, this solution had the negative effect of removing potentially valuable information that could have been retained through imputation.

Instead, we chose target encoding (Section 2.5.1). This strategy preserved the original dimensions of the categorical variables by converting categories into the average of the target variable. This integrated information from the target variable into the encoded nominal variables, enhancing the model's ability to capture more nuanced representations of the categorical data. In this experiment, we observed improved model performance using target encoding compared to one-hot encoding. As discussed in Section 3.7, using target encoding on nominal variables allowed the computation of sensible distances and effective imputations. This was particularly crucial for imputing the missing values in the dataset when using kNN imputations.

A different challenge was the question of how much information to provide to the imputers. It could potentially be more beneficial for the imputers to have access to the whole dataset, generating higher-quality imputations at the cost of data leakage. In addition to this, including the target variable would be an option. These strategies may be suitable if the objective is to maximize the quality of the imputations and not be concerned with data leakage for the survival analysis models. However, imputations may overfit in the same manner as machine learning models, and thus, being able to validate the imputations may be preferred.

6.2.5 Performance Between the Different Models

Various visualization plots can help understand and interpret the efficacy of the different survival models: Coxnet, Cox PH, RSF, and CGB. This provides a deeper insight into the model's performance, feature importance, and prediction.

The four survival models we evaluated all have advantages and disadvantages based on the kind of data they are exposed to. The high dimensionality of large datasets with many features can challenge the Cox PH model by increasing computational time and may potentially overfit.[26,46] Conversely, the small dimensional size of the GEP NEN dataset may explain the strong performance of the Cox PH model, as illustrated in Table 5.5. Despite its simplicity, the Cox PH model performed as well as they other more complex survival models. In the Cox PH survival model, higher *alpha* values (Figure 5.5) suggest a complex underlying structure in the GEP NEN dataset, potentially leading to overfitting if regularization is too weak. In Tables 5.4 and 5.5, we notice a marginal performance difference between Coxnet and Cox PH models. Specifically, the Coxnet model (alpha: 3, L1 ratio: 0.01) achieved a performance of 79.0%, while the Cox PH model (alpha: 200) achieved 78.8%. Further, the highest C-index test scores of these models were 79.6 % and 78.9 %, respectively, underscoring the minimal gap that suggests regularization has a limited impact on reducing overfitting. The similar performance of the Cox PH model to other survival models suggests that feature selection was not essential for optimal performance for the C-index test. This could be attributed to the preprocessing steps that likely eliminated features that might have negatively impacted the model's performance.

If we now look at the RSF that also used feature selection described in Section 2.3.3, it becomes evident that regardless of the hyperparameters, the top 10 models overfitted. It may not have been sufficient information to generalize well. This limitation is also reported by Katzmann et al. (2020)[71] and Qiu et al. (2020)[123], whereas the latter also reported that Cox PH outperformed RSF. This may happen since RSF uses bootstrap with replacement[60,61], which does not extend the variety of data when the original sample size is limited. Additionally, survival trees within the RSF make splits. With small sample data, these splits lead to even smaller subsamples, making calculations and predictions highly specific to small subsets rather than offering robust generalization for unseen data.[88] Similar to RSF, CGB is another ensemble model. According to Table 5.7, the C-index scores on the test folds for CGB reveal marginal differences between the RSF and CGB models, ranging from 0.777 to 0.789. These minor variations suggest that the performance of both models are nearly equivalent. However, the CGB model overfitted less than RSF from Table 5.6.

As described in Section 2.3.4, CGB is a sequential model that uses the Cox PH model as a loss function and least squares as a base learner.[91] In CGB, each base learner aims to minimize these residuals iteratively, which is crucial for effectively utilizing the limited data in small datasets. Each stage of CGB corrects the residuals left by the previous one, refining the model´s performance incrementally. This sequential learning corrects previous errors, naturally regularizing the process to prevent overfitting the training data. Additionally, since the optimal combination for CGB model had a *subsample* hyperparameter of less than 1 (SGB), it strategically reduces model variance and may decrease bias (Section 2.3.4).[36] This method effectively improves model robustness and generalizability, improving the performance of CGB models.

Brier Score

The C-index for the test folds suggested marginal differences across all models, with Coxnet, Cox PH, and CGB demonstrating less overfitting. Additionally, RSF and CGB exhibited greater stability in IBS, showing minimal variation compared to Coxnet and Cox PH. Despite Coxnet's strong C-index performance, its IBS comparison suggests potential inconsistency in predictive reliability across survival times (Figure 5.11). One issue with CGB is its computational complexity in terms of memory and runtime.[134]

Survival Curves

As shown in Figures 5.12 and 5.13, the CGB performed better than Coxnet when the observed time was short. Seow et al. (2024)[143] also report that Component-wise regression as base learner outperforms other models for short survival time. They used the same models Coxnet, Cox PH, RSF and CGB, in addition to others.

From the performance metrics evaluated across all survival models, Coxnet and CGB stood out as the top performers, achieving the highest C-index scores with minimal overfitting. Figure 5.12 compares these two survival models using their survival curves by RSKF, and Figure 5.13 compares the same models using the same hyperparameters by multiple imputations.

The Appendix Figure B.3 demonstrates that Coxnet's survival curves exhibit greater similarity compared to CGB. This difference becomes clearer in Figure B.2, where the survival curves are noticeably more similar for the Coxnet model (*alpha*: 1000, *L1 ratio*: 0.0001) than for the Coxnet model (*alpha*: 3, *L1 ratio*: 0.01). This might be linked to the higher *alpha* values, which, by shrinking the coefficient weights toward zero, align the model more closely with the baseline hazard. The survival curves resemble an exponential function, which is a reasonable approximation because patients are expected to die early for this cancer. It may suggest that the simpler survival functions in Figures 5.12 and 5.13 represented the C-index for Coxnet well, and as there was little variation in the curves, IBS was less suitable, seen for the *L1 Ratio* 0.0001 in Figure 5.9. A noteworthy observation was that as the IBS improved, there was an increase in the variability of the survival curves. This is clearly demonstrated in Appendix B, Figure B.2. Here, we compare the Coxnet model with the highest test C-index to the Coxnet model chosen earlier (Section 5.2.1) (*alpha*: 3, *L1 Ratio*: 0.01), which shows improved IBS.

This shows that reflecting on the metrics used to evaluate predictive performance in CGB and Coxnet is important, and one could also reason that choice of metric depends on the application. Consider the case where predicting the order of events is more important than the event duration. Suppose two patients are at high risk of experiencing an event and both require immediate medical attention, where the medical staff is shorthanded. By predicting that one of them experiences the event before the other, it allows healthcare providers to prioritize the interventions. In this case, the C-index is a more suitable metric compared to IBS.

For multiple imputations in Figure 5.13, it was clear that imputations influenced CGB more than Coxnet. This relates back to the proposition that Coxnet might have leaned towards the baseline hazard and not put much emphasis on the covariates. Thus, the quality of the imputations may not be as important for Coxnet as it was for CGB. Note that no information was provided on what variables were imputed or not. For patient 9007, which had accurate survival curves for both kNN and MICE, it may be that the imputations were suitable for that patient or that they had no missing values in the first place. Alternately, the fact that they died early may be an important factor for the models. This would be potential for future analysis. A quick note regarding the survival curves for multiple imputations was that they did not consider the variation when shuffling the covariates like the survival curves with RSKF. Adding this extra source of variation could provide an additional figure where both the variation from imputations and covariates are present.

6.2.6 Evaluation Using Cross-Validation

Since we worked with a small dataset, the possibilities for evaluating the models were limited. It is crucial to use a robust method for model evaluation when optimizing hyperparameters. To address this, we used repeated stratified K-fold cross-validation (RSKF)[64,76,125] described in Section 2.4.1. This technique was chosen for two primary reasons: Firstly, it reduces the variance in model evaluation by repeating the cross-validation process multiple times. Each repetition can yield slightly different results due to the random shuffling in creating the folds. Averaging the results of these repetitions provides a more consistent estimate of the model's performance. Secondly, RSKF effectively manages imbalanced data, a challenge our dataset faces due to the unbalanced

distribution of censored data. Additionally, RSKF helps prevent overfitting by creating multiple splits. In this approach, the model is repeatedly trained and tested on different subsets of data, which enhances its ability to generalize well to new unseen data. In machine learning, using an additional validation set is common practice.[175] However, due to the limited size of our dataset, the subset for the validation set would be quite small. This could lead to challenges, particularly with imbalanced features in the validation subset. RSKF also provide information on how much variation is introduced by alternating the samples used in the folds.

6.2.7 Features Coefficients for Different Alphas

The variables *Absolute Neutrophil Count* and *Chemotherapy Type* had their coefficients alternate in direction as the regularization strength increased. From a logical point of view, it makes no sense for a variable to have both a negative and positive impact on the hazard. From the permutation feature importance in Figure 5.15, we can see that they were ranked 9th and 16th in importance, which suggests that both features were relatively important for the model. This change in direction may suggest that the variables were not stable as they were sensitive to the regularization strength. This also highlights the difference between the statistical approach of comparing weights and the machine learning approach by comparing permutation importance.

6.2.8 Permutation Features Importance

In Figure 5.10, the CGB model consistently assigned higher importance to variables compared to the other survival analysis models. This is most noticeable for the top four important features. These indicate that the CGB was more sensitive to these features being permuted and relied more on them for predictions than the other models. One reason for this heightened sensitivity could be, as described in Section 2.3.4, that CGB uses gradient boosting[56,35], which builds models sequentially and corrects the mistakes of previous models. As a result, CGB may accentuate the influence of features that provide the strongest signals for correcting errors, leading to these higher-importance values. Consequently, CGB might focus more on the features that are best at fixing mistakes, which explains why these features have higher importance values. It also was supported by the survival curve Figures 5.12 and 5.13 where it is shown that CGB was more affected by the varying the variables in RSKF. This also applied to MICE, where imputations on CGB had more influence on the variation in survival curves compared to Coxnet.

As reported in Jenul et al. (2023)[65], the features *WHO Perf Stat*, *Ki-67*, and *Albumin* were identified as both the most stable and predictive. These were also some of the top contenders in Figure 5.10. As reported in Sorbye et al. (2013)[151] *LDH* is the most important negative feature in a multivariate Cox analysis. However, Figure 5.10 reveals that *LDH* is ranked as the 14th most important feature across survival models. Specifically, *LDH* was found to be most important for RSF and least for Coxnet, Cox PH, and CGB. Additionally, the average permutation feature importance for Coxnet (Figure 5.15) shows *LDH* at rank 19. This ranking clearly differs from the findings of Sorbye et al. (2013), where *LDH* was identified as an important feature. Our findings highlighted that there was a large variation in the feature importance between the different models.

During this experiment, it was noted that Albumin ranked among the top 15 most important features on average, had negative importance in the Coxnet model, unlike the other survival models. Upon analyzing the feature's distribution, it was discovered that incorrect values from two patients influenced the models. This was particularly evident in the Coxnet model, which performed approximately 1% better after correcting these errors.

The permutation feature importance and feature coefficient weights, as shown in Figures 5.15 and 5.16, illustrate the differences between the machine learning approach and the statistical approach. The three most important features of the Coxnet model, as shown in Figure 5.15 was *Number of Courses*, *Ki-67*, and *NSE*. These was also the features with the largest coefficients in Figure 5.16. Additionally, the features are approximately in a similar order of importance in both

Figures 5.15 and 5.16. This shows that there are minimal differences between the importance of features of the two approaches and indicates that the model was stable.

7 Conclusion

By reviewing the literature, conducting a robust experiment on the colon dataset, and analyzing the GEP NEN dataset, we have highlighted the importance of imputing missing data rather than discarding information in variables and samples. Removing variables led to loss of information and potentially reduced information for models, and loss of samples tends to make models prone to overfitting. In addition, we have discussed the missingness mechanisms and various imputation strategies. The violation of the MAR assumption for MICE had a negative effect on imputations. We found that kNN provided promising imputations under variables respecting the MCAR mechanism. We also briefly discussed the use cases of single versus multiple imputations; single imputation was useful when the uncertainty of imputations was not of interest, whereas multiple provided the option of assessing sample-by-sample uncertainty and agreement between imputations.

Assuming kNN imputation as the optimal single imputer, we explored various survival analysis models, including Coxnet, Cox PH, RSF, and CGB. Both Coxnet and Cox PH, alongside CGB, performed highest for the C-index and lowest for IBS, whereas RSF seemed to overfit too much. By comparing Coxnet and CGB, we found that CGB gave more variation across all survival curves as opposed to the Coxnet model. That implies that CGB was more honest with its predictive uncertainty when there was a large deviation between the survival curves. The Coxnet model´s survival curves closely resembled an exponential function, especially at higher regularization *alpha* values and lower regularization *L1 ratio*. This suggests that an approximation of exponential survival function may be suitable, which is anticipated as patients are expected to die early for this cancer. This could also be why the IBS score had little variation, and maybe the C-index was too optimistic. The choice of survival model, CGB, obtained a high C-index but also got the lowest Brier score compared to the other models, showing that CGB was the most optimal model for this dataset, even if it was more complex. Finally, features such as *Number of Courses*, *Ki-67*, and *NSE* were identified as having the most average importance across the four survival models. Additionally, features *WHO Perf Stat*, *Ki-67*, and *Albumin* were identified as equally important, consistent with the results reported by Jenul et al.(2023)[65].

8 Future work

Missing values

If time were not a bottleneck, we would have included and evaluated additional imputation strategies in the experimental setup with the colon dataset and potentially applied them to the GEP NEN dataset. In particular, it would have been interesting to dive deeper into local imputation strategies that modify kNN, especially considering how well kNN performed. Zhang et al. (2019) proposed a similar idea, where local regression models were applied to each complete tuple together with their closest neighbors, where imputations are given as the pooled predictions of each model.[180] This combats both the sparsity and heterogeneity issue presented in Section 3.9. Their implementation was done in Java, and it would be a major work converting to Python.[181] Exploring Bayesian imputations would be of interest. Imputations could be adjusted to be more accurate by including prior information based on expert knowledge. As a final point for imputations, it would have been interesting to investigate multiple imputations where all possible information is provided, such as the survival times or the cumulative hazard, as suggested by Buuren (2018).[18]

Survival analysis

Imputation strategies were paramount for the thesis. It would have been interesting to investigate the imputations on a sample-by-sample basis to reveal the strengths and weaknesses of the imputations. For example, some patients may be easier to impute, whereas others may be challenging. By collaboration with expert knowledge, it could also be possible to evaluate if the imputations were valid.

We have also determined that the number of samples was crucial for model performance. Consequently, utilizing the combined dataset to further improve the models can be future work. It would be interesting to not only analyze the shared features between the old and new studies within this dataset but also to utilize the unique features specific to each study. This approach would incorporate features that, for example, contain missing values from the first part of the dataset, corresponding to the old study, alongside features from the last part, indicative of the new study. Investigating the efficacy of imputation on this combined dataset, followed by model fitting, could provide insights into whether such strategies improve performance.

Finally, given the variability in overall survival (OS) times, one potential approach is to encode OS in months, as suggested by Jenul et al. (2023)[65], or alternatively, in weeks. This adjustment offers an interesting opportunity to observe whether the survival models leads to any improvements in performance.

References

[1] H. Abdi and L.J. Williams. "Principal component analysis". In: *WIREs Computational Statistics* 2 (2010), pp. 433–459. DOI: 10.1002/wics.101.

[2] Hussam Alkikassi and Samy I. McFarlane. "Artificial Hallucinations in ChatGPT: Implications in Scientific Writing". In: *Cureus* (2023). DOI: 10.7759/cureus.35179.

[3] Alonso Silva Allende. *Concordance Index as an Evaluation Metric*. 2023. URL: https://medium.com/analytics-vidhya/concordance-index-72298c11eac7 (visited on 02/21/2024).

[4] Paul D. Allison. *Missing Data*. Thousand Oaks: Sage Publications, 2002. DOI: 10.4135/9781412985079.

[5] Gareth Ambler, Rumana Z Omar, and Patrick Royston. "A comparison of imputation techniques for handling missing predictor values in a risk model with a binary outcome". In: *Statistical Methods in Medical Research* 16 (2007), pp. 277–298. DOI: 10.1177/0962280206074466.

[6] Christos Anagnostopoulos and Peter Triantafillou. "Scaling out big data missing value imputations: pythia vs. godzilla". In: 2014, pp. 651–660. DOI: https://doi.org/10.1145/2623330.2623615.

[7] Peter C Austin et al. "Missing Data in Clinical Research: A Tutorial on Multiple Imputation". In: *Canadian Journal of Cardiology* 37.9 (2021), pp. 1322–1331. DOI: 10.1016/j.cjca.2020.11.010.

[8] Melissa J. Azur et al. "Multiple imputation by chained equations: what is it and how does it work?" In: *International Journal of Methods in Psychiatric Research* 20.1 (2011), pp. 40–49. DOI: 10.1002/mpr.329.

[9] Axel Benner et al. "High-dimensional Cox models: the choice of penalty as part of the model building process". In: *Biometrical Journal* 52.1 (2010), pp. 50–69. DOI: 10.1002/bimj.200900064.

[10] Paul Blanche, Michael W Kattan, and Thomas A Gerds. "The c-index is not proper for the evaluation of t-year predicted risks". In: *Biostatistics* 20.2 (2019), pp. 347–357. DOI: 10.1093/biostatistics/kxy006.

[11] Leo Breiman. "Random Forests". In: *Machine Learning* 45 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[12] N. Breslow. "Covariance Analysis of Censored Survival Data". In: *Biometrics* 30.1 (1974), pp. 89–99. DOI: 10.2307/2529620.

[13] Inga Buchmann et al. "Comparison of 68Ga-DOTATOC PET and 111In-DTPAOC (Octreoscan) SPECT in patients with neuroendocrine tumours". In: *European Journal of Nuclear Medicine and Molecular Imaging* 34 (2007), pp. 1617–1626. DOI: 10.1007/s00259-007-0450-1.

[14] F Bugnard, C Ducrot, and D Calavas. "Advantages and inconveniences of the Cox model compared with the logistic model: application to a study of risk factors of nursing cow infertility". In: *Veterinary Research* 25.2-3 (1994), pp. 134–139. URL: https://hal.archives-ouvertes.fr/ffhal-00902184.

[15] F. Bugnard, C. Ducrot, and D. Calavas. "Advantages and inconveniences of the Cox model compared with the logistic model: application to a study of risk factors of nursing cow infertility". In: *Vet Res* 25.2-3 (1994), pp. 134–139.

[16] Lars Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[17] A Burton and DG Altman. "Missing covariate data within cancer prognostic studies; a review of current reporting and proposed guidelines". In: *British Journal of Cancer* 91 (2004), pp. 4–8. DOI: 10.1038/sj.bjc.6601907.

[18] Stef van Buuren. "Flexible Imputation of Missing Data, Second Edition". In: (2018). DOI: 10.1201/9780429492259.

[19] Stef van Buuren. *Get at the final model used in the MICE iterations?* 2017. URL: https://github.com/amices/mice/discussions/346 (visited on 04/08/2024).

[20] Stef van Buuren and Karin Groothuis-Oudshoorn. "mice: Multivariate Imputation by Chained Equations in R". In: *Journal of Statistical Software* 45.3 (2011), pp. 1–67. DOI: 10.18637/jss.v045.i03.

[21] Songchao Chen et al. "Probability mapping of soil thickness by random survival forest at a national scale". In: *Geoderma* 344 (2019), pp. 184–194. DOI: 10.1016/j.geoderma.2019.03.016.

[22] TG Clarke and DG Altman. "Developing a prognostic model in the presence of missing data: an ovarian cancer case study". In: *Journal of Clinical Epidemiology* 56 (2003), pp. 28–37. DOI: 10.1016/s0895-4356(02)00539-5.

[23] J. S. Cramer. *The Origins of Logistic Regression*. Technical Report. Tinbergen Institute, 2002. DOI: 10.2139/ssrn.360300.

[24] J. A. Cruz and D. S. Wishart. "Applications of Machine Learning in Cancer Prediction and Prognosis". In: *Cancer Informatics* 2 (2006). DOI: 10.1177/117693510600200030.

[25] T. Curran, S. Tulin-Silver, K. Patel, et al. "Prognostic clinicopathologic factors in longitudinally followed patients with metastatic small bowel carcinoid tumors". In: *Pancreas* 40.8 (2011), pp. 1253–1257. DOI: 10.1097/MPA.0b013e318225483c.

[26] X. Dang, S. Huang, and X. Qian. "Penalized Cox's proportional hazards model for high-dimensional survival data with grouped predictors". In: *Statistics and Computing* 31 (2021), p. 77. DOI: 10.1007/s11222-021-10052-4.

[27] Cameron Davidson-Pilon. "lifelines: survival analysis in Python". In: *Journal of Open Source Software* 4.40 (2019), p. 1317. DOI: 10.21105/joss.01317.

[28] A. Desiani et al. "Handling Missing Data Using Combination of Deletion Technique, Mean, Mode and Artificial Neural Network Imputation for Heart Disease Dataset". In: *Science and Technology Indonesia* 6.4 (2021), pp. 303–312. DOI: 10.26554/sti.2021.6.4.303-312.

[29] A. Rogier T. Donders et al. "Review: A gentle introduction to imputation of missing values". In: *Journal of Clinical Epidemiology* 59.10 (2006), pp. 1087–1091. DOI: https://doi.org/10.1016/j.jclinepi.2006.01.014.

[30] S. A. ElHafeez et al. "Methods to Analyze Time-to-Event Data: The Cox Regression Analysis". In: *Oxidative Medicine and Cellular Longevity* 2021 (2021). DOI: 10.1155/2021/1302811.

[31] T. Emmanuel, T. Maupong, D. Mpoeleng, et al. "A survey on missing data in machine learning". In: *Journal of Big Data* 8.140 (2021). DOI: 10.1186/s40537-021-00516-9.

[32] J. Engel. "Polytomous logistic regression". In: *Statistica Neerlandica* 42.4 (1988), pp. 233–252. DOI: https://doi.org/10.1111/j.1467-9574.1988.tb01238.x.

[33] J Faber and LM Fonseca. "How sample size influences research outcomes". In: *Dental Press Journal of Orthodontics* 19.4 (2014), pp. 27–29. DOI: 10.1590/2176-9451.19.4.027-029.ebo.

[34] R. Flynn. "Survival analysis". In: *Journal of Clinical Nursing* 21.19-20 (2012), pp. 2789–2797. DOI: 10.1111/j.1365-2702.2011.04023.x.

[35] Jerome H Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". In: *The Annals of Statistics* 29.5 (2001), pp. 1189–1232. URL: http://www.jstor.org/stable/2699986.

[36] Jerome H Friedman. "Stochastic gradient boosting". In: *Computational Statistics & Data Analysis* 38.4 (2002), pp. 367–378. ISSN: 0167-9473. DOI: 10.1016/S0167-9473(01)00065-2.

[37] Johannes Fürnkranz and Peter A. Flach. "An Analysis of Rule Evaluation Metrics". In: *International Conference on Machine Learning*. 2003. URL: https://api.semanticscholar.org/CorpusID:3207474.

[38] E. Graf et al. "Assessment and comparison of prognostic classification schemes for survival data". In: *Statistics in Medicine* 18.17-18 (1999), pp. 2529–2545. DOI: 10.1002/(SICI)1097-0258(19990915/30)18:17/18<2529::AID-SIM274>3.0.CO;2-5.

[39] John W Graham. "Missing data analysis: making it work in the real world". In: *Annual Review of Psychology* 60 (2009), pp. 549–576. DOI: 10.1146/annurev.psych.58.110405.085530.

[40] Grammarly. *About Grammarly*. 2024. URL: https://www.grammarly.com/about (visited on 03/25/2024).

[41] Michael Greenacre, Patrick J. F. Groenen, Trevor Hastie, et al. "Principal component analysis". In: *Nature Reviews Methods Primers* 2 (2022), p. 100. DOI: 10.1038/s43586-022-00184-w.

[42] S Greenland and WD Finkle. "A critical look at methods for handling missing covariates in epidemiologic regression analyses". In: *American Journal of Epidemiology* 142 (1995), pp. 1255–1264.

[43] Leonardo Grilli and Carla Rampichini. "Ordered Logit Model". In: *Encyclopedia of Quality of Life and Well-Being Research*. Ed. by Alex C. Michalos. Dordrecht: Springer, 2014. DOI: 10.1007/978-94-007-0753-5_2023.

[44] Frank E. Grubbs. "Procedures for detecting outlying observations in samples". In: *Technometrics* 11.1 (1969), pp. 1–21. DOI: 10.1080/00401706.1969.10490657.

[45] Damodar N. Gujarati. *Essentials of Econometrics*. 3rd ed. McGraw-Hill Irwin, 2006, p. 110.

[46] Boyi Guo and Nengjun Yi. *A scalable and flexible Cox proportional hazards model for high-dimensional survival prediction and functional selection*. 2022. arXiv: 2205.11600 [stat.ME].

[47] Humza Haider et al. *Effective Ways to Build and Evaluate Individual Survival Distributions*. 2018. arXiv: 1811.11347 [cs.LG].

[48] Michael J. Pencina Hajime Uno Tianxi Cai. "On the C-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data". In: *Statistics in Medicine* 30.10 (2011), pp. 1105–1117. DOI: 10.1002/sim.4201.

[49] F. E. Jr Harrell, K. L. Lee, and D. B. Mark. "Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors". In: *Statistical Medicine* 15.4 (1996), pp. 361–387. DOI: 10.1002/(SICI)1097-0258(19960229)15:4<361::AID-SIM168>3.0.CO;2-4.

[50] FE Harrell. *Regression Modeling Strategies*. Springer, 2001.

[51] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[52] N Hartman et al. "Pitfalls of the concordance index for survival outcomes". In: *Statistics in Medicine* 42.13 (2023), pp. 2179–2190. DOI: 10.1002/sim.9717.

[53] Edward Hearn and Brennan Whitfield. *Ordinal Data vs. Nominal Data: What's the Difference?* Updated by Brennan Whitfield on October 26, 2023. 2023. URL: https://builtin.com/articles/ordinal-data (visited on 04/10/2024).

[54] David W. Hosmer, Stanley Lemeshow, and Susanne May. *Applied Survival Analysis: Regression Modeling of Time-to-Event Data*. 2nd. Hoboken, N.J: Wiley-Interscience, 2008. ISBN: 978-0-387-95399-1.

[55] Mohammad Hossin and Mohd Nordin Sulaiman. "A Review on Evaluation Metrics for Data Classification Evaluations". In: *International Journal of Data Mining & Knowledge Management Process* 5.2 (2015), p. 1. DOI: 10.5121/ijdkp.2015.5201.

[56] Torsten Hothorn et al. "Survival ensembles". In: *Biostatistics* 7.3 (2006), pp. 355–373. DOI: 10.1093/biostatistics/kxj011.

[57] R Houari et al. "Handling missing data problems with sampling methods". In: *2014 International Conference on Advanced Networking Distributed Systems and Applications*. IEEE. 2014, pp. 99–104.

[58] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[59] G. Hutcheson. "Missing Data: data replacement and imputation". In: *Journal of Modelling in Management* 7.2 (2012). DOI: 10.1108/jm2.2012.29707baa.002.

[60] Hemant Ishwaran et al. "Random survival forests". In: *The Annals of Applied Statistics* 2.3 (Sept. 2008), pp. 841–860. DOI: 10.1214/08-AOAS169.

[61] Hemant Ishwaran et al. "Random Survival Forests for R". In: *The R Journal* 7.2 (2007), pp. 1–25. URL: https://journal.r-project.org/articles/RN-2007-015/RN-2007-015.pdf (visited on 03/02/2024).

[62] Harold R. Jacobs. *Mathematics: A Human Endeavor*. 3rd ed. W. H. Freeman, 1994, p. 547. ISBN: 0-7167-2426-X.

[63] A. Jadhav, D. Pramod, and K. Ramanathan. "Comparison of Performance of Data Imputation Methods for Numeric Dataset". In: *Applied Artificial Intelligence* 33.10 (2019), pp. 913–933. DOI: 10.1080/08839514.2019.1637138.

[64] Gareth James et al. *An Introduction to Statistical Learning: with Applications in Python*. 1st ed. Springer Texts in Statistics. Springer Cham, 2023. ISBN: 978-3-031-38747-0. DOI: 10.1007/978-3-031-38747-0.

[65] Anna Jenul et al. "Novel ensemble feature selection techniques applied to high-grade gastroenteropancreatic neuroendocrine neoplasms for the prediction of survival". In: *Computer Methods and Programs in Biomedicine* 244 (2024), p. 107934. ISSN: 0169-2607. DOI: 10.1016/j.cmpb.2023.107934.

[66] I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. New York: Springer-Verlag, 2002. ISBN: 978-0-387-95442-4. DOI: 10.1007/b98835.

[67] MP Jones. "Indicator and Stratification Methods for Missing Explanatory Variables in Multiple Linear Regression". In: *Journal of the American Statistical Association* 91.433 (1996), pp. 222–230.

[68] Tomasz Kalinowski et al. *reticulate: Interface to 'Python'*. R package version 1.35.0. RStudio. https://github.com/rstudio/reticulate, 2024. URL: https://rstudio.github.io/reticulate/ (visited on 03/04/2024).

[69] Graham Kalton and Leslie Kish. "Some Efficient Random Imputation Methods". In: *Communications in Statistics - Theory and Methods* 13.16 (1984), pp. 1919–1939. DOI: 10.1080/03610928408828805.

[70] Hyun Kang. "The prevention and handling of the missing data". In: *Korean Journal of Anesthesiology* 64.5 (2013). PMID: 23741561; PMCID: PMC3668100, pp. 402–406. DOI: 10.4097/kjae.2013.64.5.402.

[71] Alexander Katzmann et al. "Deep Random Forests for Small Sample Size Prediction with Medical Imaging Data". In: *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*. 2020, pp. 1543–1547. DOI: 10.1109/ISBI45749.2020.9098420.

[72] Joseph Kearney and Shahid Barkat. *Autoimpute: A Python Package for Analysis and Implementation of Imputation Methods*. latest. Python 3.8+. 2022. URL: https://autoimpute.readthedocs.io/en/latest/ (visited on 03/02/2024).

[73] Mark A. Klebanoff and Stephen R. Cole. "Use of Multiple Imputation in the Epidemiologic Literature". In: *American Journal of Epidemiology* 168.4 (2008). DOI: 10.1093/aje/kwn071.

[74] John P. Klein and Melvin L. Moeschberger. *Survival Analysis: Techniques for Censored and Truncated Data*. 2nd ed. Statistics for Biology and Health. New York, NY: Springer Science+Business Media New York, 2003, pp. XVI, 538. ISBN: 978-0-387-95399-1. DOI: 10.1007/b97377.

[75] David G. Kleinbaum and Mitchel Klein. *Survival Analysis: A Self-Learning Text*. 3rd. New York, NY: Springer, 2020.

[76] Ron Kohavi. "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection". In: *International Joint Conference on Artificial Intelligence*. Stanford University. Stanford, CA 94305, 1995. URL: https://www.ijcai.org/Proceedings/95-2/Papers/016.pdf (visited on 03/02/2024).

[77] Konstantina Kourou et al. "Machine learning applications in cancer prognosis and prediction". In: *Computational and Structural Biotechnology Journal* 13 (2015), pp. 8–17. ISSN: 2001-0370. DOI: https://doi.org/10.1016/j.csbj.2014.11.005.

[78] Chanyeong Kwak and Alan Clayton-Matthews. "Multinomial Logistic Regression". In: *Nursing Research* 51.6 (Nov. 2002), pp. 404–410.

[79] K. Landerholm et al. "Survival and prognostic factors in patients with small bowel carcinoid tumour". In: *British Journal of Surgery* 98.11 (2011), pp. 1617–1624. DOI: 10.1002/bjs.7649.

[80] C. S. Landry et al. "Proposed staging system for gastrointestinal carcinoid tumors". In: *The American Surgeon* 74.5 (2008), pp. 418–422. DOI: 10.1177/000313480807400511.

[81] JA Laurie et al. "Surgical adjuvant therapy of large-bowel carcinoma: An evaluation of levamisole and the combination of levamisole and fluorouracil: The North Central Cancer Treatment Group and the Mayo Clinic". In: *J Clinical Oncology* 7 (1989), pp. 1447–1456.

[82] Michael P. LaValley. "Logistic Regression". In: *Circulation* 117.18 (2008), pp. 2395–2399. DOI: 10.1161/CIRCULATIONAHA.106.682658.

[83] Ian Leboo. *Introduction to Survival Analysis*. Jan. 2023. URL: https://medium.com/@Statistician_Leboo/introduction-to-survival-analysis-992cd4520d4f (visited on 01/10/2024).

[84] JH Lee and JC Jr. Huber. "Evaluation of Multiple Imputation with Large Proportions of Missing Data: How Much Is Too Much?" In: *Iran Journal of Public Health* 50.7 (2021), pp. 1372–1380. DOI: 10.18502/ijph.v50i7.6626.

[85] WC Lin and CF Tsai. "Missing value imputation: a review and analysis of the literature (2006–2017)". In: *Artificial Intelligence Review* 53 (2020), pp. 1487–1509. DOI: 10.1007/s10462-019-09709-4.

[86] Roderick J A Little and Donald B Rubin. *Statistical analysis with missing data*. USA: John Wiley & Sons, Inc., 1986. ISBN: 0471802549.

[87] Roderick J. A. Little. “Regression With Missing X’s: A Review”. In: *Journal of the American Statistical Association* 87.420 (1992), pp. 1227–1237. DOI: 10.2307/2290664.

[88] Jing Luan et al. “The predictive performances of random forest models with limited sample size and different species traits”. In: *Fisheries Research* 227 (2020), p. 105534. ISSN: 0165-7836. DOI: 10.1016/j.fishres.2020.105534.

[89] G. S. Maddala. “Outliers”. In: *Introduction to Econometrics*. 2nd ed. New York: MacMillan, 1992, p. 89. ISBN: 978-0-02-374545-4.

[90] MathWorks. *Cox Proportional Hazard Regression*. 2023. URL: https://se.mathworks.com/help/stats/cox-proportional-hazard-regression.html (visited on 02/21/2024).

[91] A. Mayr, B. Hofner, and M. Schmid. “The importance of knowing when to stop. A sequential stopping rule for component-wise gradient boosting”. In: *Methods of Information in Medicine* 51.2 (2012), pp. 178–186. DOI: 10.3414/ME11-02-0030.

[92] Wes McKinney. “Data Structures for Statistical Computing in Python”. In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[93] Shaheen MZ. Memon, Robert Wamala, and Ignace H. Kabano. “A comparison of imputation methods for categorical data”. In: *Informatics in Medicine Unlocked* 42 (2023), p. 101382. ISSN: 2352-9148. DOI: https://doi.org/10.1016/j.imu.2023.101382.

[94] Daniele Micci-Barreca. “A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems”. In: *SIGKDD Explor. Newsl.* 3.1 (July 2001), pp. 27–32. ISSN: 1931-0145. DOI: 10.1145/507533.507538.

[95] Microsoft. *About GitHub Copilot*. 2024. URL: https://docs.github.com/en/copilot/about-github-copilot.

[96] H. C. Miller, P. Drymousis, R. Flora, et al. “Role of Ki-67 Proliferation Index in the Assessment of Patients with Neuroendocrine Neoplasias Regarding the Stage of Disease”. In: *World Journal of Surgery* 38 (2014), pp. 1353–1361. DOI: 10.1007/s00268-014-2451-0.

[97] E. M. Mirkes et al. “Handling missing data in large healthcare dataset: A case study of unknown trauma outcomes”. In: *Computers in Biology and Medicine* 75 (2016), pp. 203–216. ISSN: 0010-4825. DOI: 10.1016/j.compbiomed.2016.06.004.

[98] U. Mogensen, H. Ishwaran, and T. Gerds. “Evaluating Random Forests for Survival Analysis Using Prediction Error Curves”. In: *Journal of Statistical Software* 50.11 (2012), pp. 1–23. DOI: 10.18637/jss.v050.i11.

[99] Boris Muzellec et al. “Missing Data Imputation using Optimal Transport”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 7130–7140. (Visited on 03/02/2024).

[100] Boris Muzellec et al. *Missing Data Imputation using Optimal Transport (GitHub)*. PMLR, 2020. URL: https://github.com/BorisMuzellec/MissingDataOT/tree/master.

[101] I. D. Nagtegaal et al. “The 2019 WHO classification of tumours of the digestive system”. In: *Histopathology* 76.2 (2020). Epub 2019 Nov 13, pp. 182–188. DOI: 10.1111/his.13975.

[102] Shinichi Nakagawa and Holger Schielzeth. “A general and simple method for obtaining R2 from generalized linear mixed-effects models”. In: *Methods in Ecology and Evolution* 4.2 (2013), pp. 133–142. DOI: 10.1111/j.2041-210x.2012.00261.x.

[103] D. A. Newman. “Longitudinal modeling with randomly and systematically missing data: A simulation of ad hoc, maximum likelihood, and multiple imputation techniques”. In: *Organizational Research Methods* 6 (2003), pp. 328–362.

[104] Daniel A. Newman. “Missing Data: Five Practical Guidelines”. In: *Organizational Research Methods* 17.4 (2014), pp. 372–411. DOI: 10.1177/1094428114548590.

[105] Norwegian University of Life Sciences. *Kunstig intelligens ved REALTEK*. Norwegian University of Life Sciences (NMBU). 2024. URL: https://www.nmbu.no/fakulteter/fakultet-realfag-og-teknologi/kunstig-intelligens-ved-realtek (visited on 05/14/2024).

[106] Hendro Nugroho, Nyoman P. Utama, and Kridanto Surendro. "Smoothing target encoding and class center-based firefly algorithm for handling missing values in categorical variable". In: *Journal of Big Data* 10 (2023), p. 10. DOI: 10.1186/s40537-022-00679-z.

[107] Ziad Obermeyer and Ezekiel J. Emanuel. "Predicting the future—big data, machine learning, and clinical medicine". In: *New England Journal of Medicine* 375 (2016), pp. 1216–1219. DOI: 10.1056/NEJMp1606181.

[108] OpenAI. *GPT-4: OpenAI's Advanced Language Model*. 2023. URL: https://openai.com/gpt-4 (visited on 01/31/2024).

[109] Florian Pargent, Felix Pfisterer, Janek Thomas, et al. "Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features". In: *Computational Statistics* 37 (2022), pp. 2671–2692. DOI: 10.1007/s00180-022-01207-6.

[110] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (visited on 03/02/2024).

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

[112] F. Pedregosa et al. *Scikit-learn: Machine Learning in Python*. 2011.

[113] João Pereira et al. *Covered Information Disentanglement: Model Transparency via Unbiased Permutation Importance*. 2021. arXiv: 2111.09744 [cs.LG].

[114] Sebastian Pölsterl. *Evaluating Survival Models*. 2022. URL: https://scikit-survival.readthedocs.io/en/latest/user_guide/evaluating-survival-models.html (visited on 02/21/2024).

[115] Sebastian Pölsterl. *Random Survival Forest – scikit-survival User Guide*. 2023. URL: https://scikit-survival.readthedocs.io/en/stable/user_guide/random-survival-forest.html (visited on 04/12/2024).

[116] Sebastian Pölsterl. "scikit-survival: A Library for Time-to-Event Analysis Built on Top of scikit-learn". In: *Journal of Machine Learning Research* 21.212 (2020), pp. 1–6. URL: http://jmlr.org/papers/v21/20-729.html (visited on 03/05/2024).

[117] Sebastian Pölsterl and contributors. *Penalized Cox Models - scikit-survival*. 2023. URL: https://scikit-survival.readthedocs.io/en/stable/user_guide/coxnet.html (visited on 04/13/2024).

[118] Michael A. Poole and Patrick N. O'Farrell. "The Assumptions of the Linear Regression Model". In: *Transactions of the Institute of British Geographers* 52 (1971), pp. 145–158. DOI: 10.2307/621706. (Visited on 04/26/2024).

[119] S Prusty, S Patnaik, and SK Dash. "SKCV: Stratified K-fold cross-validation on ML classifiers for predicting cervical cancer". In: 4 (2022). DOI: 10.3389/fnano.2022.972421.

[120] PySurvival contributors. *Brier Score - PySurvival*. https://square.github.io/pysurvival/metrics/brier_score.html. 2023. (Visited on 02/21/2024).

[121] PySurvival contributors. *C-index - PySurvival*. https://square.github.io/pysurvival/metrics/c_index.html. 2023. (Visited on 02/21/2024).

[122] Python Software Foundation. *Python Documentation*. https://docs.python.org/3/. 2024. (Visited on 03/29/2024).

[123] Xingping Qiu et al. "A Comparison Study of Machine Learning (Random Survival Forest) and Classic Statistic (Cox Proportional Hazards) for Predicting Progression in High-Grade Glioma after Proton and Carbon Ion Radiotherapy". In: *Frontiers in Oncology* 10 (2020), p. 551420. DOI: 10.3389/fonc.2020.551420.

[124] S Rai, P Mishra, and UC Ghoshal. "Survival analysis: A primer for the clinician scientists". In: *Indian Journal of Gastroenterology* 40.5 (2021), pp. 541–549. DOI: 10.1007/s12664-021-01232-1.

[125] R. Bharat Rao, Glenn Fung, and Romer Rosales. *On the Dangers of Cross-Validation. An Experimental Evaluation*. IKM CKS Siemens Medical Solutions USA, 2008. URL: https://people.csail.mit.edu/romer/papers/CrossVal_SDM08.pdf (visited on 04/14/2024).

[126] Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning, 3rd Ed.* Birmingham, UK: Packt Publishing, 2019, p. 748. ISBN: 978-1789955750.

[127] RDocumentation. *impute.knn: A function to impute missing expression data*. 2024. URL: https://www.rdocumentation.org/packages/impute/versions/1.46.0/topics/impute.knn (visited on 05/14/2024).

[128] Donald B. Rubin. "Inference and Missing Data". In: *Biometrika* 63.3 (Dec. 1976), pp. 581–592. DOI: 10.1093/biomet/63.3.581.

[129] Donald B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. New York: John Wiley & Sons, 1987.

[130] B. Saha and D. Srivastava. "Data quality: The other face of big data". In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 2014, pp. 1294–1297.

[131] J. L. Schafer. *Analysis of Incomplete Multivariate Data*. London: Chapman & Hall, 1997.

[132] Joseph L. Schafer. "Multiple imputation: A primer". In: *Statistical Methods in Medical Research* 8.1 (1999), pp. 3–15. DOI: 10.1191/096228099671525676.

[133] Josepn L. Schafer and John W. Graham. "Missing data: Our view of the state of the art". In: *Psychological Methods* 7.2 (2002), pp. 147–177. DOI: 10.1037/1082-989X.7.2.147.

[134] D. Schalk, B. Bischl, and D. Rügamer. "Accelerated Componentwise Gradient Boosting Using Efficient Data Representation and Momentum-Based Optimization". In: *Journal of Computational and Graphical Statistics* 32.2 (2023), pp. 631–641. URL: https://doi.org/10.1080/10618600.2022.2116446.

[135] M. Schott et al. "Neuroendocrine neoplasms of the gastrointestinal tract". In: *Dtsch Arztebl Int* 108.18 (2011), pp. 305–312. DOI: 10.3238/arztebl.2011.0305.

[136] scikit-learn developers. *4.2. Permutation feature importance*. https://scikit-learn.org/stable/modules/permutation_importance.html. 2024. (Visited on 04/13/2024).

[137] scikit-learn developers. *Cross-validation: evaluating estimator performance*. 2024. URL: https://scikit-learn.org/stable/modules/cross_validation.html (visited on 02/19/2024).

[138] scikit-learn developers. *Preprocessing data*. Section 6.3.4. Encoding categorical features. Accessed: 2024.04.10. 2023. URL: https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-categorical-features (visited on 04/10/2024).

[139] scikit-learn developers. *Preprocessing data*. Section 6.3.4.2. Target Encoder. 2024. URL: https://scikit-learn.org/stable/modules/preprocessing.html#par (visited on 03/07/2024).

[140] Scikit-Survival contributors. *sksurv.metrics.brier_score - Scikit-Survival Documentation*. 2023. URL: https://scikit-survival.readthedocs.io/en/stable/api/generated/sksurv.metrics.brier_score.html%5C#id1 (visited on 02/21/2024).

[141] Scikit-Survival contributors. *sksurv.metrics.concordance_index_censored - Scikit-Survival*. 2023. URL: https://scikitsurvival.readthedocs.io/en/stable/api/generated/sksurv.metrics.concordance_index_censored.html (visited on 02/21/2024).

[142] scikit-survival contributors. *Integrated Brier Score*. 2024. URL: https://scikit-survival.readthedocs.io/en/stable/api/generated/sksurv.metrics.integrated_brier_score.html (visited on 02/26/2024).

[143] H. Seow, P. Tanuseputro, L. Barbera, et al. "Development and Validation of a Prognostic Survival Model With Patient-Reported Outcomes for Patients With Cancer". In: *JAMA Netw Open* 3.4 (2020), e201768. DOI: 10.1001/jamanetworkopen.2020.1768.

[144] M. Sestelo. *1.2 Censoring*. Accessed: 2024.01.26. 2023. URL: https://bookdown.org/sestelo/sa_financial/intro-censor.html (visited on 01/26/2024).

[145] N. Shrestha. "Detecting Multicollinearity in Regression Analysis". In: *American Journal of Applied Mathematics and Statistics* 8.2 (2020), pp. 39–42.

[146] Laura J. Simon. *Descriptive Statistics*. Statistical Education Resource Kit, Pennsylvania State Department of Statistics. Archived at the Wayback Machine. 2010. URL: https://web.archive.org/web/20100730/https://example.com/ (visited on 04/05/2024).

[147] N. Simon et al. "Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent". In: *Journal of Statistical Software* 39.5 (2011), p. 1. DOI: 10.18637/jss.v039.i05.

[148] *sklearn.metrics.pairwise.nan_euclidean_distances*. scikit-learn developers, 2024. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.nan_euclidean_distances.html (visited on 04/29/2024).

[149] A. B. Md Soom et al. "A Bad Idea Of Using Mode Imputation Method". In: *Journal of Information System and Technology Management* 7.29 (2022), pp. 01–09.

[150] H. Sorbye et al. "Gastroenteropancreatic high-grade neuroendocrine carcinoma". In: *Cancer* 120.18 (2014). Epub 2014 Apr 25, pp. 2814–2823. ISSN: 0008-543X. DOI: 10.1002/cncr.28721.

[151] Halfdan Sorbye et al. "Predictive and prognostic factors for treatment and survival in 305 patients with advanced gastrointestinal neuroendocrine carcinoma (WHO G3): The NORDIC NEC study". In: *Annals of Oncology* 24.1 (2013), pp. 152–160. ISSN: 0923-7534. DOI: 10.1093/annonc/mds276.

[152] Jonathan A C Sterne et al. "Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls". In: *BMJ* 338 (2009). DOI: 10.1136/bmj.b2393.

[153] Henning Langen Stokmo et al. "Volumetric parameters from [18F]FDG PET/CT predicts survival in patients with high-grade gastroenteropancreatic neuroendocrine neoplasms". In: *Journal of Neuroendocrinology* 34.7 (2022). DOI: 10.1111/jne.13170.

[154] Jianguo Sun. *The Statistical Analysis of Interval-Censored Failure Time Data*. Springer, 2006.

[155] *Survival Analysis Introduction*. lifelines, 2014. URL: https://lifelines.readthedocs.io/en/latest/Survival%20Analysis%20intro.html.

[156] Bhavisha Suthar, Hemant Patel, and Ankur Goswami. "A Survey: Classification of Imputation Methods in Data Mining". In: 2012. URL: https://api.semanticscholar.org/CorpusID:2058666 (visited on 01/10/2024).

[157] B. G. Taal and O. Visser. "Epidemiology of neuroendocrine tumours". In: *Neuroendocrinology* 80.Suppl 1 (2004), pp. 3–7. DOI: 10.1159/000080731.

[158] Azzam FG Taktak and Anthony C Fisher, eds. *Outcome Prediction in Cancer*. Elsevier, 2006. ISBN: 9780080468037.

[159] R Core Team. *R: Documentation*. 2024. URL: https://www.r-project.org/other-docs.html (visited on 03/29/2024).

[160] Terry M Therneau et al. *survival: Survival Analysis*. R package version 3.5-8. 2024. URL: https://github.com/therneau/survival (visited on 03/15/2024).

[161] Terry M. Therneau and Thomas Lumley. *Chemotherapy for Stage B/C Colon Cancer*. 2024. URL: https://stat.ethz.ch/R-manual/R-devel/library/survival/html/colon.html (visited on 04/26/2024).

[162] D. Thomas, A. V. Tsolakis, S. Grozinsky-Glasberg, et al. "Long-term follow-up of a large series of patients with type 1 gastric carcinoid tumors: data from a multicenter study". In: *European Journal of Endocrinology* 168.2 (2013), pp. 185–193. DOI: 10.1530/EJE-12-0836.

[163] Olga Troyanskaya et al. "Missing value estimation methods for DNA microarrays". In: *Bioinformatics* 17.6 (2001), pp. 520–525. DOI: 10.1093/bioinformatics/17.6.520.

[164] M Tuson et al. "Predicting Future Geographic Hotspots of Potentially Preventable Hospitalisations Using All Subset Model Selection and Repeated K-Fold Cross-Validation". In: *International Journal of Environmental Research and Public Health* 18.19 (2021), p. 10253. DOI: 10.3390/ijerph181910253.

[165] *Understanding Predictions in Survival Analysis*. scikit-survival, 2020. URL: https://scikit-survival.readthedocs.io/en/stable/user_guide/understanding_predictions.html (visited on 02/19/2024).

[166] Joseph M. Unger et al. "Comparison of Survival Outcomes Among Cancer Patients Treated In and Out of Clinical Trials". In: *JNCI: Journal of the National Cancer Institute* 106.3 (2014). DOI: 10.1093/jnci/dju002.

[167] W Vach and M Blettner. "Biased estimation of the odds ratio in case-control studies due to the use of ad hoc methods of correcting for missing values for confounding variables". In: *American Journal of Epidemiology* 134 (1991), pp. 895–907.

[168] Alexander Von Eye and Clifford C. Clogg, eds. *Categorical Variables in Developmental Research: Methods of Analysis*. Amsterdam: Elsevier, 1996.

[169] Michael L. Waskom. "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021.

[170] Ian R. White, Patrick Royston, and Angela M. Wood. "Multiple imputation using chained equations: Issues and guidance for practice". In: *Statistics in Medicine* 30.4 (2011), pp. 377–399. DOI: https://doi.org/10.1002/sim.4067.

[171] R. Williams. *Missing data Part 1: overview, traditional methods*. Notre Dame, 2015. URL: https://www3.nd.edu/~rwilliam/xsoc73994/MD01.pdf (visited on 01/15/2024).

[172] Richard Williams. *Multinomial Logit Models - Overview*. University of Notre Dame. Adapted from Menard's Applied Logistic Regression analysis and Borooah's Logit. 2021. URL: https://www3.nd.edu/~rwilliam/stats3/Mlogit1.pdf (visited on 04/22/2024).

[173] Andrew M. Wood, Ian R. White, and Simon G. Thompson. "Are missing outcome data adequately handled? A review of published randomized controlled trials in major medical journals". In: *Clinical Trials* 1.4 (2004), pp. 368–376. DOI: 10.1191/1740774504cn032oa.

[174] World Health Organization. *Cancer*. 2022. URL: https://www.who.int/news-room/fact-sheets/detail/cancer (visited on 04/16/2024).

[175] Brett Wujek and Patrick Hall. *Best Practices for Machine Learning Applications*. 2016. URL: https://api.semanticscholar.org/CorpusID:20946822 (visited on 04/09/2024).

[176] Z. Xu et al. "Epidemiologic Trends of and Factors Associated With Overall Survival for Patients With Gastroenteropancreatic Neuroendocrine Tumors in the United States". In: *JAMA Network Open* 4.9 (2021), e2124750. DOI: 10.1001/jamanetworkopen.2021.24750.

[177] Suayib Yalcin. “Advances in the systemic treatment of pancreatic neuroendocrine tumors”. In: *Cancer Treatment Reviews* 37.2 (2011), pp. 127–132. DOI: 10.1016/j.ctrv.2010.07.003.

[178] Suayib Yalcin and Kjell Öberg, eds. *Neuroendocrine Tumours: Diagnosis and Management*. 1st ed. Medicine 1. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. VII, 578. DOI: 10.1007/978-3-662-45215-8.

[179] Xue Ying. “An Overview of Overfitting and its Solutions”. In: *Journal of Physics: Conference Series* 1168.2 (2019). DOI: 10.1088/1742-6596/1168/2/022022.

[180] Aoqian Zhang et al. “Learning Individual Models for Imputation”. In: *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 2019, pp. 160–171. DOI: 10.1109/ICDE.2019.00023.

[181] Aoqian Zhang et al. “Learning Individual Models for Imputation”. In: *ICDE*. IEEE, 2019, pp. 160–171.

[182] XH Zhou, GJ Eckert, and WM Tierney. “Multiple imputation in public health research”. In: *Statistics in Medicine* 20.9-10 (2001), pp. 1541–1549.

[183] Ziwei Zhu, Tengyao Wang, and Richard J. Samworth. *High-dimensional principal component analysis with heterogeneous missingness*. 2019. arXiv: 1906.12125 [stat.ME].

[184] Hui Zou and Trevor Hastie. “Regularization and Variable Selection Via the Elastic Net”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2 (2005), pp. 301–320. DOI: 10.1111/j.1467-9868.2005.00503.x.

Appendix A

A Experimental Setup: Missing Values

A.1 MCAR

A.1.1 Bias

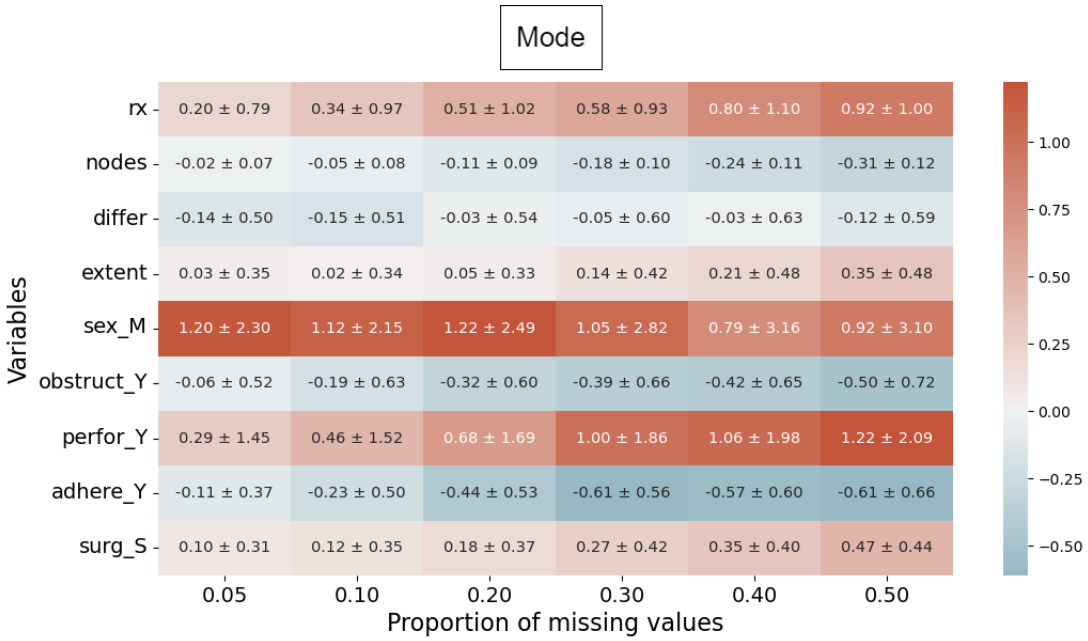


Figure A.1: Bias of coefficients from a Cox PH model of the variables as a function of the proportion of missing values after applying mode imputation on simulated missingness (MCAR) in the colon dataset, where missingness was simulated 50 times. The whitest colour implies no bias, while blue and red lean towards negative and positive bias, respectively.

A.1.2 Concordance

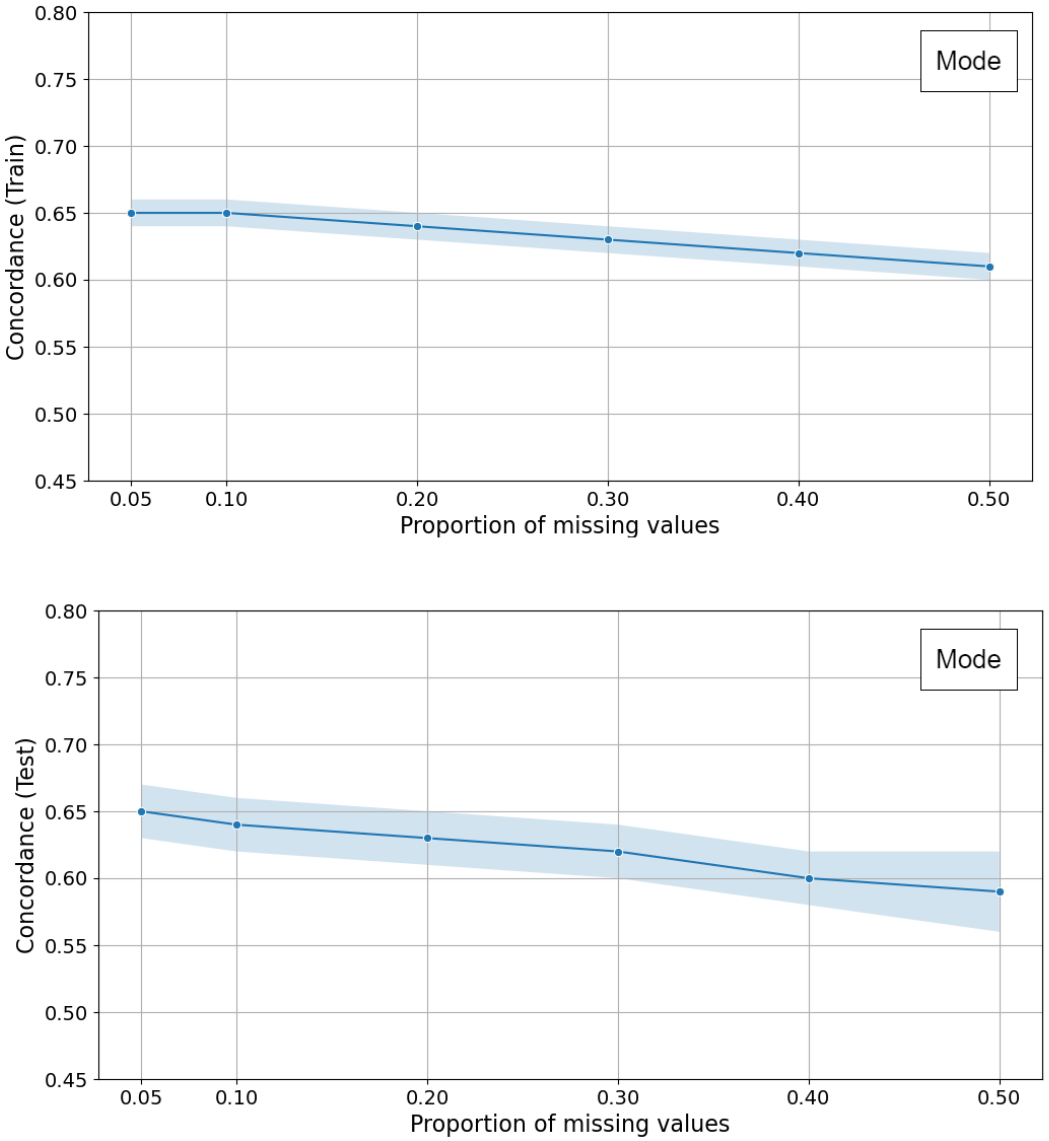


Figure A.2: Train and test concordance as a function of the proportion of missing values after applying mode imputation on simulated missingness (MCAR) in the colon dataset, where missingness was simulated 50 times. The standard deviations (shaded area) and averages (solid line) are shown.

A.2 MAR

A.2.1 Accuracy

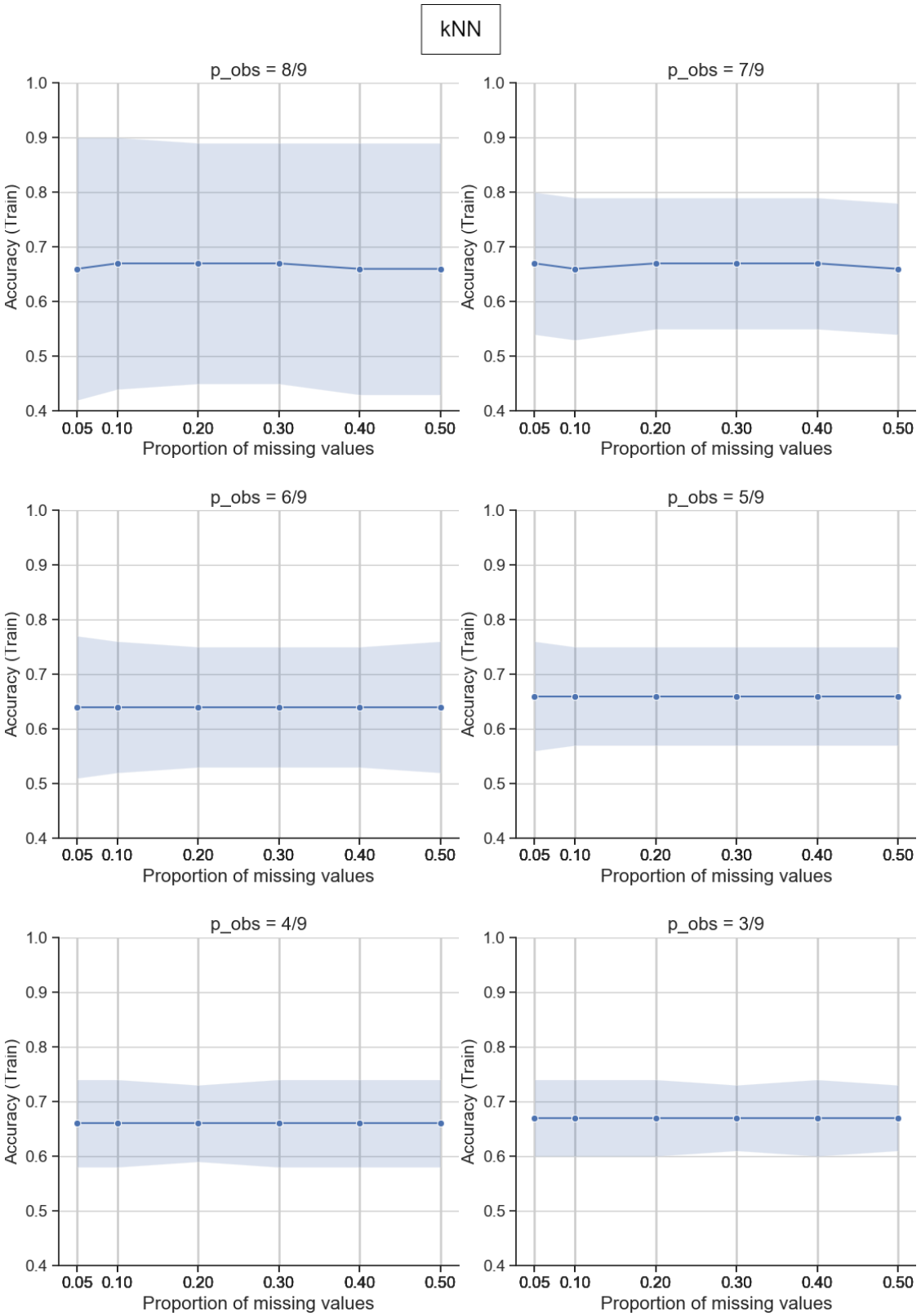


Figure A.3: Train accuracy as a function of the proportion of missing values after applying kNN imputation on simulated missingness (MAR) in the colon dataset, where missingness was simulated 50 times. The standard deviations (shaded area) and averages (solid line) are shown.

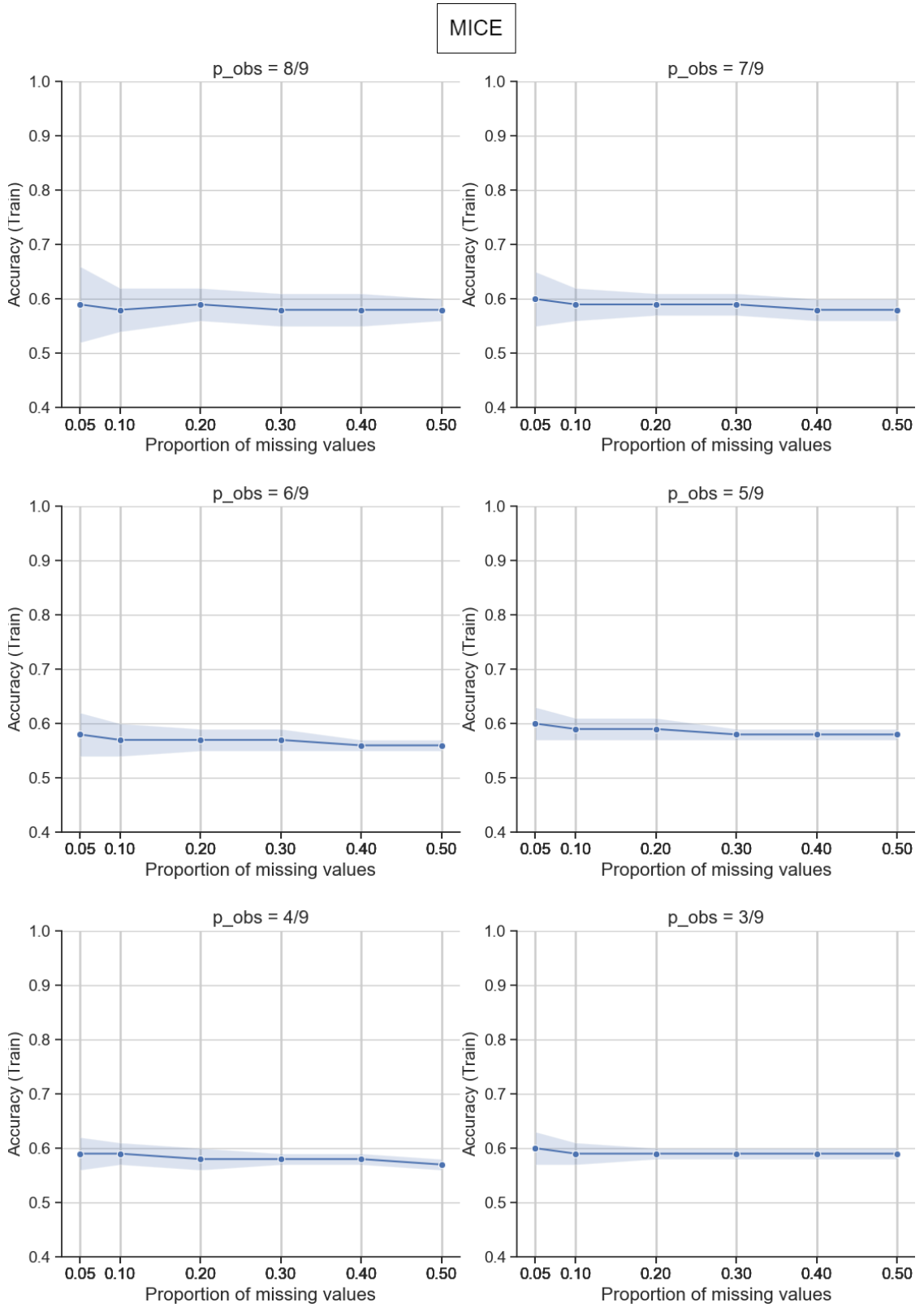


Figure A.4: Train accuracy as a function of the proportion of missing values after applying MICE imputation on simulated missingness (MAR) in the colon dataset, where missingness was simulated 50 times. The standard deviations (shaded area) and averages (solid line) are shown.

A.2.2 Bias

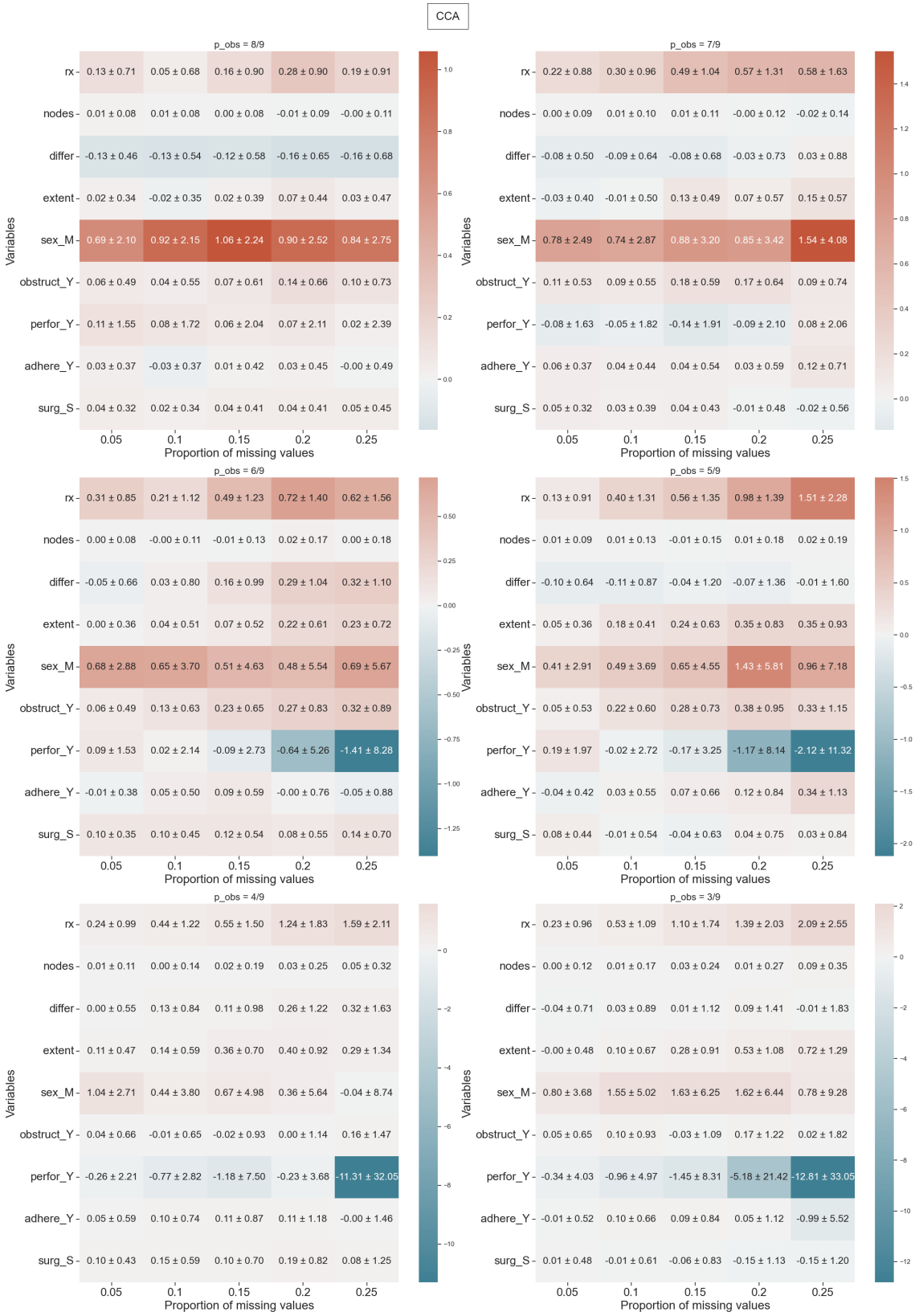


Figure A.5: Bias of coefficients from a Cox PH model of the variables as a function of the proportion of missing values after applying CCA on simulated missingness (MAR) in the colon dataset, where missingness was simulated 50 times. The whitest colour implies no bias, while blue and red lean towards negative and positive bias, respectively.

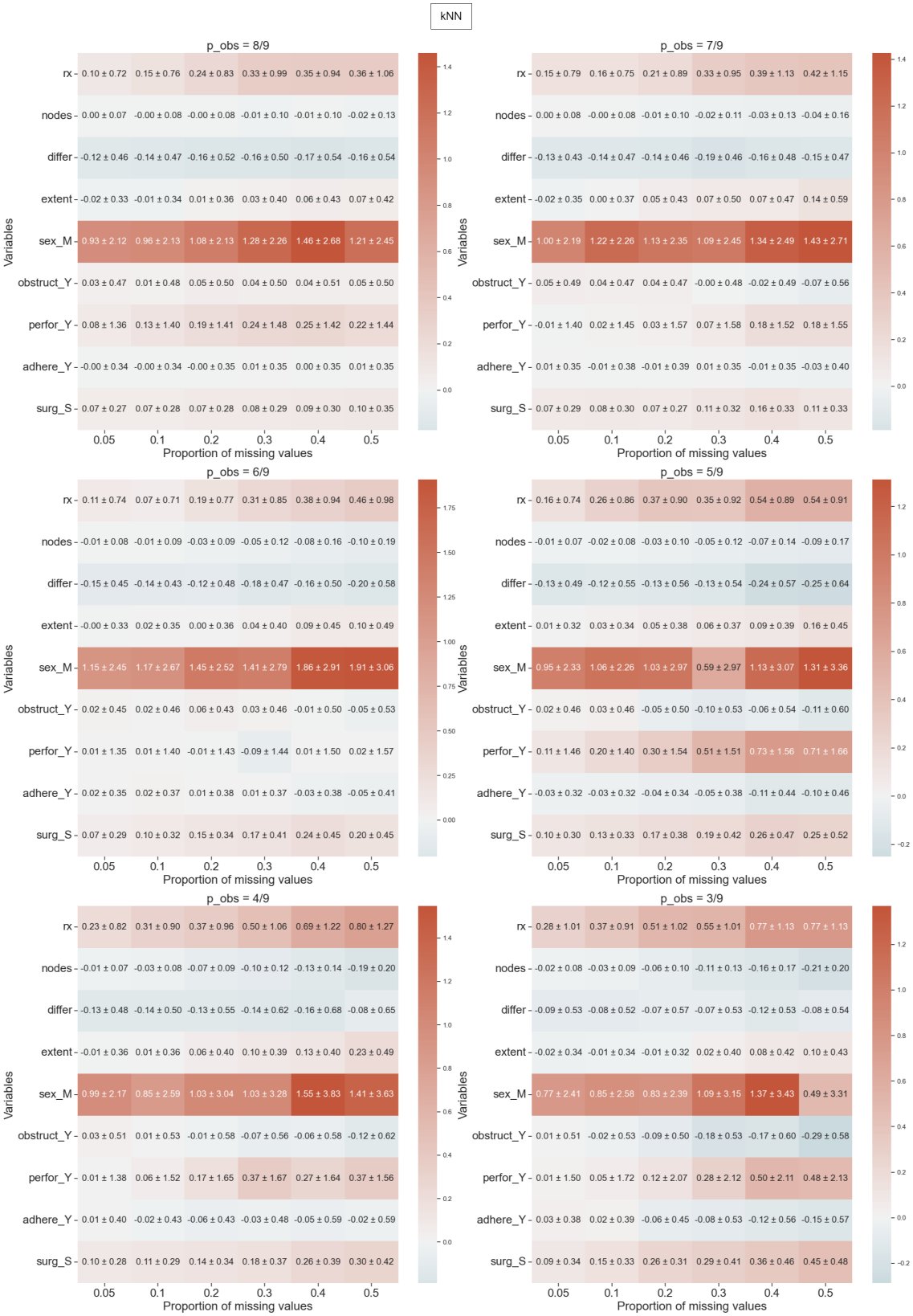


Figure A.6: Bias of coefficients from a Cox PH model of the variables as a function of the proportion of missing values after applying kNN imputation on simulated missingness (MAR) in the colon dataset, where missingness was simulated 50 times. The whitest colour implies no bias, while blue and red lean towards negative and positive bias, respectively.

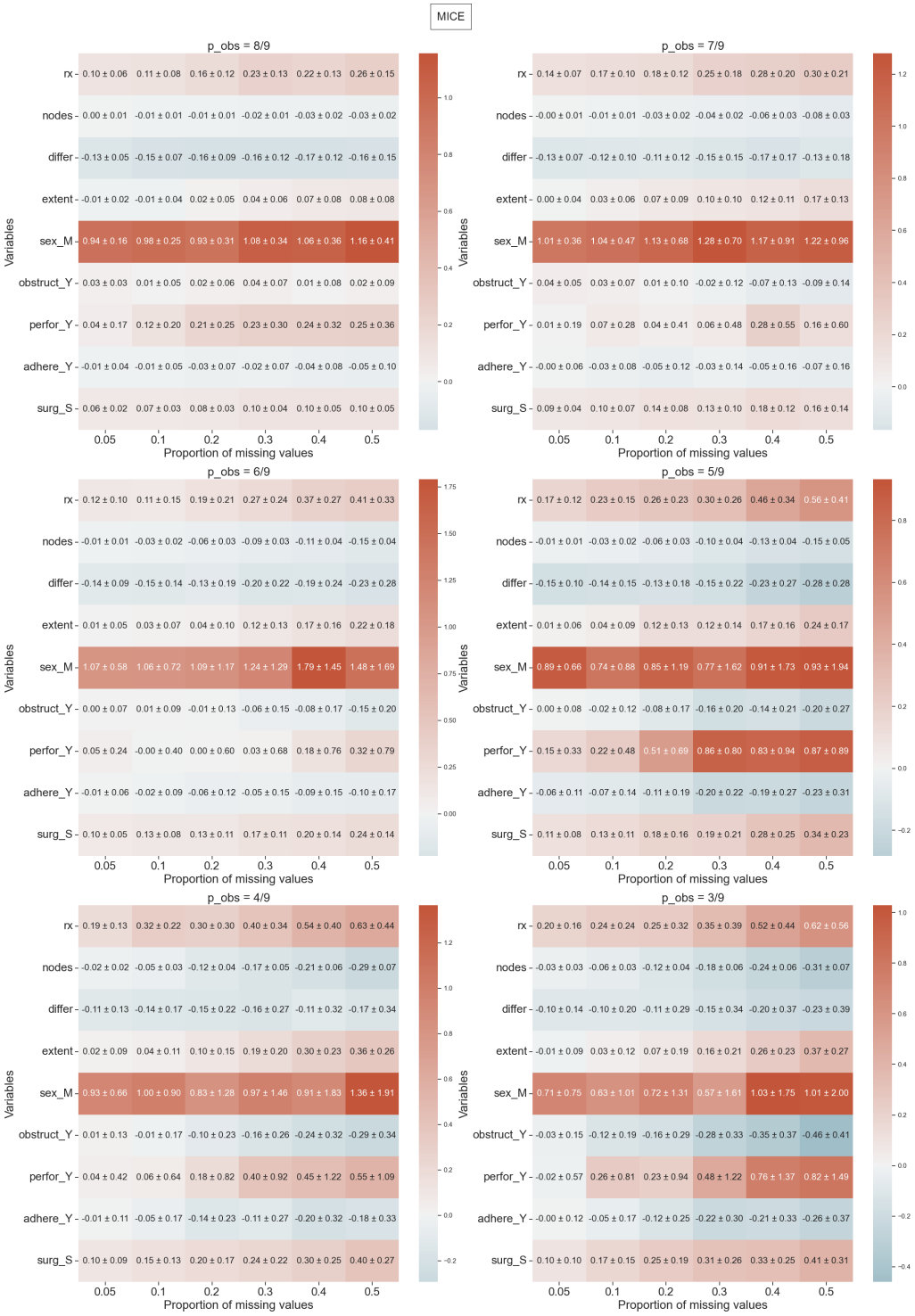


Figure A.7: Bias of coefficients from a Cox PH model of the variables as a function of the proportion of missing values after applying MICE imputation on simulated missingness (MAR) in the colon dataset, where missingness was simulated 50 times. The whitest colour implies no bias, while blue and red lean towards negative and positive bias, respectively.

A.2.3 Concordance

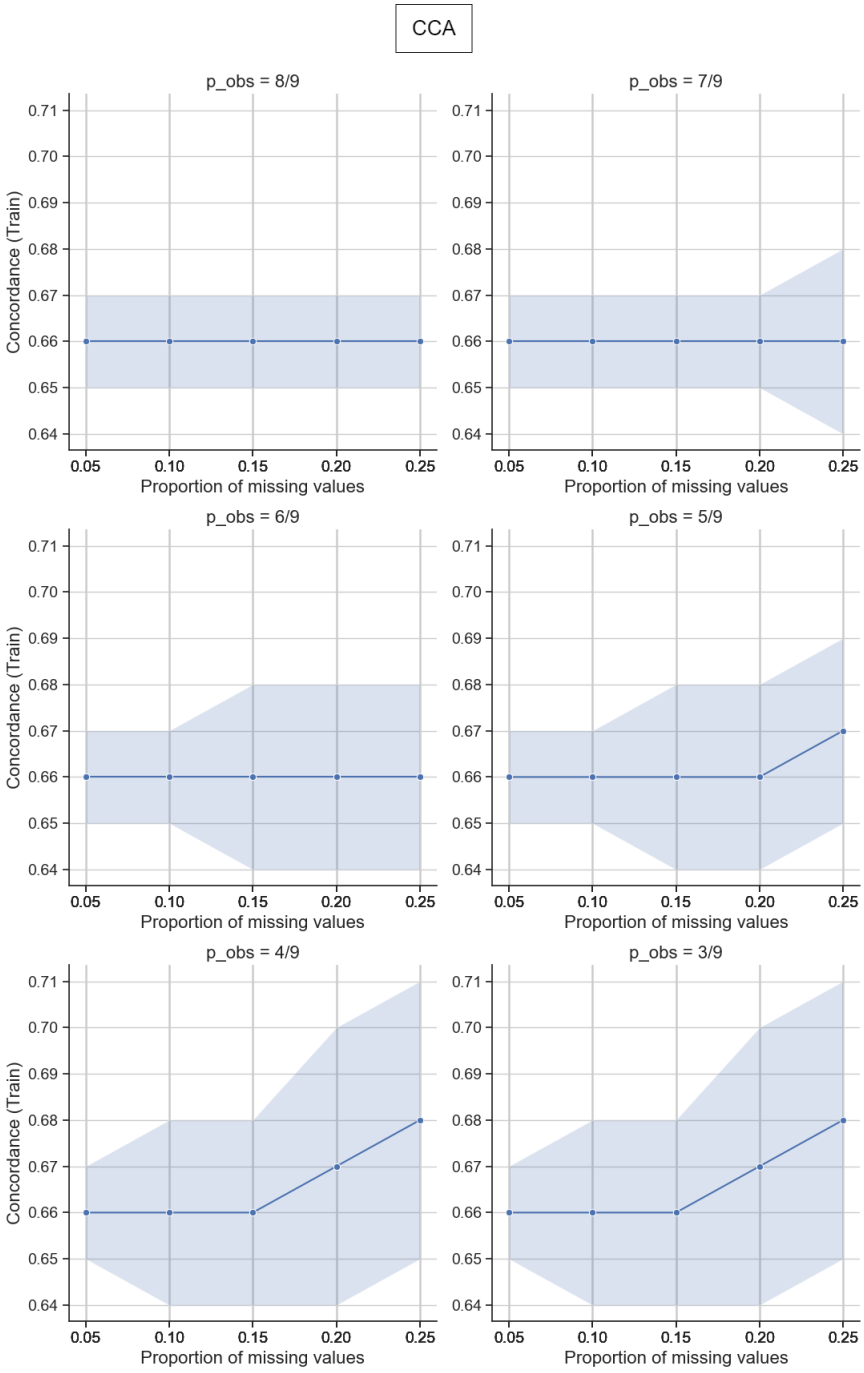


Figure A.8: Train concordance as a function of the proportion of missing values after applying CCA on simulated missingness (MAR) in the colon dataset, where missingness was simulated 50 times. The standard deviations (shaded area) and averages (solid line) are shown.

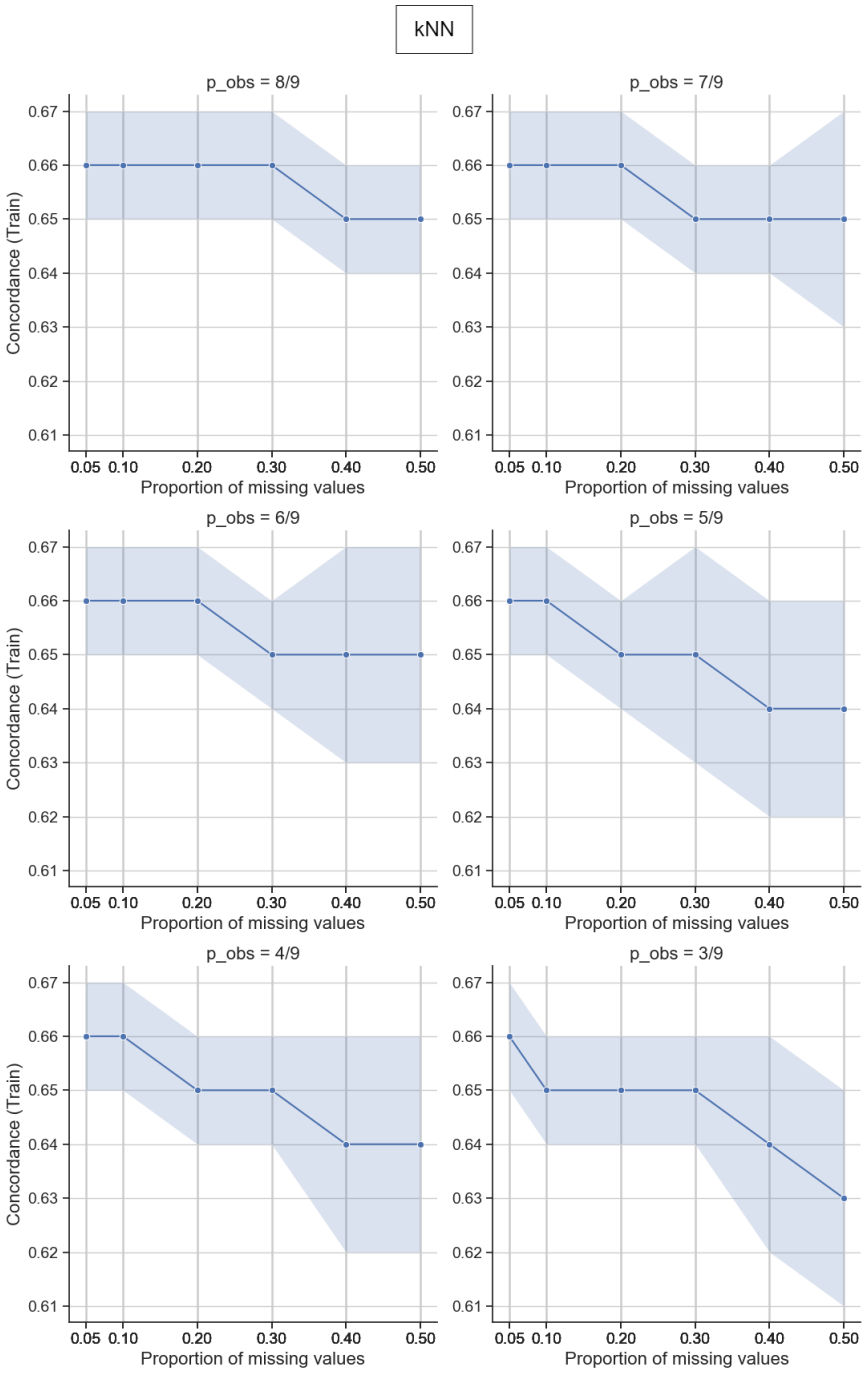


Figure A.9: Train concordance as a function of the proportion of missing values after applying kNN imputation on simulated missingness (MAR) in the colon dataset, where missingness was simulated 50 times. The standard deviations (shaded area) and averages (solid line) are shown.

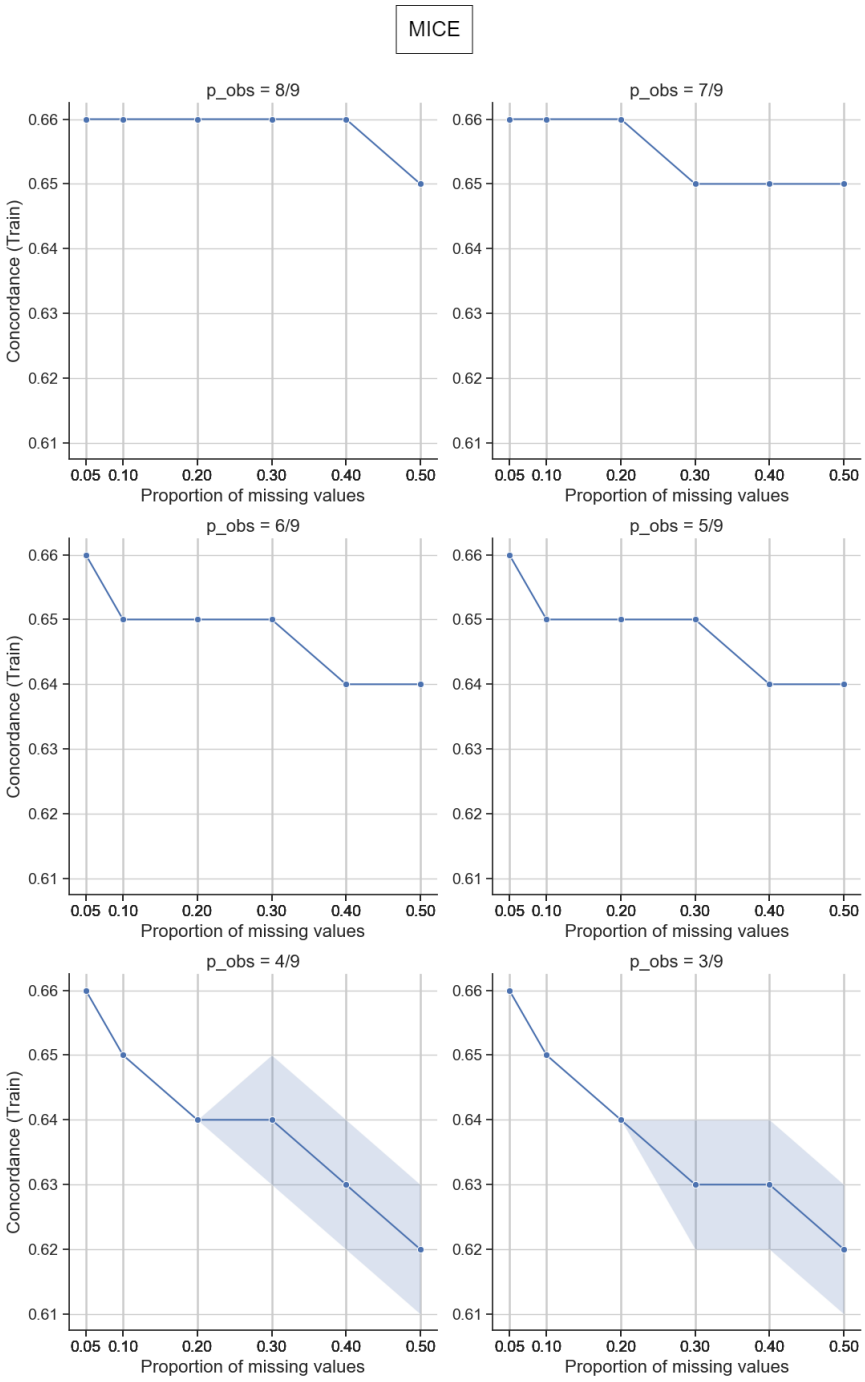


Figure A.10: Train concordance as a function of the proportion of missing values after applying MICE imputation on simulated missingness (MAR) in the colon dataset, where missingness was simulated 50 times. The standard deviations (shaded area) and averages (solid line) are shown.

Appendix B

B Survival Curves

B.1 Compare Survival Curves Between Coxnet and CGB with the Highest Test C-index

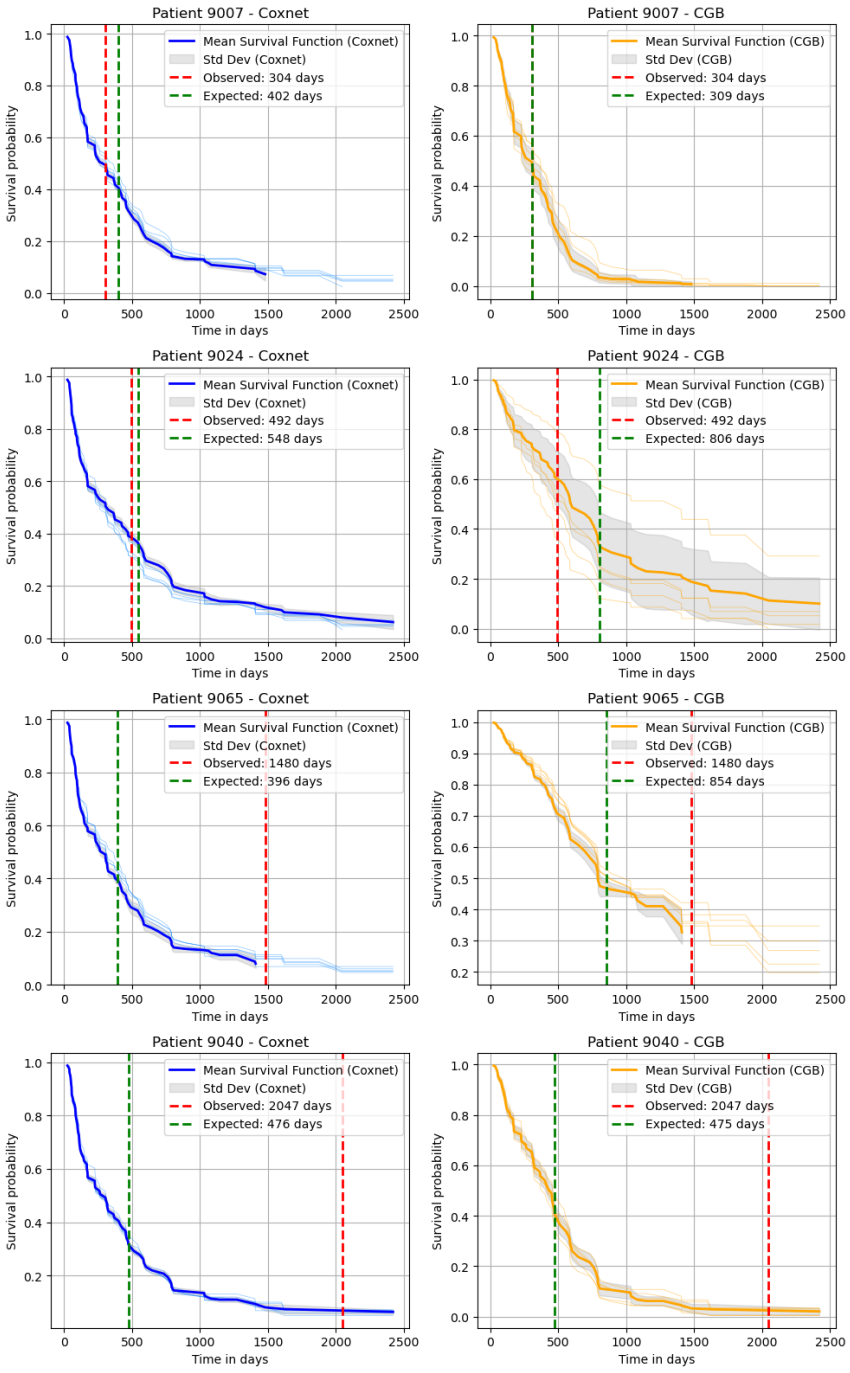


Figure B.1: Survival curves for different patients between the highest C-index model for Coxnet and CGB with kNN imputations. Survival curves for all patients who experienced an event for Coxnet and CGB models with kNN imputations. The different curves are given by repeated stratified k-fold. The blue curve is the average estimated survival curve and the shaded area (Std Dev) is the standard deviation. The red line indicates when the event of interest occurred (observed survival time) and the green line marks the model's prediction for when the event was expected to happen (expected survival time). Coxnet (alpha: 1000, L1 ratio: 0.0001), CGB (n_estimator: 10, learning_rate: 0.6, subsample: 0.6).

B.2 Compare Survival Curves Between the Coxnet Model (alpha: 1000, L1 ratio: 0.0001) with Coxnet Model (alpha: 3, L1 ratio: 0.01)

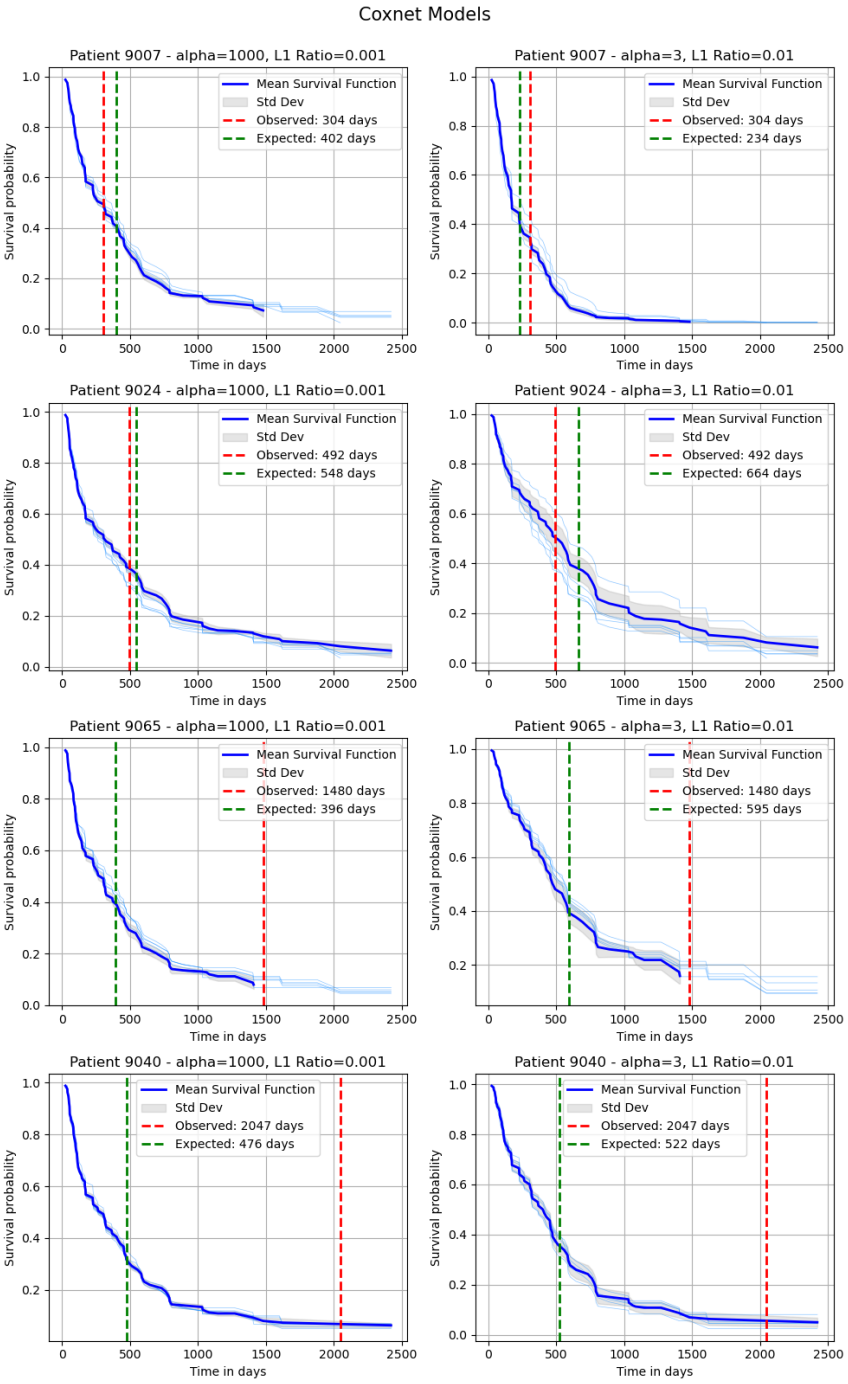


Figure B.2: Survival curves for Coxnet model with the highest C-index model and the model that were selected (Section 5.2.2) with kNN imputations. Survival curves for all patients who experienced an event for Coxnet and CGB models with kNN imputations. The different curves are given by repeated stratified k-fold. The blue curve is the average estimated survival curve and the shaded area (Std Dev) is the standard deviation. The red line indicates when the event of interest occurred (observed survival time) and the green line marks the model's prediction for when the event was expected to happen (expected survival time). Coxnet (alpha: 1000, L1 ratio: 0.0001), Coxnet (alpha: 3, L1 ratio: 0.01).

B.3 Compare Survival Curves Between Coxnet and CGB Model for All Patients.

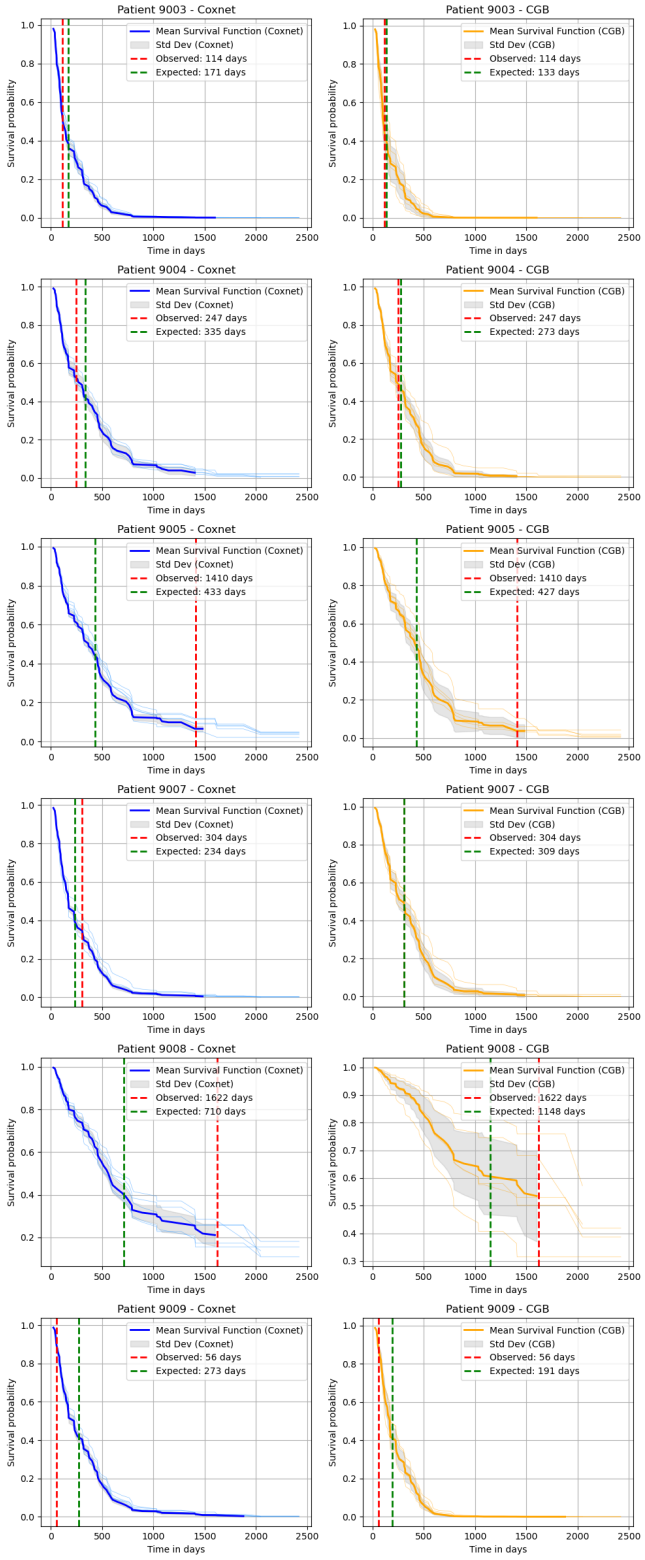
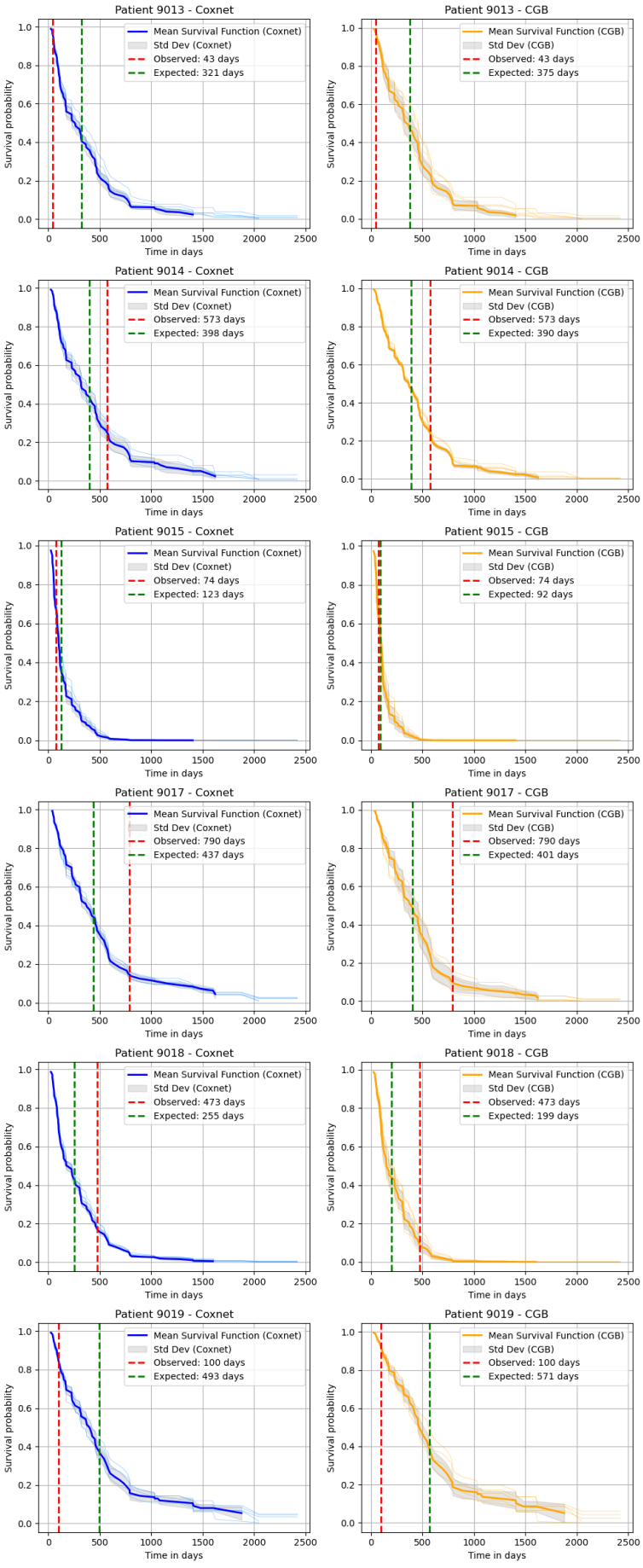
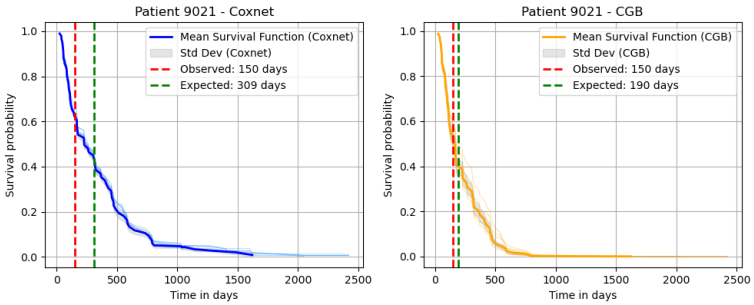
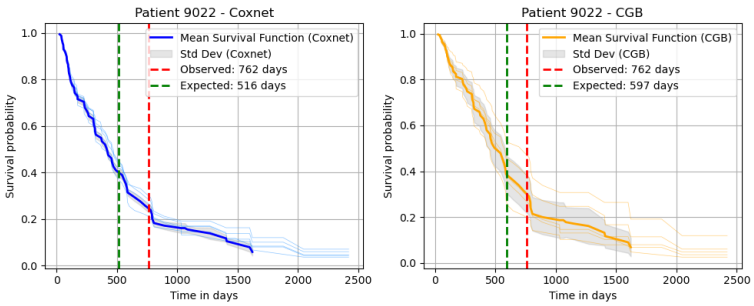


Figure B.3: Survival curves for all patients for Coxnet and CGB models with kNN imputations. Survival curves for all patients who experienced an event for Coxnet and CGB models with kNN imputations. The different curves are given by repeated stratified k-fold. The blue curve is the average estimated survival curve and the shaded area (Std Dev) is the standard deviation. The red line indicates when the event of interest occurred (observed survival time) and the green line marks the model's prediction for when the event was expected to happen (expected survival time). Coxnet (alpha: 3, L1 ratio: 0.01), CGB (n_estimator: 10, learning_rate: 0.6, subsample: 0.6).

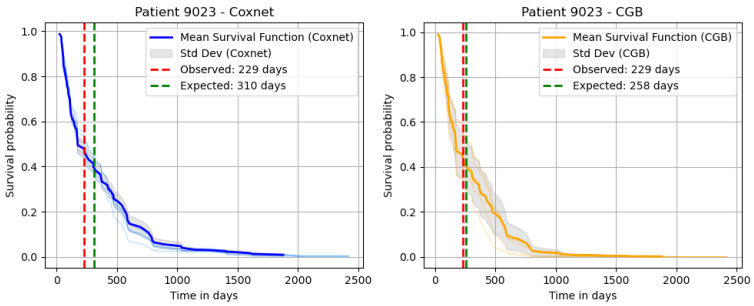

Patient 9013 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 43 days
Expected: 321 days
Patient 9013 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 43 days
Expected: 375 days
Patient 9014 - Coxnet
Observed: 573 days
Expected: 398 days
Patient 9014 - CGB
Observed: 573 days
Expected: 390 days
Patient 9015 - Coxnet
Observed: 74 days
Expected: 123 days
Patient 9015 - CGB
Observed: 74 days
Expected: 92 days
Patient 9017 - Coxnet
Observed: 790 days
Expected: 437 days
Patient 9017 - CGB
Observed: 790 days
Expected: 401 days
Patient 9018 - Coxnet
Observed: 473 days
Expected: 255 days
Patient 9018 - CGB
Observed: 473 days
Expected: 199 days
Patient 9019 - Coxnet
Observed: 100 days
Expected: 493 days
Patient 9019 - CGB
Observed: 100 days
Expected: 571 days
Survival probability
Time in days


Patient 9021 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 150 days
Expected: 309 days
Survival probability
Time in days

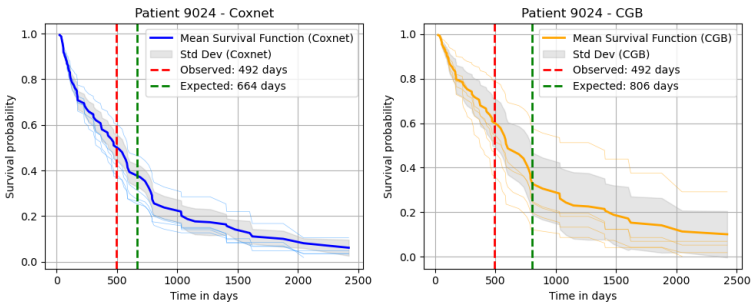
Patient 9021 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 150 days
Expected: 190 days
Survival probability
Time in days


Patient 9022 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 762 days
Expected: 516 days
Survival probability
Time in days

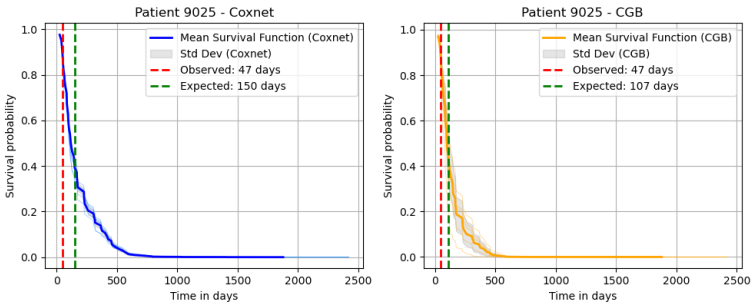
Patient 9022 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 762 days
Expected: 597 days
Survival probability
Time in days


Patient 9023 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 229 days
Expected: 310 days
Survival probability
Time in days

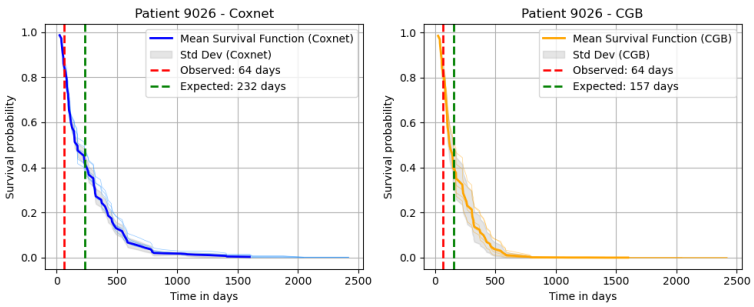
Patient 9023 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 229 days
Expected: 258 days
Survival probability
Time in days


Patient 9024 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 492 days
Expected: 664 days
Survival probability
Time in days

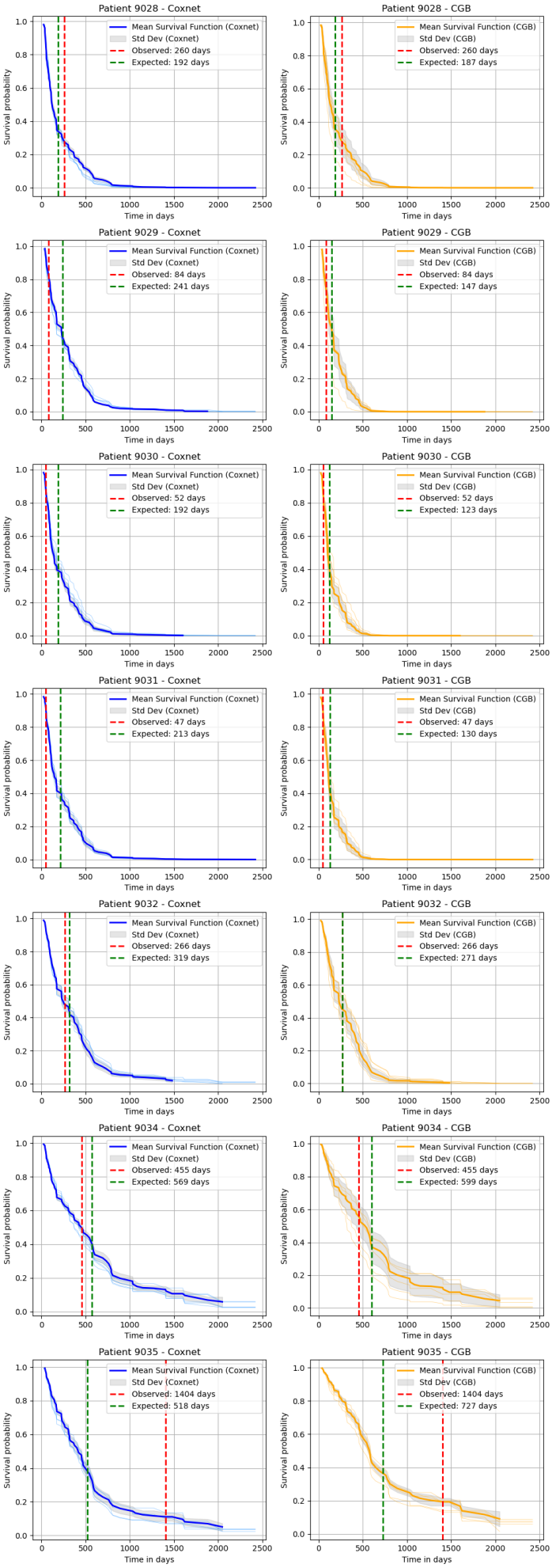
Patient 9024 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 492 days
Expected: 806 days
Survival probability
Time in days

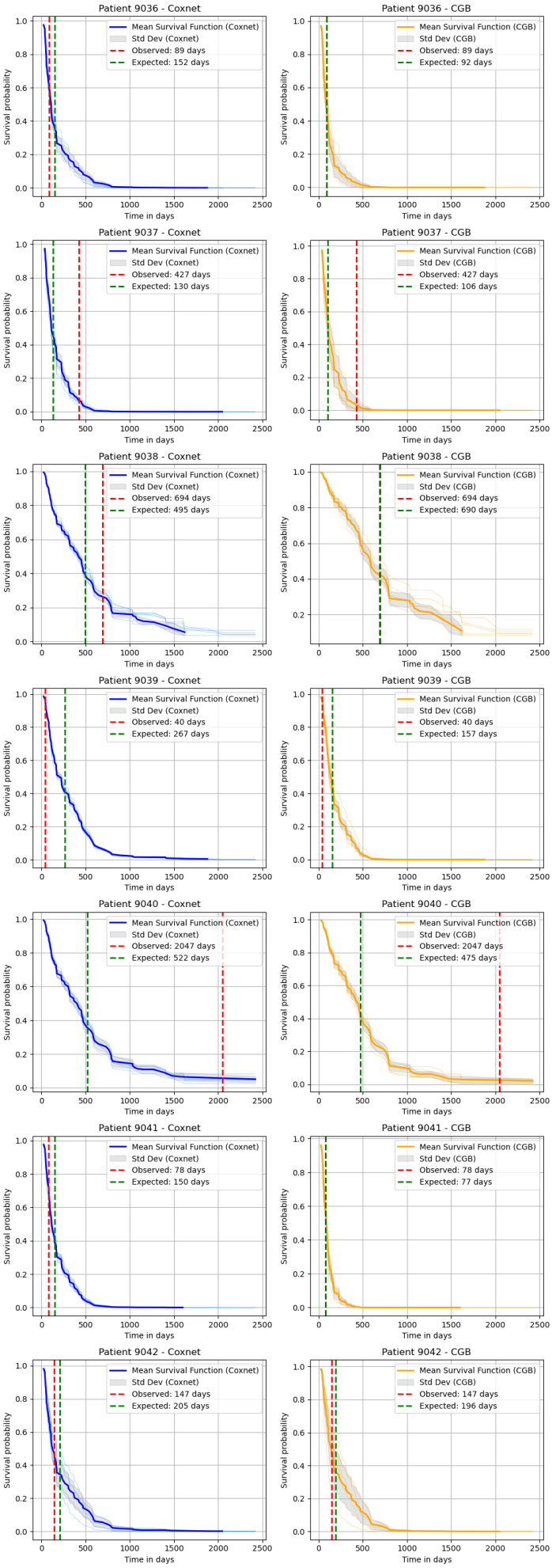

Patient 9025 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 47 days
Expected: 150 days
Survival probability
Time in days

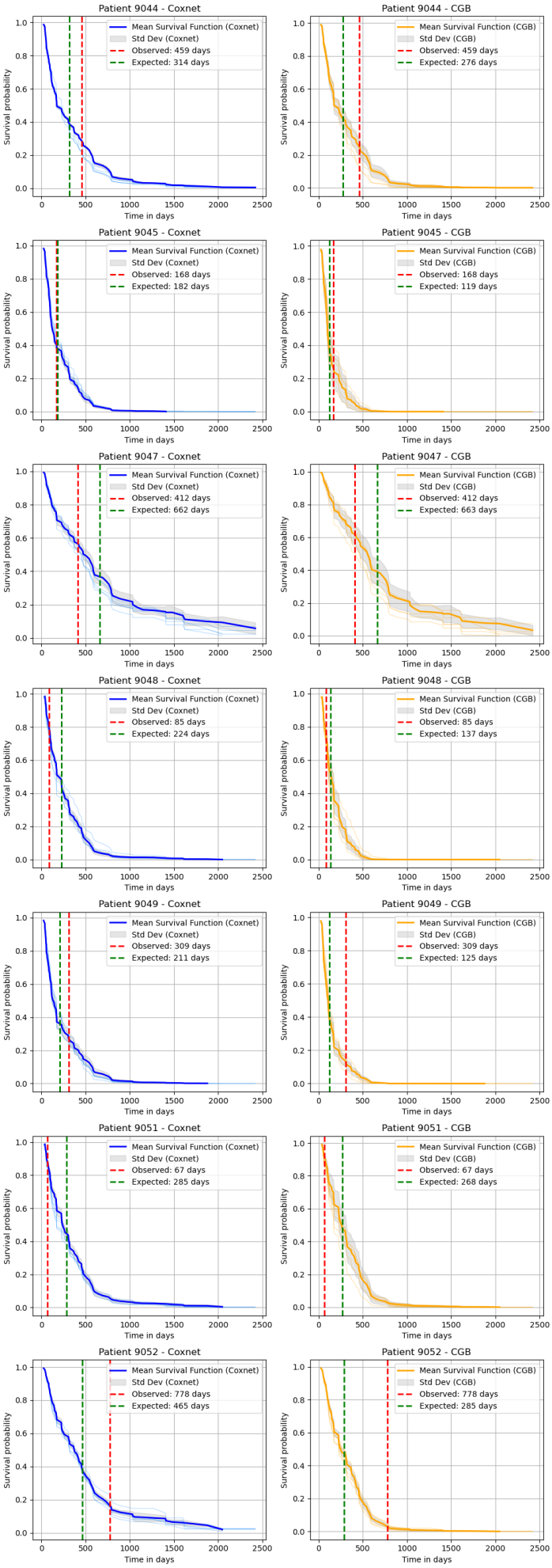
Patient 9025 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 47 days
Expected: 107 days
Survival probability
Time in days

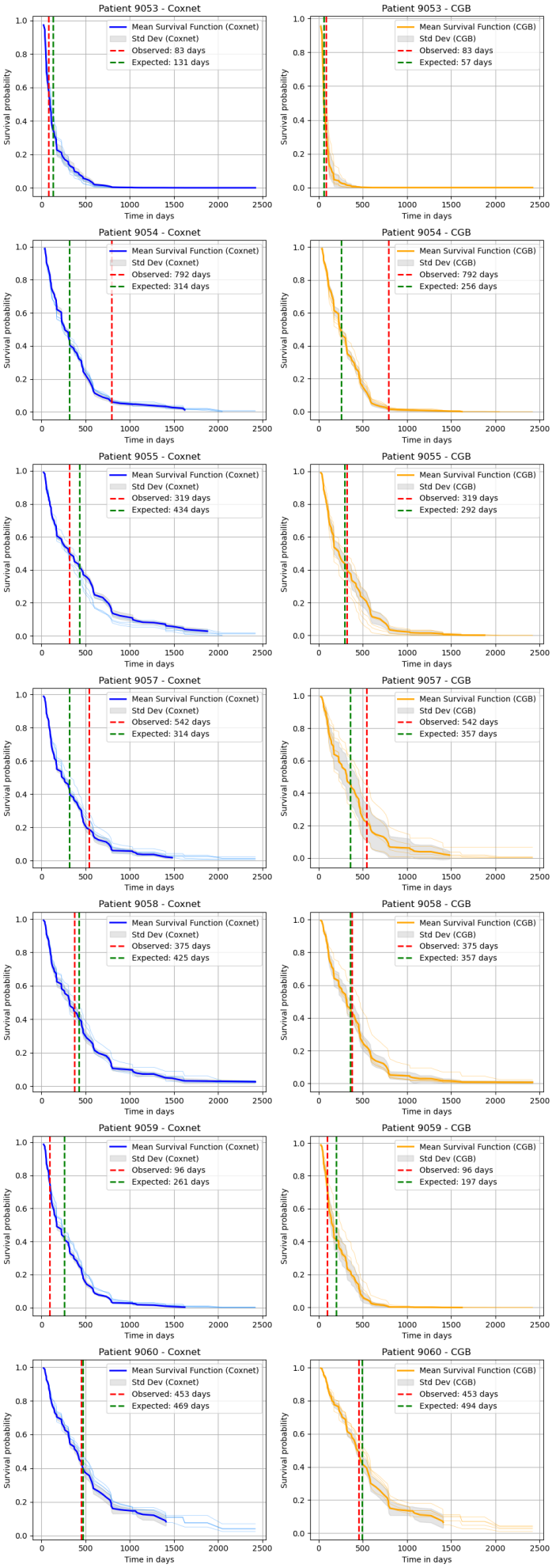

Patient 9026 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 64 days
Expected: 232 days
Survival probability
Time in days

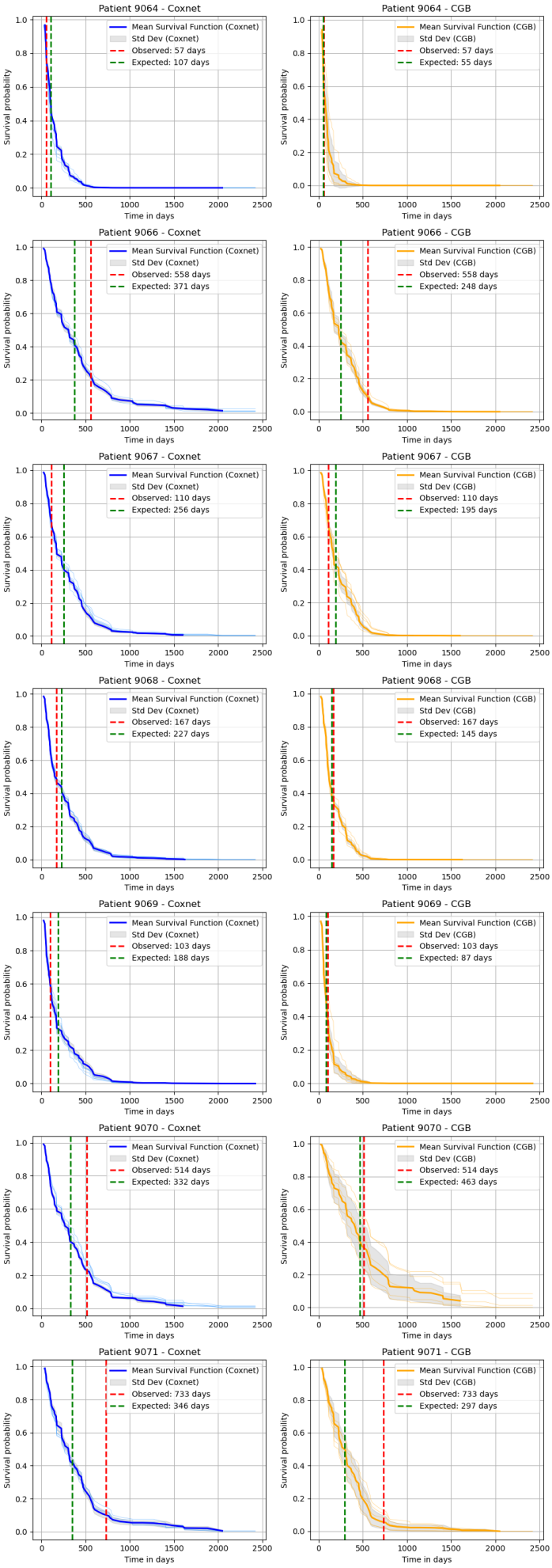
Patient 9026 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 64 days
Expected: 157 days
Survival probability
Time in days

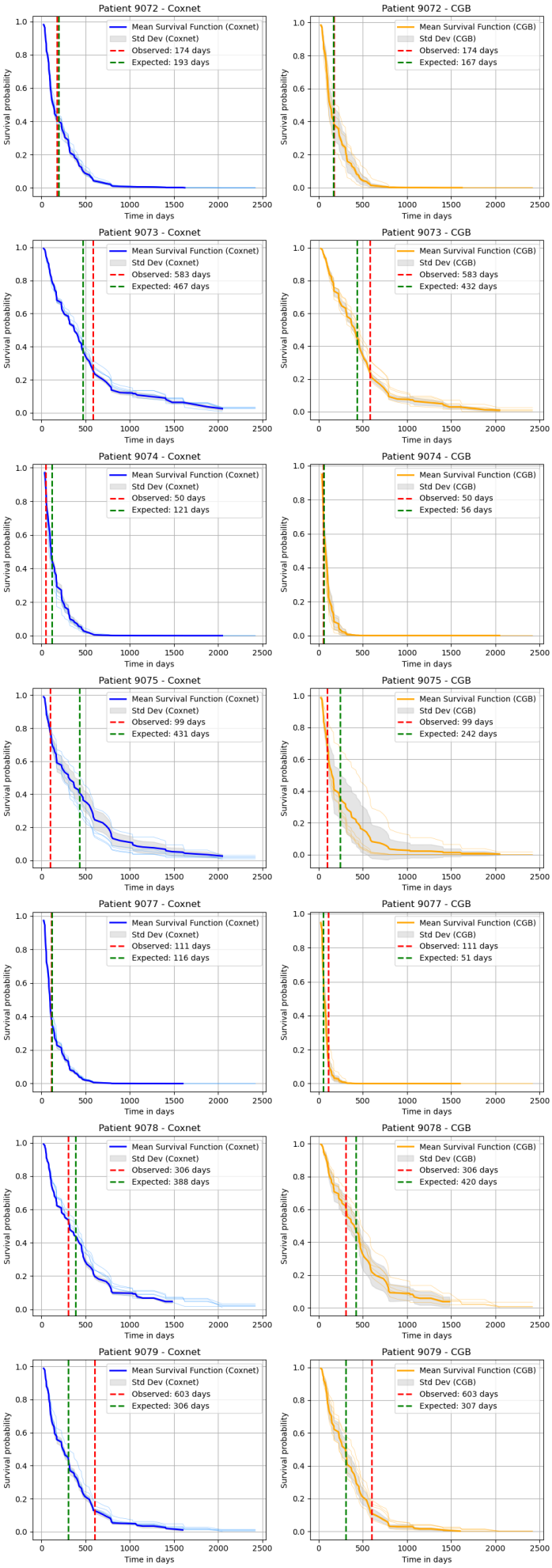

Patient 9028 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 260 days
Expected: 192 days
Patient 9028 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 260 days
Expected: 187 days
Patient 9029 - Coxnet
Observed: 84 days
Expected: 241 days
Patient 9029 - CGB
Observed: 84 days
Expected: 147 days
Patient 9030 - Coxnet
Observed: 52 days
Expected: 192 days
Patient 9030 - CGB
Observed: 52 days
Expected: 123 days
Patient 9031 - Coxnet
Observed: 47 days
Expected: 213 days
Patient 9031 - CGB
Observed: 47 days
Expected: 130 days
Patient 9032 - Coxnet
Observed: 266 days
Expected: 319 days
Patient 9032 - CGB
Observed: 266 days
Expected: 271 days
Patient 9034 - Coxnet
Observed: 455 days
Expected: 569 days
Patient 9034 - CGB
Observed: 455 days
Expected: 599 days
Patient 9035 - Coxnet
Observed: 1404 days
Expected: 518 days
Patient 9035 - CGB
Observed: 1404 days
Expected: 727 days
Survival probability
Time in days

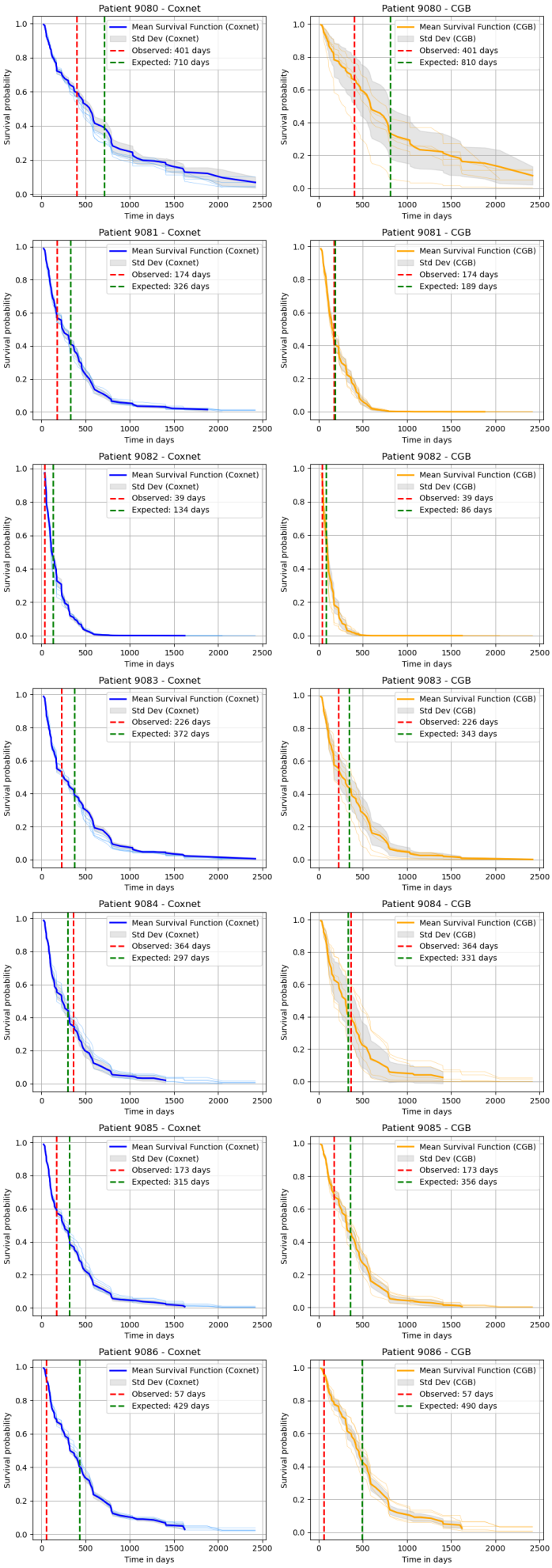

Patient 9036 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 89 days
Expected: 152 days
Patient 9036 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 89 days
Expected: 92 days
Patient 9037 - Coxnet
Observed: 427 days
Expected: 130 days
Patient 9037 - CGB
Observed: 427 days
Expected: 106 days
Patient 9038 - Coxnet
Observed: 694 days
Expected: 495 days
Patient 9038 - CGB
Observed: 694 days
Expected: 690 days
Patient 9039 - Coxnet
Observed: 40 days
Expected: 267 days
Patient 9039 - CGB
Observed: 40 days
Expected: 157 days
Patient 9040 - Coxnet
Observed: 2047 days
Expected: 522 days
Patient 9040 - CGB
Observed: 2047 days
Expected: 475 days
Patient 9041 - Coxnet
Observed: 78 days
Expected: 150 days
Patient 9041 - CGB
Observed: 78 days
Expected: 77 days
Patient 9042 - Coxnet
Observed: 147 days
Expected: 205 days
Patient 9042 - CGB
Observed: 147 days
Expected: 196 days
Survival probability
Time in days

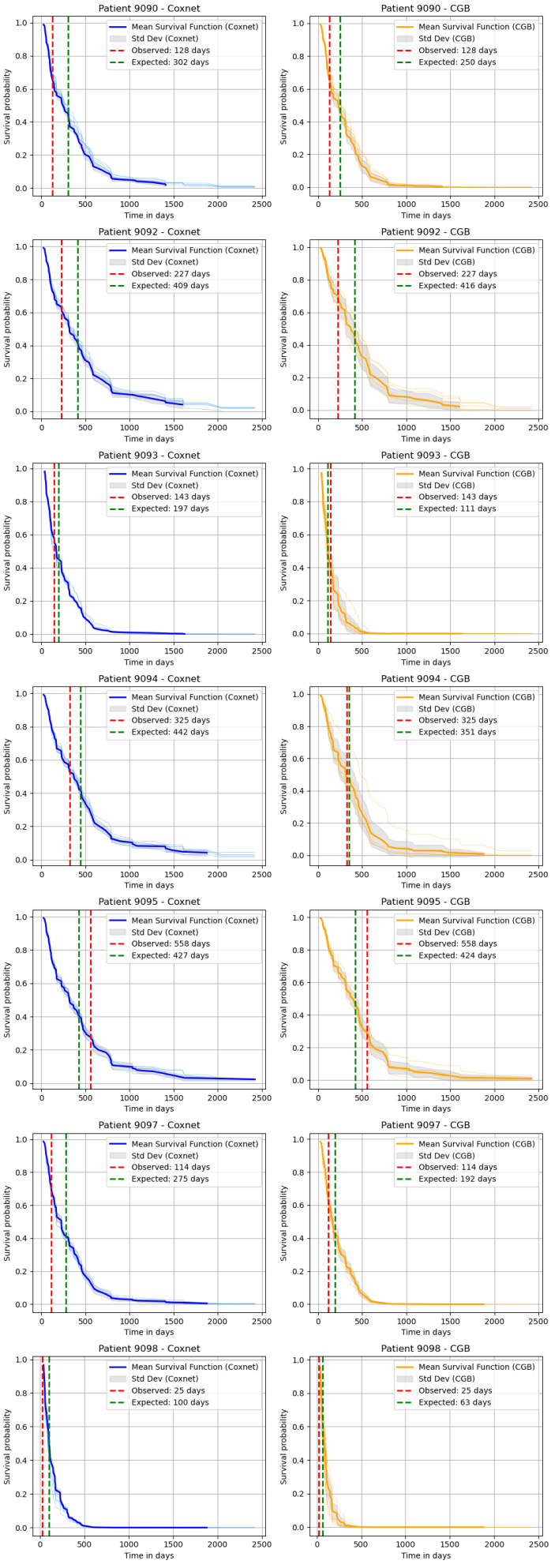

Patient 9044 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 459 days
Expected: 314 days
Patient 9044 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 459 days
Expected: 276 days
Patient 9045 - Coxnet
Observed: 168 days
Expected: 182 days
Patient 9045 - CGB
Observed: 168 days
Expected: 119 days
Patient 9047 - Coxnet
Observed: 412 days
Expected: 662 days
Patient 9047 - CGB
Observed: 412 days
Expected: 663 days
Patient 9048 - Coxnet
Observed: 85 days
Expected: 224 days
Patient 9048 - CGB
Observed: 85 days
Expected: 137 days
Patient 9049 - Coxnet
Observed: 309 days
Expected: 211 days
Patient 9049 - CGB
Observed: 309 days
Expected: 125 days
Patient 9051 - Coxnet
Observed: 67 days
Expected: 285 days
Patient 9051 - CGB
Observed: 67 days
Expected: 268 days
Patient 9052 - Coxnet
Observed: 778 days
Expected: 465 days
Patient 9052 - CGB
Observed: 778 days
Expected: 285 days
Survival probability
Time in days

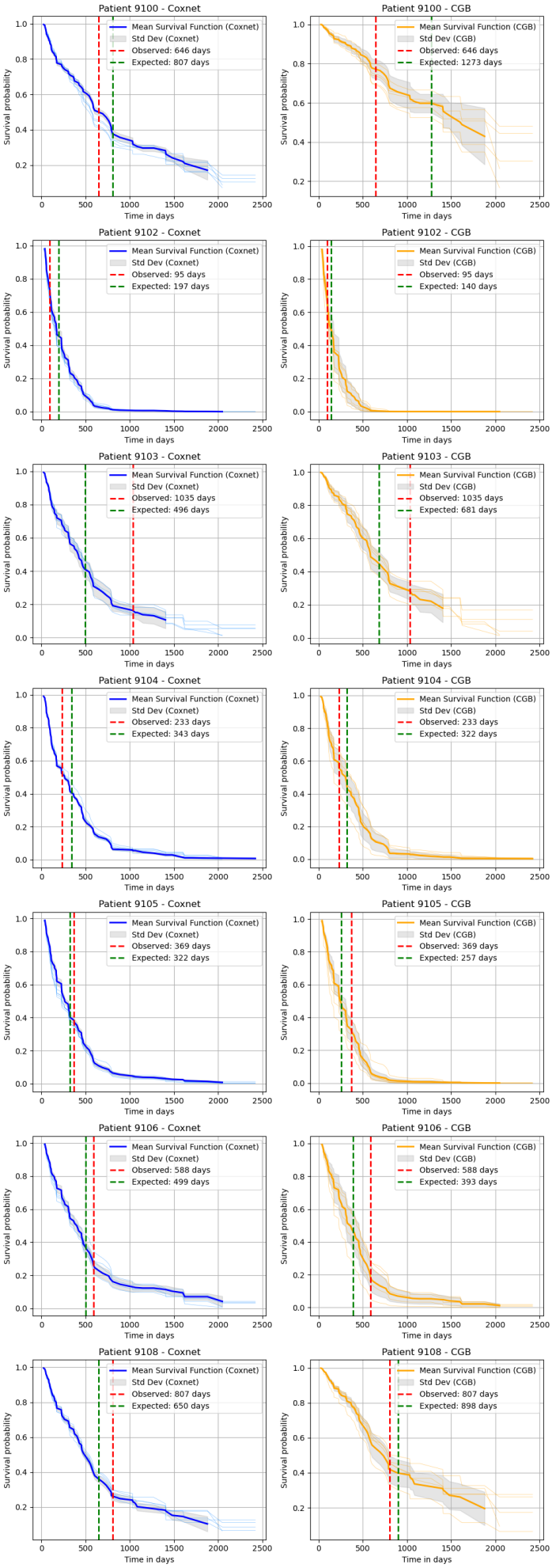

Patient 9053 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 83 days
Expected: 131 days
Patient 9053 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 83 days
Expected: 57 days
Patient 9054 - Coxnet
Observed: 792 days
Expected: 314 days
Patient 9054 - CGB
Observed: 792 days
Expected: 256 days
Patient 9055 - Coxnet
Observed: 319 days
Expected: 434 days
Patient 9055 - CGB
Observed: 319 days
Expected: 292 days
Patient 9057 - Coxnet
Observed: 542 days
Expected: 314 days
Patient 9057 - CGB
Observed: 542 days
Expected: 357 days
Patient 9058 - Coxnet
Observed: 375 days
Expected: 425 days
Patient 9058 - CGB
Observed: 375 days
Expected: 357 days
Patient 9059 - Coxnet
Observed: 96 days
Expected: 261 days
Patient 9059 - CGB
Observed: 96 days
Expected: 197 days
Patient 9060 - Coxnet
Observed: 453 days
Expected: 469 days
Patient 9060 - CGB
Observed: 453 days
Expected: 494 days
Survival probability
Time in days

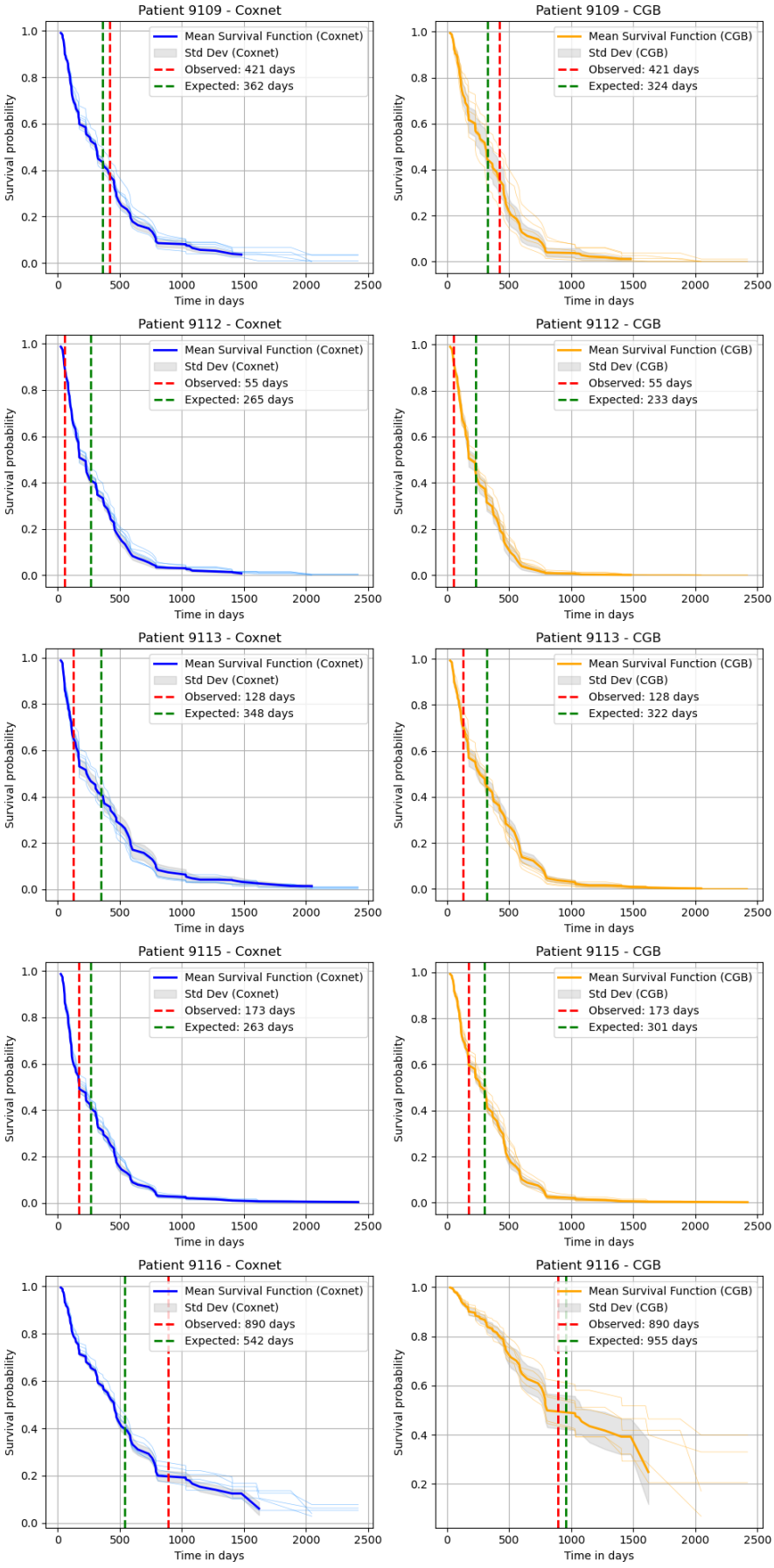

Patient 9064 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 57 days
Expected: 107 days
Patient 9064 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 57 days
Expected: 55 days
Patient 9066 - Coxnet
Observed: 558 days
Expected: 371 days
Patient 9066 - CGB
Observed: 558 days
Expected: 248 days
Patient 9067 - Coxnet
Observed: 110 days
Expected: 256 days
Patient 9067 - CGB
Observed: 110 days
Expected: 195 days
Patient 9068 - Coxnet
Observed: 167 days
Expected: 227 days
Patient 9068 - CGB
Observed: 167 days
Expected: 145 days
Patient 9069 - Coxnet
Observed: 103 days
Expected: 188 days
Patient 9069 - CGB
Observed: 103 days
Expected: 87 days
Patient 9070 - Coxnet
Observed: 514 days
Expected: 332 days
Patient 9070 - CGB
Observed: 514 days
Expected: 463 days
Patient 9071 - Coxnet
Observed: 733 days
Expected: 346 days
Patient 9071 - CGB
Observed: 733 days
Expected: 297 days
Survival probability
Time in days

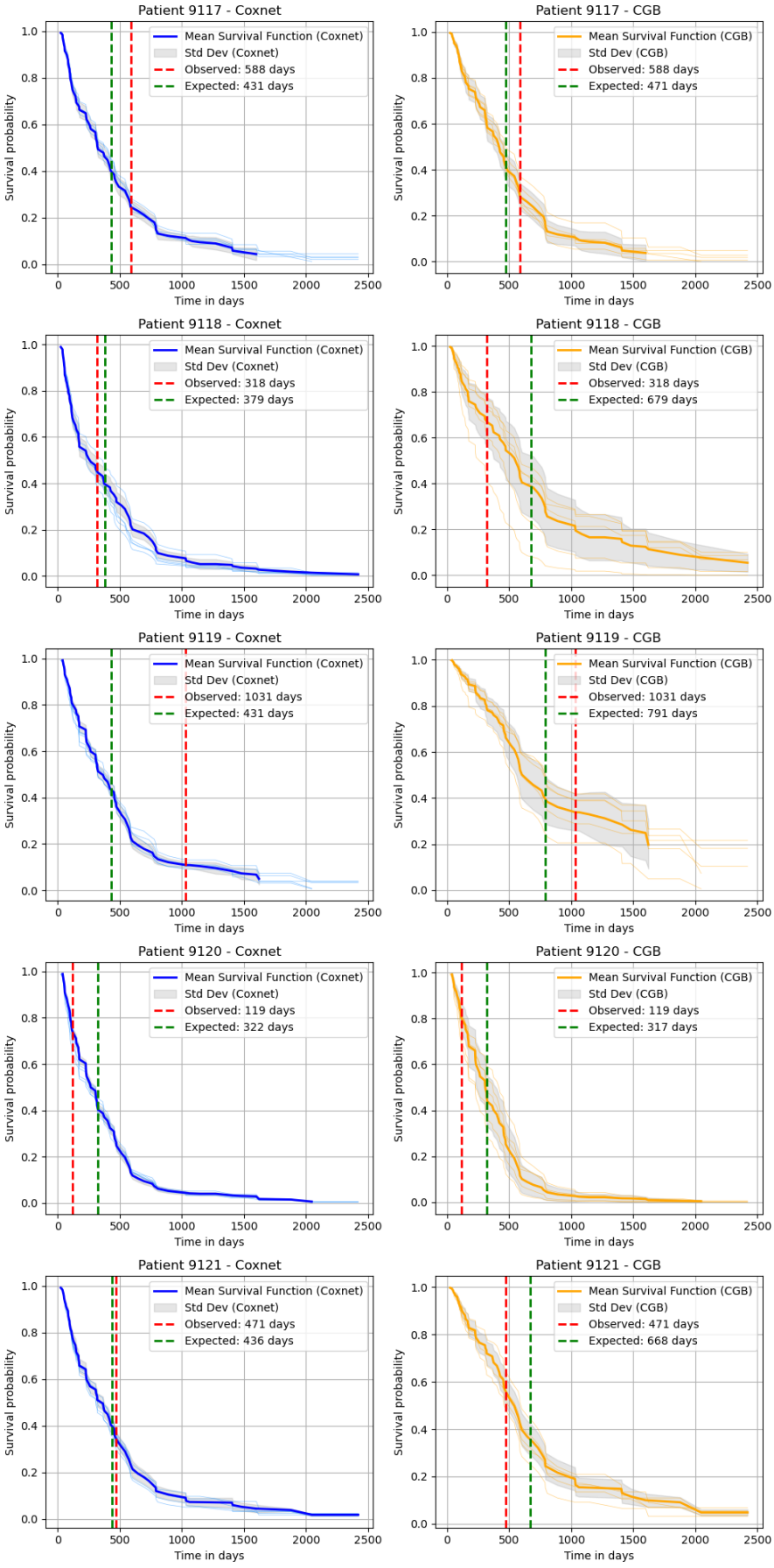

Patient 9072 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 174 days
Expected: 193 days
Patient 9072 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 174 days
Expected: 167 days
Patient 9073 - Coxnet
Observed: 583 days
Expected: 467 days
Patient 9073 - CGB
Observed: 583 days
Expected: 432 days
Patient 9074 - Coxnet
Observed: 50 days
Expected: 121 days
Patient 9074 - CGB
Observed: 50 days
Expected: 56 days
Patient 9075 - Coxnet
Observed: 99 days
Expected: 431 days
Patient 9075 - CGB
Observed: 99 days
Expected: 242 days
Patient 9077 - Coxnet
Observed: 111 days
Expected: 116 days
Patient 9077 - CGB
Observed: 111 days
Expected: 51 days
Patient 9078 - Coxnet
Observed: 306 days
Expected: 388 days
Patient 9078 - CGB
Observed: 306 days
Expected: 420 days
Patient 9079 - Coxnet
Observed: 603 days
Expected: 306 days
Patient 9079 - CGB
Observed: 603 days
Expected: 307 days
Survival probability
Time in days


Patient 9080 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 401 days
Expected: 710 days
Patient 9080 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 401 days
Expected: 810 days
Patient 9081 - Coxnet
Observed: 174 days
Expected: 326 days
Patient 9081 - CGB
Observed: 174 days
Expected: 189 days
Patient 9082 - Coxnet
Observed: 39 days
Expected: 134 days
Patient 9082 - CGB
Observed: 39 days
Expected: 86 days
Patient 9083 - Coxnet
Observed: 226 days
Expected: 372 days
Patient 9083 - CGB
Observed: 226 days
Expected: 343 days
Patient 9084 - Coxnet
Observed: 364 days
Expected: 297 days
Patient 9084 - CGB
Observed: 364 days
Expected: 331 days
Patient 9085 - Coxnet
Observed: 173 days
Expected: 315 days
Patient 9085 - CGB
Observed: 173 days
Expected: 356 days
Patient 9086 - Coxnet
Observed: 57 days
Expected: 429 days
Patient 9086 - CGB
Observed: 57 days
Expected: 490 days
Survival probability
Time in days


Patient 9090 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 128 days
Expected: 302 days
Patient 9090 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 128 days
Expected: 250 days
Patient 9092 - Coxnet
Observed: 227 days
Expected: 409 days
Patient 9092 - CGB
Observed: 227 days
Expected: 416 days
Patient 9093 - Coxnet
Observed: 143 days
Expected: 197 days
Patient 9093 - CGB
Observed: 143 days
Expected: 111 days
Patient 9094 - Coxnet
Observed: 325 days
Expected: 442 days
Patient 9094 - CGB
Observed: 325 days
Expected: 351 days
Patient 9095 - Coxnet
Observed: 558 days
Expected: 427 days
Patient 9095 - CGB
Observed: 558 days
Expected: 424 days
Patient 9097 - Coxnet
Observed: 114 days
Expected: 275 days
Patient 9097 - CGB
Observed: 114 days
Expected: 192 days
Patient 9098 - Coxnet
Observed: 25 days
Expected: 100 days
Patient 9098 - CGB
Observed: 25 days
Expected: 63 days
Survival probability
Time in days


Patient 9100 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 646 days
Expected: 807 days
Patient 9100 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 646 days
Expected: 1273 days
Patient 9102 - Coxnet
Observed: 95 days
Expected: 197 days
Patient 9102 - CGB
Observed: 95 days
Expected: 140 days
Patient 9103 - Coxnet
Observed: 1035 days
Expected: 496 days
Patient 9103 - CGB
Observed: 1035 days
Expected: 681 days
Patient 9104 - Coxnet
Observed: 233 days
Expected: 343 days
Patient 9104 - CGB
Observed: 233 days
Expected: 322 days
Patient 9105 - Coxnet
Observed: 369 days
Expected: 322 days
Patient 9105 - CGB
Observed: 369 days
Expected: 257 days
Patient 9106 - Coxnet
Observed: 588 days
Expected: 499 days
Patient 9106 - CGB
Observed: 588 days
Expected: 393 days
Patient 9108 - Coxnet
Observed: 807 days
Expected: 650 days
Patient 9108 - CGB
Observed: 807 days
Expected: 898 days
Survival probability
Time in days


Patient 9109 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 421 days
Expected: 362 days
Patient 9109 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 421 days
Expected: 324 days
Patient 9112 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 55 days
Expected: 265 days
Patient 9112 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 55 days
Expected: 233 days
Patient 9113 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 128 days
Expected: 348 days
Patient 9113 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 128 days
Expected: 322 days
Patient 9115 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 173 days
Expected: 263 days
Patient 9115 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 173 days
Expected: 301 days
Patient 9116 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 890 days
Expected: 542 days
Patient 9116 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 890 days
Expected: 955 days
Survival probability
Time in days


Patient 9117 - Coxnet
Mean Survival Function (Coxnet)
Std Dev (Coxnet)
Observed: 588 days
Expected: 431 days
Patient 9117 - CGB
Mean Survival Function (CGB)
Std Dev (CGB)
Observed: 588 days
Expected: 471 days
Patient 9118 - Coxnet
Observed: 318 days
Expected: 379 days
Patient 9118 - CGB
Observed: 318 days
Expected: 679 days
Patient 9119 - Coxnet
Observed: 1031 days
Expected: 431 days
Patient 9119 - CGB
Observed: 1031 days
Expected: 791 days
Patient 9120 - Coxnet
Observed: 119 days
Expected: 322 days
Patient 9120 - CGB
Observed: 119 days
Expected: 317 days
Patient 9121 - Coxnet
Observed: 471 days
Expected: 436 days
Patient 9121 - CGB
Observed: 471 days
Expected: 668 days
Survival probability
Time in days

Appendix C

C Coxnet Model Performance Across Hyperparameters

C.1 Coxnet Model Performance For L1 Ratio of 0.2

Table C.1: Coxnet models for different *alpha* values with a fixed *L1 Ratio* of **0.2**. This table presents the results of Coxnet models, detailing the C-index for both test and train set, and the IBS for test data across a range of *alpha* values. The table illustrates how performance metrics vary with different regularization strengths for *alpha*.

Alpha	**L1 Ratio**	**C-index test**	**C-index train**	**IBS test**
1000	0.2	0.500	0.500	0.161
700	0.2	0.500	0.500	0.161
500	0.2	0.500	0.500	0.161
200	0.2	0.500	0.500	0.161
100	0.2	0.500	0.500	0.161
70	0.2	0.500	0.500	0.161
50	0.2	0.500	0.500	0.161
20	0.2	0.500	0.500	0.161
10	0.2	0.500	0.500	0.161
5	0.2	0.500	0.500	0.161
3	0.2	0.500	0.500	0.161
1	0.2	0.784	0.813	0.129
0.5	0.2	0.786	0.834	0.115
0.1	0.2	0.764	0.875	0.114
0.05	0.2	0.753	0.883	0.122
0.01	0.2	0.718	0.889	0.148
0.005	0.2	0.710	0.889	0.161
0.001	0.2	0.700	0.889	0.173
0.0005	0.2	0.695	0.889	0.174
0.0001	0.2	0.694	0.889	0.177
0	0.2	0.693	0.889	0.177

C.2 Coxnet Model Performance For L1 Ratio of 0.3

Table C.2: Coxnet models for different *alpha* values with a fixed *L1 Ratio* of **0.3**. This table presents the results of Coxnet models, detailing the C-index for both test and train set, and the IBS for test data across a range of *alpha* values. The table illustrates how performance metrics vary with different regularization strengths for *alpha*.

Alpha	**L1 Ratio**	**C-index test**	**C-index train**	**IBS test**
1000	0.3	0.500	0.500	0.161
700	0.3	0.500	0.500	0.161
200	0.3	0.500	0.500	0.161
100	0.3	0.500	0.500	0.161
70	0.3	0.500	0.500	0.161
50	0.3	0.500	0.500	0.161
20	0.3	0.500	0.500	0.161
10	0.3	0.500	0.500	0.161
5	0.3	0.500	0.500	0.161
3	0.3	0.500	0.500	0.161
1	0.3	0.770	0.793	0.142
0.5	0.3	0.783	0.825	0.118
0.1	0.3	0.766	0.869	0.115
0.05	0.3	0.754	0.881	0.120
0.01	0.3	0.718	0.889	0.147
0.005	0.3	0.710	0.889	0.161
0.001	0.3	0.700	0.889	0.173
0.0005	0.3	0.695	0.889	0.174
0.0001	0.3	0.693	0.888	0.177
0	0.3	0.693	0.889	0.177

C.3 Coxnet Model Performance For L1 Ratio of 0.4

Table C.3: Coxnet models for different *alpha* values with a fixed *L1 Ratio* of **0.4**. This table presents the results of Coxnet models, detailing the C-index for both test and train set, and the IBS for test data across a range of *alpha* values. The table illustrates how performance metrics vary with different regularization strengths for *alpha*.

Alpha	**L1 Ratio**	**C-index test**	**C-index train**	**IBS test**
1000	0.4	0.500	0.500	0.161
700	0.4	0.500	0.500	0.161
200	0.4	0.500	0.500	0.161
100	0.4	0.500	0.500	0.161
70	0.4	0.500	0.500	0.161
50	0.4	0.500	0.500	0.161
20	0.4	0.500	0.500	0.161
10	0.4	0.500	0.500	0.161
5	0.4	0.500	0.500	0.161
3	0.4	0.500	0.500	0.161
1	0.4	0.685	0.717	0.158
0.5	0.4	0.782	0.812	0.123
0.1	0.4	0.769	0.864	0.115
0.050	0.4	0.754	0.879	0.119
0.010	0.4	0.720	0.889	0.148
0.005	0.4	0.710	0.889	0.160
0.001	0.4	0.699	0.888	0.172
0.0005	0.4	0.695	0.889	0.174
0.0001	0.4	0.693	0.889	0.177
0	0.4	0.693	0.889	0.177

C.4 Coxnet Model Performance For L1 Ratio of 0.5

Table C.4: Coxnet models for different *alpha* values with a fixed *L1 Ratio* of **0.5**. This table presents the results of Coxnet models, detailing the C-index for both test and train set, and the IBS for test data across a range of *alpha* values. The table illustrates how performance metrics vary with different regularization strengths for *alpha*.

Alpha	**L1 Ratio**	**C-index test**	**C-index train**	**IBS test**
1000	0.5	0.500	0.500	0.161
700	0.5	0.500	0.500	0.161
200	0.5	0.500	0.500	0.161
100	0.5	0.500	0.500	0.161
70	0.5	0.500	0.500	0.161
50	0.5	0.500	0.500	0.161
20	0.5	0.500	0.500	0.161
10	0.5	0.500	0.500	0.161
5	0.5	0.500	0.500	0.161
3	0.5	0.500	0.500	0.161
1	0.5	0.521	0.543	0.161
0.5	0.5	0.776	0.803	0.130
0.1	0.5	0.773	0.859	0.115
0.05	0.5	0.760	0.877	0.118
0.01	0.5	0.721	0.889	0.148
0.005	0.5	0.710	0.889	0.160
0.001	0.5	0.699	0.888	0.173
0.0005	0.5	0.696	0.888	0.174
0.0001	0.5	0.693	0.888	0.177
0	0.500	0.693	0.888	0.177

C.5 Coxnet Model Performance For L1 Ratio of 0.6

Table C.5: Coxnet models for different *alpha* values with a fixed *L1 Ratio* of **0.6**. This table presents the results of Coxnet models, detailing the C-index for both test and train set, and the IBS for test data across a range of *alpha* values. The table illustrates how performance metrics vary with different regularization strengths for *alpha*.

Alpha	**L1 Ratio**	**C-index test**	**C-index train**	**IBS test**
1000	0.6	0.500	0.500	0.161
700	0.6	0.500	0.500	0.161
200	0.6	0.500	0.500	0.161
100	0.6	0.500	0.500	0.161
70	0.6	0.500	0.500	0.161
50	0.6	0.500	0.500	0.161
20	0.6	0.500	0.500	0.161
10	0.6	0.500	0.500	0.161
5	0.6	0.500	0.500	0.161
3	0.6	0.500	0.500	0.161
1	0.6	0.500	0.791	0.161
0.5	0.6	0.767	0.791	0.138
0.1	0.6	0.777	0.855	0.115
0.05	0.6	0.760	0.874	0.118
0.01	0.6	0.723	0.889	0.147
0.005	0.6	0.710	0.889	0.159
0.001	0.6	0.699	0.888	0.172
0.0005	0.6	0.696	0.888	0.174
0.0001	0.6	0.693	0.888	0.177
0	0.6	0.693	0.888	0.177

C.6 Coxnet Model Performance For L1 Ratio of 0.7

Table C.6: Coxnet models for different *alpha* values with a fixed *L1 Ratio* of **0.7**. This table presents the results of Coxnet models, detailing the C-index for both test and train set, and the IBS for test data across a range of *alpha* values. The table illustrates how performance metrics vary with different regularization strengths for *alpha*.

Alpha	**L1 Ratio**	**C-index test**	**C-index train**	**IBS test**
1000	0.7	0.500	0.500	0.161
700	0.7	0.500	0.500	0.161
500	0.7	0.500	0.500	0.161
200	0.7	0.500	0.500	0.161
100	0.7	0.500	0.500	0.161
70	0.7	0.500	0.500	0.161
50	0.7	0.500	0.500	0.161
20	0.7	0.500	0.500	0.161
10	0.7	0.500	0.500	0.161
5	0.7	0.500	0.500	0.161
3	0.7	0.500	0.500	0.161
1	0.7	0.500	0.500	0.161
0.5	0.7	0.739	0.765	0.148
0.1	0.7	0.778	0.850	0.115
0.05	0.7	0.762	0.871	0.118
0.01	0.7	0.723	0.889	0.146
0.005	0.7	0.710	0.889	0.159
0.001	0.7	0.699	0.884	0.172
0.0005	0.7	0.696	0.888	0.174
0.0001	0.7	0.693	0.888	0.177
0	0.7	0.693	0.888	0.177

C.7 Coxnet Model Performance For L1 Ratio of 0.8

Table C.7: Coxnet models for different *alpha* values with a fixed *L1 Ratio* of **0.8**. This table presents the results of Coxnet models, detailing the C-index for both test and train set, and the IBS for test data across a range of *alpha* values. The table illustrates how performance metrics vary with different regularization strengths for *alpha*.

Alpha	**L1 Ratio**	**C-index test**	**C-index train**	**IBS test**
1000	0.8	0.500	0.500	0.161
700	0.8	0.500	0.500	0.161
500	0.8	0.500	0.500	0.161
200	0.8	0.500	0.500	0.161
100	0.8	0.500	0.500	0.161
70	0.8	0.500	0.500	0.161
50	0.8	0.500	0.500	0.161
20	0.8	0.500	0.500	0.161
10	0.8	0.500	0.500	0.161
5	0.8	0.500	0.500	0.161
3	0.8	0.500	0.500	0.161
1	0.8	0.500	0.500	0.161
0.5	0.8	0.682	0.715	0.156
0.1	0.8	0.777	0.846	0.115
0.05	0.8	0.762	0.868	0.119
0.01	0.8	0.724	0.889	0.145
0.005	0.8	0.710	0.889	0.159
0.001	0.8	0.698	0.888	0.172
0.0005	0.8	0.696	0.888	0.174
0.0001	0.8	0.693	0.888	0.176
0	0.8	0.693	0.889	0.177

C.8 Coxnet Model Performance For L1 Ratio of 0.9

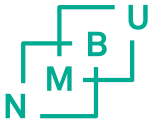
Table C.8: Coxnet models for different *alpha* values with a fixed *L1 Ratio* of **0.9**. This table presents the results of Coxnet models, detailing the C-index for both test and train set, and the IBS for test data across a range of *alpha* values. The table illustrates how performance metrics vary with different regularization strengths for *alpha*.

Alpha	**L1 Ratio**	**C-index test**	**C-index train**	**IBS test**
1000	0.9	0.500	0.500	0.161
700	0.9	0.500	0.500	0.161
200	0.9	0.500	0.500	0.161
100	0.9	0.500	0.500	0.161
70	0.9	0.500	0.500	0.161
50	0.9	0.500	0.500	0.161
20	0.9	0.500	0.500	0.161
10	0.9	0.500	0.500	0.161
5	0.9	0.500	0.500	0.161
3	0.9	0.500	0.500	0.161
1	0.9	0.500	0.500	0.161
0.5	0.9	0.594	0.626	0.160
0.1	0.9	0.775	0.843	0.115
0.05	0.9	0.765	0.866	0.119
0.01	0.9	0.723	0.889	0.145
0.005	0.9	0.712	0.888	0.159
0.001	0.9	0.698	0.888	0.172
0.0005	0.9	0.695	0.889	0.174
0.0001	0.9	0.693	0.889	0.177
0	0.9	0.693	0.889	0.177

Appendix D

D Code

Relevant code can be found at: https://github.com/erlendrisvik/Master-Thesis-2024/.



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway