



CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Nuclear Sciences and Physical Engineering



Estimating patient's life expectancy after a successful kidney transplant using machine learning methods

Odhad délky života pacienta po úspěšné transplantaci ledviny pomocí metod strojového učení

Bachelor's Degree Project

Author: **Kyrylo Stadniuk**

Supervisor: **Ing. Tomáš Kouřim**

Consultant: **Ing. Pavel Strachota, Ph.D.**

Language advisors: **PaedDr. Eliška Rafajová and Bc. Nathaniel Tobias Patton**

Academic year: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Kyrylo Stadniuk
Studijní program:	Aplikovaná informatika
Název práce (česky):	Odhad délky života pacienta po úspěšné transplantaci ledviny pomocí metod strojového učení
Název práce (anglicky):	Estimating patient's life expectancy after a successful kidney transplant using machine learning methods

Pokyny pro vypracování:

- 1) Prozkoumejte současný přístup k transplantacím ledvin, jeho problémy a výzvy. / Investigate the current approach to kidney transplantation, its problems and challenges.
- 2) Prozkoumejte příslušné metody strojového učení a metody pro hodnocení přesnosti modelu. / Explore applicable machine learning methods and model accuracy evaluation methods.
- 3) Vyčistěte, předzpracujte a rozšiřte stávající datovou sadu. / Clean, preprocess and extend the existing dataset.
- 4) Vytvořte prediktivní model strojového učení pro odhad délky života pacienta a ohodnoťte jeho přesnost. / Create a predictive machine learning model estimating a patient's life expectancy and evaluate its accuracy.
- 5) Navrhněte úpravy skórovacího algoritmu pro transplantace ledvin na základě výsledků prediktivního modelu. / Design an updated kidney matching compatibility scoring algorithm based on the prediction model.
- 6) Prozkoumejte možnost integrace dosažených výsledků do nástroje pro správu transplantací TX Matching. / Evaluate the possibility of integrating achieved results into kidney transplantation management tool TX Matching.

Doporučená literatura:

- 1) P. Bruce, A. Bruce, P. Gedeck, Practical Statistics for Data Scientists, O'Reilly, 2020.
- 2) I. H. Witten, E. Frank, M. A. Hall, Ch. J. Pal, Data Mining : Practical Machine Learning Tools and Techniques. Morgan Kaufman, 2017.
- 3) A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, 2019.
- 4) J. J. Kim, S. V. Fuggle, S. D. Marks, Does HLA matching matter in the modern era of renal transplantation? *Pediatr Nephrol* 36, 2021, 31–40.
- 5) R. Reindl-Schwaighofer, A. Heinzl, A. Kainz, et al., Contribution of non-HLA incompatibility between donor and recipient to kidney allograft survival: genome-wide analysis in a prospective cohort. *The Lancet* 393, 10174, 2019, 910-917.
- 6) M. Wohlfahrtová, O. Viklický, R. Lischke a kolektiv, Transplantace orgánů v klinické praxi. Grada, 2021.

Jméno a pracoviště vedoucího bakalářské práce:

Ing. Tomáš Kouřim
Mild Blue, s.r.o., Plzeňská 27, Praha 5

Jméno a pracoviště konzultanta:

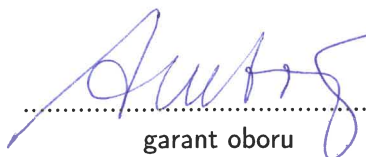
Ing. Pavel Strachota, Ph.D.
Katedra matematiky, Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze, Trojanova 13, 120 00 Praha 2

Datum zadání bakalářské práce: 31.10.2022


Datum odevzdání bakalářské práce: 2.8.2023

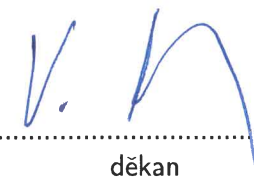
Doba platnosti zadání je dva roky od data zadání.

V Praze dne 31.10.2022


.....
garant oboru




.....
vedoucí katedry


.....
děkan

Acknowledgment:

I am grateful to Ing. Tomáš Kouřim for his expert guidance and to Dr. Pavel Strachota for his invaluable support and insightful feedback throughout this project. I would also like to extend my sincerest appreciation to PaedDr. Eliška Rafajová and Bc. Nathaniel Tobias Patton for their language assistance.

Author's declaration:

I declare that this Bachelor's Degree Project is entirely my own work and I have listed all the used sources in the bibliography. During the project, GitHub Copilot provided some assistance with coding. Additionally, I used Grammarly and ChatGPT to help with rephrasing and generating ideas for better sentence formulation. For grammar corrections, I relied on Grammarly.

Prague, August 5, 2024

Kyrylo Stadniuk

Název práce:

Odhad délky života pacienta po úspěšné transplantaci ledviny pomocí metod strojového učení

Autor: Kyrylo Stadniuk

Obor: Aplikovaná Informatika

Druh práce: Bakalářská práce

Vedoucí práce: Ing. Tomáš Kouřim Mild Blue, s.r.o.

Konzultant: Ing. Pavel Strachota, Ph.D., Katedra matematiky FJFI ČVUT

Abstrakt: Transplantace ledvin je nejlepší léčbou selhání ledvin, která ve srovnání s dialýzou přináší vyšší kvalitu a delší očekávanou délku života. Před transplantací se v naději na lepší výsledky maximalizuje kompatibilita dárce a příjemce, nicméně dlouhodobé přežití zůstává nejisté, protože je notoricky obtížně předvídatelné a chybí přesné prediktivní nástroje pro dlouhodobé výsledky po transplantaci. Tato práce se pokouší tuto mezeru překlenout trénováním několika prediktivních modelů strojového učení pro transplantace od žijících dárců (LDT) a zemřelých dárců (DDT), včetně regularizované Coxovy regrese, náhodných přežívacích lesů a rozhodovacích stromů s gradientním boostingem, s cílem odhadnout pravděpodobnost dlouhodobého přežití. Modely LDT dosáhly indexu shody Uno (c-index) 0.72 a integrovaného Brierova skóre (IBS) 0.136, zatímco modely DDT dosáhly c-indexu Uno 0.69 a IBS 0.162. S využitím těchto modelů byla vyvinuta aplikace, která má lékařům pomoci při rozhodování a potenciálně zlepšit přežití pacientů.

Klíčová slova: Gradient Boosting Survival Analysis, Předpověď přežití, Random Survival Forests, Regularizovaná Coxová regrese, Strojové učení, Transplantace ledviny

Title:

Estimating patient's life expectancy after a successful kidney transplant using machine learning methods

Author: Kyrylo Stadniuk

Abstract: Kidney transplantation is the gold standard treatment for kidney failure that provides an increased quality of life and a longer life expectancy compared to dialysis. Before transplantation, donor-recipient compatibility is maximized in hopes of better outcomes, yet long-term survival remains uncertain due to being notoriously hard to predict and lack of accurate predictive tools for long-term post-transplant outcomes. This thesis attempts to bridge this gap by training several predictive machine learning models for living donor (LDT) and deceased donor transplantations (DDT), including Cox elastic net regression, random survival forests, and gradient-boosted decision trees, to estimate long-term survival probabilities. LDT models achieved an Uno concordance index of 0.72 and an integrated Brier score (IBS) of 0.136, while DDT models achieved an Uno c-index of 0.69 and an IBS of 0.162. Leveraging these models, an application was developed to aid clinicians in decision-making, potentially improving patient outcomes.

Keywords: Gradient Boosted Survival Analysis, Kidney Transplantation, Machine Learning, Random Survival Forest, Regularized Cox regression, Survival Analysis

Contents

Introduction	8
1 Medical Background	10
1.1 Kidney — An Overview	10
1.2 Clinical Aspects of Kidney Transplantation	11
1.3 The History of Kidney Transplantation	12
1.4 The Introduction to Immunology	15
1.5 Immunology of Kidney Transplant	16
1.5.1 Immune Response to the Graft	16
1.5.2 Clinical Manifestations of Graft Rejection	18
1.5.3 Immunosuppression	19
1.5.4 Transplant Tolerance	20
1.6 Future of Kidney Transplantation	20
2 Machine Learning Background	21
2.1 Supervised Learning	21
2.1.1 Linear Regression	22
2.1.2 Logistic Regression	23
2.1.3 Support Vector Machines	25
2.1.4 Decision Trees	27
2.1.5 Stochastic Gradient Boosting	29
2.2 Unsupervised Learning	30
2.3 Data Preparation	31
2.3.1 Handling Categorical Features	31
2.3.2 Feature Scaling	32
2.3.3 Handling Missing Feature Values	32
2.4 Model Training and Hyperparameter Tuning	33
2.5 Survival Analysis	34
2.5.1 Essential Concepts	34
2.5.2 Taxonomy of Survival Analysis Methods	39
2.5.3 Statistical Methods	40
2.5.4 Machine Learning Methods	45
2.5.5 Performance Metrics	47
3 Data Preparation and Analysis	50
3.1 Data Acquisition	50
3.2 Data Loading	50

3.3	Data Preprocessing Pipeline	51
3.4	Exploratory Data Analysis	52
3.4.1	General Data	52
3.4.2	Deceased Group	57
3.4.3	Living Group	59
3.5	Dataset Building, Exclusion Criteria and Noise Reduction	63
3.6	Feature Engineering	63
4	Machine Learning Models	64
4.1	Problem Formulation	64
4.2	Model Selection and Training Approach	64
4.3	Results	65
4.3.1	Coxnet	66
4.3.2	Random Survival Forest	69
4.3.3	Gradient Boosting Survival Analysis	72
4.4	Scoring	78
4.5	Discussion	80
5	Applications	83
5.1	Existing Solutions	83
5.1.1	TX Matching	84
5.1.2	KidneyMatch	85
5.1.3	KPDGUI	85
5.2	KidneyLife	85
5.2.1	Overview	86
5.2.2	Architecture	86
5.2.3	Deployment	89
5.2.4	Functionality and UX	89
5.2.5	Future Work	89
	Conclusion	92

Introduction

Chronic kidney disease (CKD) is a common disease with an incidence of 8 to 16% of the population, often leading to kidney failure [35, 66]. *Kidney transplantation (KT)* is the best treatment for it, but organs are scarce. Only in the US, every 10 minutes, a new patient joins the transplant waiting list, and 17 people die each day while waiting [67]. Given this shortage, it is paramount to maximize the longevity of a recipient. Although some scoring techniques consider the estimated longevity [61], there is a lack of software dedicated solely to this problem.

Artificial intelligence (AI) and its subset, machine learning (ML), have gained popularity with their ability to find patterns in large amounts of data and solve practical problems, finding their uses in medicine. The most prominent ones in genetic and molecular research have led to the discovery of new therapeutic targets, enabling the development of novel treatments [65]. This thesis aims to: 1) train predictive machine learning models capable of accurately estimating patient life expectancy; 2) develop a minimum-viable-product (MVP) application based on those models.

Kidneys are vital organs with functions such as regulating fluid balance and aiding red blood cell production. Conditions like diabetes and hypertension, often linked to lifestyle choices, can impair kidney function, leading to CKD. Routine health checks usually identify this gradual decline, but in some cases, the patient starts to experience the first symptoms, such as flank pain, "foamy urine", nocturia, and decreased urine output. With advanced CKD, the patient might exhibit nausea, poor appetite, fatigue, weight loss, shortness of breath, itching, or peripheral edema [66]. If kidneys fail and a suitable donor is not available, the patient must undergo dialysis, a machine mimicking kidney function. Dialysis severely limits patient mobility and life expectancy. These factors make transplantation preferable. Moreover, performing transplantation as early as possible is crucial, as prolonged dialysis tends to adversely affect the subsequent transplantation.

There are two types of transplantation: living donor transplantation (LDT) and deceased donor transplantation (DDT). LDT provides superior results, but DDT is much more prevalent. Prior to transplantation, donors and recipients undergo compatibility assessment. Blood groups must be compatible, and the recipient immune system must not be sensitized to the donor human leukocyte antigens (HLA), proteins on cell surfaces crucial for immune system host recognition. After the transplantation, the recipient must take multiple immunosuppressive drugs and visit regular checkups to prevent rejection and other complications caused by the treatment.

Machine learning (ML) is a subset of artificial intelligence (AI) known for its ability to process large amounts of data, find complex relationships, and make predictions. It has found tremendous popularity across numerous fields, including medicine. Here, it is utilized for drug discovery, diagnostics, enhancing patient care, and personalized treatment [65]. We will utilize machine learning methods for survival analysis on the UNOS dataset to create several models to estimate the life expectancy after transplantation. These models might help clinicians make more informed decisions about transplantation.

Now, let us review the structure of this work. In Chapter 1, we explore the medical aspects of kidney transplantation. Specifically, we review the organ characteristics, common causes of its failure, clini-

cal aspects of its transplantation, and the immunology of transplantation. Chapter 2 explores machine learning and survival analysis. We review the main branches of ML and elaborate on the most prevalent techniques used in the branch most relevant to our task. In addition to that, we discuss approaches to data preparation, model training, and hyperparameter tuning. Lastly, we provide a comprehensive overview of survival analysis (SA), its methods, and machine-learning-based survival analysis techniques. Chapter 3 is devoted to data. In this chapter, we analyze the UNOS dataset, including examining value distributions and the influence of various features on survival. Moreover, we describe the data acquisition, data loading, and the preprocessing pipeline, along with the feature engineering process.

Chapter 4 presents the results of our study and thoroughly evaluates trained models for the deceased and living donor subgroups. As part of this, we examine model performance metrics and analyze their changes over time. Furthermore, we analyze feature importance to understand the factors that drive model predictions. Models are compared based on training experience, training time, and practicality. Finally, the results are discussed and compared with the state-of-the-art. The chapter concludes with the presentation of a prototype of transplantation scoring. Chapter 5 focuses on the software for KT. We review existing software solutions and introduce KidneyLife, an inference application based on the trained models. The chapter delves into its architecture and design principles.

Chapter 1

Medical Background

1.1 Kidney — An Overview

The kidneys are reddish-brown, bean-shaped organs located on both sides of the spine beneath the lower ribs. They serve several critical functions, such as filtering blood, regulating blood pressure, aiding red blood cell production, and maintaining fluid balance. Structurally, a kidney consists of about 1 million nephrons, each consisting of a small filter called a glomerulus, attached to a tubule, responsible for filtering waste products from the blood and excreting them with small amounts of fluid as urine [33].

Each kidney performs 50% of the normal kidney function. However, if one kidney is lost, the remaining kidney can adapt, increasing its capacity to as much as 75% of normal function. Kidney function is measured with *glomerular filtration rate (GFR)* [33]. There are two types of GFR: *measured GFR (mGFR)* and *estimated GFR (eGFR)*. Measuring mGFR is problematic and lengthy, so healthcare practitioners usually use a formula to calculate eGFR. An eGFR of 90 and above is the normal range. An eGFR of 15-89 is a range for different stages of chronic kidney disease. Values below 15 might indicate renal failure [38].

Impairment in kidney function can have serious health consequences. There are many reasons why kidneys may become dysfunctional, such as end-stage chronic kidney disease, heavy metal poisoning, polycystic kidney disease, infection, nephrotoxic drugs, lupus, and physical injury [36, 37]. Among these conditions, chronic kidney disease is the most prevalent — approximately 10% of the population worldwide is affected by some form of chronic kidney disease [35]. *Chronic kidney disease (CKD)* is an incurable condition characterized by a gradual reduction in kidney function over time [34, 35]. Diabetes and high blood pressure are responsible for the majority of cases of chronic kidney disease, with high blood sugar damaging the glomeruli and high blood pressure damaging the kidney blood vessels [34, 37]. The latter interaction is associated with a negative feedback loop, where it is unclear whether kidney damage causes high blood pressure or vice versa [35].

CKD progresses in 5 stages, ultimately culminating in *end-stage renal disease (ESRD)*, where kidneys fail. Treatment options include *dialysis*, a mechanical process that substitutes kidney function but cannot fully replicate it. It leads to a reduced life expectancy of, on average, 5 to 10 years. Alternatively, *kidney transplantation* offers a more permanent solution but requires a compatible donor, waiting for which can take many months or years, and the use of immunosuppressants after the transplantation [36].

In conclusion, the kidneys play an essential role in maintaining overall health with their ability to filter waste, regulate blood pressure, and balance fluids. Kidney failure has profound health implications, and there are multiple reasons for it. Chronic kidney disease, resulting from various causes such as diabetes and hypertension, progressively impairs kidney function, leading to end-stage renal disease. While dialysis offers temporary relief, it cannot fully substitute for a healthy kidney, which brings us to

the crucial role of kidney transplantation. Kidney transplantation offers a more conclusive and sustainable solution for patients with ESRD, albeit accompanied by challenges related to donor compatibility and life-long immunosuppression. Both of which will be discussed in the subsequent section.

1.2 Clinical Aspects of Kidney Transplantation

According to [32], kidney transplantation stands as the most commonly performed organ transplant worldwide. This prevalence is partly due to the high incidence of diseases such as diabetes and chronic kidney disease, which often lead to chronic renal failure. Unlike organs such as the liver and heart, kidney transplantation is relatively straightforward, contributing to its frequency of transplantation. Kidneys can be obtained from both deceased and living donors - relatives or altruistic volunteers. Living donors can donate one kidney and continue to lead normal and healthy lives, significantly expanding the potential donor pool.

ABO blood group and histocompatibility (defined in Section 1.5) matchings for kidney transplantation are of utmost importance. Proper matching substantially reduces the risk of rejection and improves transplant outcomes. Compared to organs such as the liver or bone marrow, kidneys do not present additional challenges such as graft versus host disease (GVHD), simplifying the transplantation process.

However, kidney transplantation, despite its advances, faces significant challenges. Two critical issues confront potential recipients: the scarcity of available organs and the risk of sensitization following a failed first transplant, decreasing the pool of available donors even further. In such cases, the immune system develops antibodies to the graft alloantigens – proteins on the surface of the cell that are recognized by the recipient’s immune system as foreign. An antigen is any substance that triggers an immune response, and an alloantigen is a type of antigen that varies between individuals of the same species and helps the body distinguish its cells from the cells of an invader. The antibodies bind with these alloantigens, marking the cells for destruction by the recipient’s immune system. Hyperacute rejection, a rapid and severe rejection of the transplanted organ, occurs when antibodies react immediately with the organ’s alloantigens. This happens mostly if the donor-recipient blood groups were mismatched or the recipient was sensitized to similar alloantigens. Most likely with the previous transplant. This fact often leaves many patients unable to find a compatible donor after one or two rejection episodes.

Recipients of kidney transplants typically require life-long immunosuppression. While immunosuppressants are essential to prevent organ rejection, they are associated with severe side effects, such as increased risk of cancer, hypertension, and metabolic bone disease.

Given the high incidence of renal failure in diabetic patients, which occurs in 30% of individuals with advanced diabetes, simultaneous kidney and pancreas transplants are sometimes performed. This approach addresses renal failure and underlying diabetes, offering a comprehensive solution for these patients.

In summary, kidney transplantation has become a vital and frequent medical procedure, offering improved quality of life to many suffering from kidney failure. While simpler than other organ transplants, the procedure has its own challenges. The following section will examine the history of kidney transplantation, showing its path from the initial experiments to the most frequently performed organ transplantation today. The historical perspective will provide insights into the advancements in surgical techniques, immunosuppressive therapy, and donor-recipient matching that have shaped the current state of kidney transplantation.

1.3 The History of Kidney Transplantation

Early Animal Experiments

As outlined in [1], advancements in surgical methods at the beginning of the 20th century eventually led to experiments with organ transplantation. One of the first recorded transplantations was an *autograft*, where the donor and recipient are the same individual, performed by Emerich Ullmann on March 1, 1902, at the Vienna Medical School. Ullmann successfully connected the dog's kidney to the vessels of its neck, which resulted in the urine production. The success of this experiment was notable enough to be presented to the Vienna Medical Society, sparking considerable interest.

That year, other experiments followed. Alfred von Decastello performed a dog-to-dog kidney *allograft*, a transplant between two individuals of the same species, at the Institute of Experimental Pathology in Vienna. Although initially, the transplanted kidney produced urine, it eventually ceased. Later, Ullmann performed a dog-to-goat kidney *xenograft*, a transplantation between individuals of different species, which also resulted in brief function time.

At the same time, in Lyon, Alexis Carrel and his colleagues were working on vascular suturing methods. Carrel's technique, known as Carrel's seam, was a considerable improvement over existing methods, addressing the common issues of thrombosis, hemorrhage, stricture, and embolism. His consequent move to The Rockefeller Institute for Medical Research in the United States led to further refinements of his method and the documentation of organ rejection. Carrel received the Nobel Prize in Medicine in 1912 for his contributions [2].

These early experiments, although varied in their outcomes, were crucial in shaping the future of organ transplantation. They not only illustrated the possibility of organ transplantation but also highlighted the challenges, such as rejection, that would direct the course of future research. The insights gained in animal experiments laid the groundwork for the next big milestone in transplantation medicine: the advent of human organ transplantation. This transition from animals to humans marked the beginning of a new era in medical history.

Early Human Transplantation

The first recorded human renal *xenografts* were credited to Mathieu Jaboulay in 1906. It involved a pig and a goat as donor animals. One kidney was transplanted to the arm and the second to the thigh. Although each kidney functioned only for an hour, these efforts marked the beginning of human transplantation attempts [1, 4]. Ernst Unger's xenografts in 1909 gained more attention. His first attempt, involving the transplantation of a kidney from a stillborn baby to a baboon, resulted in a lack of kidney function despite a successful connection of vessels. Inspired by a successful surgery, Unger attempted a monkey-to-human xenograft, which also failed.

These experiments demonstrated the technical feasibility of kidney transplantation but also exposed the challenge of graft rejection. In a 1914 lecture, Alexis Carrel mentioned J. B. Murphy's work on irradiation and benzol treatment, suggesting their potential to improve graft survival [1]. Inspired by these findings, Carrel conducted his experiments with irradiation, achieving prolonged graft survival. However, these findings were never formally published [1].

The 1930s and 1940s were stagnant compared to the beginning of the century. European surgical centers that studied transplantology before were in decline. Meanwhile, the Mayo Clinic in the US was conducting some cautious experiments without considering Carrel's works and attempts at immunosuppression [1].

However, it was during that period that Yurii Voronyi achieved a significant milestone. On March 3, 1933, in Kherson, Ukraine, Voronyi performed the first human kidney allograft on a woman suffering

from acute renal failure due to mercury chloride poisoning. Given the ethical concerns surrounding living donors and previous failures of xenografts, Voronyi considered a cadaveric transplant the only viable option. Although there was initial urine production, the transplant failed 48 hours post-surgery due to blood group incompatibility and prolonged warm ischemia, triggering an immune reaction. Despite this setback, Voronyi continued to perform similar transplantations. He viewed these transplants as temporary measures to bridge the gap until the recipient's kidneys could recover. Out of the six transplants he performed, two patients experienced a complete recovery, regaining normal kidney function [4].

The pioneering work of Jaboulay, Unger, and Voronyi demonstrated the technical feasibility of human organ transplantation and highlighted its main challenge of graft rejection. Despite the progress, the field has yet to witness success, primarily due to the lack of effective immunosuppression. In the next section, we will explore essential moments of the 1950s that transformed human renal transplantation from a daring experiment to a feasible medical procedure.

First Successes

In 1946, at the Peter Bent Brigham Hospital in Boston, a group of surgeons performed kidney transplantation under local anesthetic on the arm vessels. The short period of kidney functioning may have helped the patient recover from acute renal failure, igniting the hospital's interest in renal transplantation.

Meanwhile, European surgeons were making significant advancements. Notably, Simonsen in Denmark, Dempster in London, and Küss in Paris concluded that placing the kidney in the pelvis is preferable. Furthermore, both Simonsen and Dempster deduced that the immune response was responsible for graft failure and hypothesized that the humoral mechanism of rejection was probable.

The early 1950s marked a period of active experimentation. In Paris, Jean Hamburger reported the first live-related kidney transplant between a mother and her child, achieving the immediate function of the transplanted kidney for 22 days until it was rejected. Meanwhile, in Boston, a series of nine transplantations with the thigh position of the allograft was closely studied. Moreover, in 1953, David Hume introduced the pre-transplant use of hemodialysis. Although some success was achieved with the administration of the adrenocorticotrophic hormone (more known as cortisone), it was deemed clinically insignificant. Hume's further research suggested the potential benefits of prior blood transfusions, blood group compatibility, and removal of both host's kidneys for transplant success - insights later validated with further research. On December 23, 1954, in Boston, Joseph Murray performed a kidney allograft from one identical twin to another, bypassing the rejection barrier. From that time, many similar surgeries were performed in Boston [1, 3].

To conclude, the early successes during the 1950s marked a transformative period in the history of kidney transplantation. Works of Hume, Murray, and others underlined the critical role of immune response in graft survival, starting the quest for effective immunosuppression, to which the subsequent section is dedicated.

Attempts in Immunosuppression

According to [3], the exploration of immunosuppression in transplantation began as early as the late 1940s. At the Mayo Clinic in 1948, patients with rheumatoid arthritis were administered cortisone, an adrenal cortical hormone with mild immunosuppressive properties, which provided temporary relief. Although initially praised, its effects were deemed clinically insignificant for transplantation purposes. It led researchers to revisit earlier experiments with irradiation. Experiments on mice by Joan Main and Richmond Prehn showed promising results, inspiring human trials in Boston and Paris.

In 1958, Murray's team in Boston employed radiation in human transplantation, achieving a significant breakthrough with a 20-year-long kidney transplant between non-identical twins. Similarly, in Paris, Jean Hamburger's team accomplished a 26-year functioning transplant using radiation.

The quest for safer immunosuppressive methods led to the anticancer drug 6-mercaptopurine (6-MP). In 1959, Schwarz and Dameshek published a paper that described how 6-MP lowered immune response to foreign proteins in rabbits. Inspired by their work, Roy Calne performed his dog experiments, showing promising results backed by Charles Zukoski and David Hume. Despite initial setbacks, Küss and others reported prolonged graft survival from non-related donors using total body irradiation (TBI) complemented by 6-MP. The introduction of azathioprine in 1959, a derivative of 6-MP, by Gertrude Elion and George Hitchings, further improved results. Their groundbreaking work earned them the Nobel Prize, and by 1961, azathioprine was available for human use.

In 1963, a conference held by the National Research Council revealed a bleak outlook for kidney transplantation, with less than 10% survival beyond three months. It changed when Tom Starzl presented a protocol combining 6-MP with prednisone, leading to over one-year graft survival in 70% of cases. His results revolutionized the field, resulting in 50 new US transplantation programs and setting a twenty-year standard in post-transplant immunosuppression.

In summary, the late 1940s to the early 1960s was a period of discoveries and experimentation. The use of cortisone in 1948, the administration of 6-MP for immunosuppression, and the invention of azathioprine in 1959 led to better immunosuppression, culminating in Tom Starzl's protocol, which became the standard for the next two decades. The following section is dedicated to a phase of steady developments in the literature referred to as a plateau that would solidify the practice of kidney transplantation.

Plateau

[3] suggests that, from 1964 to 1980, kidney transplantation saw gradual progress. Dialysis, developed during WWII, finally became available for chronic renal failure as a result of the invention of Teflon arteriovenous conduits in the 1960s. Acceptance of brain death expanded donor pools. Organ preservation techniques also advanced: total body hypothermia was replaced by targeted cold solutions infusions that preserved organs better—by the mid-60s, longer preservation times allowed organ exchanges between centers. Concerns about equitable distribution of organs led to the National Transplant Act in 1984, establishing the United Network of Organ Sharing (UNOS) for oversight.

As techniques improved and donor pools expanded from 1964 to 1980, so did understanding of the nuances of immune compatibility. The following section describes the developments in tissue typing that would further expand the field of kidney transplantation.

Tissue Typing

As [3] further explains, the concept of tissue typing, suggested by Alexis Carrel in the early 20th century, remained unproven until Jean Dausset discovered the first human leukocyte antigen (HLA) in 1958 (more on HLA in 1.5). It was not until 1964, with Paul Terasaki's development of the microcytotoxicity assay, that testing for antibodies became reliable. The test involved mixing the donor's lymphocytes with the recipient's serum, which swiftly became the standard, known as the *crossmatch test*.

For several years, Terasaki performed typing for most U.S. transplant centers and found a couple of observations: 1) Positive crossmatch test predicts hyperacute rejection. The test is performed by mixing the recipient's serum with the donor's blood cells; if there is a reaction, the test is considered positive, and the transplantation cannot be performed. 2) Matching can reliably identify optimal donors within a family, and it was assumed that the same principle would apply to non-related recipients.

However, in 1970, Terasaki's review of his extensive database of cadaver renal allografts revealed no correlation with the typing. This raised much agitation in the tissue typing community, and his grant was temporarily suspended until others reported the same. Since then, many crucial histocompatibility antigens have been discovered (Class II locus: D and DR; more on antigens in Section 1.5). Histocompatibility matching remains essential in bone marrow transplantation and in selecting family donors.

While tissue typing and matching have played a crucial role in improving transplantation outcomes, the evolution of immunosuppressive drugs has been equally, if not more, significant. The subsequent section elaborates on new developments in immunosuppressive drugs in the coming years.

Advancements in Immunosuppression

In continuation with [3], the discovery of cyclosporine in 1976 by Jean-François Borel marked a significant milestone in transplantation. As a fungal derivative with potent immunosuppressive properties, cyclosporine drastically improved the outcomes of both renal and extra-renal transplants, surpassing the efficacy of the previously used drug, azathioprine. Similar to 6-MP, to achieve the best results, it had to be combined with prednisone. This protocol remained standard until 1989, when Tacrolimus, an even more powerful immunosuppressive agent, was introduced. Tacrolimus proved effective in cases where the combination of cyclosporine and prednisone was insufficient, effectively replacing cyclosporine as a usual baseline agent.

In conclusion, the history of renal transplantation has been marked by groundbreaking discoveries and persistent challenges. From the pioneering attempts of xenografts in the early 20th century to the technical advancements and immunological breakthroughs of the mid-century — each milestone has been essential in shaping the current landscape of organ transplantation. As we explored the history of kidney transplantation, a deeper understanding of the immune system has become crucial. The following section examines the fundamentals of immunology, laying the foundation for understanding the interplay between the immune system and the transplanted organ.

1.4 The Introduction to Immunology

As detailed in [6], the *immune system* is a sophisticated defense mechanism that evolved to protect multicellular organisms from pathogens such as bacteria, fungi, viruses, and parasites. It is a network of many cells and tissues that compose a complex system that detects, evaluates, and responds to the invader. It is essential to understand these mechanisms, as the immune response plays a crucial role in graft acceptance or rejection.

Innate and adaptive immunity are the two interconnected systems of immune response. *Innate immunity* (also *natural* or *native*) includes primitive built-in cellular and molecular mechanisms that provide rapid, albeit non-specific responses to common pathogens. In contrast, *adaptive (specific or acquired) immunity* is slower to respond but capable of providing more targeted responses.

Physical barriers, including epithelia and mucous membranes, constitute the host's first line of defense and are a part of innate immunity. This branch of the immune system prevents invaders from infiltrating the host and quickly destroys many microbes that succeed in breaching these barriers. Innate immunity provides the necessary protection until adaptive immunity is activated. It also communicates how to best respond to the invader to the adaptive immunity. Furthermore, innate immunity plays a vital role in the clearance of dead tissue and the repair initiation after the tissue is damaged.

Adaptive immunity is broadly categorized into *humoral* and *cell-mediated immunity*. *Lymphocytes*, also known as *white blood cells*, are central to both humoral and cell-mediated immune responses. *B lymphocytes (B cells)* mediate humoral response by producing antigen-specific antibodies on encounter

with the antigen. Produced antibodies then bind themselves to *antigens* — foreign molecular structures identified by common molecular patterns known as *pathogen-associated molecular patterns (PAMPs)* — to mark them for destruction. The immune system uses *pathogen recognition receptors (PRRs)*, found on the surface of T cells, in conjunction with antibodies to detect and categorize these PAMPs, which can take the form of molecules on the surface of a pathogen or its by-products. PRRs bind to PAMPs and initiate a targeted cascade of events culminating in the pathogen's elimination.

On the other hand, T lymphocytes start to proliferate when encountering an antigen, forming an army of T cells. The created 'army' will eliminate the invader and will form long-term memories about the pathogen. Activated T cells are divided into the following categories: helper T cells ($CD4^+$), which help B cells to produce antibodies and help kill ingested microbes; *cytotoxic T cells* ($CD8^+$), which target and kill infected cells; *regulatory T lymphocytes*, which prevent or limit immune responses; and *memory cells*, which remain in the body long-term to provide faster and stronger immune response if the same antigen is encountered in the future.

Pathogen-host interaction is a continuous arms race, as pathogens usually have a short life cycle and can modify their DNA to elude the host's recognition systems. The immune system counters this with the generation of host-tolerant lymphocytes with diverse PRRs during the development in bone marrow. Cells that react to the host's own cells are eliminated, ensuring that only non-self-reactive cells are allowed in circulation. The principle of recognizing self vs. non-self is called tolerance.

In conclusion, the immune system is a complex network of molecules, cells, tissues, and organs that cooperate in protecting the organism from pathogens. The system can be divided into two main branches: innate and adaptive, which cooperate in protecting the host from infections while developing long-term immunity to specific pathogens. Understanding the mechanisms of the immune system is essential to understanding the domain of kidney transplantation.

1.5 Immunology of Kidney Transplant

As outlined in [32], the degree to which the immune system responds to a graft depends on the type of graft. There are four types of grafts:

- **Autograft** is body tissue transfer from one body site to the other in the same individual.
- **Isograft** is a tissue transplanted between two genetically identical individuals. In humans, it is usually homozygotic (identical) twins.
- **Allograft** is a tissue transplanted between two genetically non-identical individuals of the same species.
- **Xenograft** is a tissue transplanted between individuals of different species.

Autografts and isografts are generally accepted because they are genetically identical. Allografts, being genetically different, are typically identified as foreign by the immune system and rejected. Xenografts, which have the most significant genetic differences, are the most vigorously rejected by the immune system.

1.5.1 Immune Response to the Graft

As detailed in [1], there are three reasons why the immune system may react to the allograft: damage due to ischemia, reaction to the incompatible blood group antigens, and reaction to major histocompatibility complex (MHC) and minor histocompatibility (miH) antigens.

Ischemia-Reperfusion Injury The transplantation process inevitably includes termination of blood flow and, as a result, oxygenation. This interruption in blood supply inhibits the cell's ability to generate sufficient energy to maintain homeostasis, leading to cellular damage or death. If this happens, the kidney is said to experience *ischemia-reperfusion injury (IRI)*. IRI is a significant factor in the success of kidney transplantation. When the blood flow to the ischemic kidney is restored, dead cells trigger an innate immune response. It is associated with the release of *danger-associated molecular patterns (DAMP)* from the compromised cells, serving as a critical alert mechanism. These DAMPs are recognized by both innate and adaptive immunity, although the former is predominately activated. Such an immune response might damage the graft even more and contribute to acute rejection. Because of that, the duration of ischemia is crucial in predicting the graft (and patient) survival.

Blood Group Compatibility ABO blood group antigens, expressed by most cell types, play a crucial role in the compatibility of organ transplants. If the transplantation between incompatible pairs were performed, it would result in a severe and often immediate reaction known as hyperacute antibody-mediated rejection (AMR or ABMR). The rejection risk is so high that the ABO blood group compatibility is the first thing checked before performing the transplant.

There are four primary blood groups: A, B, O, and AB. Within this system, individuals with blood group O are so-called "universal donors." Their organs can be transplanted to recipients with any ABO blood group. Whereas recipients in the AB group can safely receive organs from recipients with any ABO blood group and are called "universal recipients." In clinical practice, however, particularly with deceased donors, the preference is to match organ recipients with ABO-identical organs to prevent potential inequities in organ allocation and access. In the case of living donors, the principle of ABO-identity is somewhat more flexible, where organs from ABO-compatible donors are considered acceptable for transplantation.

Histocompatibility According to [32], genetically similar tissues, known as *histocompatible*, are less likely to trigger an immune response post-transplant, as the immune system does not recognize the transplanted tissue as foreign. Conversely, *histoincompatible* tissues, characterized by genetic differences, typically trigger an immune response. Although more than 40 distinct loci encode the antigens responsible for histocompatibility, the most vigorous allograft-rejection responses are attributed to loci within the *major histocompatibility complex (MHC)*. The organization of MHC in humans is called *human leukocyte antigen (HLA)*.

As detailed in [1], histocompatibility antigens, genetically encoded antigens that cover cell surfaces, play a crucial role in recognizing self versus non-self. In all vertebrates, histocompatibility antigens are categorized into a single *major histocompatibility complex (MHC)* and numerous *minor histocompatibility (miH)* systems. Mismatches in either MHC or miH result in an immune reaction, most severe in the case of MHC. Rejection in MHC-compatible donor-recipient pairs is usually delayed and, in some cases, forever. However, the miH mismatch might be so drastic that it would be comparable to the MHC mismatch.

The MHC itself is divided into class I and class II antigens. MHC class I antigens cover the surfaces of most cells and can activate cytotoxic CD8 cells. MHC class II antigens, found on specific immune cells, play an essential role in immune response coordination. In humans, each MHC class is further divided into three subgroups, as illustrated in Table 1.1.

However, [32] remarks, that class I and II mismatches differ in their impact on graft survival. A mismatch of one or two class I antigens has little effect on graft survival, whereas a single mismatch in class II antigen is equivalent to a mismatch of three or four class I antigens. When there are mismatches in both class I and II antigens, the risk and severity of the rejection are notably increased.

Table 1.1: MHC class division

MHC class I	MHC class II
HLA-A	HLA-DR
HLA-B	HLA-DP
HLA-C	HLA-DQ

HLA Typing and Matching As [1] reports, in clinical practice, clinicians assess and try to match donors and recipients according to the number of HLA-A, -B, and -DR mismatches, ranging from zero mismatches (0-0-0) to a maximum of 6 mismatches (2-2-2). Generally, more emphasis is placed on DR loci due to the capability of CD4 T cell activation, which might trigger both humoral and cellular adaptive immune responses.

As detailed in [32], HLA typing of potential donors and recipient is conducted with a *microcytotoxicity test*. The subject's white blood cells are distributed into various wells on a microtiter plate, after which specific antibodies for various class I and class II MHC alleles are added to different wells. Following incubation, complement is introduced to the wells, and cytotoxicity is assessed by the cells' absorption of various dyes. If a cell has an MHC allele for which a particular antibody is specific, then the cell will die upon the addition of a complement, and these dead cells will take up a dye.

Even when a fully HLA-compatible donor is not available, the transplantation still can be successful. In this situation, a one-way mixed-lymphocyte reaction (MLR) test can be used to assess the degree of class II MHC compatibility. The irradiated (or treated with mitomycin C) potential donor's lymphocytes play the role of stimulator, and the unaltered recipient's lymphocytes take a responder role. Then, the proliferation of recipient cells is measured. The intense recipient leukocyte proliferation indicates a poor prognosis for graft survival. The MLR gives a better understanding of the degree of CD4 cell activation. However, it takes several days to run the assay, often making it less suitable in the case of cadaver transplantation compared to a quicker microcytotoxicity test, which takes only a couple of hours. Notably, even perfect HLA matching does not guarantee rejection-free transplantation. Minor histocompatibility loci mismatches can still lead to graft rejection, necessitating at least some level of immunosuppression even in HLA-identical pairings. It is particularly critical in kidney and bone marrow transplants, where HLA matching is vital, whereas heart and liver transplantation can withstand higher levels of mismatching.

Mechanisms of Graft Rejection As [32] further explains, graft rejection is mainly due to cell-mediated immune response to alloantigens, predominantly MHC antigens, present on the graft cells. The process of cell-mediated graft rejection is divided into two phases. The first is known as the sensitization phase, which involves the recognition of MHC and miH alloantigens by helper CD4 and killer CD8 cells, leading to their proliferation. The second stage, the effector stage, is where the actual destruction of the graft occurs.

A range of effector mechanisms are involved in graft rejection. Cell-mediated reactions involving delayed-type hypersensitivity and cytotoxic T lymphocyte-mediated cytotoxicity are the most prevalent. Less common mechanisms are antibody-plus-complement lysis (cell disintegration) and destruction by antibody-dependent cell-mediated cytotoxicity (ADCC).

1.5.2 Clinical Manifestations of Graft Rejection

In continuation with [32], the onset and severity of graft rejection varies depending on the organ transplanted and the underlying immune response mechanisms involved. Rejection can be categorized

into three types: hyperacute, acute, and chronic rejections.

Hyperacute rejections occur within the first 24 hours post-transplant. This type of rejection occurs due to the prior sensitization to graft antigens, with specific antibodies already present in the bloodstream. It may happen due to several reasons: recipients of repeated blood transfusions sometimes develop significant amounts of antibodies to MHC antigens present on white blood cells of the transfused blood; women, through repeated pregnancies, can become sensitized to paternal alloantigens of the fetus; individuals with previous grafts may have antibodies against alloantigens of that graft; and naturally occurring antibodies to blood group antigens are always present, although pre-transplant blood group matching has made rejections of this nature rare.

Acute rejections typically occur within the first few weeks after transplant. They are primarily mediated by adaptive immunity via T-cell responses.

Chronic rejections, on the other hand, occur months to years after the transplantation and are mediated by both humoral and cell-mediated mechanisms. While the advancements in immunosuppression and tissue typing significantly improve short-term survival, little progress has been made in terms of long-term survival. Chronic rejections are hard to manage with immunosuppressants and may necessitate another transplantation.

1.5.3 Immunosuppression

As stated in [32], most of the available immunosuppression methods, while essential in preventing graft rejections, have the disadvantage of being non-specific, meaning they suppress immune reactions to all antigens, not only ones of the graft, placing the recipient at the risk of infection and, in some cases, more severe complications. For instance, some drugs target the speed of proliferation of activated lymphocytes. In doing so, they slow down the proliferation of all cells in the body. It can be particularly problematic for rapidly dividing cells, such as those of gut epithelia or bone marrow hematopoietic stem cells, placing patients at risk of severe or even life-threatening complications. Moreover, long-term use of immunosuppressive agents puts patients at risk of cancer, hypertension, and metabolic bone disease. In this section, we will delve into commonly used immunosuppressive methods.

A class of drugs known as *mitotic inhibitors* is used to inhibit rapid cell division across all cells. These are administered just before and after the transplantation to stop T-cell proliferation. The most commonly used mitotic inhibitors include *Azathioprine*, *Cyclophosphamide*, and *Methotrexate*. However, their broad action on cell division can lead to significant adverse effects.

Corticosteroids, another class of drugs, are often used in conjunction with mitotic inhibitors to prevent acute rejection. They have anti-inflammatory properties and act on different levels of immune response. Commonly used corticosteroids are *prednisone* and *dexamethasone*.

Fungal metabolites, such as *Cyclosporine A (CsA)*, *Tacrolimus*, and *rapamycin*, are also widely used due to their potent immunosuppressive properties in heart, liver, kidney, and bone marrow transplants. A substantial drawback is their nephrotoxicity. While CsA has been widely used, Tacrolimus and rapamycin, being much more potent, might be administered at much lower doses, minimizing the side effects.

Radiation, known for its effectiveness in destroying lymphocytes, is another lymphocyte elimination method employed just before transplantation. In one such procedure, lymph nodes, spleen, and thymus are irradiated. Later, newer, more tolerant to the graft alloantigens lymphocytes will emerge, as the bone marrow was unaffected.

Ideal immunosuppression would be antigen-specific, targeting only alloantigens of the graft, preserving the recipient's ability to respond to infections. Such specific immunosuppression has not been achieved in humans yet, but animal models suggest its feasibility.

1.5.4 Transplant Tolerance

Considering the detrimental effect of long-term immunosuppression, one of the primary objectives in transplantation is the induction of immunologic non-responsiveness (tolerance) to an allograft. The literature describes several pathways of immune non-responsiveness generation. However, it has yet to go further than animal models [1]. Tolerance is usually induced by prior exposure to donor antigens in a way that causes immune tolerance rather than sensitization in the recipient [32].

1.6 Future of Kidney Transplantation

Despite significant advances in recent years, the current approach to treating kidney failure with kidney transplantation has severe disadvantages. While it does save lives, its limitations make it an unsatisfactory long-term solution. Primary limitations are the limited lifespan of transplanted organs and a lifetime dependence on immunosuppressive drugs. These drugs come with their own set of problems, including an increased risk of infections and cancer.

Several solutions have been proposed over the years to address the limitations of kidney transplantation. The most promising solutions are 3D-printed organs, "ghost" organs, and pig-grown organs.

3D-printed Organs *3D printing* is a manufacturing technology that has gained popularity in recent years. It has also found its way into the medical field, where it holds significant promise for solving the shortage of organs for transplantation. However, according to [59], it faces the following challenges: 1) High cost in terms of time and money. 3D printers, materials, and software remain very expensive. 2) The limited ability of the material to mimic soft tissue. 3) Poor production precision. It is caused by limited imaging and 3D printing resolution. The limited space in 3D printers and the resulting need to divide large models into a couple of smaller ones and the subsequent assembling also contribute to poor precision.

Lab-grown Organs (Ott et al., 2008) [57] and (Sánchez et al., 2015) [58] removed cells from a human heart, leaving only an extracellular matrix (ECM), and then repopulated it with cells from another human. ECM is a structure that consists of connective tissue and gives the organ its form. The ECM does not have HLA or any other PAMPs, so it does not trigger an immune response. This approach will allow the use of organs that would otherwise be unusable due to incompatibilities and reduce waiting times from years to weeks or months. Also, the body will see the organ as its own, so immunosuppressants will not be needed. Theoretically, even non-human organ ECM can be utilized for transplantation to humans. Combined with recent advancements in stem cell production, this approach looks very promising.

Pig-grown organs Finally, pig-grown organs have also been researched as a potential solution to the organ shortage problem. Given the anatomical and physiological similarities between pigs and humans, xenotransplantation has emerged as a viable approach [60]. For instance, (Nagashima & Hitomi Matsunari, 2016) injected pancreatic progenitor cells into a pig fetus with an apancreatic trait and observed successful fetal development and pancreatic function. Although the approach appears promising, how it would work with human organs is unknown.

Chapter 2

Machine Learning Background

Machine learning is a subfield of computer science that consists of building algorithms capable of processing large amounts of data, finding patterns, and performing actions such as predictions or generating new data. It is an intersection of many fields of science, such as statistics, theory of probability, linear algebra, calculus, and certainly, computer science.

As detailed in [9], machine learning excels in problems that are either overly complex or have no known algorithm. It has the potential to help us create new knowledge by uncovering previously unknown correlations within the data. It might also make fewer errors in decision-making than humans.

Every field is affected by human errors, and medicine is no exception. Machine learning also makes mistakes, but achieving machine learning error even 1% below the human error rate would be a remarkable accomplishment. The human body is a complex system, and it is very difficult for the human mind to comprehend all processes and how they relate to each other. Machine learning might be able to do that better than any human could.

As [9] further explains, based on the problem and an approach to building a dataset and the model, there are four types of learning: *supervised*, *semi-supervised*, *unsupervised*, and *reinforcement learning*.

Supervised learning deals with labeled data, meaning that training data contains the desired solutions, referred to as *labels*.

Semi-supervised learning deals with partially labeled data, which needs to be labeled fully either manually, or using techniques such as *clustering*.

Unsupervised learning deals with unlabeled data.

In *reinforcement learning*, we create an environment, set up rewards for performing certain actions and punishment for others, and let the machine (actor) perform actions that produce the highest reward.

In this chapter, we will cover all theoretical backgrounds that might prove useful for solving our problem, including classical machine learning, statistical survival analysis, basic steps that are required to create machine learning systems, and data preprocessing. We will begin by exploring supervised learning.

2.1 Supervised Learning

Drawing heavily on [10] this section explores supervised learning and its algorithms. *Supervised learning* is the process of training a model on data where the outcome is known, to make predictions for data where the outcome is not known [12]. *Classification* and *regression* are common supervised learning tasks. In this section we will define these problems and the necessary terminology, and describe commonly used algorithms that are used to solve these types of problems.

In supervised learning the *dataset* is the collection of labeled samples $\{(\bar{x}_i, y_i)\}_{i=1}^N$, where each individual \bar{x}_i is called a *feature vector*. A feature vector is a vector that in each its dimension $j = 1, \dots, D$ contains a value that describes a sample in some way. This value is called a *feature* and is denoted as $x^{(j)}$. The *label* y^i might be either a finite set of classes $\{1, 2, \dots, C\}$, in case of a classification task, or a real number, a vector, a matrix or graph, in case of a regression. The goal of supervised learning algorithm is to create a model using the dataset that will take the feature vector as an input and produce a label or a more complex structure as an output.

Classification is a problem of assigning a label to an unlabeled sample. This problem is solved by a classification learning algorithm that takes a labeled set of samples as input and produces a model that takes an unlabeled sample as input and outputs a label. If the set of labels has only two classes we talk about *binary classification*. Consequently, if the set of labels has three or more classes, it is a *multiclass classification*. Some algorithms are binary classifiers by definition while others are multiclass classifiers. It is possible to create an *ensemble* out of binary classifiers that will be able to perform multiclass classification. An ensemble is a combination of algorithms that are connected to perform one task.

Regression is a problem of predicting a *target value* from an unlabeled example. The problem is solved by a regression learning algorithm that takes a set of labeled samples as input, and produces a model that takes an unlabeled sample as input and outputs a target value.

Classification and regression tasks are similar in many ways and often for each classifier there is an equivalent regressor, and vice versa. In the following subsections we are going to explore some techniques for supervised learning.

2.1.1 Linear Regression

Linear regression is a commonly used algorithm for regression learning. The resulting model is a linear combination of all features [10].

The problem formulation we are trying to solve is as follows: Given a collection of labeled samples $\{(\bar{x}_i, y_i)\}_{i=1}^N$, create a model

$$f_{\bar{w},b}(\bar{x}) = \bar{w}\bar{x} + b, \quad (2.1)$$

where N is the size of the collection, \bar{x}_i is a *feature vector* of D dimensions of sample $i = 1, \dots, N$, every feature $x_i^{(j)} \in \mathbb{R}$, $j = 1, \dots, D$, $y_i \in \mathbb{R}$ is the *target value*. \bar{w} is a D -dimensional vector of parameters and $b \in \mathbb{R}$. The notation $f_{\bar{w},b}(\bar{x})$ means that f is parametrized by \bar{w} and b .

To train the linear regression means to find optimal values (\bar{w}^*, b^*) of parameters \bar{w} and b so that the model makes as accurate predictions as possible. In graphical terms, it means finding such a hyperplane that fits data points from the training set as well as possible, as shown in Figure 2.1.

To find optimal parameters, we need to minimize

$$\frac{1}{N} \sum_{i=1 \dots N} (f_{\bar{w},b}(\bar{x}_i) - y_i)^2. \quad (2.2)$$

This expression is called a *mean squared error (MSE)*, the *loss function* that comprises of *squared error loss* $(f_{\bar{w},b}(\bar{x}_i) - y_i)^2$, another loss function that evaluates individual predictions. The loss function measures the model's overall performance (MSE) or evaluates each prediction (square error loss) [10].

There is a closed-form solution for finding optimal values (\bar{w}^*, b^*) . A *closed-form solution* is a simple algebraic expression that gives the result directly. In the case of linear regression, it is the *normal equation*, and it is given by

$$\bar{\mathbf{w}}^* = (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \mathbf{y}, \quad (2.3)$$

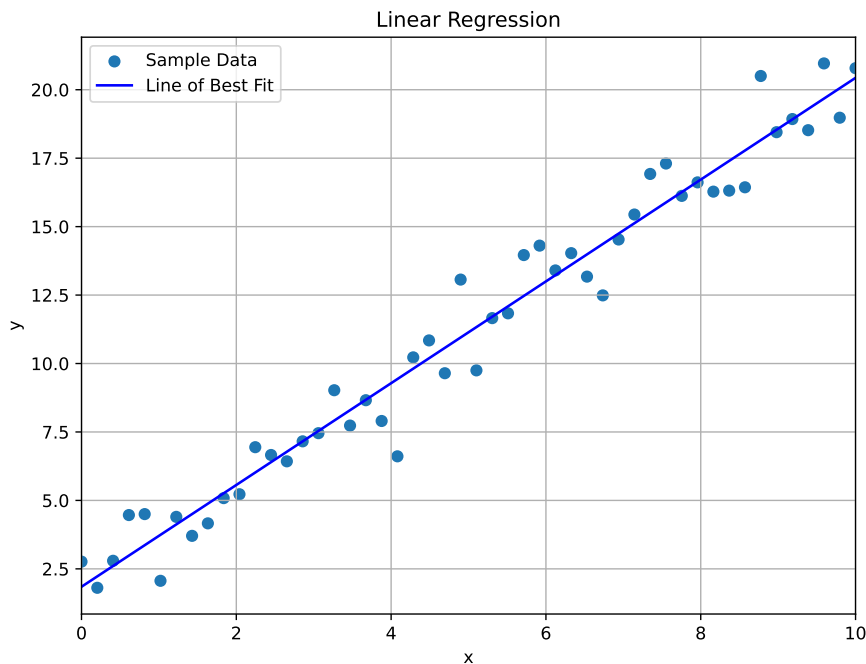


Figure 2.1: Linear regression for Two-Dimensional Data

where x^T means transposed feature matrix x .

We could select another loss function, but according to Andriy Burkov, it would be a different algorithm. For example, we could take the absolute difference between $f(x_i)$ and y_i , but that would create problems as the derivative of absolute value is not continuous. Therefore, the function is not smooth, which might create unnecessary complications during optimization.

Linear models are usually resilient to overfitting because they are simple. The model overfits when it learns the intricacies of the training dataset so well that it remembers actual values instead of learning the underlying pattern. Such a model cannot make accurate predictions when confronted with unseen data. More on overfitting in Section 2.4.

2.1.2 Logistic Regression

Logistic regression is a binary classifier that estimates the probability of an example belonging to a particular class [10]. If the predicted probability of the instance belonging to a class is greater than 50%, then the model concludes that it belongs to the class (referred to as positive class and labeled as 1). Otherwise, it predicts that the example does not belong to that class (but belongs to the negative class, labeled 0) [9]. Logistic regression comes from statistics, where its mathematical formulation is similar to a regression, hence the name [10]. Multiclass classification is available in softmax regression, a multiclass variant of logistic regression [9].

As with linear regression, in logistic regression, we want to model y as a linear combination of \bar{x} , but in this case, it is more complex [10].

The logistic regression model is given by

$$f_{\bar{w},b}(\bar{x}) \stackrel{\text{def}}{=} \frac{1}{1 + e^{-(w\bar{x}+b)}}. \quad (2.4)$$

Similar to linear regression, our task is to find optimal values (\bar{w}^*, b^*) for parameters \bar{w} and b [10].

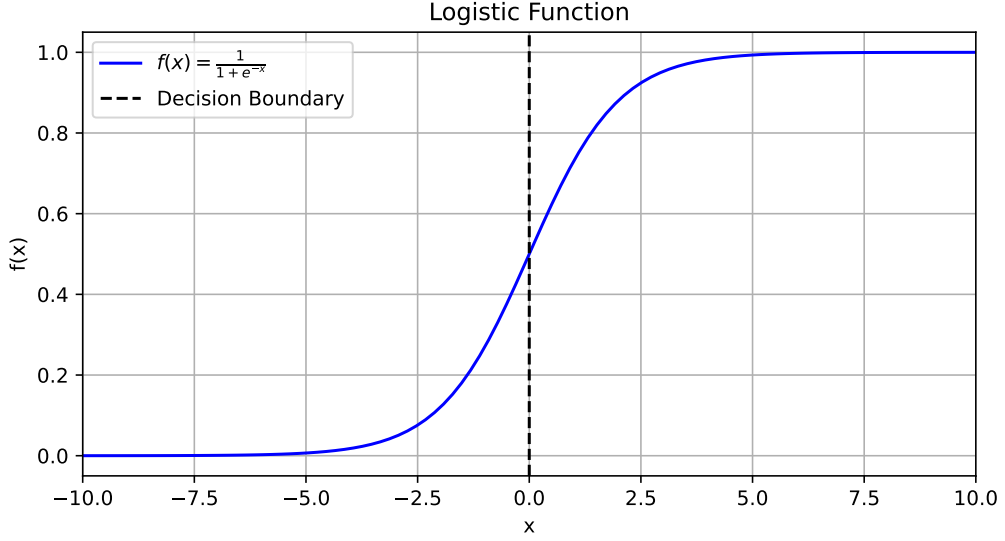


Figure 2.2: Logistic Function

Once we found (\bar{w}^*, b^*) for the Equation 2.4, in other words, we trained the model, we can apply the model 2.4 on features x_i from an example (x_i, y_i) . The output value lies in the range $0 < p < 1$. If y_i is a positive class, the likelihood of y_i being a positive class is given by p . Consequently, if y_i is the negative class, the likelihood of it being the negative class is given by $1 - p$ [10].

Figure 2.2 illustrates the logistic function. The y -axis is a decision boundary whose value corresponds to the predicted probability. The x -axis corresponds to a feature's value. In Figure 2.2, we can see that when y is less than $\frac{1}{2}$, the corresponding x values are negative, classifying it as a negative class. Conversely, if y is greater than $\frac{1}{2}$, it is classified as positive. However, the specific threshold may vary depending on the context.

According to [10], in logistic regression, instead of *minimizing* MSE, we try to *maximize* the *likelihood function*. In statistics, the likelihood function tells how likely the example is according to our model. The objective function in logistic regression is called *maximum likelihood*. It is given by

$$L_{\bar{w}, b} \stackrel{\text{def}}{=} \prod_{i=1 \dots N} f_{\bar{w}, b}(\bar{x}_i)^{y_i} (1 - f_{\bar{w}, b}(\bar{x}_i))^{(1-y_i)}. \quad (2.5)$$

On the other hand, due to the exponential function in Equation 2.5, it is better to use the *log-likelihood* instead to make calculations easier. As Log is a strictly increasing function, maximizing it is the same as maximizing its argument. The solution to this optimization problem is the same as the solution to the original problem. The log-likelihood function is expressed as

$$\log L_{\bar{w}, b} \stackrel{\text{def}}{=} \ln(L_{\bar{w}, b}(\bar{x})) \sum_{i=1}^N y_i \ln f_{\bar{w}, b}(\bar{x}) + (1 - y_i) \ln(1 - f_{\bar{w}, b}(\bar{x})). \quad (2.6)$$

Unfortunately, this optimization problem has no closed-form solution [9, 10]. Nonetheless, the function is convex. Hence, gradient descent (or any other optimization algorithm) guarantees the finding of the global minimum, provided that the learning rate is not too large and enough time is given [9].

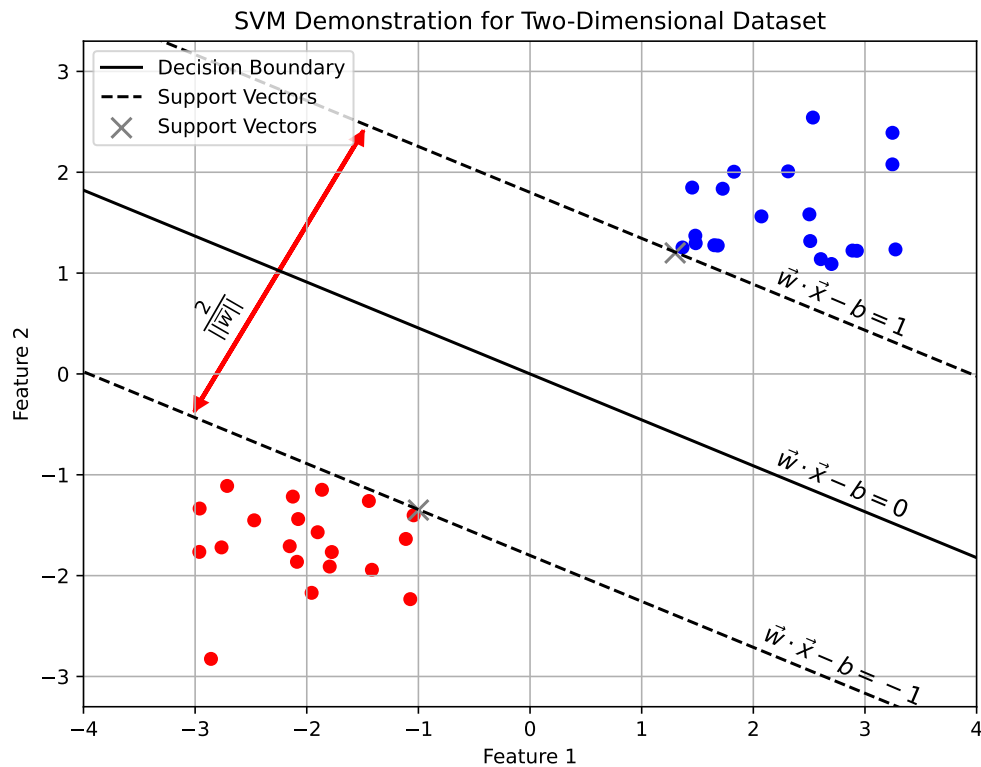


Figure 2.3: SVM Demonstration for Two-Dimensional Dataset

2.1.3 Support Vector Machines

Support vector machine (SVM) is a widely used and powerful machine learning algorithm that can perform a wide range of tasks, including linear and nonlinear classification, regression, and outlier detection on small- to medium-sized datasets [10].

Linear SVM

In its classical formulation, the support vector machine is a binary classifier. Classes are called positive and negative and are labeled +1 and -1, respectively.

The model is described by the equation

$$f(x) = \text{sign}(\bar{w}x - b).$$

The function sign returns +1 if the input is positive and -1 if negative. To train the SVM means to find optimal values (\bar{w}^*, b^*) of parameters \bar{w} and b so that the model makes as accurate predictions as possible. The process of finding (\bar{w}^*, b^*) is called training.

Figure 2.3 demonstrates the concept behind support vector machines. The image consists of two classes represented by red and blue dots, divided by a solid line termed the *decision boundary* $\bar{w}x - b = 0$, with two dashed lines by its sides known as *support vectors* $\bar{w}x - b = 1$ and $\bar{w}x - b = -1$. Support vectors are defined by the closest instances of a class to the decision boundary. These instances are emphasized in the figure.

The distance between the closest instances of two classes is called *margin* and is equal to $\frac{2}{\|\bar{w}\|}$, where $\|\bar{w}\|$ is the Euclidean norm and \bar{w} is a parameter vector of the same dimensionality as the feature vector.

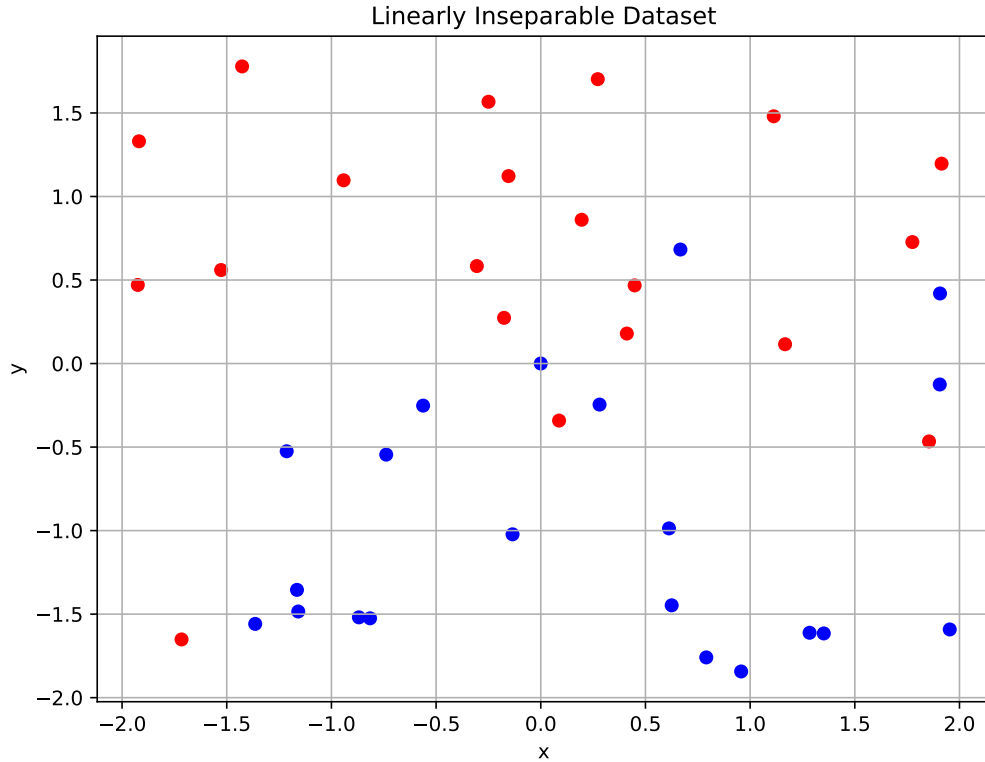


Figure 2.4: Linearly Inseparable Dataset

Thus, the smaller the norm, the larger the margin. The larger the margin, the better the model's generalization. The primary objective of the model is to find the largest possible margin $\frac{2}{\|\bar{w}\|}$, so, to do that, we need to *minimize* the Euclidian norm defined by the expression

$$\|\bar{w}\| = \sqrt{\sum_{j=1}^D (w^{(j)})^2}.$$

The fundamental assumption of support vector machines is that classes are linearly separable, implying their instances can be separated by a hyperplane (decision boundary) with no examples of one class lying among the ones of the opposite class. It is illustrated in the figure 2.4. In this case, the algorithm cannot find an optimal solution with no instances lying between the support vectors and the decision boundary. Consequently, the model is susceptible to outliers.

Every optimization problem requires constraints, and for the support vector machine, they are given by

1. $\bar{w}x_i - b \geq +1$ if $y_i = +1$
2. $\bar{w}x_i - b \leq -1$ if $y_i = -1$.

These two equations can be reduced to one $y_i(\bar{w}x - b) \geq 1$.

The optimization problem we want to solve is the following: Minimize $\|\bar{w}\|$ subject to constraint $y_i(\bar{w}x_i - b) \geq 1$ for $i = 1, \dots, N$, where N is the number of features. This problem can be modified so that the quadratic programming techniques can be used in the optimization process. The modified formula is

$\frac{1}{2}\|\bar{w}\|^2$, and minimization of it would also mean minimization of $\|\bar{w}\|$. The updated optimization problem can be expressed as

$$\min \frac{1}{2}\|\bar{w}\|^2 \text{ such that } y_i(\bar{w}x_i - b) \geq 1, i = 1, \dots, N. \quad (2.7)$$

Handling Noise

To introduce the ability of SVM to handle nonlinearly separable data (but not to the extreme), we define the hinge loss function: $\max(0, 1 - y_i(\bar{w}x_i - b))$. It is zero if the constraints 1 and 2 are satisfied. If it is not, the data point does not lie on the right side of the decision boundary. The function value is proportional to the distance from the decision boundary. The resulting cost function looks is expressed as

$$C\|\bar{w}\|^2 + \frac{1}{N} \sum_{j=1}^N \max(0, 1 - y_j(\bar{w}x_j - b)), \quad (2.8)$$

where C is the hyperparameter that determines the trade-off between increasing the size of the decision boundary and ensuring that each x_i lies on the correct side of the decision boundary. Its value is chosen experimentally. C handles the trade-off between classifying the training data well and classifying future examples well (generalization). For higher values of C , the misclassification error will be almost negligible, so the algorithm will try to find the highest margin without considering it. For lower values of C , the algorithm will try to make fewer mistakes by sacrificing the margin size. (A larger margin is better for the generalization.) Lower values lead to wider margins and more margin violations. Higher values lead to narrower margins and fewer margin violations.

SVM with the hinge loss function is called *soft-margin SVM*, while the original formulation that optimizes the Euclidean norm is referred to as *hard-margin SVM*. *Soft margin classification* tries to mitigate the downsides of *hard margin classification* by finding a balance between keeping the margin as large as possible and mitigating margin outliers (instances that lie on the margin or on the opposite side).

Handling Non-linearity

We can adapt SVM to work with nonlinearly separable datasets by applying the kernel trick. The kernel trick means transforming the original space to a higher dimensional one during the cost function optimization with the hope that, in higher dimensional space, it will become linearly separable. Mathematically, the kernel trick is mapping $\varphi : \bar{x} \rightarrow \varphi(\bar{x})$, where $\varphi(\bar{x})$ is a vector of higher dimensionality than \bar{x} . The kernel trick allows us to save a lot of non-necessary computations. Linear, polynomial, and radial basis function (RBF) are the most widely used kernel functions.

2.1.4 Decision Trees

A Decision Tree is a versatile and highly interpretable machine learning algorithm that can perform both classification and regression tasks. It is a critical component of *GradientBoostedSurvivalAnalysis*. Its extension, the Survival Tree, is also an essential component of the *Random Survival Forest*. Both ensemble methods are used in this work and are discussed in Sections 2.5.4.2 and 2.5.4.3, respectively. Scikit-learn's *DecisionTreeRegressor* is utilized in *GradientBoostedSurvivalAnalysis*, so we will focus on regression rather than classification. Scikit-learn employs the *Classification and Regression Tree (CART)* algorithm to train decision trees, so let us look into this algorithm in more detail.

In each node, the dataset is recursively split in two based on a feature k and a threshold t_k . Such pair (k, t_k) is chosen that produces the "purest" subsets. A node is pure if all training instances it applies to belong to the same class. The following impurity methods are common: Gini impurity and entropy for classification and MSE for regression.

Gini impurity is expressed as

$$G_i = 1 - \sum_{k=1}^n p_{i,k}^2,$$

where $p_{i,k}$ is the ratio of k class instances among the training instances in the i^{th} node. Entropy is a term originating from physics, where it is a measure of molecular disorder: it is zero if molecules are well-ordered and still. In machine learning, it is used as a measure of impurity and is denoted as

$$H_i = - \sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^n p_{i,k} \log_2(p_{i,k}).$$

Consequently, the entropy is zero if all instances are of the same class.

In regression, CART tries to split the dataset in each node to minimize mean squared error (MSE) (seen previously in Equation 2.2). The cost function for the decision tree regression is given by

$$J(k, t_k) = \frac{m_{\text{left}}}{m} MS E_{\text{left}} + \frac{m_{\text{right}}}{m} MS E_{\text{right}} \text{ where } \begin{cases} MS E_{\text{node}} = \sum_{i \in \text{node}} (\hat{y}_{\text{node}} - y^{(i)})^2 \\ \hat{y} = \frac{1}{m_{\text{node}}} \sum_{i \in \text{node}} y^{(i)} \end{cases}, \quad (2.9)$$

where m is the number of instances in the dataset, and *left* and *right* mean that the value relates to the left or right node of the tree. In classification, the loss function 2.9 would contain Gini (G) or entropy (H) instead of MSE.

It is worth noting that the other algorithms train the Decision Tree without restrictions (for example, the approach used in Survival Trees [71]). Then, during the *pruning* stage, all unnecessary nodes are removed. A node is considered redundant if its purity improvement is not statistically significant. Statistical tests such as χ^2 test are used to estimate the probability of the improvement being the result of a chance (the null hypothesis). If the probability (p-value) exceeds a certain threshold (often 5%), the node is deemed unnecessary, and its children are deleted.

Linear algorithms, like linear regression, assume that data are linear. In contrast, decision trees make no such assumption about the data. As a result, Decision Trees are classified as nonparametric algorithms because the number of parameters is not predefined before training. This flexibility allows the algorithm to fit the data closely, possibly leading to overfitting. Consequently, the algorithm needs to be regularized to prevent overfitting. The following hyperparameters can help regularize the model:

- *max_depth* - specifies the maximum depth of the decision tree.
- *min_samples_split* - sets the minimum number of samples a node must have before it can be split.
- *min_samples_leaf* - dictates the minimum number of samples a leaf node must have.
- *max_leaf_nodes* - determines the maximum number of leaf nodes.
- *max_features* - sets the maximum number of features evaluated for splitting at each node. Supports the following values: integer, float, square root of all features ("sqrt"), \log_2 of the number of features ("log2"), and None (all features)

The most critical parameter to limit is *max_depth*. Increasing *min_** and decreasing *max_** will help to regularize the model. There are other hyperparameters that are not related to regularization:

- *criterion* hyperparameter specifies the function to assess the quality of a split. In the scenario described above this is set to MSE.
- *splitter* defines the strategy for choose the split at each node. There are two available splitters: "best" to choose the best split and "random" to choose the random split.
- *random_state* defines the randomness of the estimator, as features are randomly permuted at each split.

For more information on the hyperparameters, refer to the scikit-learn documentation [74].

Once the decision tree is built and we need to make a prediction, the data instance is passed through the tree. It begins at the top node, known as the root node, and then it moves to the left or right node based on the instance's value for the selected feature of the node. In this way, the instance traverses through all the descendant nodes until it arrives at the terminal node, also known as the leaf node. The target value at that node will serve as the prediction, whether for regression or classification. In the case of regression, this value represents the average of all the target values that led to the leaf node.

Now, let us discuss several limitations. The first one is that the Decision Tree is a greedy algorithm, which means that finding an optimal model is nearly impossible since it's an NP-complete problem. NP-complete problems are NP (solutions can be verified in polynomial time) and NP-hard (any NP problem can be reduced to it in polynomial time). Additionally, decision trees are sensitive to small changes in data. This problem can be addressed by using PCA or opting for the ensemble models mentioned earlier, which are generally able to mitigate this instability.

2.1.5 Stochastic Gradient Boosting

Boosting is an approach to building an ensemble model, where weak learners are trained and sequentially added to the model, each trying to correct its predecessor. The most common boosting algorithms are AdaBoost and Gradient Boosting. Gradient Boosting is an ensemble machine learning method that uses Decision Trees as a base learner. Each subsequent learner is trained to fit the residuals of the previous model [9]. Residuals are typically the difference between the predicted values and the actual values. In this context, however, they are called pseudo-residuals, as they represent the negative gradients of the loss function. This section will look into the modification of Gradient Boosting, called Stochastic Gradient Boosting, presented in Friedman's work [69]. The only difference from the original Gradient Boosting is that it draws a subset of random instances from the training set to compute the negative gradient (step 2) and train the base learner (step 3) instead of using the whole training set. We have chosen it over the original to shorten the training time.

Let $\{(\mathbf{x}_i, y_i)\}_{i=0}^n$ be the training set, $L(y_i, F(\mathbf{x}))$ be a differentiable loss function, and $\{\pi(i)\}_1^N$ be a random permutation of integers $\{1, \dots, N\}$. Then a subset of random instances of size $\tilde{N} < N$ is given by $\{(\mathbf{x}_{\pi(i)}, y_{\pi(i)})\}_{i=0}^{\tilde{N}}$ and the Stochastic Gradient Boosting is denoted as follows:

Initialize the model $F_0(\mathbf{x}) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^N L(y_i, \gamma)$, where γ is the predicted value that minimizes the total error.

For each tree $m = 1, \dots, M$ (where M is the number of trees specified by the hyperparameter *n_estimators*) perform the following steps:

1. $\{\pi(i)\}_1^N = \operatorname{rand_perm}\{i\}_1^N$

2. Compute the negative gradient (pseudo-residual)

$$y_{\pi(i),m} = \left[\frac{\partial L(y_{\pi(i)}, F(\mathbf{x}_{\pi(i)}))}{\partial F(\mathbf{x}_{\pi(i)})} \right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, i = 1, \dots, \tilde{N}.$$

3. Fit a regression tree to the residuals $\tilde{y}_{\pi(i),m}$ and create terminal regions (leaves) $R_{l,m}$ for $l = 1, \dots, L$, where l is each leaf in a tree m) [70].
4. For $l = 1, \dots, L_m$ compute $\gamma_{l,m} = \underset{\gamma}{\operatorname{argmin}} \sum_{\mathbf{x}_{\pi(i)} \in R_{l,m}} L(y_{\pi(i)}, F_{m-1}(\mathbf{x}_{\pi(i)}) + \gamma)$ – determine the output value γ for each leaf l that minimizes the error.
5. Update $F_m(x) = F_{m-1}(\mathbf{x}) + \nu \cdot \gamma_{l,m} I(\mathbf{x} \in R_{l,m})$. Where $\nu \in [0, 1]$ is a learning rate (or shrinkage parameter) which regulates how much influence each trained tree has on the final result. Generally, lower values are recommended, necessitating the increase of the number of estimators. Such combination generally leads to better generalization. $I()$ is an indicator function and it is equal to one, only if \mathbf{x} is in this region (leaf) $R_{l,m}$, and zero otherwise.

$\tilde{N} = N$ will produce no randomness, effectively resulting in the original Gradient Boosting algorithm. A smaller fraction, $\frac{\tilde{N}}{N}$, introduces a trade-off: it increases randomness and decreases training time but also leads to higher variance in the learner. So, this value must be chosen wisely.

Friedman suggested various algorithms based on this template with different loss functions for regression and classification. Ridgeway [70] extended this algorithm to survival analysis, a branch of statistics that studies time until event (Section 2.5 explains it in more detail), by using the Cox Proportional Hazards loss. More on that is Section 2.5.4.2, which is dedicated to the algorithm used in the practical part.

2.2 Unsupervised Learning

Unsupervised learning deals with a dataset without labels [10]. It has three main techniques: clustering, dimensionality reduction, and anomaly detection [9].

Clustering is a method that identifies similar instances and groups them into sets. It has applications in data analysis, namely, *exploratory data analysis (EDA)*, customer segmentation, dimensionality reduction, and anomaly detection [9]. Clustering might be either soft, where an instance has a score of belonging to a particular cluster, or hard, where an instance belongs to only one class. The score might be the distance from the cluster centroid or an affinity (similarity score) [9].

Dimensionality reduction is useful for visualization and learning acceleration [9]. Datasets often have a lot of redundant data, or the task requires a lot of features. Many algorithms, such as linear models, SVMs, and decision trees, might have their performances compromised due to high-dimensional data. So-called *curse of dimensionality* states that high dimensional data can cause slow learning and prevent us from getting an optimal model [9]. Consequently, the reduction of the data dimensionality might be a good idea. However, it is worth noting that a dimensionality reduction algorithm might lose some useful information. Many modern algorithms, such as neural networks or ensemble algorithms, handle high dimensional data very well, and dimensionality reduction techniques are used less than in the past [10]. However, they are still used for data visualization and cases when we need to build an interpretable model while we are limited in the number of algorithms we can use [10].

Anomaly (outlier) detection involves the detection of instances strongly deviating from the norm [9]. These instances are called *outliers* or anomalies, while regular ones are referred to as *inliers* [9]. Anomaly detection has many applications. For example, it can be used as a data preprocessing step

to remove outliers from the dataset, which might improve the performance of the resulting model [9]. In addition, it is used in the *fraud detection* task and the detection of faulty products in manufacturing facilities [9].

Novelty detection is closely related to anomaly detection. The only difference is that novelty detection assumes that outliers did not contaminate the training dataset, while anomaly detection does not make this assumption [9].

2.3 Data Preparation

Due to factors such as the curse of dimensionality and inherent noise, we cannot load raw data to an algorithm and expect good performance [10]. Most often, the raw data has too many features, and most of them have very little predictive power. We need to build a dataset first. *Feature engineering* is responsible for transforming raw data into a dataset [10]. It is a labor-demanding process that requires creativity and, most importantly, domain knowledge [10].

The objective of this stage is to create *informative* features or features with *high predictive power* [10]. For example, in our task of predicting survival time, donor-recipient blood group compatibility or recipient's age are likely to have much higher predictive power than the donor's or recipient's citizenship. Blood group compatibility is directly linked to the successful transplantation, while the recipient's citizenship has little correlation with the genetic similarities, which we would expect from datasets from other countries, but we use the dataset from the United States - a country with very diverse population.

Moreover, it is possible to create new features with higher predictive power out of those with low predictive power [9]. For example, the calculation of *estimated Glomerular Filtration Rate (eGFR)*, the metric of kidney function estimated on a patient's age, gender, and serum creatinine level, could potentially give more information to the learning algorithm than all features separately.

In the following subsections, we will cover some popular feature engineering techniques.

2.3.1 Handling Categorical Features

The majority of machine learning algorithms primarily operate with numerical features [9]. To handle categorical features (the ones with only a few possible values), such as the age group or a blood group, we can use *one-hot encoding* to convert them to several binary ones [10]. For instance, let us consider a blood group feature comprised of four primary blood groups: A, B, AB, and O. We can convert each blood group into a vector of four numerical values:

$$A = [1, 0, 0, 0]$$

$$B = [0, 1, 0, 0]$$

$$AB = [0, 0, 1, 0]$$

$$O = [0, 0, 0, 1]$$

This technique will increase the dimensionality of the dataset, but this is a trade-off we have to make because assigning a number to each group (1 to A, 2 to B, and so on) would imply gradation or ranking among these categories, while there is none [10].

However, if the categorical feature does suggest some gradation, for example, university marks as "fail", "average", "good", or "excellent", an enumeration of each value will be appropriate. This practice of assigning a number to categories that have ranking is called *ordinal encoding* [9, 10].

Binning (or *bucketing*) is the technique used for converting numerical values into multiple binary features called *bins* or *buckets* [10]. For example, a patient's age can be transformed into age-range bins: 0 to 18, 18 to 25, 25 to 40, and so on. This technique might help the algorithm to learn better, particularly with smaller datasets [10].

2.3.2 Feature Scaling

Different ranges of feature values might pose a problem to some machine learning algorithms as they do not handle them very well [9, 10]. It might result in a slower training time or a poorer performance [10]. *Normalization* and *standardization* scaling techniques solve this problem.

Normalization (also known as *min-max scaling*) is a technique of converting an actual range of numerical feature values into a standard range of values: $[-1, 1]$ or $[0, 1]$ without losing any information [10]. The $[0, 1]$ normalization formula for value $x^{(j)}$ for feature j is expressed as

$$\bar{x}^{(j)} = \frac{x^{(j)} - \min(j)}{\max(j) - \min(j)}, \quad (2.10)$$

where $\min(j)$ and $\max(j)$ are minimal and maximal values of feature j .

Standardization is a scaling technique that scales numerical data so that after scaling, it has the *standard normal distribution* properties with the mean $\mu=0$ (average value) and the standard deviation from the mean $\sigma = 1$ [10]. The standardization formula for value $x^{(j)}$ for feature j is given by

$$\hat{x}^{(j)} = \frac{x^{(j)} - \mu^{(j)}}{\sigma^{(j)}}.$$

Typically, standardization is used for supervised learning in cases where feature values are formed by a standard distribution (bell curve) or a feature has outliers. In other cases, normalization is preferred [10].

2.3.3 Handling Missing Feature Values

Datasets frequently have missing values and, according to Andrii Burkov [10], to handle them, we have one of the following options:

1. **Removal of rows with missing values** is the most straightforward approach to managing missing data. If missing values are sparse or the dataset is large enough, this technique would be appropriate.
2. **Feature removal.** If a feature in the dataset has excessive missing values relative to its size, it is better to remove it.
3. **Regression imputation.** This technique implies filling in a missing feature value with regression algorithm predictions.
4. **Mean/median imputation.** This method involves filling missing feature values with their mean or median value.
5. **Constant value imputation.** This technique entails filling the missing values with clearly too-high or too-low values. Because of that, the algorithm will be able to discern the value as an outlier while considering other features. This method is not recommended as it can introduce bias.

6. **Adding a binary indicator for each feature with missing values.** If features with missing values are few and the dataset is large, we can add a binary indicator for each feature with missing values. The missing value can be replaced with 0 or any other number.

It is often impossible to tell which data imputation method would work best, so it should be checked experimentally.

2.4 Model Training and Hyperparameter Tuning

It is a common practice to divide a dataset into three parts:

- Training set (70% of the dataset)
- Validation set (15% of the dataset)
- Test set (15% of the dataset) [10].

The training set, being the largest, is employed to train the machine learning model. Validation and test sets, which are identical in size and often called hold-out sets, are used in subsequent stages of model evaluation [10].

The rationale behind using separate training and validation sets is to prevent *overfitting* - when the model performs well on the training data but poorly on the unseen data [10]. Overfitting can occur if the model is tested and evaluated on the same dataset. As a result, the model may memorize the training examples and fail to make accurate predictions on the unseen data. To alleviate this, we use the validation set to fine-tune the model and the test set to assess its performance before deploying it to production[10].

A typical workflow involves training the model on the training set, validating it on the validation set using the selected metric, and then adjusting the model's parameters to improve its performance. This process is repeated until no substantial improvement is observed. Finally, the model's performance is assessed on the test set [10]. This iterative process is referred to as *hyperparameter tuning*.

An alternative to the three-set technique is *k-fold cross-validation*. This technique involves splitting the dataset into k subsets, or folds, of equal size. One fold is used as a validation set, while the other $k-1$ folds constitute a training set. The model is trained exactly k times, with each fold serving as a validation set only once. The only drawback is that it is highly computationally demanding, particularly with a high k value and larger datasets, as the model will be trained k times, but with a small dataset, it is a necessity [10].

A *hyperparameter* is a parameter specified before model training, unlike regular parameters calculated during training. Each model possesses a different set of hyperparameters, and they profoundly influence the model's performance. The number of trees in Random Forest (described in Subsection 2.5.4.3) and the C hyperparameter in Support Vector Machines (described in Subsection 2.1.3) are examples of hyperparameters. The task of finding the optimal combination of hyperparameters is called *hyperparameter tuning*. One strategy might be to select hyperparameters manually and observe their impact on performance. However, utilizing the grid search is a better way [9].

Grid search is a standard way of performing hyperparameter fine-tuning. It includes defining hyperparameters to experiment with, providing values for each to be tested, and training a model for each possible combination of hyperparameters. The performance of each model is assessed using k -fold cross-validation, and the best combination of hyperparameters is selected. This approach is used in scikit-learn's implementation - GridSearchCV [9].

Grid search proves effective when dealing with relatively few hyperparameter combinations. However, with a larger number of hyperparameter combinations, it is advisable to use RandomizedSearch

(RandomizedSearchCV in scikit-learn [9]). This method is very similar to grid search, but instead of trying every possible combination of provided values, it tests only a specified number of randomly selected hyperparameter combinations. The primary advantage of this method over grid search lies in more control over computational power and the time dedicated to hyperparameter tuning [9].

2.5 Survival Analysis

Survival analysis (SA), often referred to as *time-to-event analysis*, is a set of statistical techniques employed to analyze and predict the time until an event of interest occurs. Its name originates from clinical and biological research, where these methods are used to analyze survival time. These methods, however, found their use in areas far beyond clinical settings: in business to predict the time until the customer “churns” from a subscription; in engineering to estimate the product longevity or the longevity of its parts; in social sciences to estimate the longevity of a marriage; or to estimate a student dropout rate in an academic setting.

In the context of survival analysis, *time* (also *survival time*) refers to the duration from the start of an individual’s follow-up to the occurrence of an event. It is usually measured in days, weeks, months, or years [13]. The term *event* (also referred to as *death* or *failure*) encompasses any occurrence that permanently changes the subject’s state. It can be death, the onset of a disease, a relapse from remission, a recovery, or any other specified experience of interest that an individual might encounter [13, 30]. Usually, only one event of interest is considered. When evaluating multiple events, the problem is categorized as *competing risks* or *recurrent events* problem [13].

This section will explain the fundamental survival analysis terminology, such as censoring and survival function. We will explore the taxonomy of survival analysis methods, highlighting the most popular statistical methods and one machine learning method tailored to the needs of survival analysis. Additionally, we will discuss ways to evaluate survival models effectively. Unless referenced otherwise, the contents of this section are drawn from [13].

2.5.1 Essential Concepts

In this subsection, we will cover the essential terminology required for survival analysis such as censoring, censoring assumptions, survival, and hazard functions.

Censoring

The most distinct feature of survival analysis methods is the ability to handle censored data. *Censoring* refers to a circumstance when the information about survival time is only partially known. For example, the dataset utilized in our research has 370,000 censored instances out of 500,000 performed transplantations. These patients were either alive at the last observation date or lost to follow-up. This lack of complete information indicates *censoring* in survival analysis. Censoring makes the application of standard statistical and machine-learning approaches to survival data impractical [27].

Look at the Figure 2.5. On the y-axis, we can see individual patients, while the x-axis corresponds to the study timeline (the right side is the end of the study). Cross (X) denotes an occurrence of the event, and circle (O) corresponds to the patient’s exit from the study. No sign and the line ceasing at the end of the study means the patient did not experience the event.

There are three types of censoring: *left*, *right*, and *interval* censoring. *Right censoring*, which is more common, occurs when we are sure that the event did not happen by a specific time and we do not know

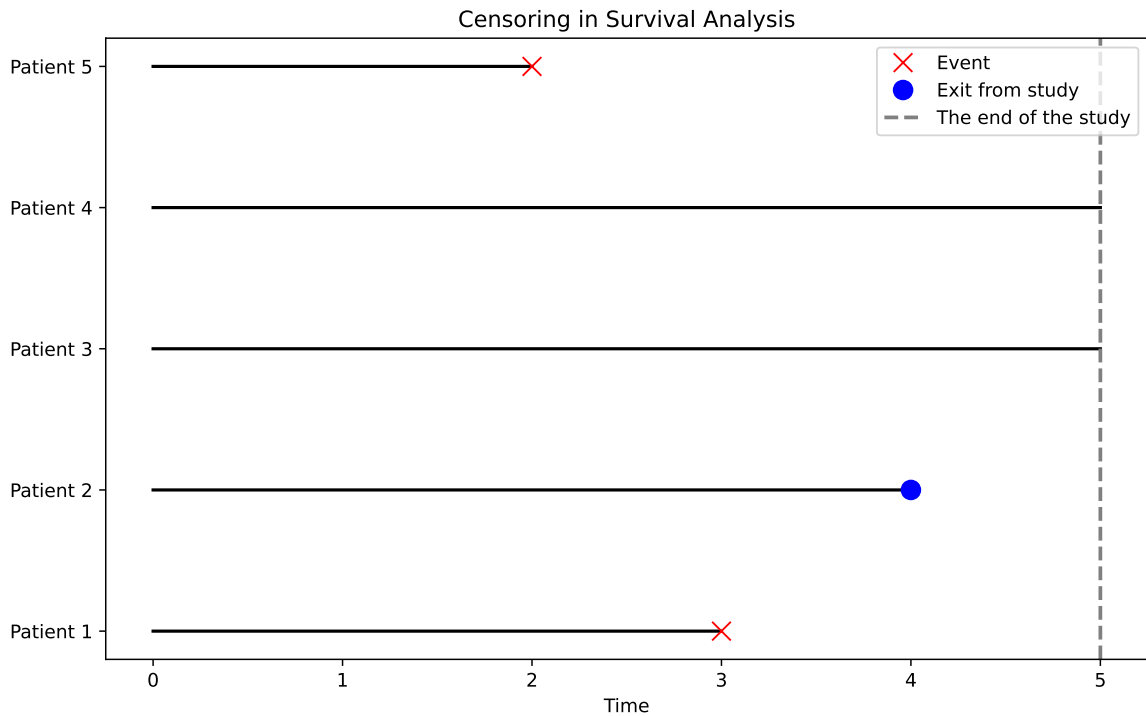


Figure 2.5: The Illustration of Censoring

when it will happen. The situation arises when the patient drops out of a study, or the study ends when they are still alive, as illustrated with patients 2, 3, and 4 in Figure 2.5.

Left censoring is less common and happens when the event occurs before the study begins or before the initial observation. We know the event happened before a specific time, but the exact time is unknown. This type is typical in cases where a patient has already experienced the event (e.g., developed a disease) before enrolling in the study.

Interval censoring happens when the event occurs within a particular timeframe, but the exact time is unknown. This can be the case in studies involving periodic patient follow-up, where the event can happen at any point between two visits.

Understanding right, left, and interval censoring is essential in survival analysis. We will next turn our attention to the assumptions associated with these censoring types. These assumptions are inherent to many survival analysis methods and are critical in selecting an appropriate technique.

Censoring Assumptions

There are three types of censoring assumptions: *random*, *independent*, and *non-informative*. Each shares certain similarities but also possesses unique distinctions, which we will explore in detail. Censoring assumptions are how censoring is managed.

1. **Random Censoring:** Subjects censored at time t are assumed to have the same failure rate as the remaining subjects, provided they have the same survival experience. The censored subjects are selected randomly, meaning the study does not influence or bias which participants are censored.

2. **Independent Censoring:** Independent censoring occurs when the censoring is random within certain subgroups defined by specific covariates. If no covariates are present, it defaults to random censoring. This distinction might not be apparent when examining a single subgroup.
3. **Non-Informative Censoring:** Non-informative censoring occurs when censored instances do not provide any information on the patient's survival prospects. In other words, whether or not the patient is censored, has no influence on experiencing the event.

Generally, it is safe to assume *non-informative* censoring when censoring is *independent* and/or *random*. However, these assumptions are not equivalent. Let us examine some examples to better understand these concepts.

Consider a three-year disease occurrence study with 100 subjects at risk (group A). By the end of the study, 20 of them contracted the disease, giving a 20% three-year disease risk. Suppose we want to extend the study for another two years on the remaining 80 individuals. However, 40 refused to continue in the study and were, therefore, lost to follow-up (censored). Of the remaining 40, five contracted the disease. Assuming those who left were representative of the remaining subjects (random and independent censoring), another five among the censored would have contracted it. Consequently, the five-year risk is 30%, and the five-year survival is 70% under random and independent censoring assumptions. In this case, random and independent censoring is the same, as no predictor variables are considered.

To illustrate the difference between random and independent censoring, let us introduce another group to the study: group B with 100 individuals. In the first three years, 40 contracted the disease, and 10 left the study. So, the calculated three-year risk for group B is 40%. In the next two years, 10 out of 50 get the disease, yielding 20% risk for years between 3 and 5. Under the independent censoring assumption, we assume that out of 10 censored, 2 contracted the disease. The five-year risk for group B is 52%, with 48% survival under independent censoring assumptions.

As we can see, the five-year risk in the two groups differs significantly (30% against 52%), and the censoring proportion is also very different (50% against 17%). Hence, the overall censoring is not random. However, it is random within each group, so the censoring is independent. Conversely, if in group B, 30 subjects out of 60 were censored at the three-year mark, the censoring proportion would be the same in both groups, and the overall censoring would be random as those censored would be the representatives of those who remained at risk.

To best illustrate non-informative censoring, let us demonstrate informative censoring. Let us take a group of subjects under random and independent censoring assumptions. Every time subject A gets an event, subject B leaves the study (e.g., B is A's relative). If the censored subjects are representative of subjects at risk, it would be random and independent censoring. Here, the censoring mechanism is directly related to the event occurrence, so the censoring is informative.

Survival Function

The *survival function*, also known as the *survivor function*, denoted by $S(t)$, represents the probability that the patient survives, in other words, does not experience the event of interest beyond a given time t . Mathematically, it is denoted by

$$S(t) = P(T > t), \quad (2.11)$$

where P represents the probability, t is any specific time of interest, and T is the random variable for the subject's survival time. For instance, if we want to know the likelihood that a patient will live for more than five years after a kidney transplant, we set t equal to 5 and evaluate $S(t)$ to determine the probability that T , actual survival time, is greater than 5 years.

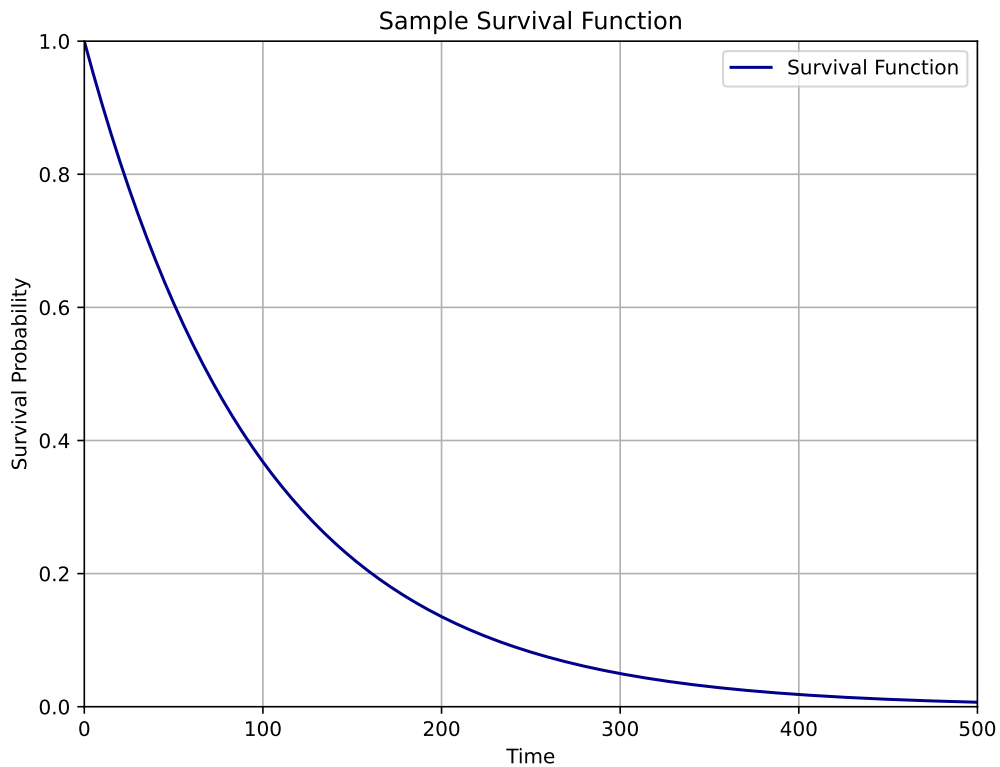


Figure 2.6: Survival Function

The survival function has several key characteristics:

1. It continually decreases or maintains its value over time, theoretically extending from 0 to infinity. If the study lasted indefinitely, the survival function would eventually fall to 0. In practical research scenarios, studies do not last forever, and not every patient experiences an event by the end of the study.
2. As it represents a probability, the function value ranges from 0 to 1. At the beginning of the observation period, it starts at 1, indicating 100% survival probability, and declines over time, potentially reaching 0 as the probability of survival decreases.
3. Although the survival function graph is smooth by definition, in reality, it is a step function. This stepwise representation is due to the nature of real-world data, where events are recorded at specific, discrete time points rather than continuously [13].

Hazard Function

The survival function provides the probability of an individual surviving at each given point in time. This function is often preferred over the *hazard function* due to its more intuitive appeal: it directly communicates the chance of survival. However, there are scenarios where knowing the risk at each point in time is necessary, entailing the use of the hazard function.

The *hazard function*, denoted as $h(t)$, represents the instantaneous potential per unit of time for the event to occur, provided the subject's survival up to time t . It is given by

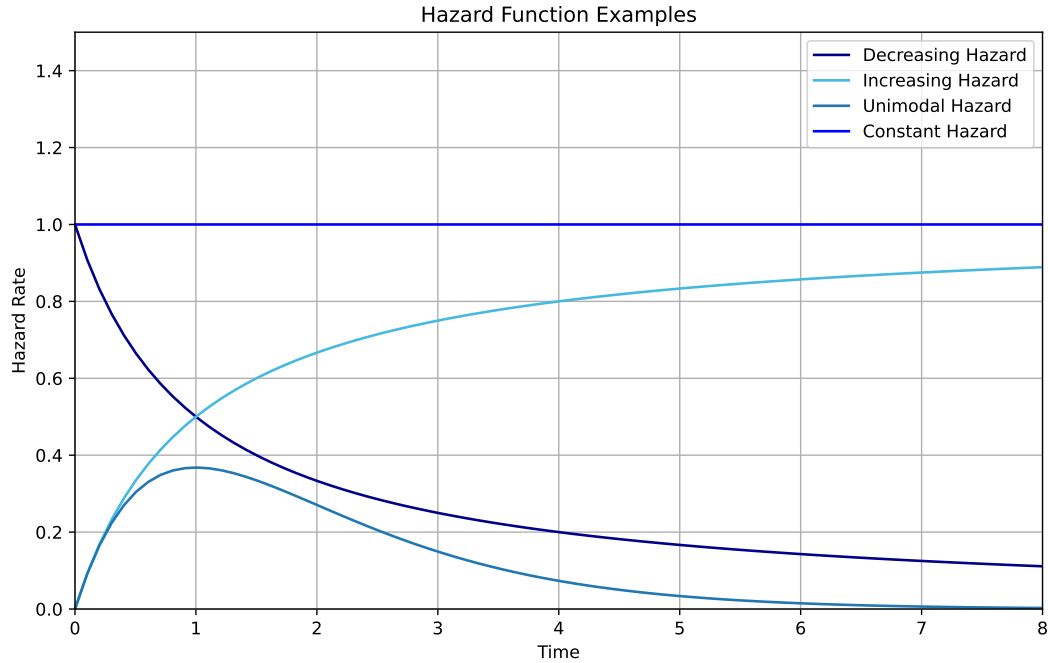


Figure 2.7: Examples of Different Hazard Function Types

$$h(t) = \lim_{\Delta t \rightarrow 0} \frac{P(t \leq T < t + \Delta t | T \geq t)}{\Delta t}. \quad (2.12)$$

Here, Δt represents an infinitesimally small increment of time. The function $h(t)$ is equal to the limit, as Δt approaches zero, of a conditional probability, divided by Δt . The conditional probability statement gives the probability that a person's survival time T will lie in the interval between t and $t + \Delta t$, given that the survival time is greater than or equal to t .

Occasionally, due to its structure, the hazard function is referred to as a *conditional failure rate*. It is a rate because it represents a conditional probability per unit of time Δt , and it is conditional on the subject surviving until time t . Unlike probability, this rate has a scale from 0 to infinity — depending on the measure of time in days, weeks, or years. By considering the limit as the Δt approaches zero, we get the instantaneous potential of failing at time t , given survival until that moment.

The *cumulative hazard function* further extends our understanding by quantifying the accumulated risk over time. It is the area under the hazard function that allows us to say which group has a greater risk. It is expressed as

$$H(t) \stackrel{def}{=} \int_0^t h(u) du, \text{ where } t > 0. \quad (2.13)$$

Where $h(u)$ represents a hazard function. This integral measure enables a comprehensive comparison of risk between groups, showing which has a greater risk from the perspective of accumulated potential for the event to occur over time.

The Relationship Between the Survival and Hazard Functions

Some models, such as Cox Proportional Hazards, are written in terms of the hazard function, and it is necessary to be able to estimate the survival function out of the hazard function to make the model more

flexible and accessible to a larger audience. Fortunately, there are ways to convert one into the other. In this subsection, we are going to describe techniques for these conversions.

Let us start by considering how to obtain the survival function $S(t)$ from the hazard function $h(t)$. The relationship is given by

$$S(t) = \exp\left[-\int_0^t h(u)du\right]. \quad (2.14)$$

Equation 2.14 illustrates that the survival function $S(t)$ is equal to the exponential of the negative cumulative hazard function from zero to t . As we can see, the integral in the equation is the $H(t)$ from the 2.13. We can simplify our equation to

$$S(t) = e^{-H(t)}. \quad (2.15)$$

On the other hand, to obtain the hazard function from the survival function, we can utilize the relationship

$$h(t) = -\frac{dS(t)/dt}{S(t)}. \quad (2.16)$$

Here, Equation 2.16 denotes that the hazard function $h(t)$ is the negative derivative of the survival function $S(t)$ with respect to time t divided by $S(t)$.

Considering the fact that the survival function describes the probability of a patient surviving up to time t , and the hazard function shows the instantaneous risk of a person dying at a specific instant t , we can say that they provide complementary information about survival and risk over time [13]. Of the two discussed functions, the survival function is used more often as it is more intuitive. In the practical part, we will estimate the survival function as well. Fortunately, we will not convert them manually. The *scikit-survival* library will do that for us.

2.5.2 Taxonomy of Survival Analysis Methods

Survival analysis methods can be broadly categorized into *statistical* and *machine learning-based* methods [23]. Both aim to estimate the survival time and the survival probability at that time [23]. Statistical methods primarily characterize the distributions of the event times and the statistical properties of the parameter estimation by estimating the survival curves. Typically, statistical methods are employed for low-dimensional data [23].

On the other hand, machine learning methods focus on predicting the occurrence of events at a given time. They harness the strengths of traditional survival analysis while integrating different machine learning techniques, leading to more potent algorithms. Machine learning methods are mostly used with high-dimensional data [23].

Statistical methods can be further subdivided based on their assumptions and parameter usage into parametric, non-parametric, and semi-parametric methods. Machine learning methods, encompassing methods like survival trees, ensembles (random survival forests), neural networks, and support vector machines, form a distinct category. Advanced machine learning techniques, such as active learning, transfer learning, and multitask learning, are also included [23].

In the following sections, we will cover selected statistical methods and one machine learning technique.

2.5.3 Statistical Methods

This section provides a concise overview of statistical techniques. Here, we introduce three different types of statistical methods that are commonly used to estimate the survival and hazard functions: *non-parametric*, *semi-parametric*, and *parametric methods*.

Non-parametric methods are preferred in situations where the event time does not adhere to any known distribution or when the proportional hazards assumption is not met [23]. There are three main non-parametric methods: the Kaplan-Meier (KM) method, the Nelson-Aalen (NA) estimator, and the Life-Table (LT) method. In the next section, we will cover Kaplan-Meier in more detail. The Life-Table method is more convenient than Kaplan-Meier for the estimation of survival curves when data subjects are segmented into distinct time intervals when dealing with an extensive number of subjects or a broad population scope. On the other hand, the Nelson-Aalen method is used to estimate hazard functions [23].

Semi-parametric models offer a middle ground between fully parametric models, which make specific distributional assumptions, and non-parametric models, which make very few assumptions. The Cox model is the most frequently employed model for survival regression analysis among semi-parametric methods. It is semi-parametric, as the distribution of the outcome is unknown. Unlike other approaches, this method is based on the proportional hazards assumption and uses partial likelihood for parameter estimation (more on that in 2.5.3.4). There are a couple of variants of the basic Cox model: the penalized Cox model, which will be used in the practical part of this work, the CoxBoost algorithm, the Time-Dependent Cox model and many more [23].

Parametric methods shine in their accuracy and efficiency when the time to the event conforms to a known distribution that can be specified in terms of certain parameters. With parametric models, estimating the time to the event is straightforward, whereas the Cox model can make this task somewhat cumbersome or unfeasible. In the domain of parametric models, linear regression is central. However, the Tobit model, Buckley-James regression, and penalized regression are the most favored. Beyond this, other parametric models, like the Accelerated Failure Time (AFT), have gained traction. The AFT model represents survival time as a function of covariates [23].

To conclude, statistical methods in survival analysis can be broadly categorized into non-parametric, semi-parametric, and parametric techniques, each with its unique strengths and applications. Among non-parametric methods, Kaplan-Meier stands out as particularly significant. It provides a robust and intuitive way to estimate survival functions.

2.5.3.1 Kaplan-Meier Survival Curves

Kaplan-Meier is a non-parametric method of creating survival functions.. It is *non-parametric* because it does not consider any covariates or parameters and requires only the survival time and the censoring indicator. It works under an independent censoring assumption. The general Kaplan-Meier formula for plotting the survival function is given by

$$\hat{S}(t_{(f)}) = \hat{S}(t_{(f-1)}) \times \hat{P}(T > t_{(f)} | T \geq f_{(f)}). \quad (2.17)$$

That can be read as the survival function \hat{S} of time $t_{(f)}$ is equal to the probability of surviving past the previous time point $t_{(f-1)}$ times the conditional probability of surviving past the time $t_{(f)}$ [13]. This function can also be expressed as a product limit

$$\hat{S}(t_{(f)}) = \prod_{i=1}^{f-1} \hat{P}(T > t_{(i)} | T \geq f_{(i)}).$$

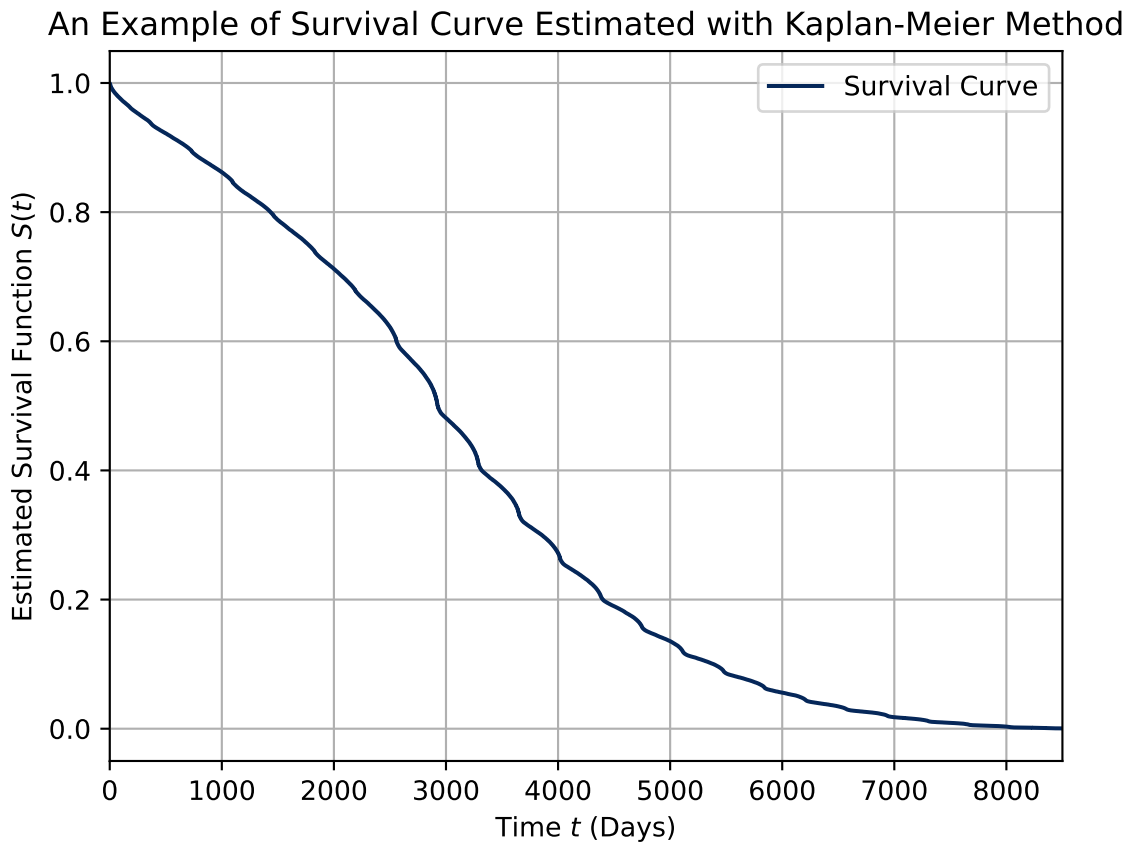


Figure 2.8: Survival Curve Estimated with KM on the Training Dataset

In the Equation 2.17, we replaced $\hat{S}(t_{f-1})$ with the product of all fractions estimating the conditional probabilities for failure times t_{f-1} and those preceding it [13]. Because of that, the Kaplan-Meier estimator is sometimes referred to as the *product-limit method* [23]. Figure 2.8 shows the survival curve created with the KM method on the dataset used in this work.

These survival curves are often compared using the log-rank test. The *log-rank test* is a way to compare two survival functions. It is often used in studies, with a target group and a placebo (control) group to assess the efficacy of the treatment or intervention in the study by comparing the survival curves of the two groups [13].

2.5.3.2 Log-rank Test

Log-rank Test is essentially a χ^2 test for two KM survival curves that can evaluate whether or not survival (KM) curves for two or more groups are statistically equivalent (χ^2 test was already touched upon in Section 2.1.4). Two curves are statistically equivalent if the produced p-value is lower than the level of significance (0.05 or 0.01). It is formed by the sum of the observed minus expected over all failure times for one of the two groups [13].

The Null hypothesis H_0 is being tested in the log-rank test, which states that groups have identical distribution curves. If the p-value from the χ^2 distribution table associated with the produced log-rank statistic is lower than the level of significance, then the Null hypothesis is rejected, and the curves are

deemed statistically inequivalent. Under the Null hypothesis, the log-rank statistic is approximately χ^2 with $n - 1$ degrees of freedom for n groups [13].

The log-rank statistic for two groups, $i = 1, 2$, is denoted as

$$\text{Log-rank statistic} = \frac{(O_i - E_i)^2}{\text{Var}(O_i - E_i)},$$

where $\text{Var}()$ is variance. The formula for more than two groups is much more complex and involves the calculation of covariates. Fortunately, the log-rank can be approximated without the need for the calculation of variance and covariance as follows

$$\chi^2 \approx \sum_i^{\text{\#of groups}} \left(\frac{O_i - E_i}{E_i} \right)^2,$$

where i is the group, O_i are summed observed, and E_i are summed expected [13].

2.5.3.3 Nelson-Aalen

The Nelson-Aalen estimator is a nonparametric estimator for cumulative hazard function. It is denoted by

$$\hat{A}(t) = \sum_{t_j \leq t} \frac{d_j}{r_j}, \quad (2.18)$$

where d_j represents the number of patients who experienced the event, and r_j is the number of patients at risk at the time t_j [75].

2.5.3.4 Cox Proportional Hazards Method

The *Cox proportional hazards* (also Cox PH) model is a widely used in survival analysis semi-parametric model. This section explores its formulation, key properties, and reasons for its widespread use in research.

The Cox proportional hazards model is defined in terms of a hazard at time t for a subject with a given vector of explanatory variables \mathbf{X} and is expressed as

$$h(t, \mathbf{X}) = h_0(t) \exp \left[\sum_{i=1}^p \beta_i X_i \right], \quad (2.19)$$

where $h_0(t)$ stands for the *baseline hazard function*. Coefficients β are the parameters of interest in the model. Since the exponential function has no t , \mathbf{X} is called *time-independent*. The exponential function ensures that the function is non-negative, satisfying the definition of the hazard function [13].

Even though Equation 2.19 contains the baseline hazard function, the function is not specified. Fortunately, we can calculate the hazard ratio, a measure of effect, without having to estimate the baseline hazard function. Similarly, the hazard function $h(t, \mathbf{X})$ and the survival function $S(t, \mathbf{X})$ can be estimated without the baseline function. So, with minimal assumptions, we can estimate everything we need (h, S, and HR).

The *hazard ratio* (HR) is a measure of the influence of an intervention on the outcome. A hazard ratio is defined as the hazard for one individual divided by the hazard for the other, as illustrated with the following

$$HR = \frac{h(t, X^*)}{h(t, X)} = \exp \left[\sum_{i=1}^p \hat{\beta}_i (X^* - X) \right]. \quad (2.20)$$

As can be seen, the equation does not contain t and the basic hazard function, as they are canceled out, making it a *proportional hazard assumption*.

Like logistic regression, the CoxPH uses the *maximum likelihood* function 2.5 to calculate its parameters ($\hat{\beta}_i$). However, since the maximal likelihood considers only a part of patients, namely those who experienced an event, the formula is called *partial likelihood* [13].

Let us define the partial likelihood. The subject's survival can be defined by $h(t, X)dt$ with $dt \rightarrow 0$. Consider J (where $J \leq N$) as the total number of events of interest observed for N instances. $T_1 < T_2 < \dots < T_J$ represents the unique and sequentially ordered times to the event of interest. Let X_j be the vector of covariates for a subject who experiences the event at time T_j . R_j is the set of risk subjects at T_j . Given that the event happens at time T_j , the individual probability associated with X_j can be expressed as

$$\frac{h(T_j, X_j)dt}{\sum_{i \in R_j} h(T_j, X_i)dt}.$$

By taking the product across all subjects' probabilities, we get partial likelihood. Based on Cox's assumption and the presence of censoring, partial likelihood is defined as follows:

$$PL(\beta) = \prod_{j=1}^N \left[\frac{\exp(X_j \beta)}{\sum_{i \in R_j} \exp(X_i \beta)} \right]^{\delta_j}. \quad (2.21)$$

δ_j is the indicator of censoring of a given instance (1 for event occurrence, 0 for censoring). Therefore, if the subject is censored, $\delta = 0$, the individual probability is equal to 1, and the subject does not affect the result. The vector of coefficients $\hat{\beta}$ is estimated by either maximizing partial likelihood, defined above, or maximizing the negative *log-partial likelihood* to improve the efficiency:

$$LL(\beta) = - \sum_{j=1}^N \delta_j \left\{ X_j \beta - \log \left[\sum_{i \in R_j} \exp(X_i \beta) \right] \right\} [23]. \quad (2.22)$$

2.5.3.5 Penalized Cox Models

According to [25], the Cox proportional hazards model is often chosen since its coefficients can be interpreted in terms of hazard ratios, offering meaningful insights. However, when estimating coefficients for many features, the standard Cox model collapses due to matrix inversion getting disrupted by correlations between features. Feature (column) correlations make a matrix singular, i.e., impossible to revert. In this section, we aim to explore the *Ridge regression*, *LASSO*, and *Elastic Net* methods as extensions or modifications to the Cox model. These techniques address the inherent challenges in the standard Cox model, offering solutions for regularization and feature selection.

Ridge As [25] further explains, we can avoid the problem of the inability to revert singular matrix by incorporating an l_2 penalty term on the coefficients, shrinking them to zero. Consequently, our objective is expressed as

$$\arg \max_{\beta} \log PL(\beta) - \frac{\alpha}{2} \sum_{j=1}^p \beta_j^2.$$

In the equation, $PL(\beta)$ denotes the partial likelihood of the Cox model (2.21), terms β_1, \dots, β_p represent the coefficients corresponding to p features, $\alpha \geq 0$ is a hyper-parameter that controls the amount shrinkage. The resulting objective is referred to as *ridge regression*. If α is set to zero, we get the regular Cox model [25].

LASSO While the l_2 ridge penalty solves the mathematical problem of fitting the Cox model, we would still need to take into account all features, no matter how many there are. Preferably, we would like to select a small subset of the most predictive features and ignore the rest, as too many features might result in overfitting. *LASSO* (Least Absolute Shrinkage and Selection Operator) does exactly that. Rather than merely shrinking the coefficients to zero, it performs feature selection as a part of the optimization process, where a subset of coefficients is set to zero and is, therefore, excluded, reducing the number of features we need for prediction. Mathematically, we would replace the l_2 penalty with the l_1 penalty, leading to the following optimization problem

$$\arg \max_{\beta} \log PL(\beta) - \alpha \sum_{j=1}^p |\beta_j|.$$

The main drawback is that we cannot directly control the number of features selected. However, the value of α essentially determines the number of selected features. To achieve a refined model that requires fewer features, we need a data-driven way to determine the appropriate α . This can be accomplished by first determining the α that would ignore all features (coefficients are set to zero) and then gradually decreasing its value, possibly down to 1% of its starting value. We would need to set $ll_ratio=1.0$ and $alpha_min_ratio=0.01$ to search for 100 α values up to 1% of the estimated maximum. Fortunately, it is implemented in scikit-survival's `skurv.linear_model.CoxnetSurvivalAnalysis` [25].

Elastic Net The LASSO method is effective for selecting a subset of discriminative features. However, it has its shortcomings. The first is that LASSO cannot select more features than there are instances in the training data set. The second is its tendency to randomly select only one feature out of a set of highly correlated ones. The *Elastic Net* alleviates these issues by incorporating the l_1 and l_2 penalties in a weighted manner, as is shown in the following optimization problem

$$\arg \max_{\beta} \log PL(\beta) - \alpha \left(r \sum_{j=1}^p |\beta_j| + \frac{1-r}{2} \sum_{j=1}^p \beta_j^2 \right),$$

where $r \in [0; 1]$ is the relative weight of the l_1 and l_2 penalty ($r = 1$ is a LASSO penalty, $r = 0$ is a ridge penalty). The Elastic Net penalty combines the LASSO's feature selection capability and Ridge's regularization power. As a result, it is more stable than LASSO, and in a situation of highly correlated features, it would select them all, while LASSO would randomly select only one. Usually, it is sufficient to give the l_2 penalty only a small weight to improve the stability of the LASSO; for example, $r = 0.9$ might suffice.

Similar to LASSO, the weight α inherently dictates the size of the chosen subset. Its optimal value is typically estimated in a data-driven way [25]. More on that in Subsection 4.3.1, where we will choose the best α for our data set.

To conclude, Ridge, LASSO, and Elastic Net are potent extensions of the Cox PH model, especially when dealing with high-dimensional datasets and highly correlated features. They help to regularize the Cox model and select only the most significant features, avoiding overfitting and assuring a more stable and interpretable model. Nevertheless, it is important to acknowledge their limitations. As linear models, they capture only linear relationships, and non-linear relationships remain uncaptured. That leads us to

the next sections dedicated to machine learning methods. Machine learning techniques are known for their ability to catch complex non-linear relationships, often outperforming traditional statistical methods in predictive accuracy.

2.5.4 Machine Learning Methods

2.5.4.1 Survival Tree

Survival Tree is an adaptation of a Decision Tree (Section 2.1.4) to Survival Analysis, presented by LeBlanc [71]. Similarly to the CART algorithm, it recursively divides the dataset at each node based on one feature that maximizes the dissimilarity in the survival distribution instead of minimizing the MSE. The dissimilarity is measured with a log-rank test (Section 2.5.3.2) or any other statistic test that can handle survival data. LeBlanc shows that log-rank performs well for splitting survival data. The scikit-survival implementation of the Survival Tree also uses log-rank. In the original paper, pruning was suggested as a way to deal with overfitting. However, after reviewing the source code for the scikit-survival's `SurvivalTree` [72] and scikit-learn's `DepthFirstTreeBuilder` [73] (used in `SurvivalTree`), we saw that they did not use pruning. The model should be regularized manually with hyperparameter tuning. All key regularization hyperparameters are the same as in Decision Trees (Section 2.1.4), namely `max_depth`, `min_samples_split`, and others.

2.5.4.2 Gradient Boosted Survival Analysis

Gradient Boosted Survival Analysis (GBSA) [56] is a Gradient Boosting algorithm (discussed in Section 2.1.5) adapted for survival analysis and implemented within the scikit-survival package. Friedman's Gradient Boosting is adapted to survival analysis by setting the partial log-likelihood as the loss function and following the algorithm [70]. The partial log-likelihood allows for the inclusion of censored instances without introducing bias. GBSA shares many hyperparameters with the Decision Trees (Section 2.1.4), such as `max_depth`, `min_samples_split` and others, for the apparent reason of trees being a base learner.

In addition to Cox Proportional Hazards ("coxph" value of `loss` hyperparameter), GBSA also allows for choosing other loss functions, such as squared regression loss ("squared") that ignores predictions beyond the time of censoring and inverse-probability of censoring weighted least squares error ("ipcwls"). In this work, we will use the Cox Proportional Hazard loss. Other hyperparameters that we have used are:

- `learning_rate` - hyperparameter that sets the weight of each learner's contribution.
- `n_estimators` - dictates how many estimators must be trained.
- `subsample` - sets the fraction of the dataset to be used to train each individual learner. Values span (0, 1], where 1.0 means Gradient Boosting, while values less than 1.0 result in the Stochastic Gradient Boosting algorithm.
- `random_state` - allows us to control the randomness during training to ensure that any performance improvements are due to deliberate changes rather than random fluctuations.

For more information on the hyperparameters, you can refer to the GBSA documentation [56].

2.5.4.3 Random Survival Forest

Random Survival Forest (RSF) is an ensemble machine-learning method for survival analysis. It is derived from Breiman's original *Random Forest* method described in [28]. RSF uses the Survival Tree (discussed in Section 2.5.4.1) as a base learner to predict the cumulative hazard function (CHF). Randomness in a Random Survival Forest takes two forms: 1) randomly drawn bootstrap samples used to grow a tree, and 2) a random number of variables is chosen as candidates for splitting.

Random Survival Forest shares similarities with the Gradient Boosting: they are ensemble models that use trees as base learners. However, beyond the differences in specific trees used – where RSF employs Survival Trees in contrast to decision trees used in GB — the two differ in approach to building the ensemble. Gradient Boosting builds its ensemble by sequentially adding base learners, with each tree correcting the errors made by its predecessor. On the other hand, Random Forests train all trees independently and then average their predictions, allowing for the parallelization of computation and possibly shorter training time.

Algorithm Random Survival Forest training is comprised of the following steps:

1. Randomly draw B bootstrap samples from the training dataset. About one-third (37%) of each bootstrap is excluded and termed out-of-bag (OOB) observations.
2. For each bootstrap sample, develop a survival tree. At each node, randomly select p candidate variables. Split the node using the candidate variable that maximizes the survival difference between children nodes.
3. Develop until terminal node has no less than $d_0 > 0$ unique deaths or a set value in one of the regularization hyperparameters (from Section 2.1.4) is reached.
4. Calculate the cumulative hazard function (CHF) with the Nelson-Aalen estimator for each tree. Calculate the average of all trees to find the ensemble CHF.
5. Use OOB data to determine the prediction error of the ensemble [26].

Cumulative Hazard Function Calculation From the Nelson-Aalen estimator defined in Equation 2.18, the CHF estimate for each terminal node h of a tree is given by

$$\hat{H}_h(t) = \sum_{t_{l,h} \leq t} \frac{d_{l,h}}{r_{l,h}}.$$

All cases within the same leaf have the same CHF. Let $H(t|\mathbf{x}_i)$ be the CHF for the sample i and \mathbf{x}_i covariates describing it. We need to drop \mathbf{x}_i down the survival tree to get this value. After that, we will get the terminal node h . Then CHF for sample i and terminal node h is denoted by

$$H(t|\mathbf{x}_i) = \hat{H}_h(t), \text{ if } \mathbf{x}_i \in h. \quad (2.23)$$

Now, we need to compute the CHF of the whole ensemble of B trees. The OOB ensemble CHF for i is an average over bootstrap samples and is given by

$$H_e^{**}(t|\mathbf{x}_i) = \frac{\sum_{b=1}^B I_{i,b} H_b^*(t|\mathbf{x}_i)}{\sum_{b=1}^B I_{i,b}},$$

where $I_{i,b}$ is an indicator function of i being in OOB case for tree b (equal to 1, if it is, 0 otherwise), $H_b^*(t|\mathbf{x}_i)$ denotes the CHF 2.23 for tree trained from the b th bootstrap. The bootstrap ensemble CHF for sample i is expressed as

$$H_e^*(t|\mathbf{x}_i) = \frac{1}{B} \sum_{b=1}^B H_b^*(t|\mathbf{x}_i). \quad (2.24)$$

Notably, it uses all trees, not only those where i is in OOB.

Prediction To get the prediction, we need to drop the feature vector \mathbf{x}_i down the 2.24. The prediction of the ensemble is the average cumulative hazard function among all learner's predictions. However, we might want to predict a survival function. Recall the Equation 2.15. From it, the formula to estimate the survival function for the given \mathbf{x}_i is given by

$$S(t, \mathbf{x}_i) = e^{-H_e^*(t|\mathbf{x}_i)}.$$

Hyperparameters Most of the available hyperparameters, such as *max_depth* and *min_samples_split*, are inherited from trees and are described in the Section 2.1.4, while some are shared with gradient boosting, such as *random_state* and *n_estimators*. A unique hyperparameter, *n_jobs*, allows for the specification of the number of CPU cores to use in training. -1 means all cores; the default is set to 1. Refer to the Random Survival Forest documentation [55] to see all hyperparameters.

In summary, the Random Survival Forest is a robust and powerful survival analysis approach that harnesses the collective power of multiple survival trees. We have explored several survival analysis methods, and it is crucial to evaluate them properly since the standard machine learning performance metrics are rarely applicable to survival analysis due to the presence of censoring. The following section will introduce essential criteria for evaluating survival analysis models and the primary performance metrics created for this purpose.

2.5.5 Performance Metrics

Survival prediction models play a vital role in healthcare. They are often used to estimate the risk of developing a particular disease and are crucial in guiding the clinical management of patients. It is, therefore, essential to assess their performance accurately. Similar to machine learning, this process of model evaluation is referred to as model validation. There are three aspects we can assess our model on:

1. **Overall performance**, which is the distance between the predicted and observed survival time.
2. **Discrimination**, or the model's ability to distinguish between high- and low-risk patients.
3. **Calibration** is the agreement between the observed and predicted survival times [21].

The absence of bias when the validation set contains censored instances indicates a good performance measure. Otherwise, in the presence of high levels of censoring, the evaluation would be unreasonably optimistic [21].

In this section, we will cover three measures of discrimination and one measure that assesses both discrimination and calibration (overall performance).

2.5.5.1 Harrel's and Uno's Concordance Indices

One way to measure discrimination is through *concordance*. A pair of patients is *concordant* if the subject who experiences an event earlier has a greater estimated risk [30]. *Measures of concordance* quantify the rank correlation between the predicted risk and the observed survival times. Typically, their values range between 0.5 and 1. A value of 0.5 indicates no discrimination, while 1 corresponds to the ideal discrimination [21].

The *concordance index*, or *C-index*, is the most widely used performance metric in survival analysis. It is defined through the concordance probability. The *concordance probability* is the probability that from the arbitrarily selected pair of patients (i, j) , the one with a shorter survival time, T has the higher predicted risk, M [21]. Mathematically, this is expressed as

$$C = P(M_i > M_j | T_i < T_j).$$

Harrell's concordance index is the most widely used implementation of the concordance index. To compute Harrel's concordance index C_H , we consider every comparable pair of patients where the one with the shorter time failed. Pair is "comparable" if we can determine which patient experienced the event first [30]. C_H is estimated as the proportion of these pairs in which the subject with the shorter survival time has a higher estimated risk. A modified version of this estimator, $C_H(\tau)$, only considers patients with $T_i < \tau$ and may provide more stable estimates [21].

Mathematically, Harrel's C-index is defined as a ratio between the number of concordant and comparable pairs and is denoted as

$$\hat{C} = \frac{\sum_{i=1}^N \Delta_i \sum_{j=i+1}^N \left[I(T_i^{obs} < T_j^{obs}) + (1 - \Delta_j) I(T_i^{obs} = T_j^{obs}) \right] \left[I(M_i > M_j) + \frac{1}{2} I(M_i = M_j) \right]}{\sum_{i=1}^N \Delta_i \sum_{j=i+1}^N \left[I(T_i^{obs} < T_j^{obs}) + (1 - \Delta_j) I(T_i^{obs} = T_j^{obs}) \right]}. \quad (2.25)$$

where $I(\cdot)$ is the indicator function. It is equal to 1 if its argument is true and 0 if it is not. T^{obs} is the observation time. Δ_i is a binary variable, and $\Delta_i = 1$ if the subject i experienced an event during the time of observation and $\Delta_i = 0$ if they did not.

According to scikit-survival's documentation [22] and Rahman et al. [21], Harrel's concordance index becomes biased in the presence of censoring. The bias increases with the level of censoring. Uno et al. [31] introduced a modified concordance index, $C_U(\tau)$, which incorporates weights based on the probability of being censored. While their estimator proved robust to the choice of τ , they noted that the error of the estimate might be quite large if there are too few instances beyond this time point [21, 22].

Having explored the concordance index and its modifications, it is evident that it provides valuable insight into the model's discriminatory abilities. However, it offers a singular value that characterizes a model's performance across different time points without considering fluctuations in concordance that tend to happen over time. To enhance our evaluation of model performance in terms of discrimination across varying time points, we turn our attention to another popular metric in survival analysis: the ROC AUC. The subsequent section explores it in detail.

2.5.5.2 Time-dependent Area under the ROC

The *area under the receiver operating characteristic curve* (ROC AUC) is a popular performance measure for binary classification tasks. In survival analysis, it is used to determine how well estimated risk scores can distinguish diseased patients from healthy ones [22].

In binary classification, the *receiver operating characteristic (ROC)* is a curve that plots the *true positive rate (TPR or sensitivity)* against the *false positive rate (FPR)*. The FPR is the ratio of negative

instances falsely classified as positive. It is equal to $1 -$ the *true negative rate* (the ratio of negative instances that are correctly classified, often referred to as *specificity*). TPR, or sensitivity, represents the ratio of positive instances classified as positive [9].

In survival analysis, we extend the ROC to continuous outcomes, where a patient is alive at the start of the observation but might experience an event later. Specificity and sensitivity, therefore, become time-dependent measures. It is especially important, as model accuracy tends to be different at different points in time. Here, we consider *cumulative cases* and *dynamic controls* at any given time t . *Cumulative cases* are all subjects who experienced an event before or at time t , while *dynamic controls* are those who have yet to experience the event after time t . By calculating the ROC AUC for any given time point t , we assess the model's ability to differentiate between patients. Specifically, how well the model can distinguish patients who fail by a given time $t_i < t$ from subjects who fail after this time $t_i > t$. The time-dependent ROC AUC is especially helpful when we want to predict an event happening in a period up to time t , rather than at a specific time-point t [22].

While the time-dependent ROC AUC provides a valuable measure of a model's discrimination ability at various time points, it falls short in providing insight into the accuracy of individual predictions – calibration. To address this limitation and provide a more comprehensive model assessment, we turn our attention to the time-dependent Brier score.

2.5.5.3 Time-dependent Brier Score

Time-dependent ROC AUC and concordance index are great for assessing the overall discrimination among all time points (mean AUC and c-index) and the discrimination at any individual time point (the ROC graph), but they tell us nothing about the accuracy of individual predictions [22]. A metric analogous to regression performance measures used in machine learning would be ideal. Fortunately, such a metric exists. *Time-dependent Brier score* is a modification of mean squared error (MSE) that handles right censored data.

While the concordance index and time-dependent ROC AUC measure only discrimination, the time-dependent Brier score measures both discrimination and calibration, making it a metric of "overall performance". It is defined as follows

$$BS^c(t) = \frac{1}{n} \sum_{i=1}^n I(y_i \leq t \wedge \delta_i = 1) \frac{(0 - \hat{\pi}(t|\mathbf{x}_i))^2}{\hat{G}(y_i)} + I(y_i > t) \frac{(1 - \hat{\pi}(t|\mathbf{x}_i))^2}{\hat{G}(t)},$$

where $\pi(\hat{t}|\mathbf{x})$ is a model's predicted probability of remaining event-free up to the time point t for feature vector \mathbf{x} , and $\frac{1}{\hat{G}(t)}$ is the inverse probability of censoring weight [22]. $I(\cdot)$ is the indicator function. y_i is the subject's survival time, δ_i is the event/censoring indicator.

The time-dependent Brier Score's limitation is that it applies exclusively to models capable of estimating the survival function. The *Integrated Brier Score* provides a scalar value for general model evaluation. It is beneficial for model comparison, as time-dependent measures are a bit harder to compare than scalar values, and for the model fine-tuning process.

In this chapter, we covered various approaches for training machine learning models. We discussed the training process, data preparation, handling missing values, and finding optimal hyperparameters for our models. Additionally, we discussed survival analysis, which offers a range of techniques for predicting the time to an event. It encompasses various models and performance metrics capable of handling censored data during training, prediction, and model evaluation. Now, we are well-equipped to perform our analysis. However, before we proceed, let us ensure that we have a dataset to work with and a solid understanding of the data .

Chapter 3

Data Preparation and Analysis

In this chapter, we will examine the study's dataset. We will explore its most important features and their influence on survival time.

3.1 Data Acquisition

The dataset provided by the IKEM (Institute of Clinical and Experimental Medicine in Prague) that we initially had was unsuitable for any meaningful analysis. It had very few records, an insufficient number of features, many missing values, and quite outdated follow-ups. That is why it was decided to look for data elsewhere.

A number of institutions were contacted, including the Scientific Registry of Transplant Recipients (SRTR), the Australia and New Zealand Dialysis and Transplant Registry (ANZDATA), the National Health Service (NHS), the Center for Research in Transplantation and Translational Immunology (CR2TI), and other organizations from Spain, Germany and Canada. However, we faced the following obstacles:

- High cost of acquiring the data
- The need for association with a local research group
- The need for local citizenship.

Fortunately, the United Network for Organ Sharing (UNOS), a US organization, agreed to provide their data for free upon signing the data use agreement. The database consists of 1,108,884 records for all kinds of transplants. The table that contained data crucial for our analysis encompassed both kidney and pancreas transplants, had 993 806 records and 457 columns for both transplanted patients and ones from the waiting list. Data span from October 1, 1987, to September 2022. Kidney transplants account for 490 172 records.

3.2 Data Loading

The data were provided in the form of a MongoDB database dump. Unfortunately, the database dump alone was insufficient for data analysis. It was necessary to run the database, import the database dump, and export data in the appropriate format. We set up the MongoDB database in a Docker container (Docker explained in Section 5.2.2) running locally on our PC, as we were not allowed to install Docker

in our university cluster. The corresponding table was then exported to CSV, compressed into a zip archive, and uploaded to the cluster.

The process of loading the data using the `read_csv()` method of the Pandas DataFrame was taking too long (about 5 minutes) due to the large size of the CSV file (80GB). The nature of data science is quite iterative, so we had to optimize this inefficiency. As a result, it was decided to store data in a Parquet file. Parquet is an efficient cloud computing format that works on the principles of databases, allowing for more efficient data loading. The `DataFrame.to_parquet()` method was used to dump the pandas DataFrame into the Parquet database file. This significantly reduced the loading time of the entire dataset to just 38 seconds. Moreover, it allows for specifying the columns to load, which further reduces the data loading time to a mere 21 seconds. Therefore, using this technology has significantly improved the workflow.

3.3 Data Preprocessing Pipeline

This section will describe the data pipeline used to create the dataset from the raw data. The pipeline can be found in the GitHub repository of this paper: `survival_pipeline.py`.

The work with the pipeline is pretty straightforward: we initialize the class and call the `load()` method. As is shown in the following block of Python code:

```
1 from surv_data_pipeline.survival_pipeline import
   ScikitSurvivalDataLoader
2
3 loader = ScikitSurvivalDataLoader()
4 X, y = loader.load()
```

Two main class constants are `categorical_values` and `numerical_values`, whose names speak for themselves. The primary class method is `load()`. This method loads the data into the pandas DataFrame, processes them, and returns X and y . X represents data transformed into numbers, while y is a tuple consisting of two elements: `PSTATUS` and `PTIME`. `PSTATUS` is the boolean censoring indicator (True - the event happened, False - otherwise), `PTIME` is the number of days survived. All scikit-survival estimators require this format of the target value.

The first step is to load the data from the parquet file into Pandas DataFrame. It is done using the Pandas method `read_parquet(path, engine, columns)`. In the `path` property, we need to specify the path to the parquet file; the `engine` parameter specifies the data loading scheme. We use `'auto'`, which tries PyArrow first; if that does not work, it uses FastParquet. PyArrow and FastParquet are just interfaces for reading data from Parquet files. In `columns`, we need to specify the columns we want to load.

Now, we need to filter out all records that might add noise to the data. Firstly, we remove patients who died of unrelated causes, such as accidents. Next, we filter out all recipients from the pediatric group, as the transplantation at that age differs significantly from that of adults. Finally, depending on the donor type for which we are training a model, we will include only living or deceased patients.

The next step is to handle missing (NaN) values. It is done with `_handle_nan()` method. We tried using mean and median imputation techniques from Section 2.3.3, but that only led to data leakage and suboptimal performance, so we just dropped all rows with empty columns. Fortunately, the size of the dataset allowed us to do that.

At this point, data were pickled and downloaded, and the next steps were carried out locally. It is more comfortable to work locally and it is good to have a control over the later stages of data processing in each iteration of training. This helped to iterate faster and to get the results quicker.

After the NaN handling step, the training set is sent to the method `_get_X_y()` (or the same steps are done locally) where the numerical columns are scaled (with `StandardScaler()`) and categorical is one-hot encoded (with `OneHotEncoder()`) in the scikit-learn transformer pipeline. The pipeline is pickled for later use in the application to transform the user input into data digestible by the model. Numerical and categorical values then comprise the X set. The target value set is constructed with the `Surv.from_arrays()` utility that accepts event and survival time and builds the y value acceptable to scikit-survival algorithms.

At this stage, the pipeline is divided in two: for models trained locally and models trained on the university cluster. For models trained locally, we pickle and download the data. And perform scaling for numerical features, one-hot encoding for categorical ones and For models trained on the cluster, data are sent to the method `_get_X_y()`. In both cases similar steps are taken: numerical columns are standardized (with `sksurv.column.standardize()`), categorical ones are one-hot encoded (with `sksurv.preprocessing.OneHotEncoder()`), and target value is constructed with the `Surv.from_arrays()` method that accepts event and survival time and builds the y value acceptable to scikit-survival algorithms.

There is only one distinction in the pipeline for models used in the application. We used custom transformers built with scikit-learn to have the ability to pickle the pipeline used for training and use it later in the application to avoid the complexity and potential errors of developing a custom pipeline. This approach ensures consistency and minimizes mistakes, thereby enhancing the reliability of our model deployment process. Models, unfortunately, need to be retrained with data from these transformers for use in the application, as our chosen approach does not preserve the column names, and we cannot assess feature importance with certainty. The approach with scikit-survival methods described before allows for preserving the column names and accurate feature importance assessment.

At this point we have the dataset to work with.

3.4 Exploratory Data Analysis

In this section, we will comprehensively cover the most significant features. It is important to note that the data were not adjusted to limit the influence of other factors. As a result, the correlations we are attempting to make here may not be entirely accurate; however, some have been confirmed in the existing literature. All plots are based on the training dataset. Features in DDT/LDT sections differ because different features were deemed significant by the permutation importance algorithm for the given transplantation type.

3.4.1 General Data

Table 3.1 lists the features used in this work along with their descriptions and types. The features used in each model differ from model to model and can be found in the feature importance figures of each model in Chapter 4.

Survival Data

In this subsection, we will explore the y -axis that will be used for the survival estimator training. As mentioned earlier, the y value consists of boolean censoring status (PSTATUS) and time (PTIME), a numerical value representing the survival time or the time of censoring. The y value is called the *survival data*.

Figure 3.1 shows the survival time box plot (PTIME column) for living and deceased donor transplantations. The first quantile (Q_1) of the deceased donor subgroup is equal to 1770 days (4.85 years), the median is 2926 days (8 years), and the third quantile (Q_3) is equal to 3828 days (11.2 years). The

Table 3.1: Table of Features and Their Types

Feature	Description	Type
ON_DIALYSIS	The recipient's pre-transplant dialysis status.	Categorical
PRE_TX_TXFUS	The recipient's history of pre-transplant blood transfusions.	Categorical
GENDER	Recipient gender	Categorical
ETHCAT	Recipient ethnicity	Categorical
DIAB	Recipient diabetes status.	Categorical
HCV_SEROSTATUS	Recipient Hepatitis C status	Categorical
DIABETES_DON	Donor diabetes status.	Categorical
ETHCAT_DON	Donor ethnicity.	Categorical
ABO_MAT	Donor-recipient blood type match type.	Categorical
HBV_CORE	Recipient Hepatitis B status.	Categorical
LIV_DON_TY	Living donor type.	Categorical
AGE	Recipient age at transplant.	Numerical
BMI_CALC	Recipient BMI at transplant.	Numerical
AGE_DON	Donor age at transplant.	Numerical
CREAT_TRR	Recipient serum creatinine at transplant.	Numerical
NPKID	Number of previous transplants.	Numerical
COLD_ISCH_KI	The duration kidneys are preserved at low temperature.	Numerical
KI_CREAT_PREOP	Living donor preoperative serum creatinine (mg/dL)	Numerical
HGT_CM_CALC	Recipient height in centimeters	Numerical
BMI_DON_CALC	Donor BMI	Numerical
DIALYSIS_TIME	Days on dialysis.	Numerical
KDPI	Kidney Donor Profile Index.	Numerical

interquartile range (*IQR*) is 2058 days, which forms the box. The whiskers extend from 0 days to 7397.3 (20.3 years) days. Any value above 7397.3 is an outlier, as it is above the 99th percentile. The statistics for the living are generally better: Q_1 corresponds to 2287.75 days (6.3 years), the median is 3280 days (9 years), and Q_3 is equal to 4426 days (12.1 years). However, the 99th percentile is much closer to the value in the deceased subgroup compared to the quantiles: 7623 days (20.9 years), which is only 226 days of difference. As can be seen the deceased donor subset has more outliers than the living donor one. This might be attributed to the larger number of instances.

Figure 3.2 shows pie charts of the PSTATUS column values for living and deceased donor groups. As can be seen, the percentage of censoring in deceased donor transplantation subset is 51.6%, and in living donor transplantation 68.6%. This difference might be attributed to the fact that the number of LDT has started to grow only relatively recently (Figure 3.5) and less recipients experiences the event, and that LDT tends to be slightly more successful [16, 17], as also illustrated in Figure 3.1. Now, let us talk in more detail about the influence of the donor type on survival.

Donor Type

In this subsection, we will explore the influence of donor type (living or diseased) on survival. It is a well-established fact that recipients who receive kidneys from living donors have higher life expectancy [16, 17]. Let us confirm that on the data. Figure 3.3 plots two Kaplan-Meier survival curves for all patients from the dataset. On the graph, we can see that the LDT survival probability is indeed higher

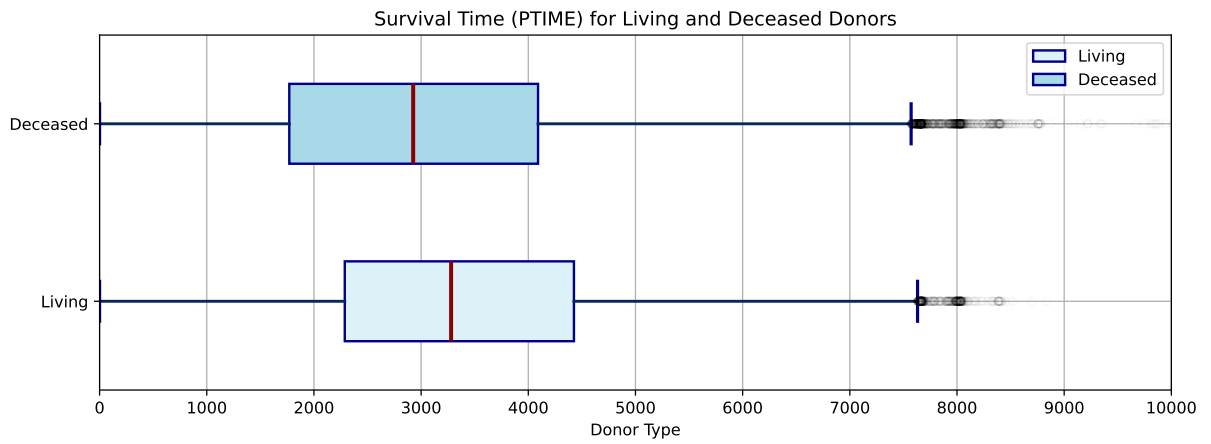


Figure 3.1: Box Plots for the Survival Time for Deceased and Living Donor Groups

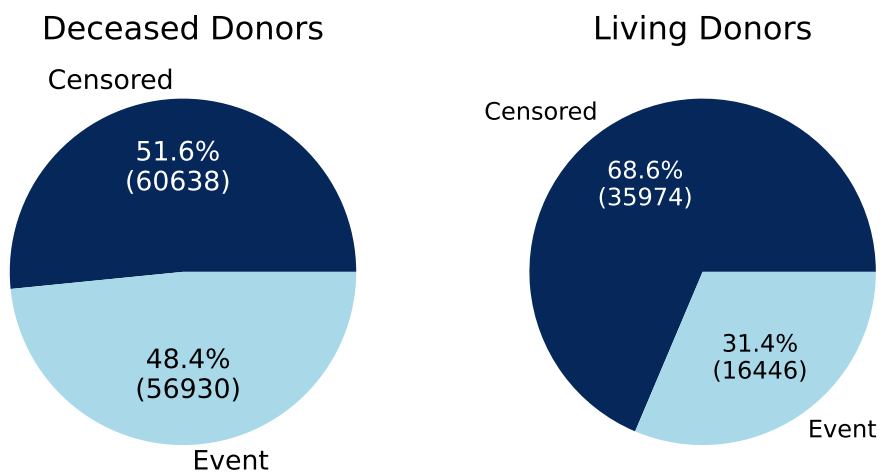


Figure 3.2: Pie Charts of Censored/Non-censored Values for Deceased and Living Donor Groups

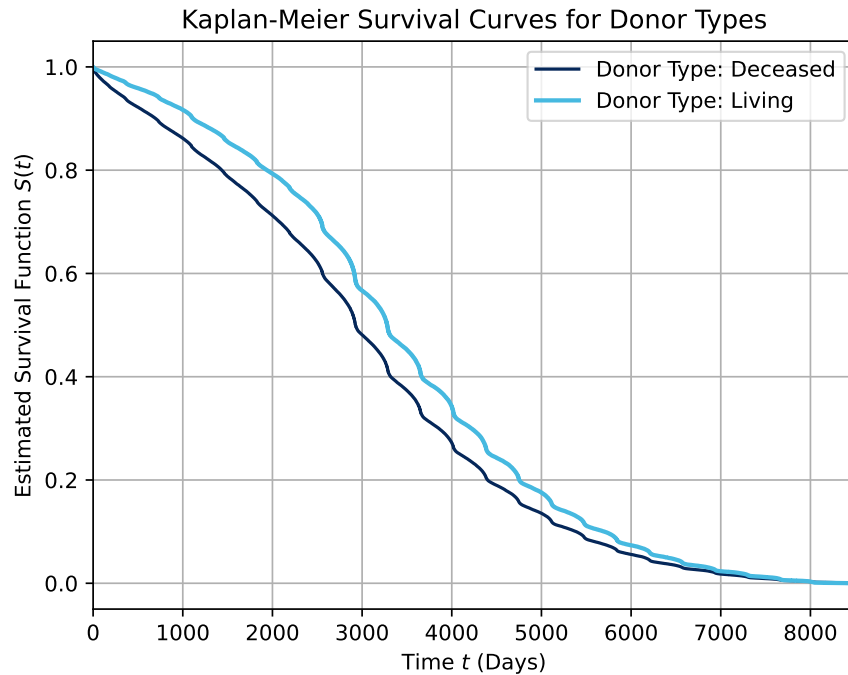


Figure 3.3: Kaplan-Meier Survival Curves for Donor Types

than the DDT survival probability. This is the case because often, there is no time to make full, in-depth HLA screening, allowing for some HLA mismatches. Additionally, deceased transplants may suffer from mild kidney damage due to the delay in transplantation. Living donor transplants are most often performed between blood relatives who share similar HLA, or there is more time to find a compatible not related donor, resulting in better compatibility.

Figure 3.6a illustrates the distribution of donor types in a pie chart. As can be seen, deceased donor transplantation is much more prevalent. Now let us look at the number of deceased/living donor transplants by year on Figure 3.5. Although the number of living transplantations mostly grows each year, deceased transplantations still prevail.

Recipient Gender

In this subsection, we will explore the gender distribution in our dataset and its influence on survival. Figure 3.6b illustrates the distribution of gender in our dataset. As can be seen, there are more males than females. Even though chronic kidney disease is more common in women, end-stage kidney disease and, therefore, the need for kidney transplants is more common in men [14].

Now, let us take a look at gender's influence on survival. Figure 3.4 illustrates the Kaplan-Meier survival curves for men and women on the whole dataset. As can be seen from the graph, females generally have a lower risk than their male counterparts. Women usually live longer [14]. Quite a significant factor is the difference between male and female immune responses – males have a greater risk of getting an infection than females, and the intensity of the infection is higher [15]. Furthermore, the influence of immunosuppressants makes the risk of infection even worse.

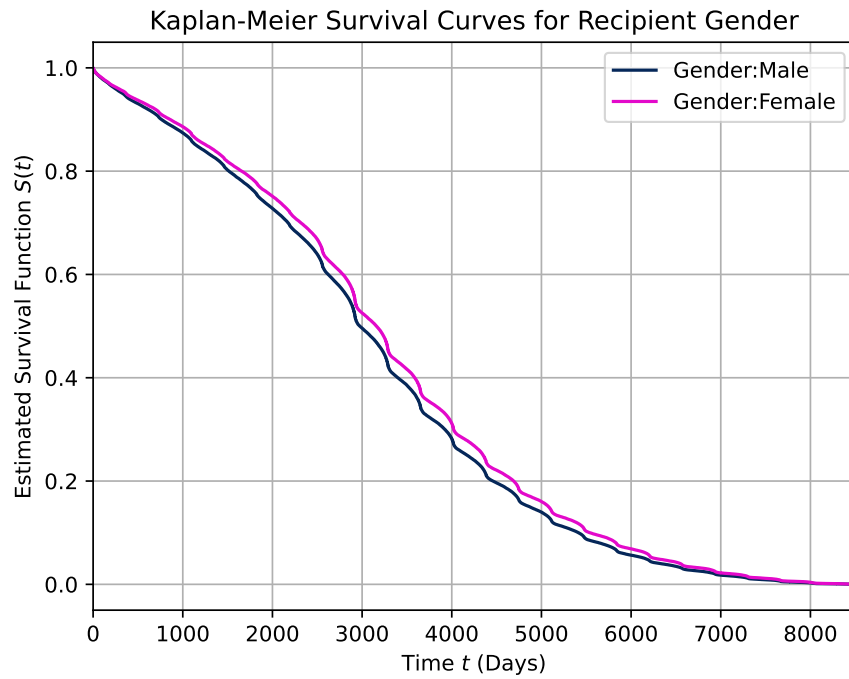


Figure 3.4: Kaplan-Meier Survival Curves for Genders

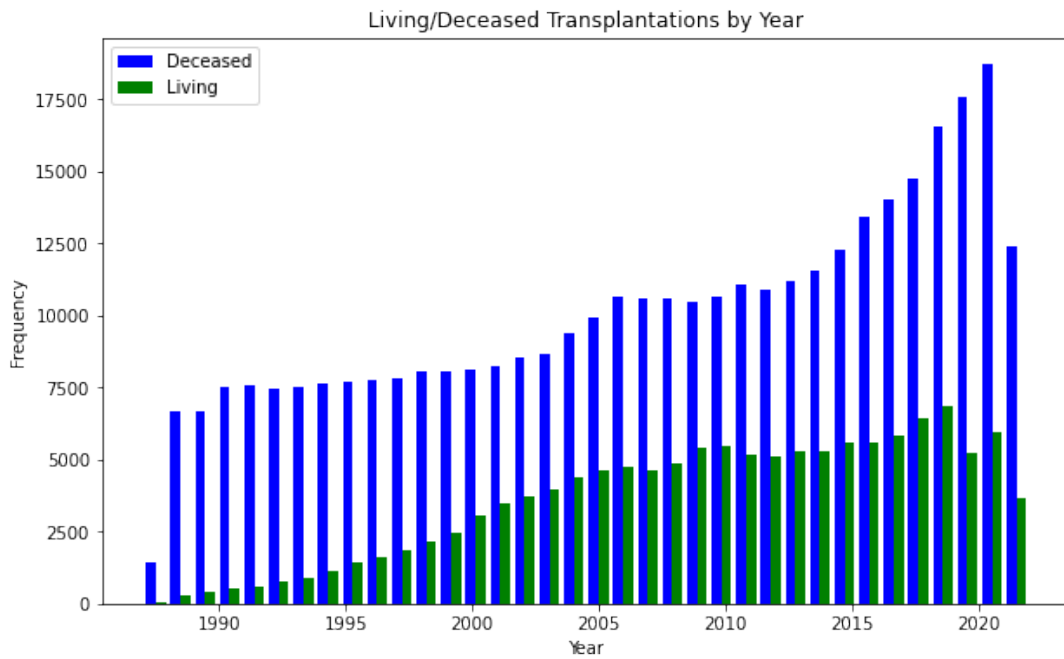


Figure 3.5: Histogram of Deceased/Living Donor Transplantations by Year

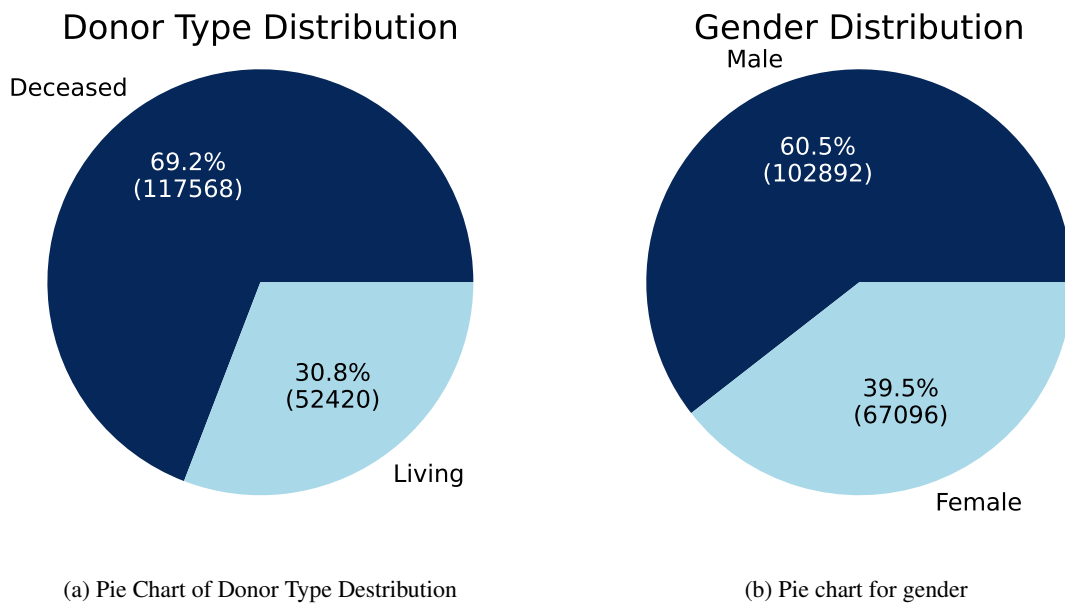


Figure 3.6: Pie Charts of Donor Type (Living/Deceased) and Gender

The Recipient Use of Dialysis

In this subsection, we will explore the influence of dialysis on survival. In Figure 3.7, we can see two survival curves for the patients who were on dialysis and who were not. As can be seen, the patients who were on dialysis before the transplantation have a greater risk than those who were not. It agrees with [20, 48].

Recipient Ethnicity

This subsection explores the survival curves of different ethnic groups. Figure 3.8 plots seven survival curves for various ethnicities are plotted. As can be seen in the image, five ethnicities share about the same survival probability, while three groups slightly differ from the rest other. Multiracial has the best survival curve, yet later falls under the other curves. Native and Native Hawaiian have curves slightly below others. For these three ethnicities, these survival curves are not enough to make any conclusions, as these are also the least populated groups. As for other ethnicities, they have mostly similar survival curves with the exception of Asian group, which is slightly higher than the rest, yet later converges with them.

3.4.2 Deceased Group

In Table 3.2, we can see the numerical value statistics for the deceased subset, which includes a total of 117 568 records.

Moving on to Table 3.3, we see the distribution of categorical Yes / No features. As we can see, the majority of kidney recipients from deceased donors were on dialysis and did not receive blood transfusions. The fact that the majority of recipients from the deceased donor subset were on dialysis may be attributed to the fact that the suitable living donor was not found in time, and the deceased donor's kidney was received more or less as a last resort.

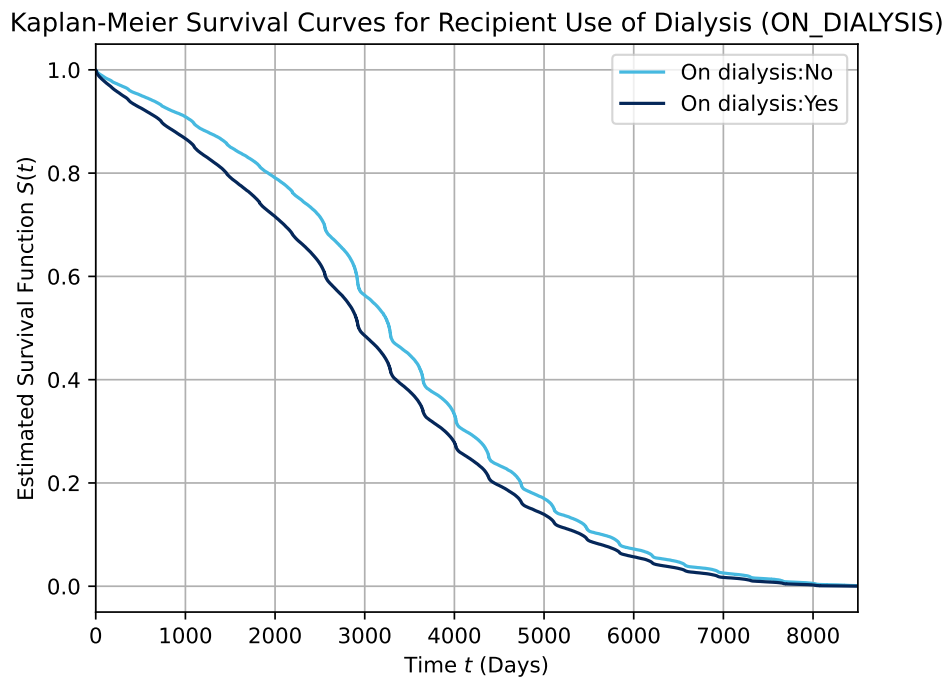


Figure 3.7: Kaplan-Meier Survival Curves for the Pre-transplant Usage of Dialysis or not

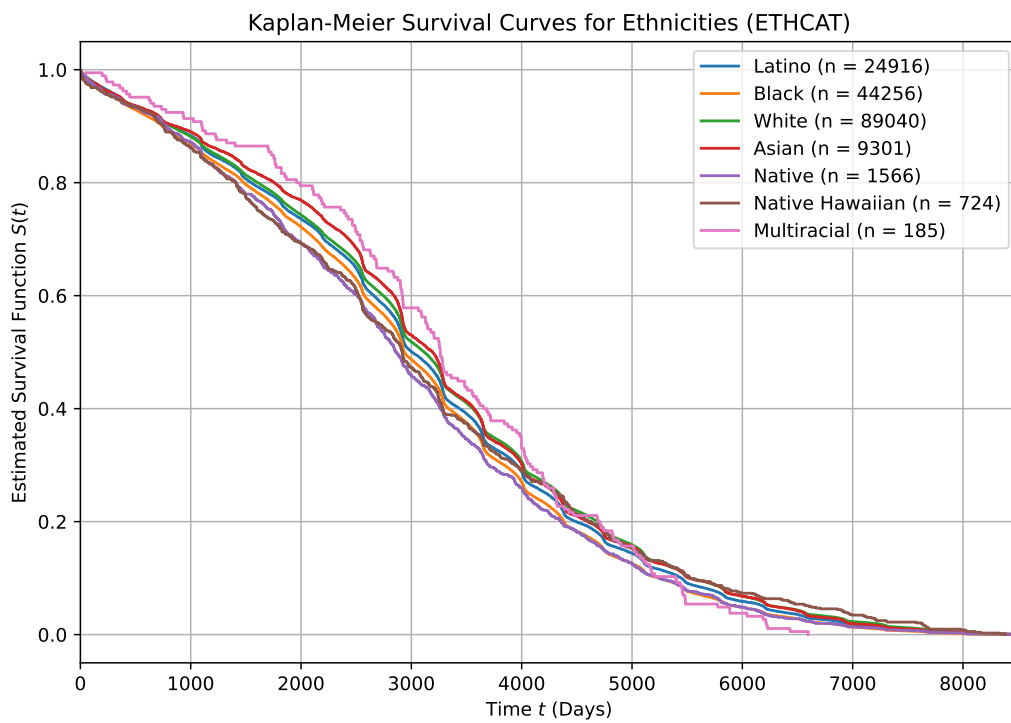


Figure 3.8: Kaplan-Meier Survival Curves for Ethnicities

Table 3.2: Descriptive Statistics for Numerical Features for Deceased Donors

	mean	std	min	25%	50%	75%	max
AGE	51.88	13.11	18.0	43.00	53.00	62.0	90.0
BMI_CALC	27.77	5.42	15.0	23.80	27.30	31.3	72.2
AGE_DON	38.10	16.59	0.0	24.00	40.00	51.0	88.0
CREAT_TRR	8.252	3.51	0.1	5.70	7.82	10.3	28.2
NPKID	0.12	0.36	0.0	0.00	0.00	0.0	5.0
COLD_ISCH_KI	18.02	8.98	0.0	11.87	17.00	23.0	99.0

Table 3.3: Frequency of Yes/No Variables in Deceased Donor Group

Value	ON_DIALYSIS	PRE_TX_TXFUS	DIABETES_DON
Yes	25881 (22.01%)	88215 (75.03%)	109920 (93.49%)
No	91687 (77.99%)	29353 (24.97%)	7648 (6.51%)

Figure 3.9 shows the ethnicity distribution of individuals who received a kidney from a deceased donor. It is worth noting that Black recipients make up 31.1% despite constituting only 12.6% of the US population. White recipients account for 46.7% of the total despite representing 58.9% of the US population. Hispanic/Latino recipients represent 14.8% of the total, while 19.1% of the overall US population. The only group with similar percentages are Asians/non-Hispanic, who make up 6.1% of total deceased donor kidney recipients, which is almost equivalent to their representation in the population at 5.6%. The remaining ethnicities constitute only 1.5%. US demographics data were taken from [68].

Figure 3.10 shows distributions for donor diabetes, and recipient hepatitis C status. 65.8% of the recipients do not have diabetes. Hepatitis C is a viral infection causing liver inflammation that can lead to severe liver damage. 33.5% have different forms of diabetes. The diabetes status of the remaining 0.8% is unknown. Almost 90% of the recipients did not have Hepatite C, while 6.4% tested positive.

3.4.3 Living Group

Table 3.4 illustrates the basic statistics for the numerical values for the living subset that contains 52420 records.

Table 3.4: Descriptive Statistics for Numerical Variables in Living Donor Group

	mean	std	min	25%	50%	75%	max
KI_CREAT_PREOP	0.87	0.38	0.10	0.70	0.80	1.00	25.0
SERUM_CREAT	1.78	1.42	0.10	1.10	1.40	1.90	21.0
NPKID	0.10	0.33	0.00	0.00	0.00	0.00	4.0
AGE	47.46	13.78	18.00	37.00	48.00	58.00	85.0
HGT_CM_CALC	171.11	10.67	109.00	162.60	170.20	178.00	213.4
BMI_DON_CALC	26.90	4.37	15.01	23.71	26.60	29.71	71.9
AGE_DON	41.34	11.55	15.00	32.00	41.00	50.00	77.0

Moving on, Table 3.5 presents the ethnicity distributions of donors and recipients in the living donor group. Notably, Whites constitute 65-68% of donors and recipients. It is higher than the corresponding

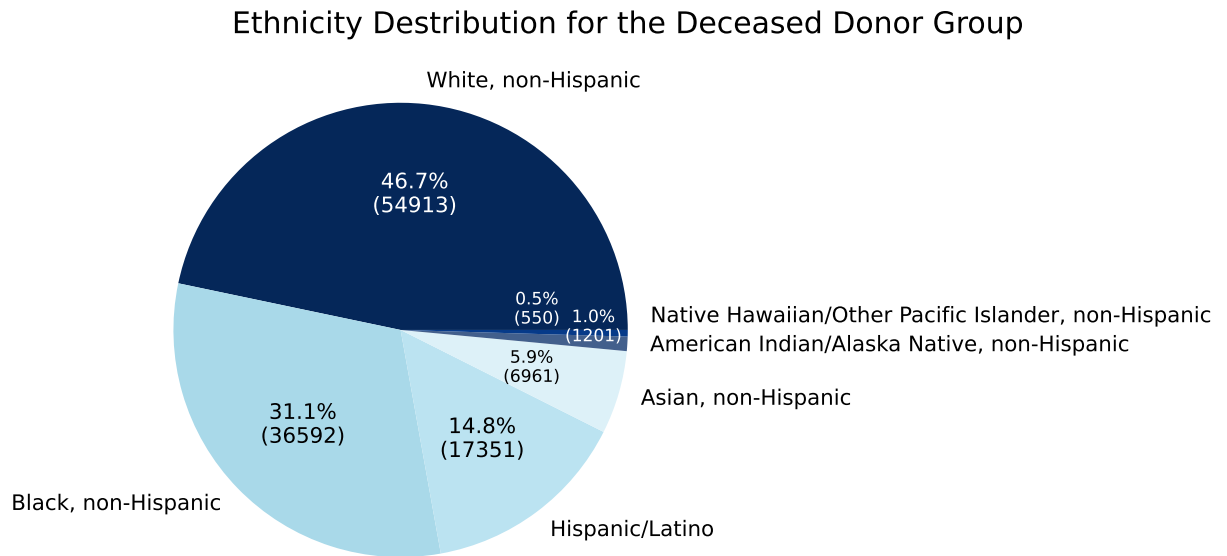


Figure 3.9: Pie Chart of Ethnicity Distributions for the Deceased Donor Subset

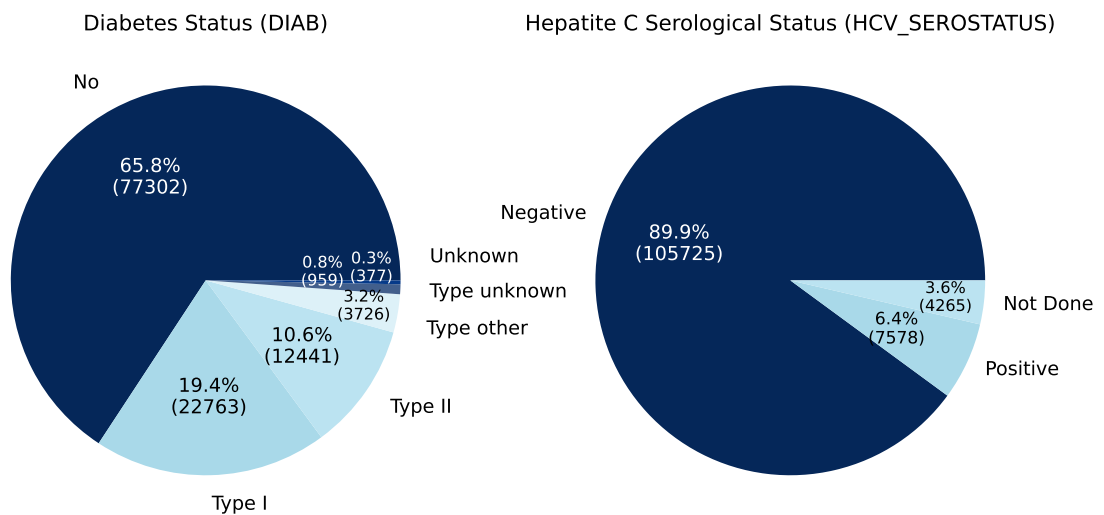


Figure 3.10: Pie Charts of Diabetes Status and Hepatitis C Serological Status for the Deceased Donor Subset

percentage in the deceased donor group and in the US population, which may be attributed to the larger pool of potential donors and the resulting relative ease of finding a match. On the other hand, Blacks are close to their percentage in the population, while the percentage of Hispanics/Latinos is lower than their percentage in the population. Other ethnicities constitute less than 2%.

Table 3.5: Frequency Table for ETHCAT and ETHCAT_DON in Living Donor Group

Category	ETHCAT (%)	ETHCAT_DON (%)
White, non-Hispanic	34127 (65.1%)	35778 (68.25%)
Black, non-Hispanic	7664 (14.6%)	7386 (14.09%)
Hispanic/Latino	7565 (14.43%)	6639 (12.67%)
Asian, non-Hispanic	2340 (4.46%)	1958 (3.74%)
American Indian/Alaska Native, non-Hispanic	365 (0.7%)	291 (0.56%)
Native Hawaiian/Other Pacific Islander, non-Hispanic	185 (0.35%)	216 (0.41%)
Multiracial, non-Hispanic	174 (0.33%)	152 (0.29%)

Table 3.6 provides information on the distribution of two Yes/No categories in the living donor subset, namely ON_DYALISIS and PRE_TX_TXFUS. The distribution of the ON_DYALISIS feature differs from that of the deceased donor subset. Here, the distributions of "Yes" and "No" are much closer to each other. This can be attributed to the fact that chronic kidney disease was caught in time, and the appropriate living donor was found before the use of dialysis was needed. Furthermore, the data indicate that the receivers of blood transfusions (PRE_TX_FUS) are in the minority.

Table 3.6: Frequency of Yes/No Variables in Living Donor Group

Value	ON_DIALYSIS (%)	PRE_TX_TXFUS (%)
Yes	22488 (42.9%)	42388 (80.86%)
No	29932 (57.1%)	10032 (19.14%)

Table 3.7 illustrates the distribution of recipient hepatitis. Recipients tested negative for hepatitis C and B are in the majority. Categories "Not Done" and "Positive" are in the minority; however, the amount of "Not Done" for hepatitis B is rather significant.

Table 3.7: Frequency of N/P Variables in Living Donor Group

	Value	HCV_SEROSTATUS (%)	HBV_CORE (%)
N	N	50237 (95.84%)	43276 (82.56%)
ND	P	801 (1.53%)	6429 (12.26%)
P	ND	1382 (2.64%)	2715 (5.18%)

Figure 3.11 illustrates the distributions of GENDER, DIAB, and ABO_MAT in the living donor subset. The gender distribution is the same as in the full dataset. Furthermore, 71.8% of recipients do not have diabetes; the remaining patients have some form of it. Additionally, three-quarters of all transplants are of identical blood groups, while 24% of living donor transplants are of compatible blood groups, and 1.4% are of incompatible blood groups.

Finally, Figure 3.12 shows the distribution of the living donor type feature that describes a recipient's biological/social relationship to a donor.

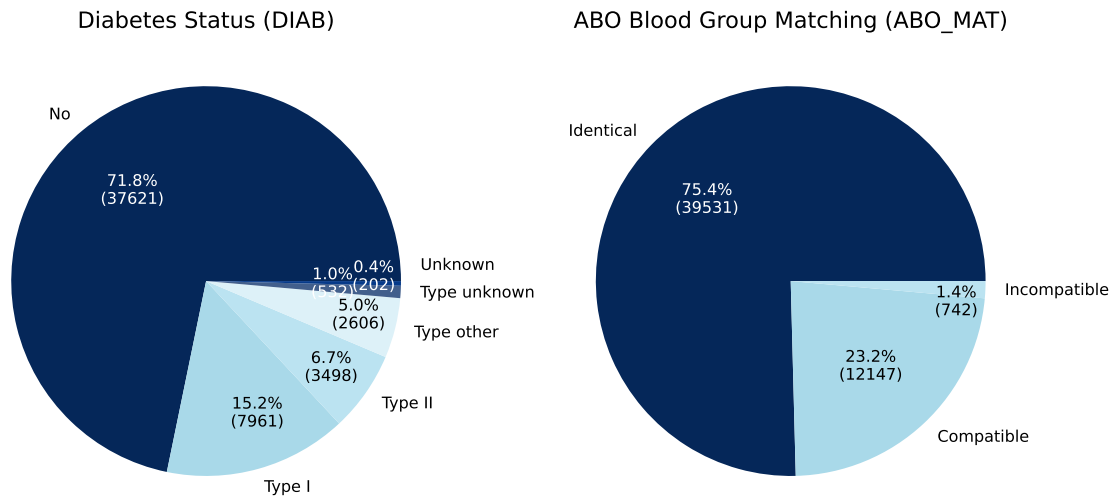


Figure 3.11: Pie Chart of Diabetes Status and ABO Match in Living Donor Subgroup

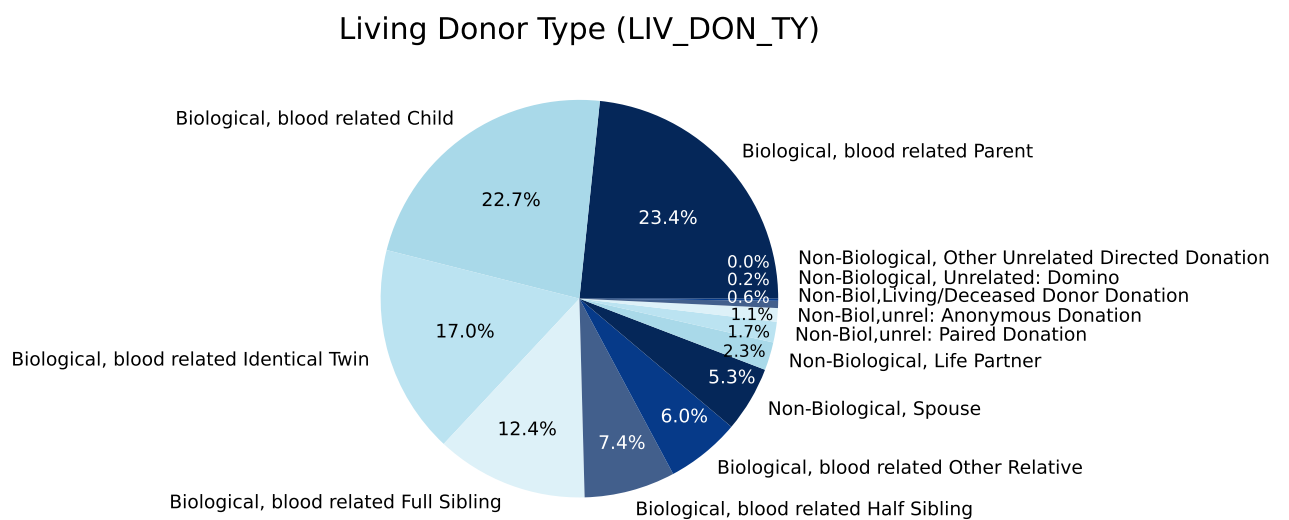


Figure 3.12: Pie Chart for Living Donor Type for the Living Donor Subset

3.5 Dataset Building, Exclusion Criteria and Noise Reduction

Exclusion Criteria Certain exclusion criteria were applied to the datasets to ensure better performance. Individuals under the age of 18 or those deceased from unrelated causes were removed. While censoring such deceased patients may have been an option, this was deemed unnecessary. Additionally, records with categorical values such as "Unknown" or "Not Done" were removed as they added noise to the models and hindered performance. However, these values were still used for training application models.

Handling of missing values When handling missing values, it was decided to remove entire rows rather than use mean imputation, which introduced bias to the model. Fortunately, due to the abundance of available data, this approach did not negatively impact the results.

3.6 Feature Engineering

We attempted to do feature engineering with donor-recipient height, weight, BMI, and age differences, as was done in [47]. However, these features had an adverse effect on performance, leading us to exclude them. The engineered feature that actually improved performance was the number of days the patient was on dialysis (DIALYSIS_TIME). The inspiration was drawn from [48, 61].

Chapter 4

Machine Learning Models

4.1 Problem Formulation

Predicting the survival time after a successful kidney transplant can be approached in three ways: as a regression problem, classification problem, or survival analysis.

A *regression* model may seem an intuitive choice, as we want to predict a numerical value – the survival time. However, it is not the best option for the following reasons:

1. **The censored dataset.** The dataset has a high level of censoring – 76%. Including living and deceased patients would introduce too much noise to the model, making it highly inaccurate because the survival time is unknown for the living recipients.

2. **Censoring removal would produce bias and significantly reduce the dataset.** We could remove all censored instances, but that would significantly reduce the dataset and introduce significant bias, as the dataset would contain only deceased patients, and most of them passed away before the introduction of modern techniques for treating rejection. Additionally, the more recent transplants are all unsuccessful, that would contribute to the bias. As a result, the model trained on such a dataset would be highly inaccurate.

3. **Regression predicts only one single number.** It poses a problem, especially over extended time frames, as there are too many factors that we cannot account for, leading to incorrect predictions.

Another way of formulating the problem is classification. We can theoretically divide the dataset into groups: "less than one year", "one to five years", "five and more," or even more groups and train a classifier based on them, as Naqvi et al. [47] did. Again, we would face the problems of censoring and bias mentioned above, so classification is not the best option.

A more appropriate way of problem formulation is survival analysis. Survival analysis methods handle censoring and provide a better form of prediction: the survival function or hazard function, which represents survival probability or the failure rate at each moment in time, respectively. This formulation is superior to others because it acknowledges the inherent uncertainty in predicting life expectancy. Instead of providing overly specific predictions, survival analysis allows for a probabilistic approach that accommodates natural variability. By focusing on survival probability or hazard, we can make more meaningful and realistic predictions.

4.2 Model Selection and Training Approach

In our study, we used the scikit-survival library algorithms to train our models. When choosing the models we would fully train and fine-tune in the end, we tried most of the models that scikit survival offers on a small subset of data and chose the ones that perform best while taking adequate time. We

had high hopes for SVM-based survival models, but despite them demonstrating superior performance on the small subset of data, we decided against using them due to their inherent inability to handle large datasets, which results in extended training times and frequent crashes when attempting to train on the slightly larger than 1000 samples subset, not even mentioning the entire training set.

After rigorous experimentation, we ultimately selected `CoxnetSurvivalAnalysis` (Cox Elastic Net, later referred to as Coxnet) [54], Random Survival Forest (RSF) [55], and Gradient Boosted Survival Analysis (GBSA) [56] as our preferred models. The latter two are ensemble models widely recognized for their high accuracy and robustness despite prolonged training time. The former demonstrated a notably low training time but worse results than the ensemble models. It allowed us to experiment with less time invested than the other models. Further research led us to discover an approach to training the Coxnet in the scikit survival documentation [25], which offered comparable performance to other models, requiring only a tiny fraction of the training time and computational resources. This showed that the Cox Elastic Net is more than a viable alternative to powerful and robust ensemble models, especially when time and computational resources are limited.

However, we encountered some challenges due to the relatively large dataset, even after all exclusion criteria and the removal of NaN values. RSF and GBSA demonstrated a prolonged training time compared to Coxnet (hours instead of minutes), and RSF demonstrated high computational demands. The latter resulted in frequent kernel crashes in our Jupyter notebooks after long hours of training due to the RAM shortage (on machine with 384 GB RAM). To mitigate these issues, we divided our dataset based on living and deceased donors and reduced the number of features used. These subsets are later referred to by the name of the type of transplantation, namely the deceased donor transplantation (DDT) and the living donor transplantation (LDT). The models were trained for each subset separately. To reduce the number of features, we trained models on smaller subsets of data and removed ones that showed negative or zero feature importance. This approach allowed us to account for features unique to a particular subset (KDPI for deceased and LIV_DON_TY for living) and to tailor our models to a specific type of transplantation. Sizes of the resulting datasets are as follows: 53,082 instances in living (42,465 in training) and 117,536 instances in deceased (94,028 training). By training separate models for each subset, we were able to develop potentially more accurate and effective models.

After the removal of missing values, the combined datasets contain 170,618 instances, which is a significant reduction from the initial dataset of nearly 500,000 samples. Despite this reduction, the remaining samples are likely representative of the omitted data, and the dataset size remains sufficient for effective model training.

4.3 Results

In this study, six models were fine-tuned and trained, three for each subset. We utilized the Cox Elastic Net, Random Survival Forest, and GradientBoostedSurvivalAnalysis models from the scikit-survival library. Two models were trained locally (Coxnet and GBSA) on Apple MacBook Air 16GB RAM, and one was trained in a university cluster (RSF). The preferred model in this study is Coxnet due to its impressive performance and short training time of 30 to 180 seconds compared to other models that can take several hours to train. All notebooks can be found in this project's GitHub repository in Code folder.

As numerical performance measures, the Uno concordance index (c-index), integrated Brier score (IBS), and the mean cumulative/dynamic area under curve (AUC) were chosen. The performance of trained models for living and deceased datasets can be observed in Tables 4.1 and 4.2. As can be seen, the DDT models perform poorer than the living ones. This is likely related to the inherent donor-recipient compatibility limitations within the DDT dataset and the presence of older transplants performed with less refined techniques. These dataset shortcomings are reflected in the models. Moreover, models

Model	Uno c-index	IBS	Mean AUC
Coxnet	0.723	0.136	0.743
Gradient Boosting	0.722	0.136	0.742
Random Survival Forest	0.723	0.139	0.744
Kaplan-Meier (for IBS reference)	-	0.247	-

Table 4.1: Model Comparison for LDT Models

Model	Uno c-index	IBS	Mean AUC
Coxnet	0.695	0.163	0.729
Gradient Boosting	0.699	0.162	0.734
Random Survival Forest	0.691	0.164	0.724
Kaplan-Meier (for IBS reference)	-	0.247	-

Table 4.2: Model Comparison for DDT Models

have more or less the same performance for the same dataset, which might mean that models capture data well, and the only bottleneck is the dataset. Consequently, the only potential way to improve the performance is to expand the dataset with further feature engineering. For more detailed information on the training of each model, refer to the corresponding section from the following sections.

For LDT, RSF shows slightly superior discrimination (AUC), but the overall prediction accuracy (BS) is inferior to Coxnet and GBSA. For the deceased, GBSA is the best model across all metrics. Due to its low training time, Coxnet is the preferred model for both LDT and DDT. However, if training time is not a concern, GBSA may be preferred for the DDT cohort due to its slightly higher discriminative ability.

As time-dependent metrics, ROC AUC and Brier Score were chosen. Figure 4.1 plots AUC over time for all living models. Even though the mean AUCs are almost identical, values vary over time. From day 1000 to 2800, RSF performs better, then from day 2900 to day 4000, coxnet and GBSA overperform RSF. All models performed similarly from day 4000 to 5000 except for a slight rise in coxnet and GBSA, which later converged with RSF. Day 5000 to 6000 RSF outperforms the two models again. RSF outperforms Coxnet and GBSA on 3/5 of the timeline, underperforms on 1/5, and is mostly similar on 1/5.

In Figure 4.2, we can see time-dependent Brier scores for all living models. Coxnet and GBSA are identical here despite having different learners. We had to adjust the visualization settings for the other to be seen. The RSF curve is slightly higher than the two, meaning overall performance is slightly worse despite mostly superior discrimination in the previous graph.

In Figure 4.3, we can see the AUC graph for the deceased models. GBSA shows superior discriminative ability, Coxnet is in the middle, and RSF shows the worst performance. Although all similarly rise over time.

Figure 4.4 plots the Brier Score for the deceased models. Similarly to LDT, the performance of Coxnet and GBSA is identical, but RSF is slightly worse, although by a small margin.

4.3.1 Coxnet

Coxnet, also known as a *Cox elastic net* or *Cox regularized regression*, is a linear model that efficiently handles large datasets and performs well.

As Tables 4.1 and 4.2 show, Coxnet performed better for the living group and worse for the deceased.

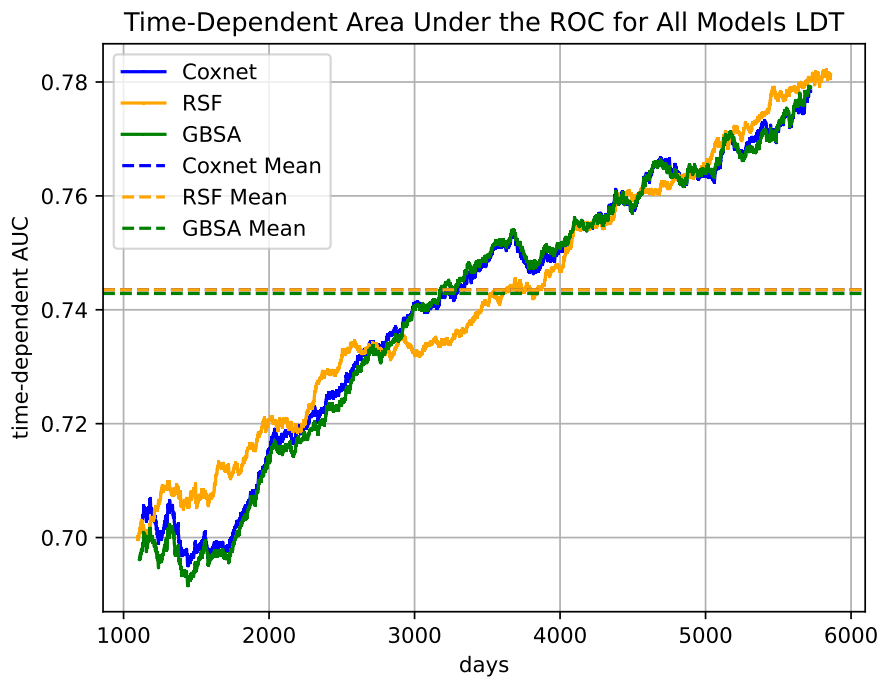


Figure 4.1: Time-Dependent AUC for LDT models

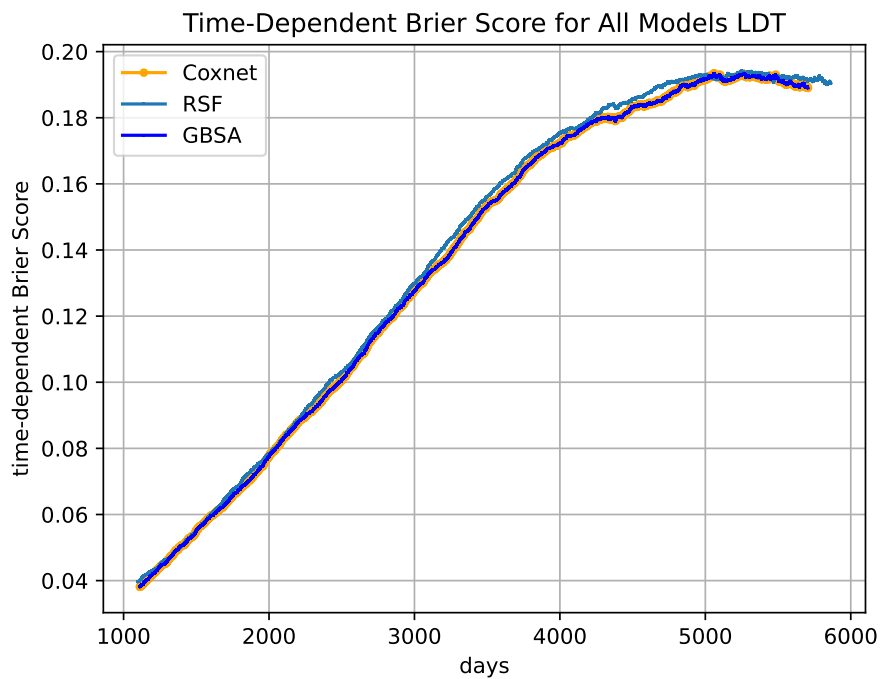


Figure 4.2: Time-dependent Brier Score for LDT models

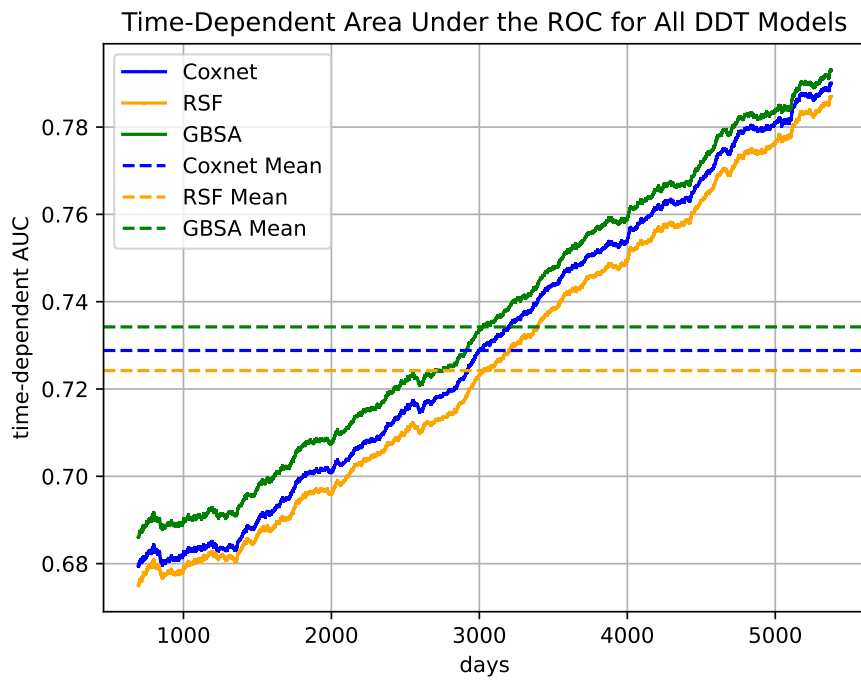


Figure 4.3: Time-Dependent AUC For DDT

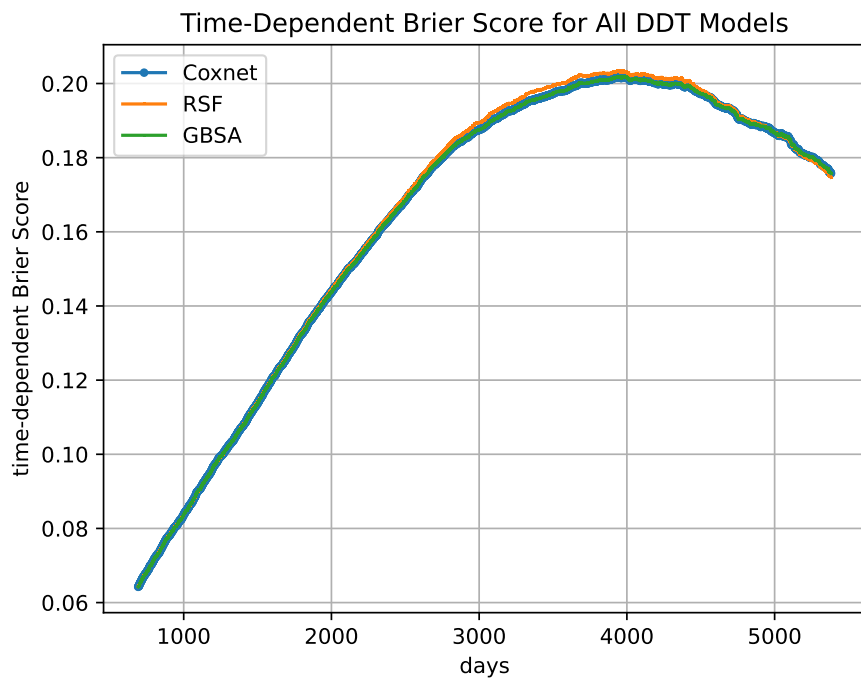


Figure 4.4: Time-Dependent Brier Score For DDT models

Hyperparameters The coxnet has only two hyperparameters: L_1 ratio and α . The L_1 ratio defines the relative weight of the l_1 and l_2 penalty. We have found that the choice of L_1 ratio did not significantly impact the c-index, IBS, or mean AUC values. At both high (0.9) and low (<0.1) values of the L_1 ratio, those values were about the same. However, a higher value of L_1 ratio (0.9) was selected for stability purposes, as suggested by the documentation of the scikit survival library [25].

To find the best α , we used the approach from the documentation [25]. In short, it is calculated by fitting the model and choosing α that generalizes best from the resulting list *CoxnetSurvivalAnalysis._alphas*. As was covered in 2.4, the hyperparameter tuning is usually performed with either GridSearch or RandomizedSearch. As there are few alphas, we used GridSearchCV from scikit-learn to find the best one. The best α for the DDT was 0.000772369005962216, while 0.00021510636871239242 was the best for LDT.

Feature importance The importance of more than 60 features was examined with the permutation importance method. Only a fraction of them had any importance. Those of negative and zero importance were removed, increasing model performance. Interestingly, HLA had little influence on long-term survival, similar to what Terasaki noted in Section 1.3. So, they were removed, and the model performance improved. The feature importances are visualized in Figures 4.5 and 4.6.

As we can see, the most predictive features are the patient age, patient diabetes status, dialysis, and recipient ethnicity.

Feature Engineering We attempted to do feature engineering with donor-recipient height, weight, BMI, and age differences inspired by [47]. The introduction of these features worsened the performance, so they were excluded. The engineered feature that improved the performance is the number of days the patient was on dialysis (DIALYSIS_TIME), inspired by [48].

Data Pipelines Coxnet models are the only two models used in KidneyLife (they are the most lightweight and fast in prediction of all trained). That necessitates a data input pipeline to provide the data the model expects. Fortunately, it is possible to pickle scikit-learn's transformer pipelines used in training and use them later to process the input values. However, they have a downside of not preserving the column names (at least we were not able to preserve them), making the feature importance analysis impossible. Because of that, we use two data pipelines for different use cases. The second pipeline is suggested in the scikit-survival documentation approach: directly using the scikit-survival's functions *encode_categorical()* and *encode_numerical()*.

4.3.2 Random Survival Forest

The *Random Survival Forest* is a robust ensemble machine-learning algorithm with Survival Trees as a base learned (discussed in more detail in Section 2.5.4.3). However, its training time is longer than Coxnet, and it can be challenging to work with due to its high computational requirements (depending on the number of learners), especially during hyperparameter tuning. Moreover, RSF is known to be highly memory-hungry during training and prediction, making it unsuitable for use in the application.

On the positive side, RSF can learn well from smaller datasets. Although Coxnet performs well on large datasets, it does worse than RSF on smaller ones. It was observed during hyperparameter tuning, where we used a subset of 1000 transplantations from the training dataset. RSF fitted them well, and when evaluated on the whole test set, the performance was comparable to that of the entire dataset. In fact, the difference in performance between the subset and the whole dataset was minimal,

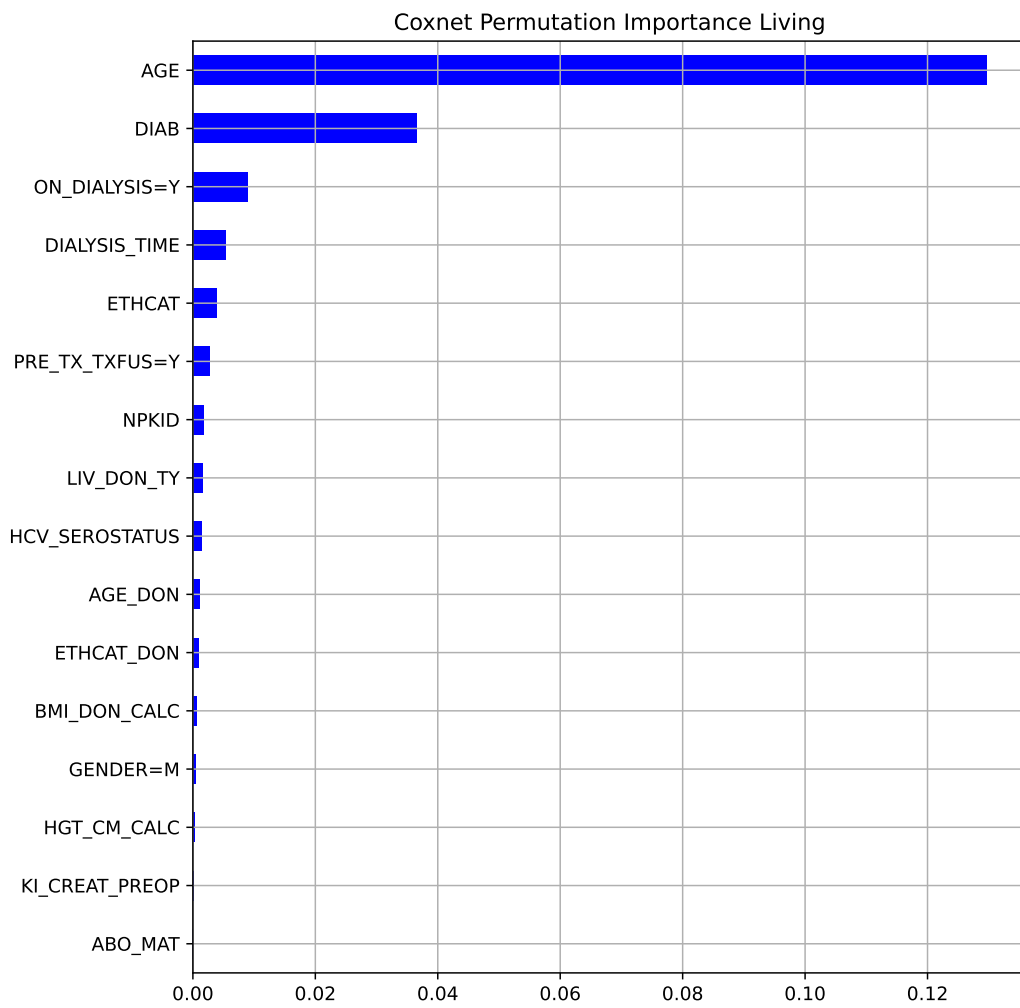


Figure 4.5: Feature Permutation Importance for Coxnet Living

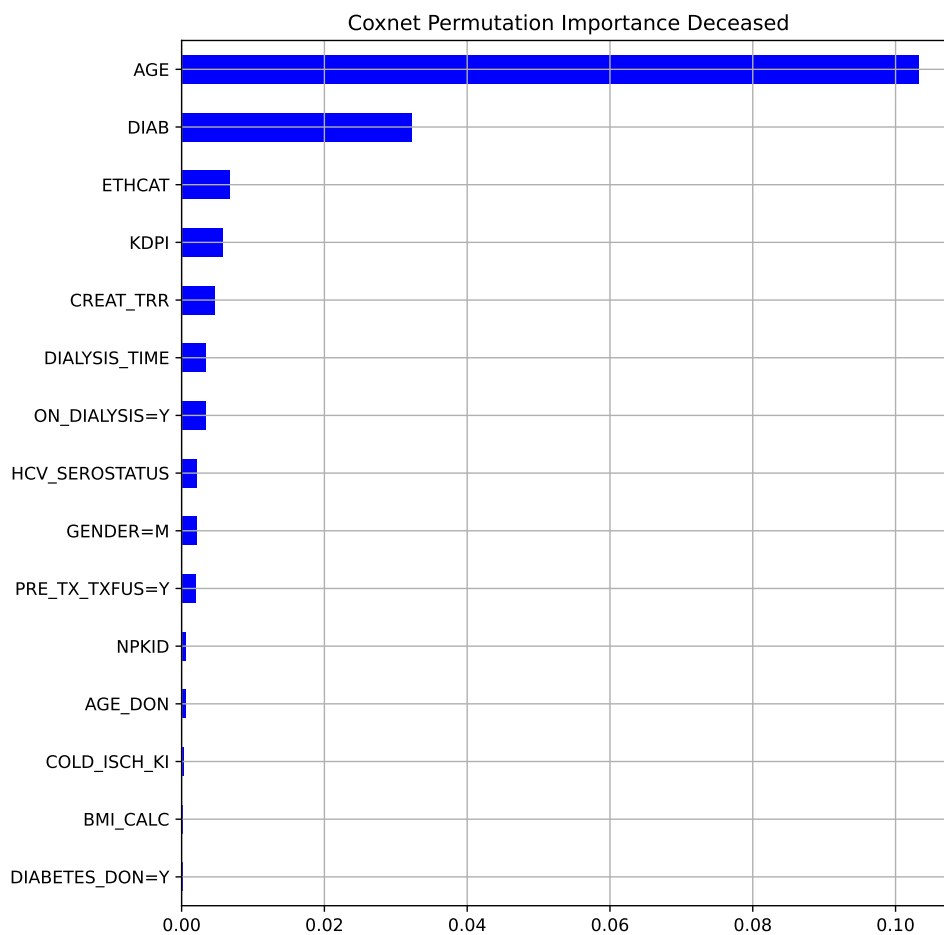


Figure 4.6: Feature Permutation Importance for Coxnet Deceased

Hyperparameter	Values Tried	Chosen Value
Number of Estimators	1, 5, 10, 30, 50, 70, 80	50
Max Depth	2, 4, 6, 8, 10, 12, 14	12
Min Samples Split	2, 4, 6, 8, 10, 12, 14, 16, 18	16

Table 4.3: RSF hyperparameters

as evidenced by the integrated Brier score (IBS) of 0.13906552 (1000 instances) compared to 0.139001 (whole dataset).

Tables 4.1 and 4.2 present RSF's performance. Similarly to Coxnet and GBSA, it performs worse for the deceased than for the living. However, overall, the performance is quite similar between RSF and the other two methods.

Hyperparameters The hyperparameters were finetuned with a script that mimicked GridSearchCV without cross-validation for fine-tuning. The table 4.3 shows the fine-tuned hyperparameters, as well as the values that were tested and ultimately selected. After experimenting with higher values (80), we kept the number of estimators at 50 since we did not observe any noticeable improvement beyond that point, and the training time increased significantly from 60 to 101 minutes. Nonetheless, it is generally accepted that the greater the number of estimators, the better the results.

Feature Importance Figures 4.7 and 4.8 illustrate the feature importances. The feature selection process can be found at "FeatureSelection/rsf_living.ipynb" and "FeatureSelection/rsf_living.ipynb" (calculated in cluster file deceased. ipynb). For the living, the most important ones are, as before, recipient age, diabetes status, dialysis status, recipient ethnicity, and dialysis time. For the deceased, recipient age, recipient diabetes status and recipient pretransplant serum creatinine are top three features.

Training Time The LDT trained for 65 minutes, and the DDT for about 80 minutes. Both are on the 50 estimators. Upping the estimators is a potential avenue for improvement, but one would need to be careful not to overfit the model. Moreover, the training would take a long time: 200 estimators would require 4 hours of training, and one might run out of RAM. Overfitting can be avoided with regularization, but it is unlikely to surpass current performance by a large margin, as we already have reached values observed in the literature (discussed in more detail in Section 4.5).

4.3.3 Gradient Boosting Survival Analysis

GradientBoostedSurvivalAnalysis is a model with gradient-boosted Cox proportional hazard loss with regression trees as the base learner, as discussed in Section 2.5.4.2. It is a relatively slow algorithm, as it can only run on one CPU core and trains only one estimator at a time. The training time might be considerable as we cannot fully utilize the cluster computing power. However, that can also be advantageous, as we can train the model on a PC. Additionally, the algorithm provides valuable insights during training, such as the predicted training time and loss and out-of-bag improvement for each estimator. This information is logged regularly throughout the training process, approximating the model's accuracy well at that point of training.

Tables 4.1 and 4.2 show the performance of two GBSA models for living and deceased subsets. As with the two previous models, the living model performs better than the deceased model. It is the best among the DDT models with c_index 0.699 and IBS 0.162.

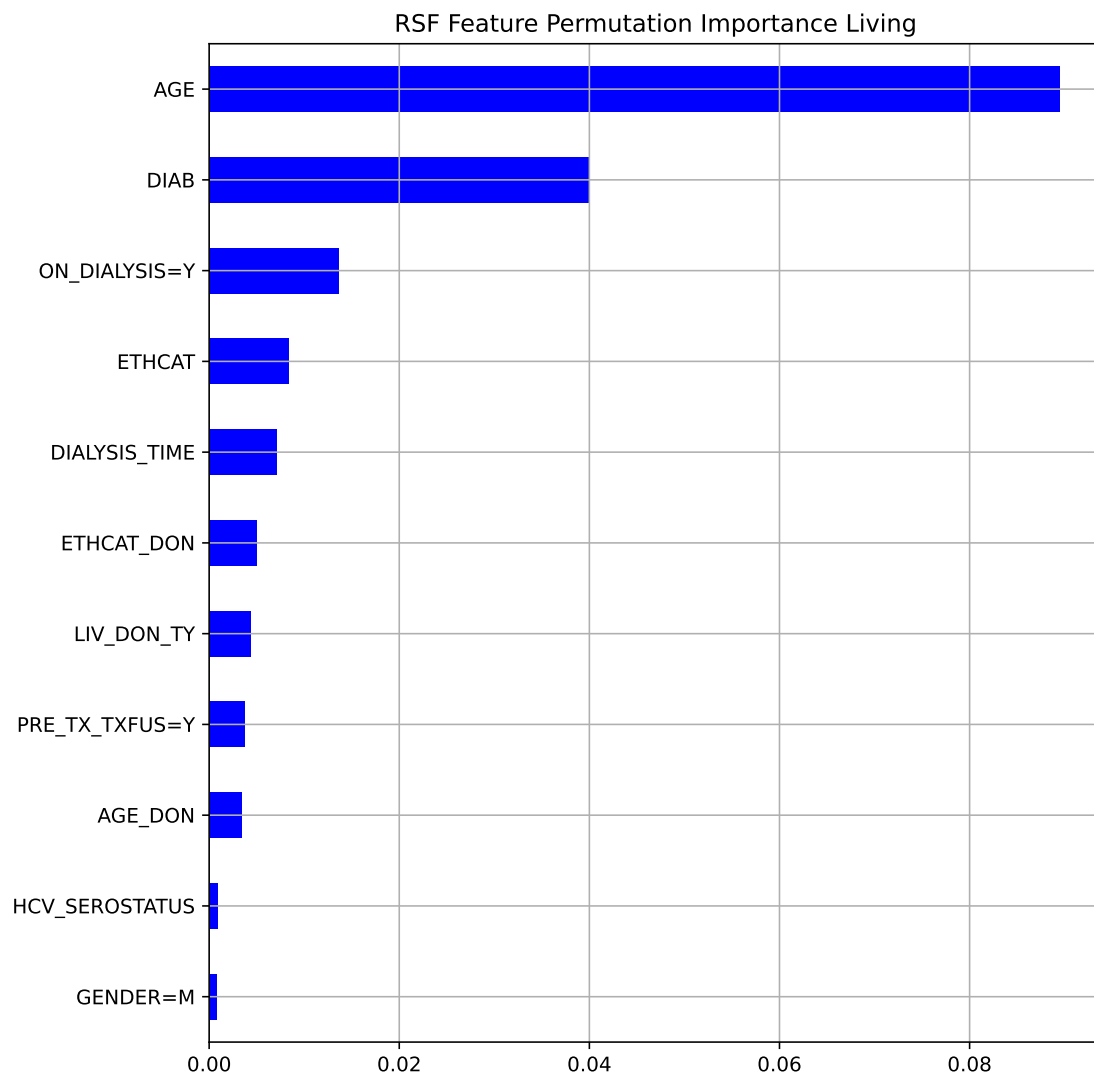


Figure 4.7: Feature Importance for RSF LDT

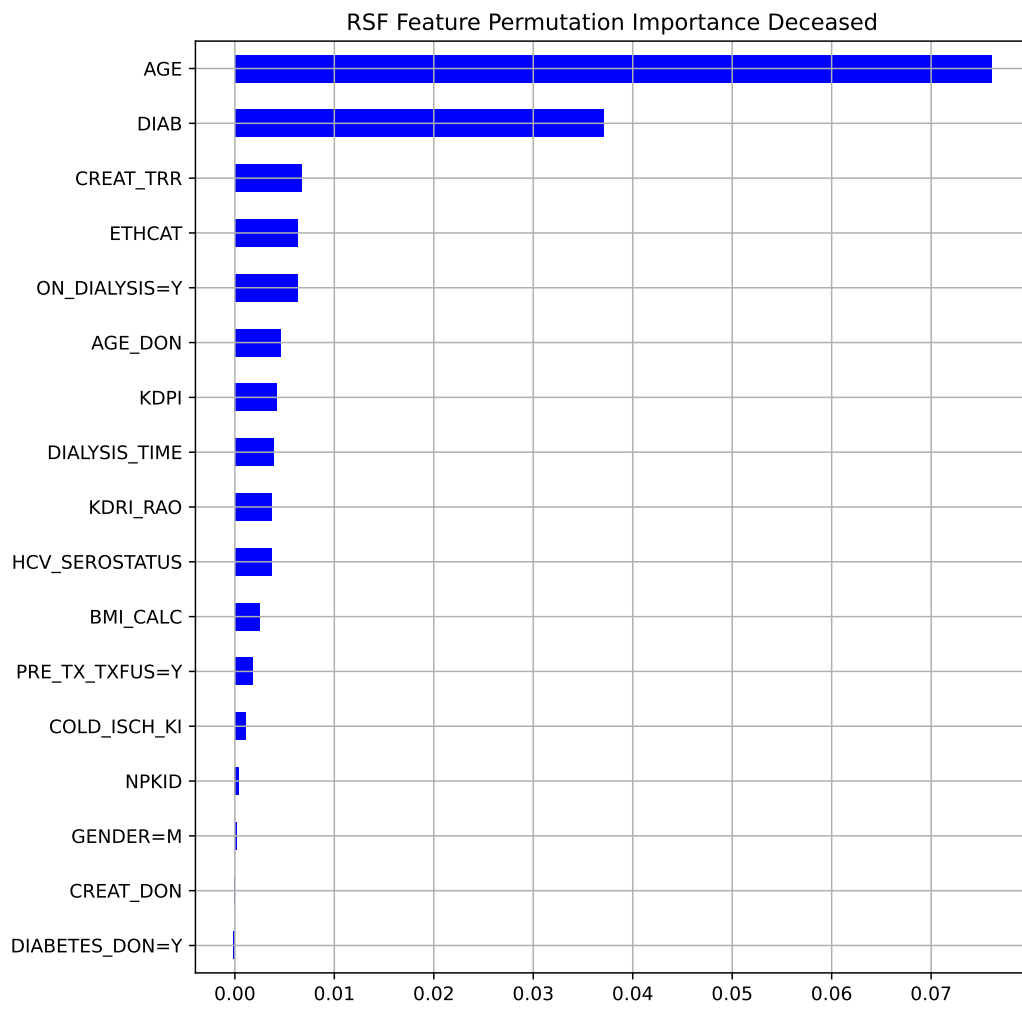


Figure 4.8: RSF Feature Importance DDT

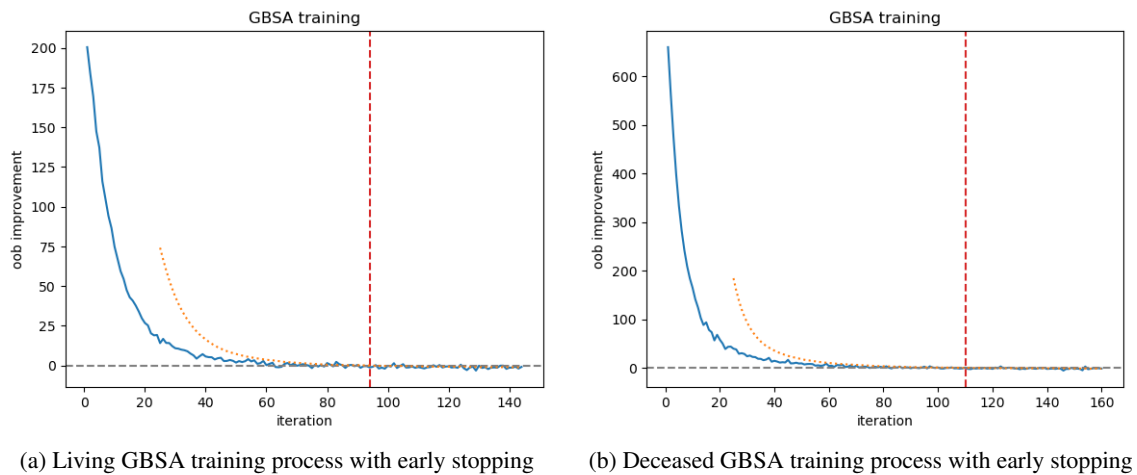


Figure 4.9: GBSA training process with early stopping

Hyperparameters We used three primary hyperparameters: the number of estimators, learning rate, and `max_depth`. Grid search estimated the best `max_depth` to be 6. However, experimentally, we found that 4 gives better results. Through trial and error, we selected a learning rate of 0.2, which resulted in the best outcomes. With lower values (0.05, 0.1, 0.15), the model was underfitted, with higher (0.5, 1) overfitted. The number of estimators was determined using an early-stopping monitor, a widely used approach for GB algorithms. The monitor analyzes the average improvement of the last 25 iterations and stops training if negative for the last 50. The monitor implementation and further explanation can be found in the documentation [11]. The optimal number of estimators for the deceased was 110, and for the living, it was 94. Figures 4.9a and 4.9b show the plot of OOB improvement per learner, with the red dashed line representing the cutoff point at which performance begins to decline.

Training Time The models were trained locally, and the training time was 75 minutes for the living and 60 minutes for the deceased. Using a lower learning rate resulted in a significantly longer training time (396 minutes for living at a 0.05 learning rate).

Feature Selection Initially, we performed feature selection on models with a small number of learners, and it produced an interesting result: only a small selection of features was deemed important. The result did not seem accurate. Since we already had several models trained with many more learners, we decided to assess their feature importance. Subsequently, we discovered that many features initially deemed irrelevant were, in fact, relevant. Therefore, this feature selection approach is unsuitable for GBSA, and feature importance must be evaluated on a fully trained model. For instance, RSF is unaffected by this issue, as feature importance remains more or less consistent for both trained and under-trained models. This makes sense, as the model is improving with each trained estimator (the algorithm is described in Sections 2.1.5 and 2.5.4.2), contrary to RSF, where it can perform with a relatively low quantity of learners because each learner is an individual estimator.

Feature Importance Figures 4.10 and 4.11 display the estimated feature importances for living and deceased GBSA using permutation importance. Notably, in the living GBSA, a few features such as `PRE_TX_TXFUS`, `KI_KREAT_PREOP`, `HCV_SEROSTATUS`, and `GENDER` have negative importance. However, their removal only results in a minor performance decrease from 0.722 to 0.720.

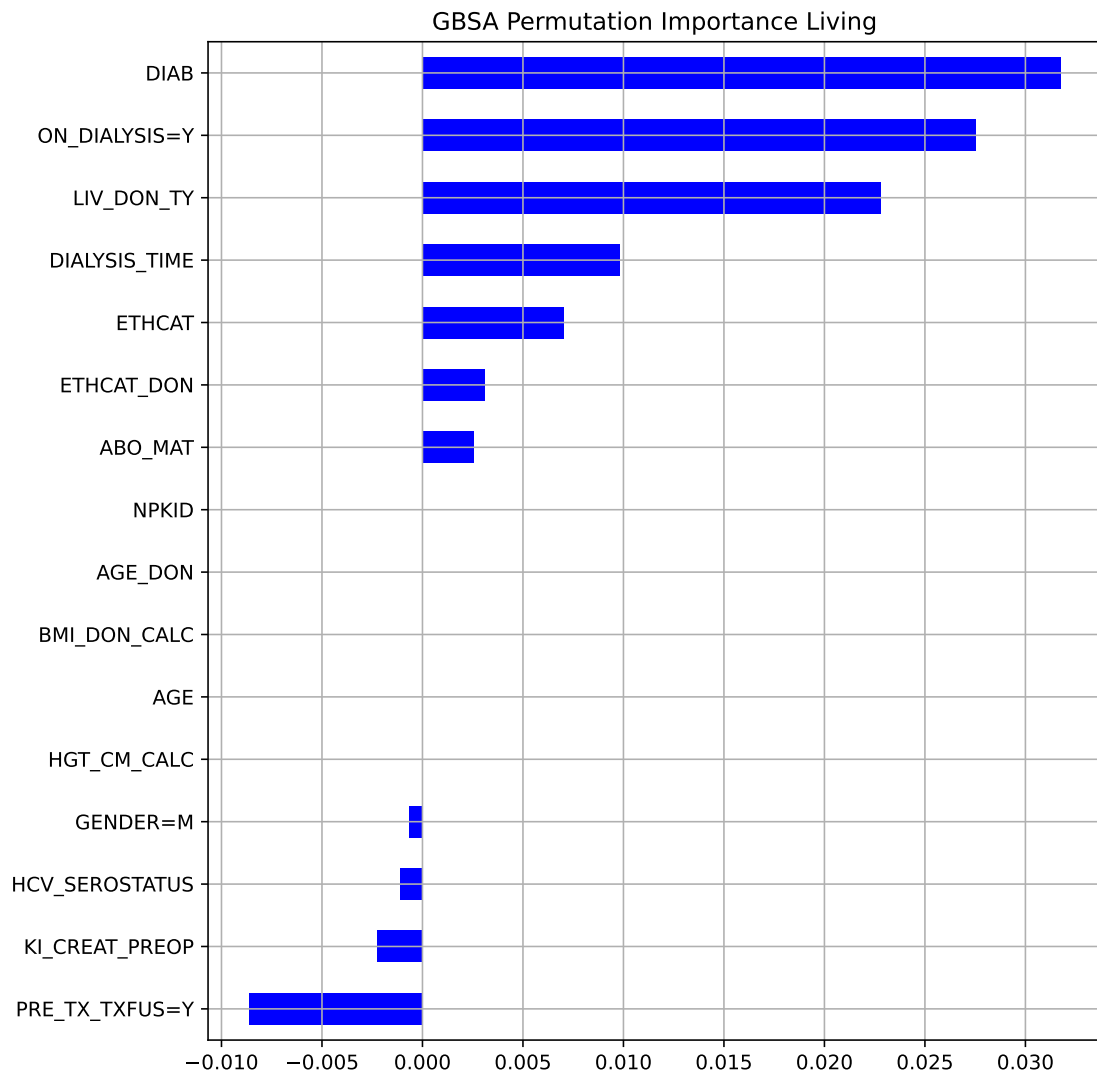


Figure 4.10: GBSA Feature Importance LDT

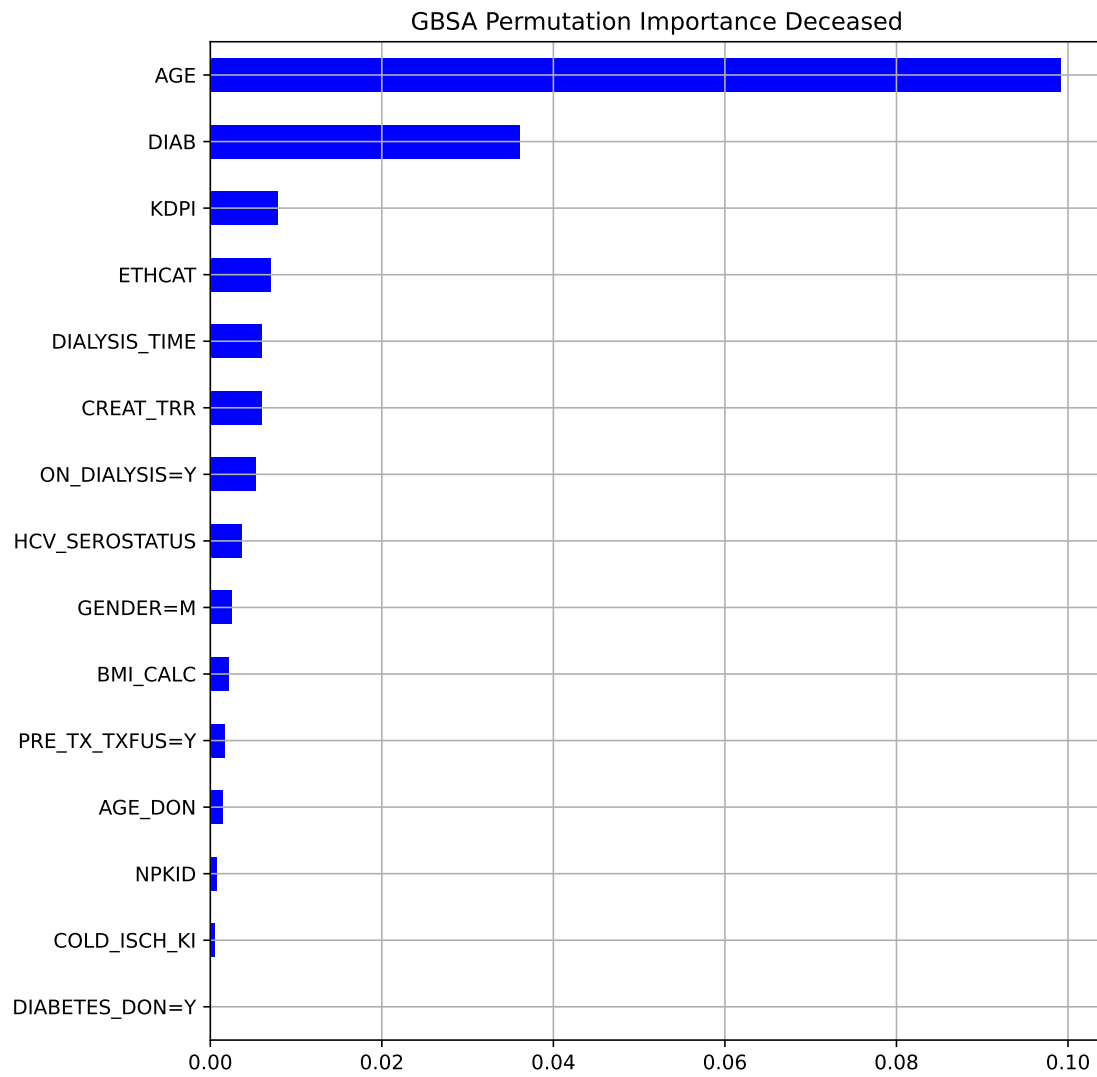


Figure 4.11: GBSA Feature Importance DDT

4.4 Scoring

Various methods have been proposed to evaluate donor-recipient compatibility, such as the Estimated Post-Transplant Survival Score (EPTS) score [61], which is presently utilized in the US kidney allocation system and is based on survival analysis techniques. (Mark et al., 2019) [64] mentioned other scoring systems, such as Life Years from Transplant (LYFT) [63] and Recipient Risk Score (RRS) [62].

Our bachelor project task included the development of a new scoring technique for TX Matching. Even though we diverged from integration with it, we still thought developing a scoring technique was worthwhile. TX Matching (discussed in Subsection 5.1.1) uses the following formula for scoring:

$$\begin{aligned} \text{score} = & 1 * (\text{number of A and B antigen matches}) \\ & + 10 (\text{number of DR antigen matches}) \\ & + (\text{bonus for compatible blood group}) * (\text{indicator of compatible blood group}) \end{aligned}$$

It focuses on histo- and blood group compatibility but does not consider other factors, such as potential life expectancy. We wanted to address this.

We aimed to develop a scoring technique based on the hazard predicted by the GBSA model. Our primary assumption was that patients who lived the longest would have the lowest hazard. Based on this hazard, we would create a formula that would produce a score ranging from 0 to 100, with 0 being the worst possible match and 100 being the best possible. So, let us look at the predicted hazard for each time group.

Figure 4.12 illustrates the mean predicted hazard for four groups: "less than 1 year", "1-5 years", "5-10 years", and "more than 10 years". As can be seen, there is an expected trend of decreasing predicted hazards. However, let us look closer. Figure 4.13 illustrates the mean predicted hazard for each year and patients, with patients who have survived more than ten years categorized into a separate group. As can be seen, in the group "less than 1 year", the hazard is expectedly high. The "2 years" and "3 years" groups are similar and have lower hazards than the first group. Then, however, in "4 years" it starts to increase, culminating in "5 years". Only then does the mean hazard start declining, as expected.

Based on this, we can speculate that the score might be somewhat accurate only for patients with lower predicted hazards, who are likely to live for more than five years. Even though there is an expected trend, these are mean values. The actual predictions have quite a significant standard deviation (0.86) in all groups, suggesting that the predicted hazard cannot be the sole deciding factor. As discussed at the beginning of this Chapter (Section 4.1), it seems that it is indeed hard to reduce such a complex topic as survival to a single number. The survival, hazard, and cumulative hazard functions should be assessed alongside other factors to make a well-informed decision.

The Scoring As part of our task to create a scoring, we are presenting the following scoring formula:

$$\text{score} = 100 * \left(\frac{1}{2} C_{\text{hazard}} + \frac{1}{2} C_{\text{age}} \right), \quad (4.1)$$

where C_{hazard} is the hazard coefficient, and C_{age} is the age coefficient. $\frac{1}{2}$ is the weight of the coefficient. The weight of the coefficient for n features is calculated as $\frac{1}{n+1}$, with 1 accounting for the hazard. C_{hazard} is denoted by

$$C_{\text{hazard}} = 1 - \frac{H_{\text{predicted}} - H_{\text{min}}}{H_{\text{max}} - H_{\text{min}}},$$

where H is a hazard. The fraction is the normalization equation, seen in Equation 2.10, subtracted from one to reflect the fact that the lower the hazard the better. The normalization formula was chosen as it

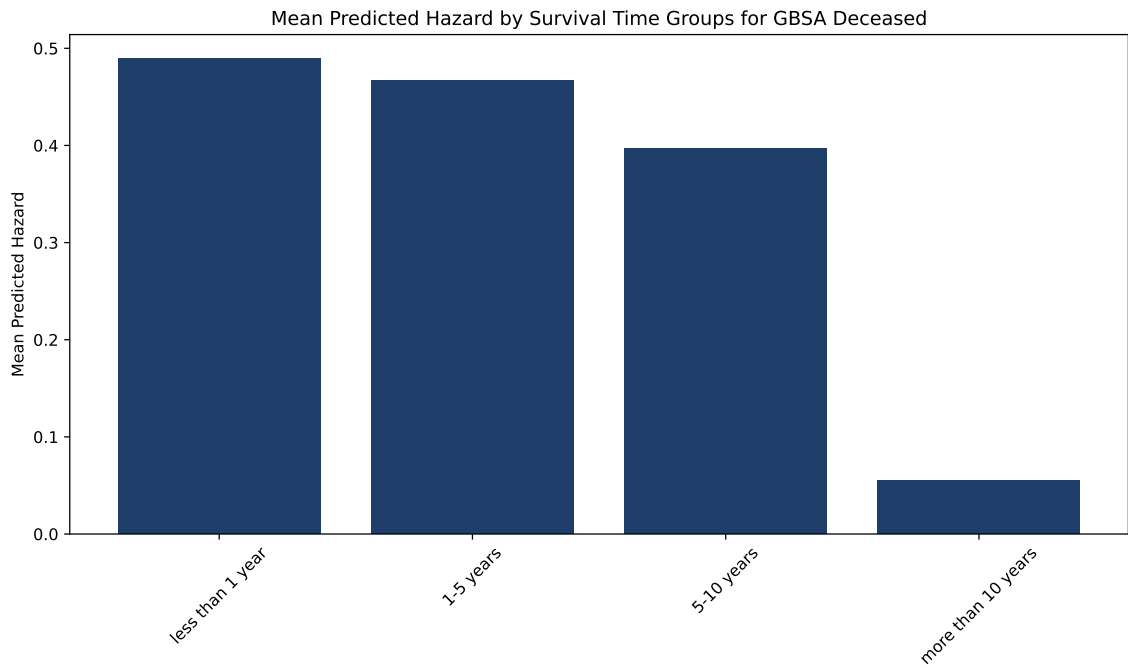


Figure 4.12: Mean Predicted Hazard by Survival Time Groups for GBSA Deceased

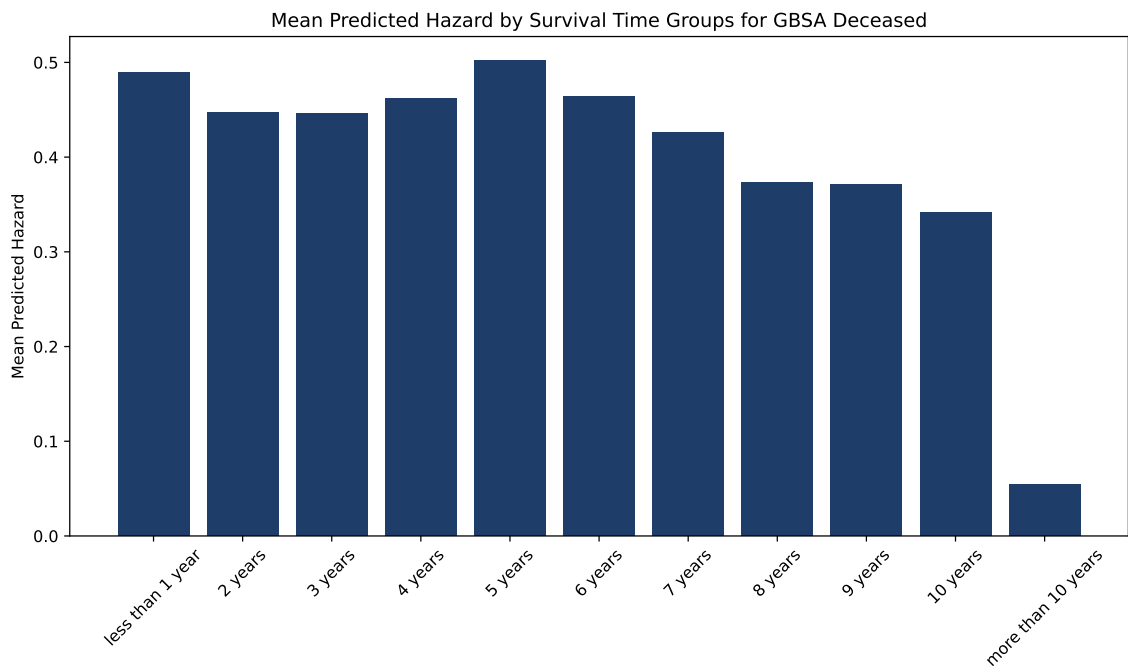


Figure 4.13: Mean Predicted Hazard by Detailed Survival Time Groups for GBSA Deceased

reflects the statistical properties of the feature, while keeping it in range from 0 to 1, resulting in a good coefficient.

C_{age} is given by

$$C_{age} = \begin{cases} 0 & \text{if } age \geq 80 \\ 1 - \frac{age-18}{80-18} & \text{otherwise} \end{cases},$$

where 80 is an arbitrarily chosen marginal age, after which the transplantations are not performed. In real-life situations, it could and should differ. 18 is the minimum age because pediatric transplantation is a unique field on its own. The reason for subtracting the normalization fraction from 1 is the same as previously - the younger the age, the better the outcome.

We have focused on the recipient age feature, one of the most predictive features in all models except GBSA Living. However, additional features can and should be included to improve scoring accuracy. We suggest incorporating the following features into the formula: dialysis status and/or dialysis time and recipient diabetes status. These features had a high importance in all our models. By doing this, one would be following the approach used in the EPTS. In EPTS, factors such as years on dialysis, diabetes, prior transplants, and age were included. Although the hazard prediction was not directly included in the formula, the equation was derived from Cox proportional hazards.

Table 4.4: Calculated Score for Given Predicted Hazard

Predicted Hazard	Score	Survival (Days)
-0.854124	67.079316	2113.0
0.203719	41.642521	5537.0
0.593991	32.258065	2513.0
-0.979759	70.100331	964.0
-0.344581	54.826901	4793.0
0.258669	40.321190	1428.0
0.058312	45.138960	5561.0
-1.074110	72.369084	2142.0
1.313032	14.968083	2.0
-1.266887	77.004577	2685.0

Examples Table 4.4 shows the score for each predicted hazard for ten random samples from the test set. As can be seen, the score caught a lousy transplantation with only two days survived by giving it a score of 14. However, for more successful transplantations, it is less accurate.

4.5 Discussion

Our goal was to create a model that could estimate the life expectancy of transplant recipients for subsequent use in our application. We developed six models: three for living donor transplantation (LDT) and three for deceased donor transplantation (DDT). Our models' performance was comparable to the state of the art, with the highest c-index of 0.699 achieved for DDT using GBSA and 0.723 for LDT using Coxnet over a 15-year period. However, we found that the DDT model performance was inferior to that of LDT, likely due to inherent limitations in donor-recipient compatibility present in the DDT dataset. Another reason might be related to the data itself. The DDT dataset might have more

of older records when the techniques of transplantation and immunosuppression were not that refined, which might negatively influence the prediction. The distributions of living/deceased transplantations by year on the whole dataset can be seen in Figure 3.5.

Mark et al. [49] achieved 0.724 Harrel’s c index over a 5-year period with RSF and CoxPH on the UNOS dataset. Paquette et al. [50] achieved c index of 0.65, 0.661, and 0.659 for the neural network–based models (DeepSurv, DeepHit, and RNN, respectively) on the SRTR dataset. Senanayake et al. [51] achieved a concordance index of 0.67 with a Random Survival Forest on the ANZDATA dataset. Other SA studies from the Ravindhran et al. [52] meta-analysis showed similar results of sub-0.70 c-index. Only one other study [53] surpassed it and achieved a 0.80 concordance index with survival trees, but with post-transplant 3-month serum creatinine and on a smaller dataset (3,117 instances).

While the importance of individual features varied between models, some were consistently deemed more important than others. Across all models, recipient diabetes, age, and dialysis status or time were consistently among the most important features. Surprisingly, HLA features only hindered the performance or were unimportant enough.

We designed and implemented a prototype of a scoring algorithm (Equation 4.1) focused on life expectancy. It is based on predicted hazard and the recipient’s age. We have found that it is not very accurate for people with successful transplants, but it can catch potentially unsuccessful ones. Refer to Section 4.4 for further details.

Limitations and Further Work The inherent models’ assumption is the compatibility of donor and recipient. Consequently, models should be used after the donor-recipient compatibility is confirmed. Otherwise, the predictions might be inaccurate.

We achieved good performance for the pre-transplant prediction, possibly close to the best possible. However, it might not be enough. Therefore, we suggest training additional models for post-transplant observation and follow-up to predict the lifespan of the transplant recipient with higher accuracy. Similar to [53], which trained a model with the post-transplant 3-month serum creatinine, but we advise using more features.

Our models achieved performance very close to each other for each transplantation type, which means that models fitted data reasonably well, and the performance is limited only by the data. We have observed minor improvements after introducing the engineered feature DIALYSIS_TIME, so one could try to engineer better features. For example, calculate eGFR from age, serum creatinine, and gender for both recipient and donor and see how it influences performance. This feature can be more informative than all those features combined and can have less noise.

Training and testing were performed on data from the same UNOS dataset. We do not know how models will perform on other datasets, so evaluating the model performance on another dataset would be beneficial. Ideally, even to train a model on data from multiple datasets.

Upping the estimators in random survival forest is a potential avenue for improvement, but one would need to be careful not to overfit the model. That would take a long time: 200 estimators would require 4 hours of training, and we might overfit the model or run out of RAM. Overfitting can be avoided with regularization, but it is unlikely to surpass the achieved performance by a large margin.

The scoring is not finished and needs more work. While it can catch really bad transplantations, it is less accurate with more successful ones (that do not experience hyperacute or acute rejection). We suggest adding more features to the formula or trying a different approach altogether.

Initially, we considered training a survival neural network. However, after reviewing the literature, we found that the models we had already used had provided similar or better performance than neural networks. Furthermore, neural networks perform best with a large amount of highly complex data, such as images or text. Our dataset is relatively simple, consisting of tabular data with no more than 40 features

after one-hot preprocessing. Therefore, the algorithms we use should be at least as effective. However, training a neural network is still a viable option for those willing to invest the time necessary to achieve better performance.

Chapter 5

Applications

This chapter focuses on the software used in kidney transplantation. We will start by exploring *kidney paired donation* (KPD), with its intricacies and necessary terminology. Then, we will examine three solutions for KPD. Finally, we will introduce KidneyLife - a survival estimation application to complement KPD software to aid medical professionals in decision-making.

5.1 Existing Solutions

To the best of our knowledge, there is no survival estimation software for kidney transplantation as of yet. So, let us review other adjacent solutions: kidney pair matching software. We will review three options: TX Matching, KidneyMatch, and KPDGUI. It is worth noting that there is limited information available online about alternatives. This shortage may stem from the fact that such software is developed for each transplantation program individually or from the preference of some developers to market their solutions directly rather than on the Internet. Nevertheless, some hospitals still rely on more traditional and manual matching techniques with no aid from specialized software.

When finding a compatible donor within the recipient's social circle is not feasible, the patient has an option to register in the *kidney paired donation* (KPD) program, also known as the *kidney exchange program* (KEP). KPD is an approach to living donor transplantation, where exchanges between pairs with incompatible donors are arranged in a way that allows each recipient to obtain a compatible kidney. While, theoretically, a large number of pairs can participate in the exchange, the higher the number of patients in the sequence of transplantations, the higher the risk of someone refusing to donate their kidney or being unable to due to a medical condition, which could compromise the entire sequence. So, the number of pairs in one sequence is usually limited to some number that makes sense for the center.

An example of how KPD works is as follows. Suppose A, B, and C are incompatible donor-recipient pairs, and a recipient is compatible with the donor from the next pair. Donor A donates their kidney to recipient B; donor B donates theirs to recipient C; and donor C donates theirs to recipient A. Such a sequence of transplantations is called a loop. An illustration of that can be seen in Figure 5.1a.

In some cases, altruistic or non-directed donors (NDD) are available. These donors are willing to donate a kidney to anyone compatible without being bound to a specific recipient. In such instances, a sequence of transplantations called a chain is formed, progressing forward from one recipient to the next, culminating in the recipient from a regular waiting list, as can be seen in Figure 5.1b.

Such software also performs so-called virtual cross-match. It is a digital simulation of a test that predicts hyperacute rejection (briefly explained in 1.3). However, a virtual cross-match test is not a substitution for a regular one and must be complemented by a physical test, as occasional disparities may arise between the two.

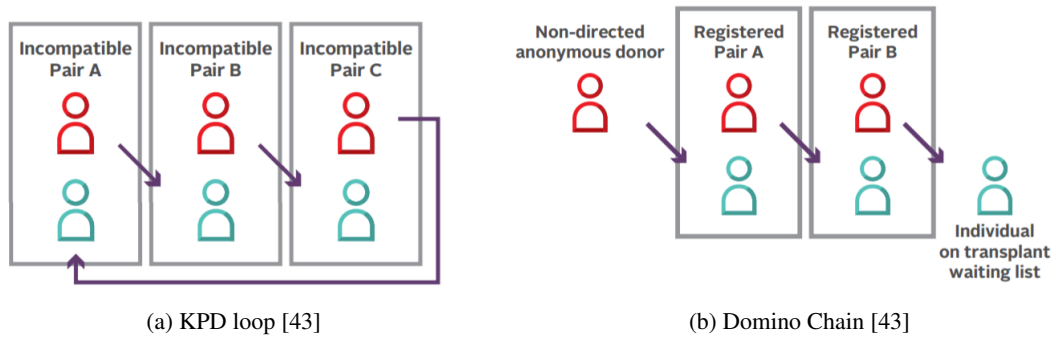


Figure 5.1

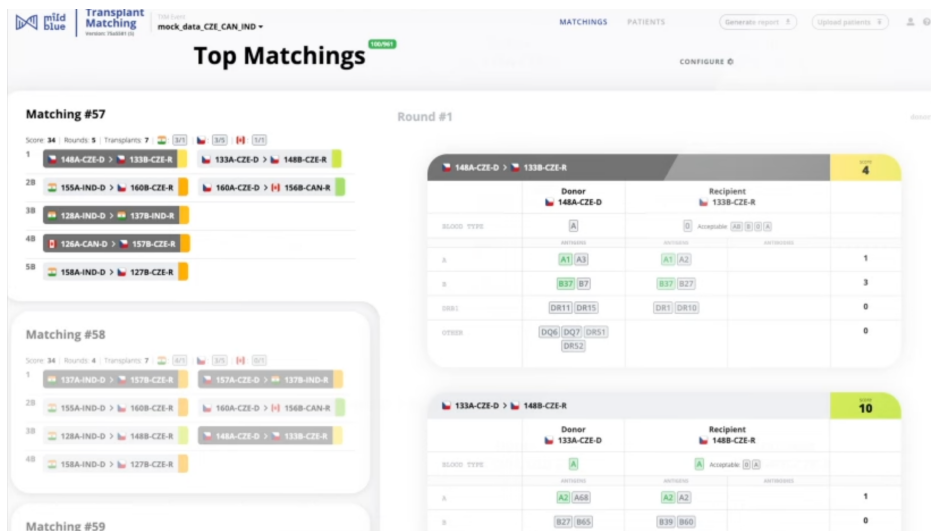


Figure 5.2: Txmatching

5.1.1 TX Matching

TX Matching is open-source software that allows KPD centers to find optimal matches from a pool of incompatible donor-recipient pairs and altruistic donors. It utilizes one of two algorithms (solvers): integer linear programming (ILP) solver and All Solution Solver to find the best possible loops and chains.

One of its core features is multi-national support that allows for collaboration in the organ exchange between the Czech Republic, Austria, and Israel. Additionally, it offers customizable pools of patients, referred to as "TXM events", allowing for the segregation of patients based on many factors, including logistic feasibility. This feature ensures that matches are not only biologically compatible but also feasible according to other factors.

Furthermore, TX Matching has a wide range of configurable settings, allowing users to adjust parameters to their needs and requirements, enhancing the relevance of solved matchings. Matching scoring mechanism includes evaluation of blood group and HLA compatibility. Although, TX Matching is used primarily by the Czech Institute of Clinical and Experimental Medicine, efforts have been made to expand to other transplantation centers, but unfortunately, they were unsuccessful so far.

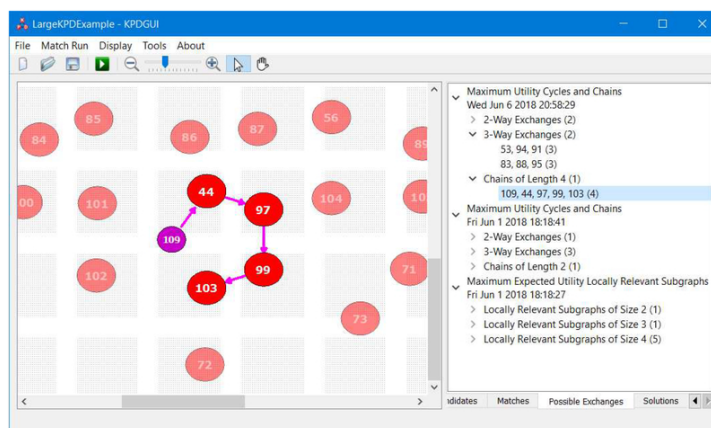


Figure 5.3: KPDGUI interface [44]. On the image you can see a chain.

5.1.2 KidneyMatch

KidneyMatch is the software developed and used by The Alliance for Paired Kidney Donation (APKD). They highly emphasize the security of their app: multifactor authentication, role-based authorization, and audit trails on every action in the software. KidneyMatch allows specifying the level of the match run: internal, regional, or national. The matches may be specified even further with various matching criteria, blood group, HLA, and discretionary exclusion criteria (DEC). They also allow specifying the expected solution: from a single match to chains and loops of different sizes.

To ensure the best possible matches, KidneyMatch collects more than 80 data points about each donor and more than 90 data points about each recipient. Such extensive data collection not only ensures good matching but also contributes to the refinement and development of matching methods. Furthermore, KidneyMatching supports additional medical data such as scans and charts, enhancing user experience and allowing for more well-informed decision-making [45].

5.1.3 KPDGUI

KPDGUI (Kidney Paired Donation Graphical User Interface) is an interactive open-source software designed for the optimization and visualization of KPD programs. It addresses the aforementioned problem of donor withdrawal from KPD by arranging fallback options in case someone indeed left KPD and the exchange cannot proceed. What distinguishes it from the rest is that the application provides interactive visualization of the pool of incompatible pairs, non-directed donors, and the suggested loops and chains [44].

5.2 KidneyLife

Our supervisors originally suggested creating a module for Txmatching using the developed models, as is reflected in the bachelor project task. However, due to differences in project philosophy and the kind of data used in Txmatching, it was decided to create a separate application. In the following section, we will present the application, its architecture, how it works, the process of its development, and what can be done further.

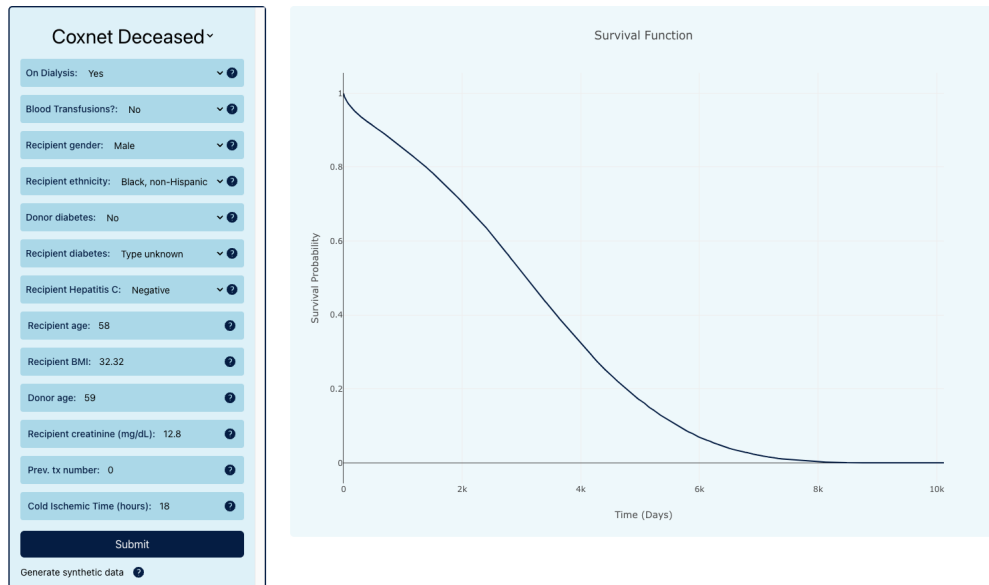


Figure 5.4: Kidney Life UI

5.2.1 Overview

We present KidneyLife, an inference application dedicated to the user-friendly use of machine learning models for survival analysis. The application aims to aid physicians in decision-making by comprehensively estimating a kidney transplant recipient’s lifespan. Due to the models’ assumption of donor-recipient compatibility, it should be used with the previously described KPD software or other compatibility-matching software.

Figure 5.4 features KidneyLife’s user interface. On the left, you can see the form for the input of the data. On its top, you can notice a dropdown menu for the model selection. Categorical features can be selected through dropdowns, while numerical data can be manually entered into text inputs. On the bottom of the form, you can see the Submit button, which sends the data to the server to make a prediction and performs checks for correct input and the absence of missing values. For more details on the validation and the front-end application, refer to Section 5.2.2 under the "Front End" subsection. Further insights into the UI can be seen in Section 5.2.4. Details regarding the prediction methods can be found in Section 5.2.2 under the "Back End" subsection.

Under the Submit button, you can find a button fetching synthetic data from the server. This allows for quick model illustration without the need for manual data entry. The details of the synthetic data generation can be found in Section 5.2.2 under the "Back End" subsection.

It is important to note that KidneyLife should be perceived as a minimum viable product (MVP) or, rather, a proof of concept, offering a minimum usable set of features for user feedback. This approach was adopted to address the rapidly growing development complexity and uncertainty of user needs and preferences. Section 5.2.5 outlines the application’s initial conceptualization and potential avenues for further development.

5.2.2 Architecture

The application was built with Docker. Docker is a software platform that allows developers to package their applications in containers, a standard unit of software that bundles code and its dependencies

to ensure fast and reliable execution across various computing environments [39]. Docker compose is a utility for building and distributing multi-container applications. Each container within the Docker Compose application stack is called a service [40]. KidneyLife consists of three services: front end, back end, and the reverse proxy.

Front End The front-end service contains an application developed with React, the popular JavaScript front-end framework developed by Facebook in 2013. The architecture of the front-end application is rather simple, as it consists only of a few key components. A component is an independent and reusable piece of code that takes the form of a JavaScript function that returns HTML. The main page component contains all other components and all logic related to the temporary data storage and sending requests.

The InputField component renders the input type text. Any characters apart from numbers and a dot are replaced with an empty string. It was done so, as the input type number provided inconsistent behavior: it worked as expected in browsers based on Chromium, such as Brave and Chrome, but allowed other characters in other browsers, such as Firefox and Safari. Moreover, the input type number allows the character "e", as it is a part of scientific notation. Furthermore, it contained unappealing arrows for increasing and decreasing an input number whose styles could not be altered.

The Select component renders the dropdown menu that allows one to select values from the provided list, which is particularly useful for categorical values. The ModalContainer serves as a wrapper and a background for the form. Finally, the PlotComponent renders the interactive plot from the Plotly.js package.

The front-end validation includes checks for the provided values. First and foremost, it checks if the value was indeed provided. It also includes min/max value validation, ensuring that the value falls into a specific range. Moreover, there is float/integer validation, depending on the data type the field requires. Finally, it verifies that the selected categorical value is indeed from the provided list of values, preventing users from adding their own category using the browser's built-in developer tools and sending the request to the server.

The styling was done with Tailwind, a CSS library that significantly simplifies the process of transforming the design into code compared to vanilla CSS, which can be cumbersome and time-consuming.

As a part of the front-end service build process, React code is compiled into production-ready optimized code, which is then served with the Nginx server.

Back End The Django Rest Framework, a popular Python web framework for building RESTful APIs, was used for the back end. An application program interface (API) is a set of guidelines that dictate the procedures for connectivity and data exchange between applications or devices, while a REST API specifically adheres to the principles of the representational state transfer (REST) architectural style. Essentially, API serves as a mechanism for accessing a resource in one application or service from another. The application or service that is accessing the resource is called the client. The application or service providing the resource is referred to as a server. The resources accessed here are the machine learning models trained previously, which were serialized or "pickled" with Python package pickle and are deserialized when we need to make a prediction.

The back-end REST API consists of two endpoints: one for prediction and the other for synthetic data generation. The prediction endpoint expects the request body with two parameters: the model name and the data dictionary. It then validates the data, passes data through the data pipeline to make the data usable by the model, loads data into the model, makes a prediction, and sends the results to the front end as a response.

The synthetic data generation endpoint has only one parameter: the model name. When a request is sent, it loads the proper model description JSON document and looks at its statistical data, particularly

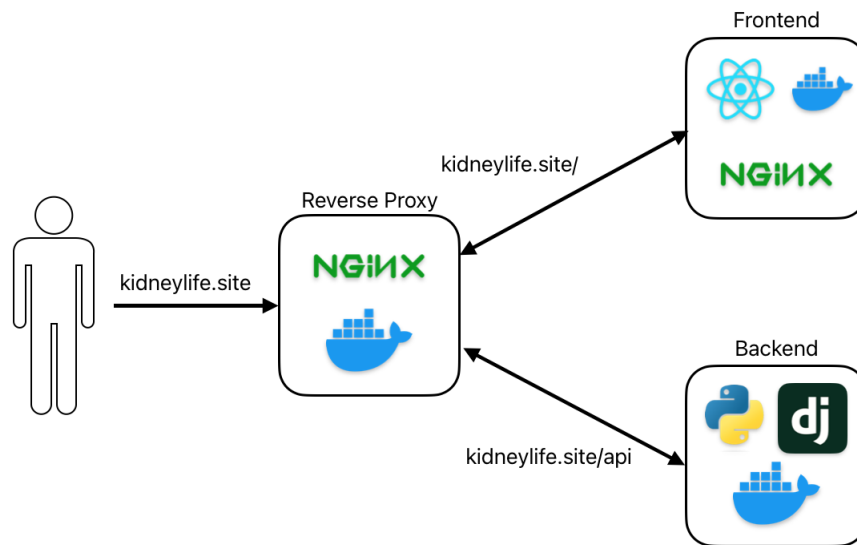


Figure 5.5: Application architecture.

the 10th and 90th quantiles for numerical data and the frequencies for categorical data. Then, under the assumption of normal distribution, we generate data for each numerical feature and, under given value frequencies, the values for categorical features. These quantiles are used to avoid peculiar situations when minimal values are generated along with the maximal (e.g. it could generate a pair of 6-year-old donor and 86-year-old recipient).

Reverse proxy Even though the front-end code is hosted in the docker container on the server, it is executed in the user's browser. Cross-Origin Resource Sharing (CORS) is a mechanism that allows an application on one URL to request data from another URL. The browser implements the Same-origin policy as a part of its security measures, allowing requests from its own URL and blocking requests from other URLs unless certain conditions are met. When a browser sends a request to the server, it has an Origin property in its header. The server adds the header Access-Control-Allow-Origin to its response. If Origin and Access-Control-Allow-Origin are not the same, the browser will prevent the server from sharing the contents of the response with the client.

It is not possible, however, to identify or specify the URL of each user, nor is it secure to allow calling the request from all sources. To solve this problem, we use the reverse proxy. A *reverse proxy* is an intermediary server that directs the client request to the appropriate backend server [41]. When we access the applications's URL we get redirected to the front end. When we want to make a prediction, we send a request to the proxy, it redirects the request to the back end, and we get our prediction and render it on the front end. This schema is illustrated on the Figure 5.5.

In addition to the CORS handling, reverse proxy increases security, as it serves as a barrier between the internet and the backend. It also increases the application's scalability, as we can create multiple backends on different servers and balance the load between them.

5.2.3 Deployment

To deploy the app, we first need a place to deploy it to. We rent a virtual private server (VPS) or a virtual machine on the cloud. Initially, we used Google Cloud on the free trial, but it proved to be too expensive, so we later migrated to another VPS hosting provider.

After securing the VPS, we bought the domain name "kidneylife.site" from the domain registrar Namecheap. Then, we registered the domain in the domain name system (DNS), associating the VPS IP address with the acquired domain.

The application was then deployed to the VPS using a Shell script. It goes through the following steps:

1. Check if the connection with the VPS can be established
2. Zip the application folder
3. Send the zip to the VPS
4. Unzip the folder
5. Build and run the docker compose stack with the production environment variables.

5.2.4 Functionality and UX

KidneyLife is a standard machine learning inference application. Inference application stands for an application whose sole purpose is user-friendly access to the machine learning model. As such, it is a one-page application that has select input with two models, the inputs are rendered based on the selected model. And the interactive graph made with Plotly.js. Finally, it features a button "Generate synthetic data" made for testing and illustration purposes.

The design of the application was crafted with consideration of several design principles. Shades of blue were chosen because of their calming effects and their association with reliability [42]. Moreover, blue is generally a safe color, as it appeals to a larger audience. Simplicity, being one of the main principles of modern web design, was naturally prioritized as inference applications inherently have little complexity. Furthermore, the larger margins and paddings were employed to provide 'breathing room' within the interface, preventing it from feeling overcrowded. Lastly, the design exhibits responsiveness, accommodating a wide range of devices, from tablets to large 2k monitors, delivering a consistent and comfortable user experience across different screen sizes.

5.2.5 Future Work

There are several avenues for enhancing the application's functionality and user experience. First, integrating additional machine learning models is a promising avenue for providing more comprehensive and accurate predictions. While models mostly from scikit survival were used in this work, other packages, such as lifelines, can offer a wider range of survival analysis models.

Secondly, implementing a robust login system is essential to ensure secure access to the application's features and confidential data. It is especially important if we aim to implement the features outlined in the subsequent paragraphs, as they imply the storage of sensitive information. User authentication mechanisms, such as username-password authentication or OAuth, can restrict access to unauthorized individuals, protecting sensitive data associated with subsequent features.

Moreover, introducing patient (recipient and donor) and pair management systems can improve user experience by enabling the saving of the predictions for the provided pair of patients and models in the

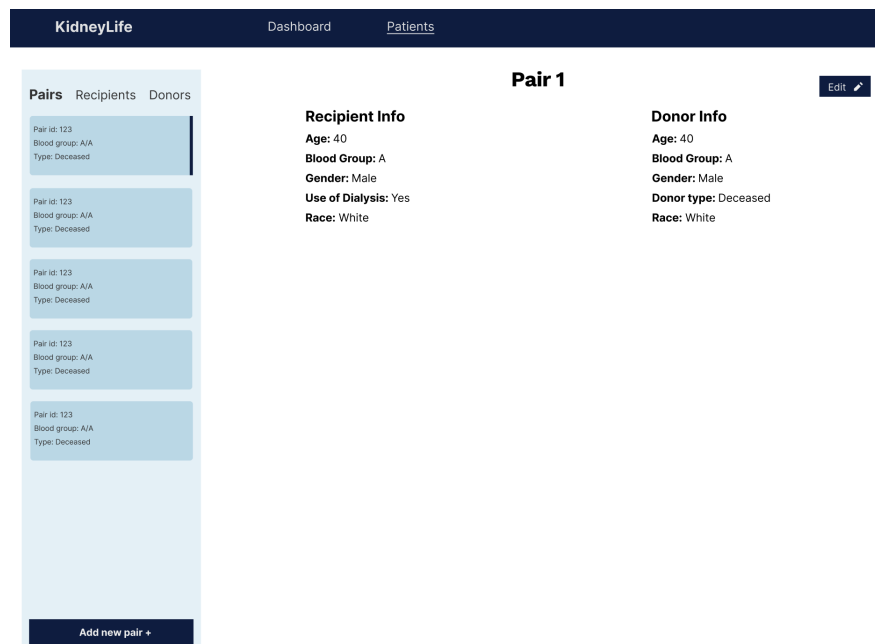


Figure 5.6: Patient and pair management system concept

database. The feature will allow users to look at the predicted survival curve later only with a couple of clicks, eliminating the need to enter the data manually each time. The UI concept of this system can be seen in Figure 5.6.

Another feature worth implementing is an option of estimating the cumulative hazards. It is relatively easy to implement, as it will only include minor changes to the front end and creating a new endpoint, which will access the method of already trained model and return the results. This might be useful if a user wants to see the graph of cumulative risk in time, rather than the probability of survival.

Furthermore, integrating the HL7 FHIR standard would enhance interoperability and data exchange between the application and the hospital's ecosystem. FHIR, which stands for *Fast Healthcare Interoperability Resources* (FHIR) is a *Health Level Seven* (HL7) standard for electronic healthcare data exchange.

If any of these changes are to take place, we need to add continuous integration (CI) and continuous delivery (CD). CI/CD automates testing and deployment processes, improving efficiency and reducing the risk of errors. While the deployment has already been automated, the testing has not. GitHub Actions can be utilized to test the application automatically on the server every time the changes are pushed to GitHub, ensuring that pull requests cannot be merged into the main branch if the tests fail. While the tests for the backend already exist, automating them with GitHub actions would be the next step.

Cross-browser compatibility is another avenue for enhancing the application. It is crucial to ensure that the application looks the same across different operation systems and browsers. It can be achieved by designing custom scrollbars, improving the select fields by creating custom dropdown menus, and ensuring they look consistent across all browsers. However, it is possible that we may have missed something, so further testing is needed.

Before implementing any of the mentioned points, we need to reach out to institutions to see if this application is what actually transplant centers need. In case it is, the further development of the application can take place, making sure that it aligns with the needs of any given institution.

During the later stages of development, it was discovered that a nonprofit organization with the name KidneyLife already exists. If someone is interested in the application, we would need to rename and

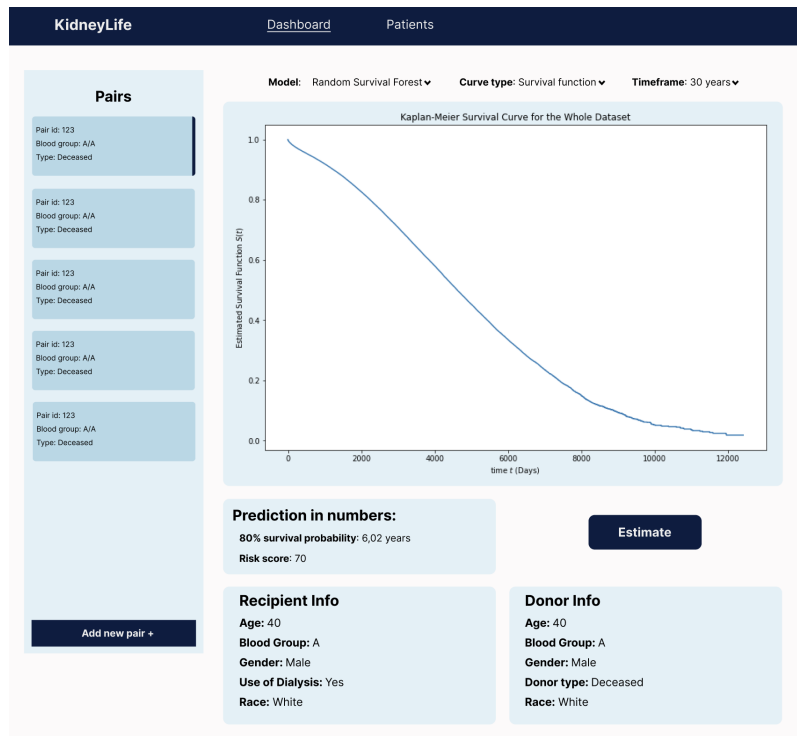


Figure 5.7: Kidney Life Dashboard page concept

rebrand it to prevent any confusion and potential legal issues.

Conclusion

The study involved training six models to predict the survival outcome of kidney transplantation, three for each type of transplantation – living donor transplantation (LDT) and deceased donor transplantation (DDT). The study used regularized Cox (Coxnet) [54], Gradient-Boosted Survival Analysis (GBSA) [56], and Random Survival Forest (RSF) [55] algorithms from the scikit-survival package.

The performance of the models was evaluated using the Uno concordance index (Uno c-index), Integrated Brier Score (IBS), and the mean cumulative/dynamic area under the curve (mean AUC) as numerical performance metrics. The Brier score and the cumulative/dynamic area under curve (AUC) were used as time-dependent measures.

The DDT models achieved an Uno c-index of 0.691 - 0.699, while the LDT models achieved a 0.722-0.723. However, the results are comparable to those of other papers that trained models without separation.

Hyperparameter tuning was performed with GridSearchCV and some manual tweaking. Parameters from Table 5.2 were found to be the best. However, further fine-tuning might be needed, especially with the number of estimators in the Random Survival Forest.

The feature selection was performed by evaluating the importance of the feature with the importance of permutation and the iterative exclusion of features with negative and zero importance. More than 70 features were considered and engineered, only for most of them to be excluded. The following features were most important across models: DIAB, AGE, ETHNCAT, HCV_SEROSTATUS, ON_DIALYSIS, and GENDER. Table 6.1 provides a description and type for each of these features.

A prototype of a transplant scoring system was developed using trained models. While it can catch bad transplantations, it cannot tell which successful transplantations are better. This might be due to the inherent difficulty of accurately predicting survival, as too many factors influencing survival cannot be included in the dataset. In any event, more work is needed to make any conclusions.

An application named KidneyLife has been developed to assist physicians in their decision-making process by offering a comprehensive estimate of the expected lifespan of a kidney transplant recipient. The application employs Coxnet models for LDT and DDT. Users can enter values for the selected fea-

Living	Uno C-index	IBS	Mean AUC
Coxnet	0.723	0.136	0.743
GBSA	0.722	0.136	0.742
RSF	0.723	0.139	0.744
Deceased			
Coxnet	0.695	0.163	0.729
GBSA	0.699	0.162	0.734
RSF	0.691	0.164	0.724

Table 5.1: Model performance for DDT and LDT. The best model is highlighted.

Model	Hyperparameter Set
RSF	{num_estimators=50, max_depth=12, min_samples_split=16}
GBSA	{learning_rate=0.2, num_estimators=94(living), 110(deceased), max_depth=6 }
Coxnet	{l1_ratio=0.9, alpha=*calculated* }

Table 5.2: The best hyperparameters for trained models

tures, click "Submit," and obtain a prediction in the form of a survival function. Synthetic data generation for testing purposes is also available. The application is considered a minimum viable product (MVP), and its functionality may be significantly expanded according to the needs of interested institutions. For further details, refer to the Section 5.2.5.

It is worth noting that the data assume well-matched pairs despite HLA features being excluded in the end. Consequently, models and the application should be used only with KPD software and prior negative crossmatch results. Otherwise, the results might be inaccurate.

Data Source and Source Code

The data reported here have been supplied by the United Network for Organ Sharing as the contractor for the Organ Procurement and Transplantation Network. The interpretation and reporting of these data are the responsibility of the author(s) and in no way should be seen as an official policy of or interpretation by the OPTN or the U.S. Government.

The source code for both the application and trained models can be found in the following GitHub repository: <https://github.com/krllstdn/BachelorProject> Data and trained models are, of course, excluded.

Bibliography

- [1] Knechtle, S. J., Marson, L. P., & Morris, P. (2019). *Kidney transplantation - principles and practice: Expert consult - online and print (8th ed.)*. Elsevier - Health Sciences Division
- [2] Nobel prize in physiology or medicine (2022) Our Scientists. Available at: <https://www.rockefeller.edu/our-scientists/alexis-carrel/2565-nobel-prize/> (Accessed: February 6, 2023).
- [3] Barker, C. F., & Markmann, J. F. (2013). Historical Overview of Transplantation. *Cold Spring Harbor Perspectives in Medicine*, 3(4). <https://doi.org/10.1101/cshperspect.a014977>
- [4] Matevossian, Edouard, et al. "Surgeon Yurii Voronoy (1895-1961)-a pioneer in the history of clinical transplantation: in memoriam at the 75th anniversary of the first human kidney transplantation." *Transplant International* 22.12 (2009): 1132.
- [5] PUNT, Jenni et al. *Kuby immunology*. Eight. vyd. New York: Macmillan Education, 2019. ISBN 9781319114701;1319114709;
- [6] ABBAS, Abul K., Andrew H. LICHTMAN a Shiv PILLAI. *Basic immunology: functions and disorders of the immune system*. Sixth. vyd. Philadelphia: Elsevier, 2020. ISBN 9780323549431;0323549438;
- [7] NCI Dictionary of Cancer terms (no date) National Cancer Institute. Available at: <https://www.cancer.gov/publications/dictionaries/cancer-terms/def/abo-blood-group-system> (Accessed: March 6, 2023).
- [8] Dean L. Blood Groups and Red Cell Antigens [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2005. Chapter 2, Blood group antigens are surface markers on the red blood cell membrane. Available from: <https://www.ncbi.nlm.nih.gov/books/NBK2264/>
- [9] Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., Sept. 2019.
- [10] Andriy Burkov. *THE HUNDRED-PAGE MACHINE LEARNING BOOK*. Andriy Burkov, 2019.
- [11] Gradient Boosted Models — scikit-survival 0.22.2. (2015). Readthedocs.io. https://scikit-survival.readthedocs.io/en/v0.22.2/user_guide/boosting.html
- [12] Bruce, P., Bruce, A., & Gedeck, P. (2020). *Practical statistics for data scientists: 50+ Essential concepts using R and python (2nd ed.)*. O'Reilly Media. p. 141
- [13] Kleinbaum, D. G., & Klein, M. (2011). *Survival analysis: A self-learning text, third edition (3rd ed.)*. Springer.

- [14] Ostan R, Monti D, Guerresi P, Bussolotto M, Franceschi C, Baggio G. Gender, aging and longevity in humans: an update of an intriguing/neglected scenario paving the way to a gender-specific medicine. *Clin Sci (Lond)*. 2016 Oct 1;130(19):1711-25. doi: 10.1042/CS20160004. PMID: 27555614; PMCID: PMC4994139.
- [15] vom Steeg LG, Klein SL. SeXX Matters in Infectious Disease Pathogenesis. *PLoS Pathog*. 2016 Feb 18;12(2):e1005374. doi: 10.1371/journal.ppat.1005374. PMID: 26891052; PMCID: PMC4759457.
- [16] Rodrigues S, Escoli R, Eusébio C, Dias L, Almeida M, Martins LS, Pedroso S, Henriques AC, Cabrita A. A Survival Analysis of Living Donor Kidney Transplant. *Transplant Proc*. 2019 Jun;51(5):1575-1578. doi: 10.1016/j.transproceed.2019.01.047. Epub 2019 Jan 21. PMID: 31155195.
- [17] Nemati E, Einollahi B, Lesan Pezeshki M, Porfarziani V, Fattahi MR. Does kidney transplantation with deceased or living donor affect graft survival? *Nephrourol Mon*. 2014 Jul 5;6(4):e12182. doi: 10.5812/numonthly.12182. PMID: 25695017; PMCID: PMC4317718.
- [18] Pisavadia B, Arshad A, Chappelow I, Nightingale P, Anderson B, Nath J, Sharif A. Ethnicity matching and outcomes after kidney transplantation in the United Kingdom. *PLoS One*. 2018 Apr 13;13(4):e0195038. doi: 10.1371/journal.pone.0195038. PMID: 29652887; PMCID: PMC5898720.
- [19] Guillermo García García, Arpana Iyengar, François Kaze, Ciara Kierans, Cesar Padilla-Altamira, Valerie A. Luyckx, Sex and gender differences in chronic kidney disease and access to care around the globe, *Seminars in Nephrology*, Volume 42, Issue 2, 2022, Pages 101-113, ISSN 0270-9295, <https://doi.org/10.1016/j.semnephrol.2022.04.001>. (<https://www.sciencedirect.com/science/article/pii/S0270929522000092>)
- [20] Mange KC, Joffe MM, Feldman HI. Effect of the use or nonuse of long-term dialysis on the subsequent survival of renal transplants from living donors. *N Engl J Med*. 2001 Mar 8;344(10):726-31. doi: 10.1056/NEJM200103083441004. PMID: 11236776.
- [21] Rahman MS, Ambler G, Choodari-Oskooei B, Omar RZ. Review and evaluation of performance measures for survival prediction models in external validation settings. *BMC Med Res Methodol*. 2017 Apr 18;17(1):60. doi: 10.1186/s12874-017-0336-2. PMID: 28420338; PMCID: PMC5395888.
- [22] Evaluating survival models — scikit-survival 0.21.0. (n.d.). Readthedocs.io. Retrieved July 18, 2023, from https://scikit-survival.readthedocs.io/en/stable/user_guide/evaluating-survival-models.html
- [23] Ping Wang, Yan Li, and Chandan k. Reddy. 2019. Machine Learning for Survival Analysis: A Survey. *ACM Comput. Surv.* 51, 6, Article 110 (February 2019), 36 pages.<https://doi.org/10.1145/3214306>
- [24] Definitions:, 1. 1. (n.d.). Survival distributions, hazard functions, cumulative hazards. Stanford.edu. Retrieved October 23, 2023, from <https://web.stanford.edu/~lutian/coursepdf/unit1.pdf>
- [25] Penalized Cox Models — scikit-survival 0.22.1. (2015). Readthedocs.io. https://scikit-survival.readthedocs.io/en/stable/user_guide/coxnet.html

- [26] Ishwaran, H., Kogalur, U. B., Blackstone, E. H., & Lauer, M. S. (2008). Random survival forests.
- [27] Wang, H., & Li, G. (2017). A selective review on random survival forests for high dimensional data. *Quantitative bio-science*, 36(2), 85.
- [28] Breiman, L. (2001). Random forests. *Machine learning*, 45, 5-32.
- [29] Gönen M, Heller G. Concordance probability and discriminatory power in proportional hazards regression. *Biometrika* 2005;92(4):1799–09.
- [30] Enrico Longato, Vettoretti, M., & Barbara Di Camillo. (2020). A practical perspective on the concordance index for the evaluation and selection of prognostic time-to-event models. *Journal of Biomedical Informatics*, 108, 103496–103496. <https://doi.org/10.1016/j.jbi.2020.103496>
- [31] Uno H, Cai T, Pencina MJ, D'Agostino RB, Wei LJ. On the C-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data. *Stat Med*. 2011 May 10;30(10):1105-17. doi: 10.1002/sim.4154. Epub 2011 Jan 13. PMID: 21484848; PMCID: PMC3079915.
- [32] Kindt TJ Goldsby RA Osborne BA Kuby J. *Kuby Immunology*. 6th ed. New York: W.H. Freeman; 2007.
- [33] Health. (2022). Kidneys. [Vic.gov.au. https://www.betterhealth.vic.gov.au/health/conditionsandtreatments/kidneys](https://www.betterhealth.vic.gov.au/health/conditionsandtreatments/kidneys)
- [34] Facts About Chronic Kidney Disease. (2020, May 15). National Kidney Foundation; <https://www.kidney.org/atoz/content/about-chronic-kidney-disease>
- [35] Kamyar Kalantar-Zadeh, Jafar, T. H., Nitsch, D., Neuen, B. L., & Perkovic, V. (2021). Chronic kidney disease. *The Lancet*, 398(10302), 786–802. [https://doi.org/10.1016/s0140-6736\(21\)00519-5](https://doi.org/10.1016/s0140-6736(21)00519-5)
- [36] Holland, K. (2019, May 23). Everything You Need to Know About Kidney Failure. *Healthline*; Healthline Media. Retrieved 5 August, 2024 from <https://www.healthline.com/health/kidney-failure#outlook>
- [37] Causes of chronic kidney disease. (2022, August 29). National Institute of Diabetes and Digestive and Kidney Diseases; NIDDK - National Institute of Diabetes and Digestive and Kidney Diseases. <https://www.niddk.nih.gov/health-information/kidney-disease/chronic-kidney-disease-ckd/causes>
- [38] Estimated Glomerular Filtration Rate (eGFR). (2015, December 24). National Kidney Foundation; Retrieved 5 August, 2024 from <https://www.kidney.org/atoz/content/gfr>
- [39] What is a Container? | Docker. (2023, October 26). Docker. Retrieved 5 August, 2024 from <https://www.docker.com/resources/what-container/>
- [40] Use Docker Compose. (2024). Docker Documentation. Retrieved 5 August, 2024 from https://docs.docker.com/get-started/08_using_compose/
- [41] What is a Reverse Proxy Server? | NGINX. (2023, June). NGINX. Retrieved 5 August, 2024 from <https://www.nginx.com/resources/glossary/reverse-proxy-server/>

- [42] Kendra Cherry, Msc. (2022, November 22). How the color blue impacts moods, feelings, and behaviors. Verywell Mind. Retrieved 5 August, 2024 from <https://www.verywellmind.com/the-color-psychology-of-blue-2795815>
- [43] Kidney Paired Donation (KPD) Program. (2024, January 3). Professional Education. Retrieved 5 August, 2024 from <https://professionaleducation.blood.ca/en/organs-and-tissues/programs/kidney-paired-donation-kpd-program>
- [44] Bray, M., Wang, W., Rees, M. A., Peter X-K. Song, Leichtman, A. B., Ashby, V. B., & Kalbfleisch, J. D. (2019). KPDGUI: An interactive application for optimization and management of a virtual kidney paired donation program. *Computers in Biology and Medicine*, 108, 345–353. <https://doi.org/10.1016/j.combiomed.2019.03.013>
- [45] 4 Key Features of the APKD Software Program | Alliance for Paired Kidney Donation. (2022, December 9). Alliance for Paired Kidney Donation | Retrieved 5 August, 2024 from <https://paireddonation.org/4-key-features-of-the-apkd-software-program/>
- [46] Aufhauser, D. D., Peng, A. W., Murken, D. R., Concors, S. J., Abt, P. L., Sawinski, D., Bloom, R. D., Reese, P. P., & Levine, M. H. (2018). Impact of prolonged dialysis prior to renal transplantation. *Clinical Transplantation*, 32(6). <https://doi.org/10.1111/ctr.13260>
- [47] Naqvi SAA, Tennankore K, Vinson A, Roy PC, Abidi SSR. Predicting Kidney Graft Survival Using Machine Learning Methods: Prediction Model Development and Feature Significance Analysis Study. *J Med Internet Res*. 2021 Aug 27;23(8):e26843. doi: 10.2196/26843. PMID: 34448704; PMCID: PMC8433864.
- [48] Aufhauser DD Jr, Peng AW, Murken DR, Concors SJ, Abt PL, Sawinski D, Bloom RD, Reese PP, Levine MH. Impact of prolonged dialysis prior to renal transplantation. *Clin Transplant*. 2018 Jun;32(6):e13260. doi: 10.1111/ctr.13260. Epub 2018 Jun 25. PMID: 29656398; PMCID: PMC6023748.
- [49] Mark E, Goldsman D, Gurbaxani B, Keskinocak P, Sokol J. Using machine learning and an ensemble of methods to predict kidney transplant survival. *PLoS One* 2019; 14.
- [50] Paquette FX, Ghassemi A, Bukhtiyarova O, Cisse M, Gagnon N, Della Vecchia A et al.. Machine learning support for decision-making in kidney transplantation: step-by-step development of a technological solution. *JMIR Med Inform* 2022;10:e34554
- [51] Senanayake S, Kularatna S, Healy H, Graves N, Baboolal K, Sypek MP et al.. Development and validation of a risk index to predict kidney graft survival: the kidney transplant risk index. *BMC Med Res Methodol* 2021;21:1–11. (0.60-0.63)
- [52] Ravindhran B, Chandak P, Schafer N, Kundalia K, Hwang W, Antoniadis S, Haroon U, Zakri RH. Machine learning models in predicting graft survival in kidney transplantation: meta-analysis. *BJS Open*. 2023 Mar 7;7(2):zrad011. doi: 10.1093/bjsopen/zrad011. PMID: 36987687; PMCID: PMC10050937.
- [53] Yoo, K.D., Noh, J., Lee, H. et al. A Machine Learning Approach Using Survival Statistics to Predict Graft Survival in Kidney Transplant Recipients: A Multicenter Cohort Study. *Sci Rep* 7, 8904 (2017). <https://doi.org/10.1038/s41598-017-08008-8>

- [54] `sksurv.linear_model.CoxnetSurvivalAnalysis` — `scikit-survival` 0.22.2. (2015). [Readthedocs.io](https://scikit-survival.readthedocs.io/en/v0.22.2/api/generated/sksurv.linear_model.CoxnetSurvivalAnalysis.html). Retrieved 5 August, 2024 from https://scikit-survival.readthedocs.io/en/v0.22.2/api/generated/sksurv.linear_model.CoxnetSurvivalAnalysis.html
- [55] `sksurv.ensemble.RandomSurvivalForest` — `scikit-survival` 0.22.2. (2015). [Readthedocs.io](https://scikit-survival.readthedocs.io/en/v0.22.2/api/generated/sksurv.ensemble.RandomSurvivalForest.html). Retrieved 5 August, 2024 from <https://scikit-survival.readthedocs.io/en/v0.22.2/api/generated/sksurv.ensemble.RandomSurvivalForest.html>
- [56] `sksurv.ensemble.GradientBoostingSurvivalAnalysis` — `scikit-survival` 0.22.2. (2015). [Readthedocs.io](https://scikit-survival.readthedocs.io/en/v0.22.2/api/generated/sksurv.ensemble.GradientBoostingSurvivalAnalysis.html). Retrieved 5 August, 2024 from <https://scikit-survival.readthedocs.io/en/v0.22.2/api/generated/sksurv.ensemble.GradientBoostingSurvivalAnalysis.html>
- [57] Ott, H. C., Matthiesen, T. S., Saik Kia Goh, Black, L. D., Kren, S. M., Netoff, T. I., & Taylor, D. A. (2008). Perfusion-decellularized matrix: using nature’s platform to engineer a bioartificial heart. *Nature Medicine*, 14(2), 213–221. <https://doi.org/10.1038/nm1684>
- [58] Sánchez PL, Fernández-Santos ME, Costanza S, et al. Acellular human heart matrix: A critical step toward whole heart grafts. *Biomaterials*. 2015;61:279-289. doi:10.1016/j.biomaterials.2015.04.056
- [59] Jin, Z., Li, Y., Yu, K., Liu, L., Fu, J., Yao, X., Zhang, A., & He, Y. (2021). 3D Printing of Physical Organ Models: Recent Developments and Challenges. *Advanced Science*, 8(17). <https://doi.org/10.1002/advs.202101394>
- [60] Nagashima, H., & Hitomi Matsunari. (2016). Growing human organs in pigs—A dream or reality? *Theriogenology*, 86(1), 422–426. <https://doi.org/10.1016/j.theriogenology.2016.04.056>
- [61] A Guide to Calculating and Interpreting the Estimated Post-Transplant Survival (EPTS) Score Used in the Kidney Allocation System (KAS) https://optn.transplant.hrsa.gov/media/1511/guide_to_calculating_interpreting_epts.pdf. Accessed 16 Apr. 2024.
- [62] Baskin-Bey ES, Kremers W, Nyberg SL. A recipient risk score for deceased donor renal allocation. *Am J Kidney Dis*. 2007 Feb;49(2):284-93. doi: 10.1053/j.ajkd.2006.10.018. PMID: 17261431.
- [63] Wolfe RA, McCullough KP, Leichtman AB. Predictability of survival models for waiting list and transplant patients: calculating LYFT. *Am J Transplant*. 2009 Jul;9(7):1523-7. doi: 10.1111/j.1600-6143.2009.02708.x. PMID: 19656143.
- [64] Mark E, Goldsman D, Gurbaxani B, Keskinocak P, Sokol J. Using machine learning and an ensemble of methods to predict kidney transplant survival. *PLoS One*. 2019 Jan 9;14(1):e0209068. doi: 10.1371/journal.pone.0209068. PMID: 30625130; PMCID: PMC6326487.
- [65] Hamet, P., & Tremblay, J. (2017). Artificial intelligence in medicine. *Metabolism, Clinical and Experimental*, 69, S36–S40. <https://doi.org/10.1016/j.metabol.2017.01.011>
- [66] Chen, T. K., Knicely, D. H., & Grams, M. E. (2019). Chronic Kidney Disease Diagnosis and Management. *JAMA*, 322(13), 1294–1294. <https://doi.org/10.1001/jama.2019.14745>
- [67] Organ Donation Statistics | [organdonor.gov](https://www.organdonor.gov). (2024, March 29). [Organdonor.gov](https://www.organdonor.gov). Retrieved May 20, 2024 from <https://www.organdonor.gov/learn/organ-donation-statistics>

- [68] USAFacts. (2024, June 15). US population by year, race, age, ethnicity, & more. USAFacts; USAFacts. Retrieved June 23, 2024 from <https://usafacts.org/data/topics/people-society/population-and-demographics/our-changing-population/>
- [69] Friedman, J. (1999, March). Stochastic gradient boosting. Technical report, Stanford University, Statistics Department.
- [70] G. Ridgeway, "The state of boosting," *Computing Science and Statistics*, 172–181, 1999.
- [71] Leblanc, M., & Crowley, J. (1993). Survival Trees by Goodness of Split. *Journal of the American Statistical Association*, 88(422), 457–467.
- [72] scikit-survival/sksurv/tree/tree.py at v0.23.0 · sebp/scikit-survival. (2024). GitHub. Retrieved 5 August 2024 from <https://github.com/sebp/scikit-survival/blob/v0.23.0/sksurv/tree/tree.py#L38-L676>
- [73] scikit-learn/sklearn/tree/_tree.pyx at main · scikit-learn/scikit-learn. (2024). GitHub. Retrieved 5 August, 2024 from https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/tree/_tree.pyx#L140
- [74] DecisionTreeRegressor. (2024). Scikit-Learn. Retrieved 5 August 2024 from <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html>
- [75] Nelson-Aalen Estimator. (n.d.). Retrieved 5 August 2024 from <https://www.medicine.mcgill.ca/epidemiology/hanley/c609/material/NelsonAalenEstimator.pdf>