A Systematic and Data-Driven Approach for Improving Survival Analysis of Heart Transplantation

by

Hamidreza Ahady Dolatsara

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 3, 2019

Keywords: Heart, Survival, Transplant, Prediction, Monotonicity

Approved by

Fadel M. Megahed, Co-Chair, Assistant Professor of Industrial and Systems Engineering
Ashish Gupta, Co-Chair, Associate Professor of Business Analytics
John Evans, Professor of Industrial and Systems Engineering
Richard Sesek, Associate Professor of Industrial and Systems Engineering

Abstract

This dissertation is developed to improve survival analysis of heart transplant surgeries by first, investigating factors that are associated with the performance of predictive machine learning algorithms, then creating multiple experiments of predictions and selecting the best combination of the factors. These factors are multiple combinations of data preparation methods (imputation, encoding, and data cleaning), feature selection methods (filter, wrapper, and hybrid), resampling unbalanced data (synthesized oversampling, and undersampling), and well-known predictive algorithms (Logistic Regression, Linear Discriminant Analysis, Multivariate Adaptive Regression Spline, Neural Network, Naive Bayes, Support Vector Machines, eXtreme Gradient Boosting, Stochastic Gradient Boosting, and Random Forest).

The goal of the second part is introducing an approach that could deliver monotonic survival probabilities. Therefore, the predicted survival probabilities could be more easily interpreted by practitioners. The best model that yields the highest predictive performance in the first part of this study is considered for a post-calibration phase and producing monotonic predictions. In addition, a tool is developed to empower practitioners to employ the best predictive model for investigating the survival of patients with heart transplantation in a 10-year time window after the transplantation surgery.

This study is presented in five major sections. The first section highlights the necessity of considering a systematic approach for developing machine learning-based research with its application in transplant surgeries. The second section reviews the major challenges in predicting survival of heart transplantation surgeries. In the third section, a systematic approach based on

Design of Experiments (DoE) for developing and optimizing predictive models in the heart transplantation domain is presented. In the fourth section, the identified challenges in heart transplantation studies are addressed. Conclusions, limitations, and future studies are discussed in the last section. To facilitate the reproduction of these results and to explain the details of the analysis, the codes and instructions are provided in the appendix.

Acknowledgments

First and foremost, I am sincerely grateful to my adviser Dr. Fadel Megahed for his endless support, patience, and motivations throughout my Ph.D. studies at Auburn University. His academic advice and wisdom helped me to learn how to systematically develop good academic research. He never gave up on me and I could not have imagined a better Ph.D. adviser for my academic adventure.

I would like to express my deep thanks to my co-advisor Dr. Ashish Gupta for his continuous support during the development of this study. The resources that he provided at Auburn University led to a productive research environment for me. Likewise, I am overwhelmed in humbleness to acknowledge my research mentor, Dr. Ying-Ju Tessa Chen from Dayton University for all of her significant contributions. Also, I would also like to thank Dr. John Evans, Dr. Richard Sesek, Dr. David Paradice, and Dr. LuAnn Carpenter for their encouragement and support during my fruitful Ph.D. education at Auburn University.

Last but not least, I appreciate and thank my parents (Yahya and Tooba), my brothers (Hesam and Majid), and my sisters (Bahareh and Gelareh) for their love, prayers, and unbelievable efforts. Without their support and love, I could never have reached this point.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ADT	Alternating Decision Tree
BBN	Bayesian Belief Network
CART	Classification & Regression Trees
CDR	Cadaver Donor Registration
CRISP-DM	CRoss Industry Standard Process for Data Mining
DDR	Deceased Donor Registration
DM	Data Mining
DoE	Design of Experiments
DS	Decision Stump
DT	Decision Tree
GB	Gradient Boosting
GLMNET	Lasso and Elastic-Net Regularized Generalized Linear
HT	Heart Transplantation
KNN	K-Nearest Neighbors
KPLS	Principal Component Regression Kernel
LDA	Linear Discriminant Analysis
LR	Logistic Regression
ML	Machine Learning
NB	Naïve-Bayes
NNET	Neural Network
NNS	Nearest Neighbors

REPT	Reduced Error Pruning Tree
RF	Random Forest
RH	Recipient Histocompatibility
RotF	Rotation Forest
RS	Random Subspace
RUS	Random Under Sampling
SMOTE	Synthetic Minority Over-Sampling
SVM	Support Vector Machines
TCR	Transplant Candidate Registration
TRF	Transplant Recipient Follow-ups
TRR	Transplant Recipient Registration
UNOS	United Network for Organ Sharing
WL	Waiting List
XGB	eXtreme Gradient Boosting

Preface

This is developed and submitted to Auburn University's Graduate School in partial fulfillment of the requirements for the Doctoral degree in Industrial and Systems Engineering. The major incentive behind this dissertation was having an impact in the heart transplantation field by using data analytics tools. I have discussed with Dr. Megahed in several meetings to discover how we can improve the previous data-driven studies. Then we developed a stream of research ideas and approaches for the excellence of survival studies in the heart transplantation field.

In the second chapter, the current issues for the development of a data-mining study in the heart transplantation field are discussed. We think this is a fundamental requirement for developing a reproducible framework for future studies. Because any researchers would face similar issues. In addition, we provided the details and the codes to facilitate the future researches. This chapter would be presented in "2019 INFORMS Annual Meeting" in Seattle to receive the communities' feedback for a future publication. In the following two chapters, we discussed two approaches for addressing the issues.

The third chapter is dedicated for introducing a systematic approach that improves the quality and accuracy of machine learning algorithms in long term survival prediction. Dr. Megahed advised me to use the Design of Experiments concept as the systematic approach. Especially reducing the gap between sensitivity and specificity in the first year survival prediction after a transplantation surgery. We discussed with Dr. Ying-Ju Tessa Chen who is an assistant professor at University of Dayton and developed multiple scenarios that consider the major contributing factors for this type of data mining study. We are preparing this chapter for future conferences and publications.

Finally, in the fourth chapter, we introduced the isotonic regression algorithm as a post calibration method to provide a moronically decreasing pattern for predicting survival chance in consecutive timestamps. This approach eventually resolve the non-monotonically decreasing pattern issue in the literature and facilitate the interpretation of survival chance. Dr. Ying-Ju Tessa Chen and Dr. Megahed has contributed significantly in the development of the codes and consulted in the statistical interpretation of the results. We discussed about the development of study with Dr. Ashish Gupta and we made it ready for submission in *Decision Support Systems Journal (DSS).*

Besides learning a lot in the field of heart transplantation, completing this dissertation empowered me with a great deal of research skills in development of a data-driven study. I learned how to develop a research question and scientifically resolve the research challenges. In addition, I provided a reproducible platform for heart transplantation survival studies and addressed a major research challenge in the field, as the major contributions.

Chapter 1

Problem Description and Significance

This study proposes a framework for improving survival prediction for heart transplantation patients after the surgical procedure. The framework discusses how to construct, optimize, and calibrate a data mining study for investigating the survival of the patients. The major application of this data mining study is delivering a superior and reproducible model for predicting the survival chance of the patients over consecutive timestamps after the surgery. This study developed by investigating the transplantation data is achieved from the United Network for Organ Sharing (UNOS) organization [1]. Pre-transplantation data of the patients and the donors are utilized for model development. Therefore, the projection of the described survival pattern could demonstrate the long-term quality of the match between the donated organ and the patient. The latter part of the platform is dedicated for calibration of the consecutive survival prediction to guarantee a monotonically decreasing pattern.

A survival prediction process is considered as an experiment with a predictive performance as its outcome. The dimension of the timestamps is one year and the first year survival is the dependent variable (Target). Since the majority of the patients survive the first year, the distribution of Target is highly imbalanced. Meanwhile, the literature shows a lower outcome for the first year Target in compare with the following years [2, 3]. The framework investigates the optimization of the experiments by analyzing different factors that potentially affect the outcome of predicting the Target. Design of Experiments (DoE) [4] is the major methodology behind the framework. The factors consist of several options for data preparation, feature selection, sample balancing, and training models. DoE is utilized to demonstrate the significance of those factors and their interactions in the outcomes of the experiments. Several combinations of the significant factors

are investigated to obtain the factors' level that yields the highest outcome for predicting the Target. This information then used for model development in the other timestamps. In the following time- stamps, important features are investigated and combined with the previous information to develop survival models. The models then employed to predict the survival chance of each patient in all the timestamps. The predicted survival chances are investigated for the occurrence of a potential non-monotonicity decreasing pattern [5, 6]. In the case of the occurrence, isotonic regression [7] as a post-calibration method is employed to deliver a monotonically decreasing survival chance pattern. The main deliveries of the proposed platform for the heart transplantation:

- Introduce an objective and reproducible approach for selecting the best combination of factors that yield the highest predictive performance
- Address the potential non-monotonicity issue in the survival prediction for the consecutive timestamps.
- Demonstrate the most important factors' level that affects the prediction of Target.
- Compare the importance of features for predicting survival chance in consecutive timestamps.
- Predict long-term survival chance of heart transplant surgeries by utilizing pre-transplant data to investigate the quality of patient-donor matches.
- Provide a high and balanced prediction performance for the years that the dependent variable is highly imbalanced.
- Develop a tool for longterm survival prediction in the field.

1.1 Significance

Establishing a solid, state of the art, and reproducible framework for investigating the survival of heart transplantation (HT) is a necessity for patients in end-stage heart disease, transplant surgeons, and scholars that intend to scientifically study patients' conditions after transplantation surgery. Heart failure is an expensive and relatively common disease. The Heart Failure Society of America (HFSA) reported that about 6.5 million adults have heart failure in the USA with an annual accumulation rate of 960,000 new patients [8]. The demand and supply equilibrium is not balanced for HT [9, 10]. At any given time, there are about 3,000 patients registered on the waiting list for transplantation. However, there are only about 2,000 possible heart donors [11]. In addition, about half of the offered organs are not transplanted due to a conservative policy regarding organ allocation [12]. On the other hand, organ donation policy should be prioritized based on the patients' benefits from the surgery [13, 14]. Effective policies have demonstrated a reduction in the mortality rate [14, 15]. Data mining techniques are widely used for predicting the survival chance of patients. A survival model based on pre-transplantation variables can provide an estimate of the benefits to the candidate recipients of a transplantation surgery [16]. However, potential non-monotonic patterns in the predicted pattern of survival chance of patients in the consecutive timestamps are reported [5, 6]. Interpretation of a non-monotonic survival pattern is challenging which is not encouraging for the adoption of a survival model in the transplantation domain.

The first research question is: "What are the current issues in the development of a data mining study for predicting survival of heart transplantation surgeries?" Numerous data mining approaches for predicting survival of heart transplantation are offered. These approaches could be divided into different combinations for three major parts of data preparation, feature selection and

prediction. Each of them can be divided into different subsets (factors). The major factors of the data preparation part include a data quality check, handling of missing/unknown numerical values, and handling missing/unknown categorical values. In the feature selection, the most important variables that contribute to the prediction of the dependent variables are selected. The major elements of prediction are resampling and model training. Although the general outline of the studies is reported, most researches have not discussed the details in a manner that facilitates replication of the studies. In particular, research in this area is lacking a discussion of the challenges related to data collection and analysis and how those challenges were addressed. Therefore, it is difficult to reproduce the same research and achieve the presented results. Reproducibility of previous research's procedures and results is an essential principle of science [17, 18]. Although the majority of survival analyses were developed by investigating UNOS dataset [19] the literature lacks a reproducible and publically available framework that addresses the issues and publically shares release the details.

The second research question is: "How can an objective approach be utilized to optimize the performance of survival models?" A major significance of this study is providing a reproducible framework for improving the quality of survival prediction studies that potentially could optimize organ allocation [12, 20]. The previous literature has not reported the factors that could affect the performance of machine learning algorithms in the field. Instead, typically different machine learning algorithms are examined and the performance is compared (see papers with multiple methods in Table 1). In addition, a lower performance for prediction of death cases in the earlier years that data is highly imbalanced is reported (e.g. see Dag [2],Dag, Oztekin [3],Tang, Hurdle [14], Brown, Elster [21],Ohno-Machado and Musen [5], Ohno-Machado and Musen [6], and Lin, Horn [22] in table2). The first step in optimizing the performance is

investigating the factors that are associated with model performance and then constructing a response area for the possible results. The second step is searching for the response area in order to find a corner that yields the highest performance. In some studies, DoE is used for optimizing the modeling section of a data mining study (see: *Table 9*). However, complete research that considered all the possible factors in data preparation, feature selection, and prediction has not been reported in the field. As the goal of optimizing prediction performance, through a screening step, all the contributing factors is investigated, then a surface response containing all the possible combination of the factors is searched to find the best combination that yields the best predictive performance. The selection criteria is having a high predictive performance for both survival and death cases.

1.2 Research Objectives

The main goal of this dissertation is delivering a superior predictive model for predicting the survival of heart transplantation surgery by optimizing the contributing factors in a data mining study. This dissertation consists of three major research studies. The first study reviews the major challenges in transplant survival studies and highlights areas for improvement. The second study presents the development of a framework that considers factors associated with the various steps of a data mining study in the transplantation field and the optimization of the survival prediction performance for both death and survival cases. The last study is dedicated to investigating the pattern of predicted survivals for consecutive timestamps and how best to deliver a monotonically decreasing survival pattern that does not reduce the performance of predictions. The summary of the dissertation' objectives is as below:

- Present current issues for the development of a data mining study in the transplantation field, and deliver objective countermeasures to address issues that impede data analysis.
- Investigate the factors that are contributing to the performance of a data mining model.
- Study the pre-transplant variables that are significant in predicting post-transplantation survival.
- Compare the performance of various machine learning algorithms applied to the contributing factors and select the best combination of factors.
- Facilitate interpretation of predicted survival over consecutive timestamps by providing a monotonically decreasing survival pattern.
- Provide the source codes and details of the steps to facilitate reproduction of the results by other researchers.

1.3 Dissertation Layout

The remaining of the dissertation is structured as follows: Chapter 2 reviews the literature and discusses issues related to the development of data mining projects in the transplantation field. Chapter 3 presents the application of DoE in identifying the factors that contribute to the performance of a survival prediction model. Then, the significant factors are selected to construct a response surface. The corners of the response surface are investigated to find the point that yields the highest predictive performance. Chapter 4 highlights the monotonicity issue and addresses it by employing isotonic regression. Chapter 5 summarizes the conclusions and discusses the implementations and limitations of the contributions. Finally, recommendations and outlines for future studies in the field are provided in this chapter.

Chapter 2

Current Challenges in Development of a Survival Analysis Data Mining Study

2.1 Abstract

This chapter highlights the issues and challenges that are associated with the development of a data mining (DM) study in the transplantation field. The UNOS dataset was investigated for the development of a data mining study to predict survival of patients within one year after heart transplantation surgery. CRISP-DM (CRoss Industry Standard Process for Data Mining) [23] was adopted as the outline of the DM based survival analysis. These challenges are focused on the three parts of CRISP-DM: 1- the data understanding part: understanding missing data (systematic, and random), and data definitions, 2- the data preparation part: handling uninterpretable data (such as extremely light patients etc.), encoding the data, and missing data, and 3- the development of predictive models part: feature selection and developing machine learning algorithms. For the sake of readily reproducing the results, the details of how data were processed and handled are documented using R programming and provided publically.

2.2 Introduction

The major goal of this chapter is to highlight the major challenges in machine learning (ML) based heart transplantation (HT) survival prediction and analysis over the UNOS database. The main motivations are the importance of HT, lack of a standard framework for addressing the issues, and improving data mining research for the survival of HT. Because, improving performance of such studies could potentially improve the organ allocation procedures [1, 16]. Despite several recent studies for transplantation survival analysis, the challenges of these studies

are scarcely mentioned and based of the author's best knowledge no detailed and reproducible approach for addressing these challenges in the field has been reported.

HT is a critical surgery because a) there are about 6.5 million people with heart disease [24] and, heart failure is a relatively prevalent disease in the USA since b) there is no balanced equilibrium between demand and organ availability [25, 26] and demand is significantly more than the available organs, and c) patients that need HT are in the end-stage of heart failure and all other potential treatments have failed [3]. Survival analysis is a data-driven approach for addressing the mentioned issues. This framework is developed by adopting the CRoss Industry Standard Process for Data Mining (CRISP-DM) [23, 27]. It is a well-established model for ML-based study projects [23, 27]. This standard provides a common language for research scholars to reproduce the procedures and facilitates the development of a specialized ML-based tool for survival analysis.

To use CRISP-DM [23, 27], the first steps in such studies is understanding the purpose ("business understanding") and then understanding the data (background knowledge and/or literature review). There is a close link between them [23]. However, "business understanding" could be updated after investigating the data. In addition, "business understanding" has a pivotal role in the development of the next two steps of data preparation and modeling. Modeling itself has two subsets of feature selection and model training.

According to CRISP-DM [23, 27], data understanding is the next step in ML-based survival analysis. This step is associated with understanding the data source, the data gathering process, itself, and developing insight regarding the data. Most of the transplantation survival analyses are developed by investigating the UNOS data repository [1]. UNOS is a nationwide private, and non-profit organization [28]. UNOS maintains a comprehensive dataset that contains heart transplantation information and subject demographics. The data is derived from data gathering

forms such as waiting list (WL), transplant candidate registration (TCR), transplant recipient registration (TRR), deceased donor registration (DDR), recipient histocompatibility (RH), donor histocompatibility, post-¬transplant Malignancy, transplant recipient follow-ups (TRF), cadaver donor registration (CDR). The gathered data could be categorized into three major groups: 1) numerical data (such as weight), categorical data (such as blood type), and 3) useless data (such as patient ID).

Following instructions of CRISP-DM [23, 27] the next step is data preparation. In this step, the raw UNOS data will be transferred to the proper format for the ML algorithms. Also, after the modeling step, some new ideas and feedbacks may rise for developing new data [23]. However, pre-requisites for the data preparation step are data quality and validation checks. Importance of data quality for the predictive performance of machine learning algorithms has been highlighted in several studies [29-32]. Typos and missing values are common error sources in many data-driven studies [29]. The UNOS database has maintained data of all recipients and donors in the USA since October/01/1987 [33]. Since then some UNOS data has been updated (e.g. see [34]). The updates include adding new variables and removing other variables. However, UNOS keeps all the variables and presents them in spreadsheets of the transplantation data. It potentially could lead to systematic (non-random) missing data. Patients who received transplantation after the obsoletion date of the variables do not have the associated value. Also, data of the patients that had the surgery before the adding of new variables may not have any value for those variables. Human involvement in measurement and data entry results in some random missing data [35] or outliers [36] in the UNOS data. Addressing the mentioned quality issues requires an investigation for ensuring the quality of data for either excluding or repairing them. The exclusion rules could be developed through denial constraints [29] and functional dependencies [37]. Examples of those

are expected thresholds for patient's weight (denial constraints), and correlation of weight and height (functional dependencies). Other major reasons for the missing values in the UNOS dataset are revisions and development of this dataset over the years. Several variables have been introduced, dropped, or redefined [2]. Following the data quality investigation is the handling of missing values. The major approaches use a heuristic method for dropping rows and columns until getting a solid dataset, and/or imputation [38, 39]. Numerical and categorical variables require different techniques for imputation. There are two major imputation approaches that are predictive methods and statistical driven constants. The proposed predictive methods in the literature are highly associated to the domain and characteristics of data and there is no clear and proven indication towards application of them [40]. In the predictive approach, the missing data are predicted by employing machine learning algorithms [41]. However, for a large data set such as UNOS that has more than 400 variables (the number of variables depends on data acquisition time), this approach is very time consuming and is not practical. Instead, statistical constants such as median and mode might be used to impute the missing values [42]. Another approach is labeling the missing value and considering it as a category [42]. This approach is significantly less time intensive for large datasets such as UNOS. If the proportion of unavailable data on importation variables is very large, the analysis might provide biased results [43]. There is no established threshold for an acceptable portion of unavailable data in the literature [44]. However, there are some rules of thumb. For example, Bennett and health [45] stated that statistical analysis when more than 10% of data is missing is likely biased. In addition in some studies, excluding variables or observations that have too many missing values are considered (e.g. see: [46, 47]).

Data encoding is another significant aspect of data preparation. Label encoding and one-hot encoding [48] are the main encoding methods for addressing multi-class variables. In one-hot

encoding, the categories of a variable transform to binary variables for representing the value of the categorical variable. Whereas, in the label encoding a number is assigned to each variable based on the category number. In the UNOS dataset, the majority of variables are categorical and many of them have several levels. In other words, the importance of the variables is distributed over several subcategories. Data-driven dimension reduction methods such as principal component analysis (PCA) could yield good predictive performance [49] however investigating the variables' significance is more challenging. Reducing the dimension by merging the homogenous levels facilitates the interpretation of the predictive models but consolidating the categories requires a solid background knowledge and a comprehensive literature review (data understanding).

Feature selection is an important step for data-driven studies since infusing many variables into a model could greatly increase the training time and reduce the predictive performance by getting into a local "min (max) trap" [50]. Other benefits of feature selections are: improving predictive performance, reducing training time and computing requirements, facilitating data visualization, improving interpretation of data, and preventing overfitting [51, 52]. The feature selection algorithms are generally categorized into three major groups of filter, wrapper and hybrid methods [53-56]. Wrapper algorithms include a machine learning algorithm that selects the features based on predictive performance. However, filter algorithms select the features based on the characteristics of the training dataset [56]. Although the wrapper methods demonstrated superior outcomes, they are computationally expensive [56, 57]. The UNOS dataset is a large database that includes several features and thousands of records. Therefore, many wrapper methods may not be suitable for such kinds of studies.

The last step is training a model. These predictive algorithms are of three main types: ML-based algorithms [58], statistical methods that are mainly derived from Kaplan-Meier [59], and Cox [60],

hybrid models that are developed from ML and statistical models (e.g. IHTSA [61]). Although statistical and hybrid methods could provide acceptable predictive performance however there are some caveats about those models. Interpreting non-linear and complex relationship with the statistical methods is challenging [62]. Interpretation of survival curves developed from Kaplan-Meier are subject to possible distortions and imperfect conclusions [63], and presenting differences between the groups of observations, is prone to bias [64]. Although proportionality of hazard is a fundamental condition for the reliability of Cox based models, it is not typically investigated in the literature [65]. ML-based algorithms have garnered significant attention in recent years. They relax the limitations of statistical and hybrid models. In addition, they could interpret the complex and non-linear relationships among variables [62]. These advantages, have motivated researchers to apply ML algorithms for survival analysis of transplant surgery data.

In the literature, most of the machine learning based survival analysis is designed for predicting the survival chance over a window of time (e.g. see: [3, 5, 6, 8, 14, 20-22, 62]). Intuitively, the chance of survival is decreasing over time. In other words, the majority of people would survive to earlier times than longer periods of time. Therefore, the dependent variable is more imbalanced in the earlier windows of time. Synthetic minority over-sampling (SMOTE) [66], random undersampling (RUS) [67], ROSE [68], random sampling with replacement over the minority class (up-sampling) [69], random subletting of the majority class (down-sampling) [69] are among the most used resampling methods in the literature that could be considered as the countermeasures for this issue. An oversampling algorithm, such as SMOTE is more useful for a balanced algorithm such as logistic regression and support vector machine and it is not beneficial for ensemble models such as random forest [70]. Applying SMOTE for a large dataset like UNOS [71] could be challenging because it intensifies computational complexity [70, 72] for an already large scale

problem and might deteriorate the predictive performance [72, 73]. An oversampling algorithm increases the minority class for balancing the dataset. However, it might cause overfitting that decreases the predictive performance of the machine learning algorithm [72, 73]. In addition, SMOTE potentially yields noise rather than true resembling of the minority class [20, 74, 75]. Machine learning algorithms could generally be categorized in two groups of black-box (such as neural network, and support vector machines) and white-box (such as logistic regression, decision trees based models, and k-nearest neighbors) algorithms [76]. Black-box models often outperform the white-box models however white-box model enables the interpretation of parameters [76]. The interpretation of models' structure is the key reason for the attractiveness of the white-box models over the black-box models in the clinical cases [49, 76]. However, in the heart transplantation case, a higher predictive performance is more desirable [1, 77] therefore the black-box models are a viable alternative for survival analysis. Table 1 provides an overview of the ML-based survival analysis' literature.

Following the modeling step, the next step is evaluation. The outcomes of this step measure the alignment of the results with the project's purpose already identified in the "business understanding" step. The common predictive performance measures for evaluating ML algorithms are: 1) accuracy [58], 2) sensitivity [58], 3) specificity [58], 4) area under the curve (AUC) [93], and geometric mean (GMEAN) [94]. Since in the earlier time majority of the patients survive, and in the long windows of time majority of patients die, having high accuracy does not necessarily reflect a higher prediction performance in both survive/death cases. Yielding simultaneous high sensitivity, and specificity brings confidence about the capability of predicting both survived/death cases [58]. Higher GMEAN or AUC is analogous to higher performance in predicting both classes [93, 94].

Table 1: Overview of Literature in the ML based Survival Analysis

Paper	Method(s)	White Box	Black Box
Yoon, Zame [78]	NNET, LR, DT, RF, AdaBoost, DeepBoost, LogitBoost, XGBoost	Yes	Yes
Oztekin, Al-Ebbini [79]	KNN, NNET, SVM	Yes	Yes
Li, Serpen [80]	BBN	No	Yes
Akl, Ismail [81]	NNET	No	Yes
Lasserre, Arnold [82]	LR, NNET, SVM, RF	Yes	Yes
Khan, Choudhury [83]	GB, RF, NB, LR	Yes	Yes
Goldfarb-Rumyantzev, Scandling [84]	LR, CART	Yes	Yes
Tang, Poynton [85]	LR, NNET, SVM	Yes	Yes
Krikov, Khan [86]	CART	Yes	No
Chi, Street [20]	NNET	No	Yes
Ohno-Machado and Musen [6]	NNET	No	Yes
Ohno-Machado and Musen [5]	NNET	No	Yes
Tang, Hurdle [14]	CART	Yes	No
Oztekin [87]	NNET, SVM	No	Yes
Lin, Horn [22]	LOG, NNET	Yes	Yes
Dag, Topuz [8]	BBN, NNET, CART	Yes	Yes
Brown, Elster [21]	BBN	No	Yes
Agrawal, Al-Bahrani [88]	SVM, NNET, DT, KStar, REPT, RF, ADT, DS, NB, BBN, LR, AdaBoost, LogitBoost, Bagging, RS, RotF	Yes	Yes
Dag, Oztekin [3]	LR, NNET, SVM, CART	Yes	Yes
Misiunas, Oztekin [89]	NNET	No	Yes
Medved, Nugues [90]	NNET	No	Yes
Raji and Chandra [91]	NNET	No	Yes
Cucchetti, Vivarelli [92]	NNET	No	Yes

The last step of CRISP-DM [23, 27] is deployment. This step is associated with documenting the results and distributing the produced knowledge. Reproducibility of the procedures and results is an essential principle of science [17, 18]. Although a significant portion of transplant survival studies is developed by investigating UNOS dataset, the literature lacks a reproducible and publically available framework that addresses the issues and publically releases the details.

2.3 Methodology

This study adopts CRISP-DM [23, 27] as the common language for developing a reproducible ML-based framework. The challenges that arise in each part of such studies are pointed out. Then the details of the procedures for resolving the issues are explained. For replicating the results, the procedures are programmed by R programming language and the R codes provided as an appendix of this study. Figure 1 demonstrates the CRISP-DM model [23] for data mining.

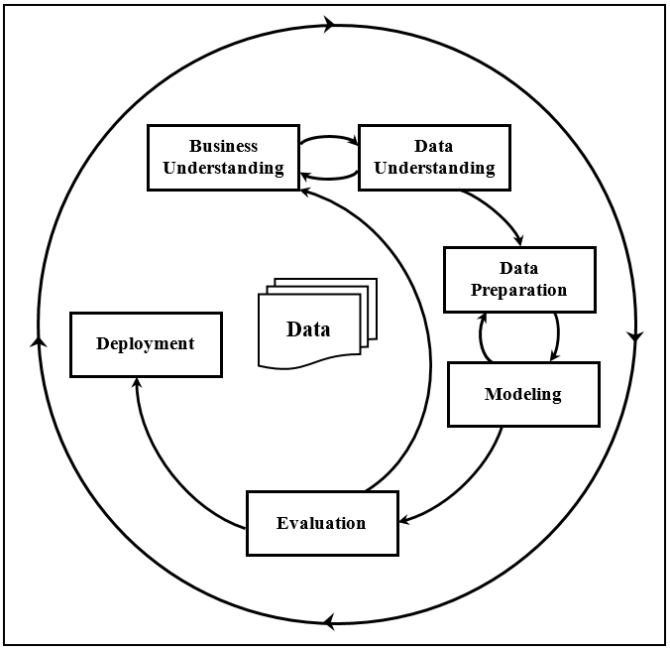


Figure 1: CRISP-DM Model [23]

An extensive literature review is performed for the "business understanding" step. It revealed the scope of these studies and the details of the research processes explained in the field. In the "data understanding" step in addition to performing an extensive descriptive statistics, the data dictionary and Waiting List (WL), Transplant Candidate Registration (TCR), Transplant Recipient Registration (TRR), Deceased Donor Registration (DDR), Recipient Histocompatibility (RH), donor histocompatibility, post-transplant Malignancy, Transplant Recipient Follow-ups (TRF),

and Cadaver Donor Registration (CDR) forms are studied to understand the interpretation of the variables' values.

The data preparation step consisted of exclusions (dropping undesired variables, and the observations that are outliers), solidifying the data-table, and encoding. The exclusions have two major significant benefits: a) reduces the size of the problem, b) improve the predictions' interpretation, validation, and calculation time. Validity and quality of models that have irrelevant variables such as patients' identification number or adult (age more than 18) patients with 12 kg (26.5 lb) weight are questionable. Solidifying the data-table means addressing the missing values. There are two major approaches for solidifying the data-table: a) imputation b) heuristically dropping rows and/or columns.

For reproducing the results detail of this chapter is provided in the following GitHub link: https://github.com/transplantation/unos-ht

2.3.1 Variable Exclusion

For avoiding systematic missing values, the variables that are ended or added after 2000 are excluded from the study. The associated variables in UNOS are "VAR.END.DATE" and "VAR.START.DATE". here is the pseudocode:

$$\begin{aligned} &\textit{if ``VAR.END.DATE'' exist} \rightarrow \textit{drop the variable} \\ &\textit{if ``VAR.START.DATE''} > 2000 \rightarrow \textit{drop the variable} \end{aligned} \tag{1}$$

The second variable exclusion rule is based on the proportion of unavailable data. Variables that have too many missing values are excluded as well. The denial threshold is 90 percent.

$$\textit{if for more than 90\% of the observations are not available} \rightarrow \textit{drop the variable} \tag{2}$$

The third variable exclusion rule is based on the distribution of levels in the categorical variables. Categorical variables that most of the observations just have one level of their categories are excluded as well. Similar to the previous variable exclusion rule, the denial threshold is 90 percent.

if more than 90% of the observations belong to one category $\rightarrow$ *drop the variable* (3)

2.3.1 Observation Exclusion

Through descriptive statistics, some observations were labeled suspected of non-systematics errors. Such as adult patients who are potentially unnaturally too short or too light. In addition, patients whose donors were unnaturally too light. They are considered as outlier observations and excluded from further investigations. For this purpose, the patients that belong to the lightest and shortest 0.01-th percentile of the observation are considered as the outliers and excluded from further investigations. Other constraints for excluding observations were: only adult patients and only Heart organ for investigating the thoracic database of UNOS.

The pseudocode s of observations' exclusion are:

- focusing on adult patients, the UNOS variable is "AGE".

$$if\ "AGE" < 18\ \rightarrow\ drop\ the\ variable \quad (4)$$

- too light patients are excluded, the UNOS variable is "WGT_KG_DON_CALC":

$$if\ \text{"WGT\_KG\_DON\_CALC" is less than } 0.01-\text{th percentile} \rightarrow \text{drop the observation} \quad (5)$$

- too short patients are excluded, the UNOS variable is "HGT_CM_TCR":

$$if\ \text{"HGT\_CM\_TCR" is less than } 0.01-\text{th percentile} \rightarrow \text{drop the observation} \quad (6)$$

- exclusion if the donor is too light, the UNOS variable is “WGT_KG_DON_CALC”:

$$if \text{ "WGT\_KG\_DON\_CALC " is less than } 0.01 - \text{th percentile} \rightarrow \text{drop the observation} \quad (7)$$

- focusing on heart transplantation only, the UNOS variable is “WL_ORG”:

$$if \text{ WL\_ORG" is not HR" (heart)} \rightarrow \text{drop the observation} \quad (8)$$

In addition, there are some variables such as patients ID, and waiting list code, and etc. that are not considered as relevant for training a predictive model are excluded from further investigation. The details of the initial decision about inclusion/exclusion of the variables are provided in the appendix.

2.3.2 Solidifying the Data-Tables through Imputation

Imputation by through statistical driven constants is adopted for imputing unavailable data. In addition, for the categorical variables, if the categorical variables labeling the unavailable is considered. The labeling could be performed in two different ways: a) label all the unavailable data cells as a label (“UNKNOWN”), b) differentiate between the data cells that are missing and are unknown (two labels of “MISSING” and “UNKNOWN”). The statistically driven constants for the numerical variables is mode, and median. The statistically driven constant for the categorical variables is mode.

2.3.3 Solidifying the Data-Tables through Dropping Rows and Columns

Some researchers avoided imputation, and rather to work with the reported data (e.g. see [2]). One approach is dropping the rows and columns and trying to maximize the amount of the remaining data. In this study, a heuristic approach programmed in R is adopted. In this approach, through a loop every time the emptiest rows and columns are calculated and consequently the emptiest one is dropped until a solid data-table remains. The pseudocode is provided below:

while *there are rows or columns with unavailable cells* **do**:

calculate the portion of unavailable cells for each row

calculate the portion of unavailable cells for each column (9)

drop the row or columns that has the most unavailable cells

end

2.3.4 Encoding

Categorical variables are processed through both label encoding and one-hot encoding [48] in this study. In the UNOS dataset, some of the variables have several categories. Merging the levels of categorical variables that are homogenous with each other is a primary step before hard encoding [95]. For instance, the state of the patients is a categorical variable that has several levels. One-hot encoding and driving more than 50 dummy variables not only makes the problem much more difficult. It may reduce the importance of patients' location. Instead, states levels are consolidated into geographical regions that have significantly fewer categories. Then hard encoding and label encoding are applied.

2.3.5 Feature selection

In this study, RF, Fast Correlation-Based Features (FCBF) [56], and LASSO [96] are employed as a wrapper, filter, and embedded algorithm for feature selection, consequently.

In feature selection with random forest the variable importance of X^j is articulated as below [97]:

$$\frac{1}{ntree}\sum_{t}(err\ O\tilde{O}\ B_t^j - errOO\ B_t) \tag{10}$$

where:

t: sample tree

$errOO\ B_t$: misclassification rate

Boruta package of R software as a fast implementation of RF feature selection is adopted [98]. Boruta shuffles some features and selects them as the shadow variables. Then it constructs the maximum Z score among shadow attributes (MZSA) and assigned a hit to all attributes that have a better MZSA. Variables that are significantly more than MZSA are confirmed as the selected features. Through the iterations, it randomly selects new shadow variables until either only confirmed variables are left or it reaches the maximum number of runs [98].

In FCBF, features are selected based on their correlation to the dependent variable. The correlation measure is based on the concept of entropy and information gain [56]. The amount of X entropy that decreases after exposing to variable Y is considered as information gain (IG):

$$IG(X|Y) = H(X) - H(X|Y) \tag{11}$$

where H refers to entropy.

LASSO fits a generalized linear model through a penalized likelihood function [96]. The elastic mixing parameter (α) is 1 for LASSO. The penalty function is defined as:

$$penalty = \frac{1 - \alpha}{2}||\beta||_2^2 + \alpha||\beta||_1 \quad (12)$$

The algorithm solves the following problem with a set of features that optimize it [96].

$$\min_{\beta_0,\beta} \frac{1}{N}\sum_{i=1}^{N} w_i l\,(y_i, \beta_0 + \beta^T x_i) + \lambda * penalty \quad (13)$$

where:

β_j: jth row of the coefficient matrix β

x_i: independent variable *i*

$l(y, \eta)$: negative log-likelihood contribution for observation *i*

λ: tuning parameter

w_i: weight

The ridge penalty reduces the coefficient of predictors that are correlated while LASSO picks the one that discards the others.

2.3.6 Training the prediction model

There is no rule of thumb for selecting the best algorithm, some of the black box models are very time consuming versus white box models that are generally faster. LR, NNET, and RF that represent white box, black box, and ensemble models are selected, respectively. The intention of this study is discussing the challenges of delivering the best model with the highest performance.

Optimization algorithms for such these experiments might bring better insights into the performance of them.

LR is a simple and relatively fast algorithm and it has shown promising results for the large datasets in comparison with recently developed classification algorithms [99].

Suppose dependent of X has M features and R rows, the logistic curve that models the probability of belonging to either category of X is written as below [99]:

$$\mu_i = \frac{\exp(\beta_0 + \beta_1 x_{i1} + \cdots \beta_M x_{iM})}{1 + \exp(\beta_0 + \beta_1 x_{i1} + \cdots \beta_M x_{iM})} \quad (14)$$

The maximum likelihood equations for logistic regression are:

$$\sum_{i=1}^{R} (y_i - \mu_i) = 0 \quad (15)$$

$$\sum_{i=1}^{R} (x_{ij} - \mu_i) = 0, \text{ j} = 1 \ldots \text{M} \quad (16)$$

where:

β_i: coefficients of the independent variables

NNET mimics the architecture of human neurons for transferring the received data into information. It consisted of an input layer, a hidden layer(s), and an output layer. The input layer receives the data and then the weighted sum of the input layer's node transferred to the nodes of the hidden layer(s) [100].

$$X_j = \sum_{i=1}^{j} W_{ij} X_i \quad (17)$$

In cases where there are multiple hidden layers, the first hidden layer is considered as an input layer and with similar logic, its nodes transfer the weighted sums to the next layer. Finally, The

output layer receives the weighted sum of all the previous layers. If the dependent variable is dichotomous, this value could be formed as a probability of belonging to either category of the variable. It also could a trigger function that decides about the category of the data. Figure 2 represents a simple neural network with one hidden layer.

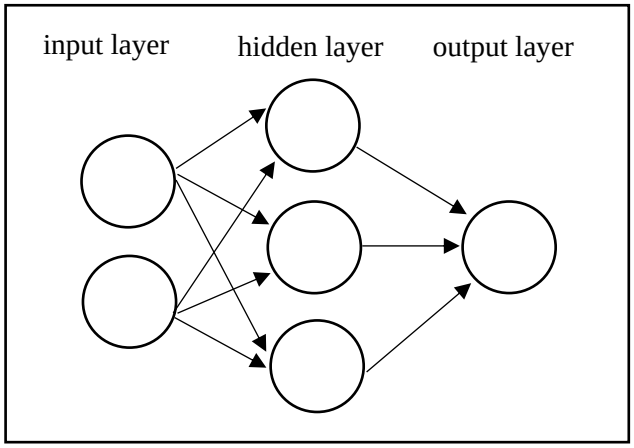


Figure 2: A simple neural network [100]

RF is an ensemble model that includes the development of multiple decision trees (forest) through bagging (bootstrap aggregation). Each tree of the forest vote for a class of the dependent variable. RF decides about the class of the dependent variable based on the majority of the votes [101].

The margin function of the random forest classifier is [102]:

$$mr(X,Y) = P_{\Theta}(h(X,\Theta) = Y) - max_{j \neq Y} P_{\Theta}(h(X,\Theta) = j) \quad (18)$$

The strength of the classifiers' strength is:

$$s = E_{X,Y}\, \mathrm{mr}(X,Y) \quad (19)$$

where:

Θ_k: a random generated for kth tree

$h(X, \Theta_k)$: a classifier generated for the input (x)

2.3.7 Evaluation

For comparing the performance of the algorithms, four performance measures of accuracy, sensitivity, specificity, AUC, and GMEAN are considered. Table 2 reflects the distribution of predictions and the actual dependent variable with two class of positive and negative.

Table 2: Confusion Matrix

		Prediction	
		negative	**positive**
Actual Class	**negative**	TN	FP
	positive	FN	TP

Then:

$$accuracy = \frac{TN + TP}{(TN + TP + FP + FN)} \tag{20}$$

$$sensitivity = \frac{TP}{(TP + FN)} \tag{21}$$

$$specificity = \frac{TN}{(TN + FP)} \tag{22}$$

$$GMEAN = \sqrt{sensitivity \times specificity} \tag{23}$$

As demonstrated in Figure 3, the area under the receiver operating characteristics (ROC) curve (AUC) is a measure for performance of a model in distinguishing between two classes of the dependent variable. The curve is constructed based on changing the threshold level for making a decision about the class of the dependent variable. The higher the AUC, the higher overall accuracy in the model [103].

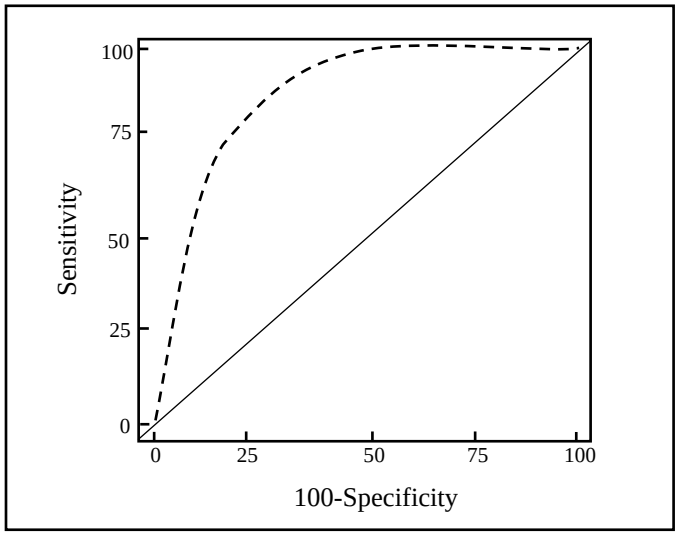


Figure 3: ROC curve [103]

2.4 Results

The acquired UNOS dataset consisted of 159318 heart transplantation records. The earliest and latest transplantation date were “1987-10-01 “ and “2016-09-30 “. There are 494 variables in the dataset. These data are either gathered from UNOS forms or calculated based on the data in the forms. Here are the distributions of variables’ source and the added date.

Different dates for adding and dropping variables causes systematic missing (unavailable) data. To minimize this effect, only the variables that are not ended and added before 2000 are considered in this study. In addition, the undesired variables are excluded. The undesired variables are a) the ones that are not relevant (such as patient ID), b) variables that more than 90% of their data is unavailable, c) categorical variables that more than 90% of data belongs just to one category, and d) post-transplantation variables. The details of these exclusions are provided in the appendix of this chapter. Tables, 4, and 5 provide a summary about source of entering and removal daes gathering and date distribution if the UNOS variables.

Table 3: UNOS variables' Source

Form	Frequency
CALCULA-TED	89
CALCULA-TED TCR	1
CALCULA-TED TRR	1
CDR/LDR	3
DDR	136
DDR/LDR	5
DDR/LDR-CALCULA-TED	2
LDF	1
LDR	19
RH	16
NO SOURCE DATA	3
TCR	78
TCR/TRR	1
TCR-CALCULA-TED	1
TRF	5
TRF/TRR	6
TRR	84
TRR/TRF-CALCULA-TED	4
TRR>TCR	3
TRR-CALCULA-TED	2
WAITING LIST	34
Total	494

Table 4: adding dates of the variables

Year	Frequency
1987	56
1990	9
1994	129
1995	10
1997	2
1999	60
2003	9
2004	75
2005	4
2006	4
2008	4
2015	10
2016	1
NO YEAR DATA	121

Table 5: ending dates of the variables

Year	Frequency
1999	4
2003	6
2004	26
2007	3
2015	9
NO YEAR DATA	446

After excluding the undesired observation, 133 variables and 45,089 observations remained. The last step of data preparation was solidifying the data table (either impute or drop the rows and columns and encoding). Numerical data were either imputed by the median or dropped (a 2-level factor), the categorical variables are either imputer by mode or labeled as MISSING, and UNKNOWN, or dropped (a 4-level factor), and then both label and one-hot encoding (a 2-level factor) were performed. Therefore, the total different combinations of the data preparation are $2 \times$

$4 \times 2 = 16,$ that is equal to the number of developed datasets. The dimension of the dataset is provided in Table 6 (including an added ID column):

Table 6: detail of data preparation scenarios

Scenario No.	Encoding	Categorical Process	Numerical Process	Rows no.	Columns No.
1	label	labeled "UNKNOWN"	drop	26829	134
2	label	labeled "MISSING"	drop	26829	134
3	label	imputed by mode	drop	26829	134
4	label	drop	drop	29214	90
5	label	labeled "UNKNOWN"	imputed by median	45089	134
6	label	labeled "MISSING"	imputed by median	45089	134
7	label	imputed by mode	imputed by median	45089	134
8	label	drop	imputed by median	37502	90
9	one-hot	labeled "UNKNOWN"	drop	26829	264
10	one-hot	labeled "MISSING"	drop	26829	269
11	one-hot	imputed by mode	drop	26829	269
12	one-hot	drop	drop	29214	152
13	one-hot	labeled "UNKNOWN"	imputed by median	45089	278
14	one-hot	labeled "MISSING"	imputed by median	45089	286
15	one-hot	imputed by mode	imputed by median	45089	220
16	one-hot	drop	imputed by median	37502	152

RF, FCBF, and LASSO algorithms are applied for feature selection and the details of the selected variables are provided in the GitHub link of this study. LR, NNET, and RF (a 3-level factor) with 5-fold cross validation over all the combinations of encoding (a 2-level factor), feature selection (a 3-level factor), numerical treatment (a 2-level factor), categorical treatment (a 4-level factor), and resampling (a 5-level factor) is performed. The total number of models that trained in this study is:

$$3 \times 5 \times 2 \times 3 \times 2 \times 4 \times 5 = 3600 \quad (24)$$

Table 7 provides information about the best models based on yielding the best accuracy, the best sensitivity, the best specificity, the best AUC, and the best GMEAN:

Table 7: the best model performance based on different measures

LR	encoding	feature selection	numerical treatment	categorical treatment	resampling	AUC	sensitivity	specificity	accuracy	GMEAN
best_accuracy	LABEL	RF	DROP	UNKNOWN	none	62.84%	0.43%	99.88%	87.93%	5.57%
best_sensitivity	HARD	LASSO	MEDIAN	UNKNOWN	down	65.08%	60.15%	61.68%	61.47%	60.91%
best_specificity	LABEL	RF	DROP	DROP	none	59.40%	0.11%	99.97%	87.75%	1.48%
best_AUC	HARD	LASSO	MEDIAN	UNKNOWN	up	65.47%	59.27%	62.88%	62.39%	61.04%
best_GMEAN	HARD	LASSO	MEDIAN	UNKNOWN	up	65.47%	59.27%	62.88%	62.39%	61.04%

NNET	encoding	feature selection	numerical treatment	categorical treatment	resampling	AUC	sensitivity	specificity	accuracy	GMEAN
best_accuracy	LABEL	FFS	DROP	UNKNOWN	none	62.12%	0.37%	99.95%	87.99%	4.53%
best_sensitivity	HARD	LASSO	DROP	MODE	up	60.26%	69.09%	46.22%	49.00%	47.50%
best_specificity	HARD	FFS	DROP	DROP	none	59.68%	0.00%	100.00%	87.76%	0.00%
best_AUC	HARD	LASSO	DROP	MODE	none	64.04%	0.18%	99.94%	87.95%	1.89%
best_GMEAN	HARD	LASSO	DROP	UNKNOWN	up	61.56%	61.86%	56.19%	56.92%	58.65%

RF	encoding	feature selection	numerical treatment	categorical treatment	resampling	AUC	sensitivity	specificity	accuracy	GMEAN
best_accuracy	LABEL	LASSO	DROP	MODE	none	62.81%	1.40%	99.85%	88.02%	11.76%
best_sensitivity	LABEL	LASSO	MEDIAN	DROP	down	64.18%	61.23%	59.52%	59.75%	60.36%
best_specificity	HARD	LASSO	DROP	DROP	none	60.71%	0.25%	99.94%	87.75%	4.36%
best_AUC	HARD	LASSO	MEDIAN	UNKNOWN	down	64.93%	59.84%	61.44%	61.22%	60.63%
best_GMEAN	HARD	LASSO	MEDIAN	MISSING	down	64.92%	60.33%	61.29%	61.16%	60.81%

In the previous table, each row is associated with the logic for selecting the best model. Despite, being the simplest and fastest algorithm, LR was also the best performing model. In addition, the models that had the best accuracy, AUC, and specificity could not always bring balanced prediction capabilities in predicting both survivals (measured by specificity), and death (measured by sensitivity) cases.

Table 8 demonstrates the standard deviation of all the models.

Table 8: standard deviation of the models

standard deviation	AUC	sensitivity	specificity	accuracy	GMEAN
LR	9.92%	21.16%	21.21%	16.45%	20.06%
NNET	6.69%	20.24%	16.26%	11.69%	20.26%
RF	2.41%	21.20%	15.69%	11.00%	17.60%

The performance measures have a fairly large standard deviation. However, AUC has the smallest variability. The fairly large standard deviation shows how different configurations of the factors, could lead to a wide range of predictive performance in these data mining studies.

2.5 Discussion and Future Studies

In this research, different treatments for addressing common issues in developing a data mining study are provided. The issues are categorized based on the CRISP-DM model [23] for data mining. To reproduce the results, details of the development process that is programmed in R is provided in the appendix. Distinct decisions for handling the issues are provided. The decisions are named factors. There are different methods for each decision (factor) that are called levels. Five performance measures of "accuracy," "sensitivity," "specificity," "accuracy," and "GMEAN" were elected for determining the best model. However, selecting the best model based on AUC and GMEAN led to models that have more balanced capabilities in predicting in both death, and survival cases. In addition, the variability of the results is fairly high, that demonstrate the importance of selecting appropriate factors' level for addressing the issues. Therefore, the data mining issues and the way that researchers respond to them may significantly affect the quality of survival models. Another highlight of the results was the competitive performance of LR in comparison with more sophisticated models of NNET and RF. In other words, in addition to an advanced training algorithm, other factors in the development of a data mining study may significantly contribute in the predictive performance of survival model and their interaction with the training models should be studied. Further study to understand the significance of the factors and optimizing the performance of survival models could yield better models. Design of experiments (DoE) [4] could be employed for reaching this goal.

Chapter 3

A systematic approach for Optimizing Performance of Data Mining Studies in Survival Analysis of Heart Transplantation

3.1 Abstract

This study presents an approach for optimizing the predictive performance of data mining studies that are developed for the survival of heart transplantation surgeries. This data-mining project is viewed as an experiment with consecutive steps (factors) as described in CRISP-DM (CRoss Industry Standard Process for Data Mining) [23]. Since there are multiple options (levels) for development of each step, Design of Experiments (DoE) [4] is employed to analyze the significance of the factors and optimize the performance of the studies. The desired performance is defined as the highest accuracy for predicting both survival and death cases after the transplantation surgery with the only pre-transplantation variables. Survival outcome of surgeries (survival/death) within one year after the surgery is the dependent variable. The introduced approach showed the significance of the data mining factors that are employed in this study and yielded 1-year post-transplantation survival prediction with 59.27% sensitivity and 62.88% specificity for this challenging problem.

3.2 Introduction

This study adopts DoE as a systematic approach for improving the predictive performance of heart transplantation (HT) survival studies over United Network for Organ Sharing (UNOS) [19]. The scope of the study is limited to the survival chance of patients within one year after transplantation surgery. Improving the predictive performance is an important problem. Because

it would contribute to the optimization of the organ allocation process [1, 16]. HT is the last treatment of a severe heart failure disease [3], organ supply is substantially less than the demand [25, 26], and about half of the offered organs could not be transplanted due to the conservative allocation policies [12]. The previous research has shown that the predictive performance of machine learning algorithms could be improved by selecting a better combination of the factors (e.g. see [2, 3, 88]).

In accordance with CRISP-DM [23], the HT survival analysis is considered a data mining experiment. The elements of data preparation and modeling in CRISP-DM is considered as the potential factors that could contribute to the predictive performance. Then the significance of the factors and their interactions in survival predictive performance are investigated through DoE. Then the best combinations of the contributing factors' levels with the highest predictive performance is presented. The factors that are investigated in the data preparation section of CRISP-DM are 1) the method for addressing unavailable numerical (factor-1) and categorical (factor-2) data, and 2) the encoding method (factor-3). The factors that are investigated for modeling section of CRISP-DM are 1) feature selection method (factor-4), 2) resampling method (factor-5), and 3) training algorithm (factor-6).

There are many unavailable numerical and categorical data in UNOS. The unavailable data is comprised of missing, unknown, or non-applicable observations. A major reason for systematic unavailable observations in UNOS is multiple updates [34] that have occurred since the establishment date of this data repository. UNOS has stopped a certain number of variables (the "obsolete" variables) and defined new variables for data gathering in certain occasions [34]. However, UNOS delivers the data repository with all the defined variables even if they are currently obsolete. Therefore, for the patients that had surgery in the earlier dates, and in the latest

dates, the newly defined, and the obsolete variables are not available consecutively. Exclusion (variables and/or observations) and imputation are the major methods (levels) for the (factor-1 and factor-2) (e.g. see: [3] and [10]).

Data encoding is another significant aspect of data preparation. Label encoding and one-hot encoding (levels of factor-3) [48] are the main encoding methods to prepare the multi-class categorical variables for the following steps of a data mining study. In the UNOS dataset, the majority of variables are categorical and many of them have several levels. Therefore, the importance of the variables is distributed over several levels. Recoding the categorical variables into a lower number of the levels [104, 105] reduces the number of created dichotomous variables after one-hot encoding. However, it requires a solid understanding and background knowledge of the field. The next step (factor-4) is feature selection. It could potentially reduce the training time, improve performance, and prevent overfitting problems [51, 52]. Feature selection does not affect the originality of data. Therefore, this is a preferable approach in comparison with other dimension reduction algorithms such linear discriminant analysis (LDA) [106] and Principal component analysis (PCA) [107, 108] when the interpretation of the variables is important. Feature selection algorithms are generally categorized into three major groups (levels of factor-4) [109]: filter, wrapper and hybrid methods [53-56]. Wrapper algorithms include a machine learning algorithm that selects features based on predictive performance and is associated with the characteristics of the training dataset [56]. Although the wrapper methods could demonstrate superior outcomes, they are computationally expensive [56, 57]. The UNOS dataset is a large database that includes several features and thousands of records. Therefore, many wrapper methods might be overwhelmingly time-consuming. As presented in Table 9, many researchers used the concept of DoE in tuning the training algorithms and feature selection.

Table 9: Application of DoE in Data Mining Studies

Paper	DoE Application	Predictive Algorithm
Allias, Megat [114]	Feature Selection	SVM
Chuang, Yang [115]	Feature Selection	KNN
Kwak and Choi [116]	Feature Selection	NNET
Yang, Huang [117]	Feature Selection	NNS
Suzuki and Ryu [118]	Feature Selection	Regression
Maji, Mitra [119]	Feature Selection	NB
Packianather, Drake [120]	Model Tuning	NNET
Wang, Stockton [121]	Model Tuning	NNET
Yang, Lee [122]	Model Tuning	NNET
Khaw, Lim [123]	Model Tuning	NNET
Peterson, Clair [124]	Model Tuning	NNET
Packianather, Drake [120]	Model Tuning	NNET
Tortum, Yayla [125]	Model Tuning	NNET
Bashiri and Geranmayeh [126]	Model Tuning	NNET
Otok, Ulama [127]	Model Tuning	NNET
Ortiz-Rodríguez, Martínez-Blanco [128]	Model Tuning	NNET
Pontes, Amorim [129]	Model Tuning	NNET
Tyasnurita, Özcan [130]	Model Tuning	NNET
Ardalani-Farsa, Zolfaghari [131]	Model Tuning	Ensemble NNET
Kumar, Gupta [132]	Model Tuning	NNET
Balestrassi, Popova [133]	Model Tuning	NNET
Chan, Khadem [134]	Model Tuning	NNET
Lin, Ping [135]	Model Tuning	NNET
Peterson, Clair [124]	Model Tuning	NNET
Sukthomya, Tannock [136]	Model Tuning	NNET
Kim and Yum [137]	Model Tuning	NNET
Hsu and Yu [138]	Model Tuning	SVM
Huang, Hung [139]	Model Tuning	SVM
Huang, Hung [140]	Model Tuning	SVM
Erfanifard, Behnia [141]	Model Tuning	SVM
Zare, Behnia [142]	Model Tuning	SVM

Currently, 84.5% of the patients survive the one-year timestamps [110]; Therefore, the dependent variable is quite imbalanced and the training process is challenging [111]. Resampling algorithms

(factor-5) could be an option to address the issue of the imbalanced dataset in this study. These methods adjust the distribution of datasets [112, 113]. Synthetic minority over-sampling (SMOTE) [66], random undersampling (RUS) [67], ROSE [68], random sampling with replacement over the minority class (up-sampling) [69], random subletting of the majority class (down-sampling) [69] (levels of factor-5) are among the most used resampling methods in the literature.

The last factor is training an appropriate algorithm for predicting the chance of survival within one-year after the transplantation surgeries. This type of survival analysis is a supervised learning problem for classifying patients in two groups of death and survival. Supervised learning algorithms for the dependent variable could be categorized in statistical algorithms such as Cox (for continuous dependent variables) [60], Logistic regression (LR) [143], Linear Discriminant Analysis (LDA) [144], and Lasso and Elastic-Net Regularized Generalized Linear (GLMNET) [145], single classifiers such as Neural Network (NNET) [146, 147], Naïve-Bayes (NB) [148], Support Vector Machines (SVM) [146, 147], and Classification & Regression Trees (CART) [146, 147], and Partial Least Squares and Principal Component Regression Kernel (KPLS) [149], and ensemble algorithms such as Random Forest (RF) [150], Gradient Boosting (GB) [151], and eXtreme Gradient Boosting (XGB) [152]. Any of these algorithms could be considered as a level of the factor in a DoE study.

Design of Experiments (DoE) [4] is the major methodology behind the framework of this study. The factors consist of several options for data preparation, feature selection, resampling, and training models. DoE is utilized to demonstrate the significance of those factors and their interactions in the outcomes of the experiments (i.e. prediction performance). Several combinations of the significant factors are investigated to obtain the factor level that yields the highest outcome.

In the manufacturing context, DoE is developed to study the effects of multiple factors and their interactions on a specific response or quality of a process or a product [153]. In the data mining literature, DoE provided promising improvements in several data mining studies through investigating modeling or feature selection sections of a data mining study (see Table 9). However, based on the author's best knowledge, a comprehensive DoE that considers all the possible factors of data preparation, feature selection, and prediction parts has not been reported in the HT field nor has such methodology been applied to the UNOS dataset. The described factors in this study consisted of many levels that may contribute to the predictive performance. In other words, the variation of the factors might affect the variation of the predictive performance outcome. Identifying the significant factors and investigating them rather than all of the possible factors significantly reduces the number of required experiments [154]. Screening design facilitates the identification of the key factors with a substantially lower number of experiments [155]. A two-level DoE is a suitable tool for the screening process [154].

As the goal of improving prediction performance, through a screening step, all the contributing factors are investigated, then a set containing all the possible combination of the factors is searched to find the best combination that yields the highest predictive performance. The selection criteria is a high predictive performance for both survival and death cases.

3.3 Methodology

This study adopted DoE in order to investigate the significance of factors associated with data preparation and modeling parts of CRISP-DM in the variability of predictive performance. The significant factors are identified through a screening process with a two-level design for each variable. Then a full factorial design is developed to search for the best combination of the factors'

level that maximized the prediction performance. Both screening and full factorial design model the observed predictive performance as the dependent of the factors through a multiple linear regression equation. The general format of the question is presented as [4]:

$$y_i = \beta_0 + \sum_{j=1}^{k} \beta_j x_{ij} + \epsilon_i \quad i = 1,2, \dots, n \tag{25}$$

Where:

x_{ij} : ith level of variable x_j

$\in$: the error term

y : response

The hypothesis and the associated test for the significance of the regression are [4]:

H_0: $\beta_1 = \beta_2 = 0$

H_a: $\beta_j \neq 0$ [for at least one j]

$$F_0 = \frac{SS_R/k}{SS_E/(n-k-1)} \tag{26}$$

Where:

β_j : $factor\ jth$

SS_R : $regression\ some\ of\ squares$

SS_E : $residual\ some\ of\ squares$

The hypothesis and the associated test for the significance of each factor are [4]:

H_0: $\beta_j = 0$

H_a: $\beta_j \neq 0$

$$t_0 = \frac{\widehat{\beta}_j}{\sqrt{\hat{\sigma}^2 C_{jj}}} \tag{27}$$

Where:

$\beta_j : factor\ jth$

$\sqrt{\hat{\sigma}^2\ C_{jj}} : standard\ error\ of\ the\ resgression$

The considered factors are associated with a) data preparation (numerical treatment, categorical treatment, and encoding); b) feature selection (filter, wrapper, and hybrid methods); and c) modeling (resampling, training algorithm methods). The numerical treatment is a two-level factor. The unavailable numerical variables are either imputed by a median or dropped. The unavailable categorical variables are either imputed by mode, labeled as an unavailable category, or dropped. Some of the unavailable categorical values are due to unknown values or missing. Then the unavailable categorical variables are either labeled as "UNKNOWN" or two-class label of "UNKNOWN/MISSING" which is considered for differentiating between the unknown and the missing values. Therefore, categorical treatment is a four-level factor. One-hot encoding and label encoding [48] (a two-level factor) are adopted for preparing categorical variables for the feature selection and training. In one-hot encoding, the levels of categorical variables are represented by a dichotomous variable. However, in the numerical encoding, instead of multiple new binary variables, a numerical value represents a categorical variable.

Feature selection is an important step for data-driven studies since infusing many variables into a model could highly increase the training time and reduce the predictive performance by getting into the local min (max) trap [50]. The feature selection has three levels of wrapper, filter, and embedded method. Implementation of RF through the Boruta package [98] of R is selected as the wrapper method [97]. In this method importance of variable X^j is calculated as below [97]:

$$\frac{1}{ntree}\sum_{t}(err\ O\tilde{O}\ B_t^j - errOO\ B_t) \tag{28}$$

where:

t: sample tree

$errOO\ B_t$: misclassification rate

The Boruta package calculates the mean and standard deviation of the importance to calculate the Z score. It then randomly selects shadow variables and finds the maximum Z score and names it MZSA. If the variable's Z-score is significantly larger than MZSA, it will be selected as an important feature.

FCBF is assigned as the filter method [56]. In this method, features are selected based on their correlation to the dependent variable. The correlation is developed through the information gain theory [56]. The conditional information gain equation is:

$$IG(X|Y) = H(X) - H(X|Y) \tag{29}$$

where:

H: entropy

For the embedded level, LASSO [96] feature selection algorithm is adopted. LASSO select the features that maximize the following equation [96]:

$$\min_{\beta_0,\beta} \frac{1}{N}\sum_{i=1}^{N} w_i l\,(y_i, \beta_0 + \beta^T x_i) + \lambda * penalty \tag{30}$$

The penalty equation is:

$$penalty = \frac{1-\alpha}{2}||\beta||_2^2 + \alpha||\beta||_1 \tag{31}$$

In this study, five different approaches are considered for dealing with imbalanced data: no-sample balancing, RUS (down-sampling) [156], SMOTE [157], ROSE, and up-sampling (a five-level factor). With RUS, a subset of the majority class with the size of the minority class was randomly selected to balance the distribution of the dependent variable [156]. SMOTE constructs a vector between a data point and one of the k nearest neighbors then multiplies the vector to a random number between 0 and 1 to produce a synthetic data point [157]. ROSE is a bootstrap-based technique that creates artificial random samples through the kernel density estimate of the observations' conditional densities [68]. In up-sampling, extra samples (with replacement) from the minority class is selected to match the size of both classes [146, 147].

The last step is the algorithm training. Three categories of the white box [76], black box [76], and ensemble models [158] are selected as the training algorithms. The white box models are LR, LDA, GLMNET, and MARS. The black box models are NNET, NB, KPLS, and SVM. The ensemble models are: RF, XGB, CART, and GB [146, 147].

LR fits a sigmodal logistic curve that represents the probability of an observation belonging to the threshold of the curve [83]. The probability is formulated as below [159]:

$$\Pr = \frac{\exp(\beta_0 + \beta^T x)}{1 + \exp(\beta_0 + \beta^T x)} \tag{32}$$

Similar to LR, LDA results in a linear logit [159]. The general formulation of LDA for K classes is [159]:

$$Pr(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l} \tag{33}$$

where:

$f_k(x)$: *a multivariate Gaussian density function*

GLMNET fits a linear model through assigning a penalized maximum likelihood function. Since the dependent variable is a binary variable (death/survival), this algorithm solves the below problem over a grid value of the tuning parameter (λ) that controls the strength of the penalty [96].

$$\min_{(\beta_0,\beta)\epsilon R^{p+1}} -\left[\frac{1}{N}\sum_{i=1}^{N} y_i.\left(\beta_0 + x_i^T\beta\right) - \log\left(1 + e^{\left(\beta_0 + x_i^T\beta\right)}\right)\right] + \lambda[(1-\alpha)\left|\left|\beta\right|\right|_2^2/2 + \alpha\left|\left|\beta\right|\right|_1] \quad (34)$$

For the binomial dependent variable the predicted probability is estimated through a logistic function [96]:

$$\Pr = \frac{\exp(\beta_0 + \beta^T x)}{1 + \exp(\beta_0 + \beta^T x)} \quad (35)$$

The MARS algorithm is implemented through the EARTH package of R software [146]. This algorithm divides the dataset into some partitions and develops a regression line for each partition. MARS utilizes the basis functions and searches for all possible hinge locations [160]. Here is the general format of the model:

$$y = \beta_0 + \sum_{j=1}^{P}\sum_{b=1}^{B}[\beta_{jb}(+)MAX\left(0, x_j - h_{bj}\right) + \beta_{jb}(-)MAX(0, h_{bj} - x_j)] \quad (36)$$

Where:

$MAX(0, x - H)$, $(0, H - x)$: basis functions

H: hinge

The algorithm utilizes a stepwise search to reach a number of basis functions that minimize the lack of fit.

NNET algorithm is a network of interconnected layers that mimics the human neuron system. The first layer receives the incoming data (x_i). A weight assigned to incomings of each neuron and summation of them transferred to neurons of the next layer (j).

$$neuron\ input_j = \sum_{i=1}^{n} weight_{ij}\ x_i \tag{37}$$

The weight assignment continues until the information reaches the activation function in the last layer [161].

$$output = f(I_j) \tag{38}$$

NB is based on Bayes theorem. NB investigates the relation of each variable to the dependent variable independently. Assuming C is the matrix of "observation × variables", and Y is the dependent variable, the structure of a general NB algorithm is:

$$\hat{Y}_b = argmax_y[\frac{P(y)\prod_{j=1}^{d} P(C_j|y)}{\sum_y P(y)\prod_{j=1}^{d} P(C_j|y)}] \tag{39}$$

Kernel PLS maps the original data to another space that has more dimension [162].

$$\Phi: \chi \rightarrow F$$
$$\chi \rightarrow \Phi(\mathrm{x}) = (\sqrt{\alpha_1}\,\psi_1(x), \sqrt{\alpha_2}\,\psi_2(x), \ldots, \sqrt{\alpha D_H}\,\psi_H(x)) \tag{40}$$

SVM creates hyperplanes to classify observations in different classes [88]. The associated function for predicting the dependent variable is [163]:

$$f(x) = sign\left(\sum_{i=1}^{N} \alpha_i y_i K(x_i.x_j) + b\right) \tag{41}$$

$K(x_i.x_j)$ is the kernel function and estimated through the following function [163]:

$$\max(\sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j K(x_i.x_j)) \tag{42}$$

where:

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad \& \quad 0 \ll \alpha_i \ll C$$

C: the regularization parameter

The RF uses bootstrapped sampling to develop multiple trees. Then RF classifies the observation based on the majority of the trees' votes [101]. The following equation represents the margin function of an RF [102]:

$$mr(X,Y) = P_{\Theta}(h(X,\Theta) = Y) - max_{j \neq Y} P_{\Theta}(h(X,\Theta) = j) \tag{43}$$

where:

$h(X)$: a classifier

XGB is the boosted ensemble of K CARTs models developed over the dataset. The summation of all the votes creates the prediction. A general XGB equation is shown as follow [164]:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \qquad f_k \, \epsilon \, F \tag{44}$$

where:

f_k: independent tree

F: space of all CARTs

CART develops a set of rules to split the data into the smaller and smaller fractions based on their homogeneity [165]. The rules are a set of questions that drive the parent point into the children until it reaches to the leaves that represent the classes' labels. Figure 4 demonstrates a simple split rule in a CART model where t_p is the parent node, t_l, t_r are the childer nodes, x_j is variable j, and x_j^R is the best splitting values [165].

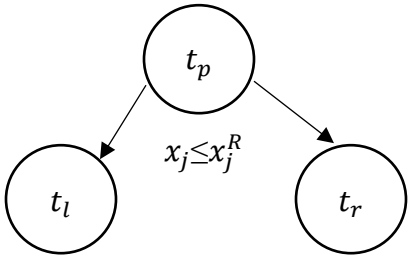


Figure 4: CART Splitting Process [165]

The following equation demonstrates the splitting rule in each parent node [165]:

$$argmax[i(t_p) - P_l i(t_l) - P_r i(t_r)] \tag{45}$$

where:

$x_j \leq x_j^R$, j=1,…,M

$i(t)$: impurity function

P_l: the probability of the left node

P_r: the probability of the right node

GBM maps a probabilistic input ($x = [x_1, \dots, x_r]$) to a random response (y) through $F^*(x)$ function. The general form of the mapping function is as follows [151]:

$$F^*(x) = argmin F(x)\, E_{y,x}\, \Psi(y, F(x)) \tag{46}$$

where:

$\Psi(y, F(x))$: the loss function

The general GBM algorithm is then [151]:

$$\begin{aligned} &1 \quad F_0(x) = argmin_\gamma \sum 1 - N\, \Psi(y_i, \gamma) \\ &2 \quad For\ m = M\ do: \\ &3 \qquad [\pi(i)]_1^N = rand\_perm\ [i]_1^N \end{aligned} \tag{47}$$

4 $\hat{y}_{\pi(i)m} = -\delta\, \Psi(y_{\pi(i)}, F(x_{\pi(i)}))\delta F(x_{\pi(i)})_{F(x)=F_{m-1}(x)}, i = 1, N$

5 $[R_{lm}]_1^L = L - terminal\ node\ tree\ ([\hat{y}_{\pi(i)m}, x_{\pi(i)}]_1^N)$

6 $\gamma_{lm} = argmin_\gamma \sum_{x_{\pi(i)}\ \epsilon\ R_{lm}} \Psi(y_{\pi(i)}, F_{m-1}(x_{\pi(i)}) + \gamma)$

7 $F_m(x) = F_{m-1}(x) + \nu.\gamma_{lm}\ l(x \epsilon R_{lm})$

8 $endFor.$

where:

$[y_i, x_i]_1^N$: entire training data

$[x_{\pi(i)}]_1^N$: random permutation from [1,2,..,N]

Among the above-mentioned algorithms, SVM and GBM have not converged to a result for a significant portion of the scenarios. Therefore, in this study LR, LDA, GLMNET, NNET, NB, KPLS, RF, XGB, and CART (a nine-level factor) are utilized for training the model.

Table 10: Total No. of Data Mining Project Developed

	CRISP-DM Sections					
	Data Preparation			**Training Model**		
Factors	**Categorical Imputation**	**Numerical Imputation**	**Encoding**	**Feature Selection**	**Resampling Method**	**Training Algorithm**
No. of levels	4	2	2	3	5	9
Total Combinations	4 × 2 × 2 × 3 × 5 × 9 × (5 replications) = 10,800					

There are several combinations of options that could be considered in the development sequence of data preparation and model training of CRISP-DM. Considering the 5-fold cross validation that is considered for avoiding overfitting [166], the Table 10 demonstrates how many different

combinations of data-mining projects that could be developed using the factors that are considered for the DoE.

However, before going through full factorial design, a two-level DoE is developed as the screening step to identify the contributing factors [154]. Then the significant factors are utilized for a full factorial design. Table 11 demonstrates the screening design and labels of their levels.

Table 11: Screening Design

Factors	Categorical Imputation	Numerical Imputation	Encoding	Feature Selection	Resampling Method	Training Algorithm
Level-1 (label)	Drop (DRO)	Drop (DRO)	Label (LAB)	FFS (FFS)	NONE (non)	LR (glm)
Level-2 (label)	Mode (MOD)	Median (MED)	One-hot (HAR)	LASSO (LAS)	SMOTE (smo)	RF (ran)

The main factors and their first interactions are investigated for significance. The response variable is GMEAN [94] to ensure a balanced performance for both death and survival the following equation demonstrate the calculation of GMEAN:

$$GMEAN = \sqrt{sensitivity \times specificity} \tag{48}$$

The Ohio supercomputer center [167] is a world-class computing center utilized for training the models in the parallel mode. The analysis is performed through the R programming language and for reproducing the results, the screening, and the full factorial design codes are provided in the appendix section of the dissertation and in the following GitHub link: https://github.com/transplantation/unos-ht

3.4 Results

A two-level factorial design is developed for the screening design. The response variable is GMEAN. The significance of the screening factors and their first interactions are

investigated. All of the main factors are significant. From the first-level interactions: training algorithm×resampling, training algorithm × feature selection, training algorithm × numerical imputation, resampling × feature selection, resampling × numerical imputation, resampling × encoding, feature selection × numerical imputation, feature selection × categorical imputation, feature selection × encoding, numerical imputation × categorical imputation, and numerical imputation × encoding are significant. Since all of the main factors are significant, they are incorporated in a full factorial design.

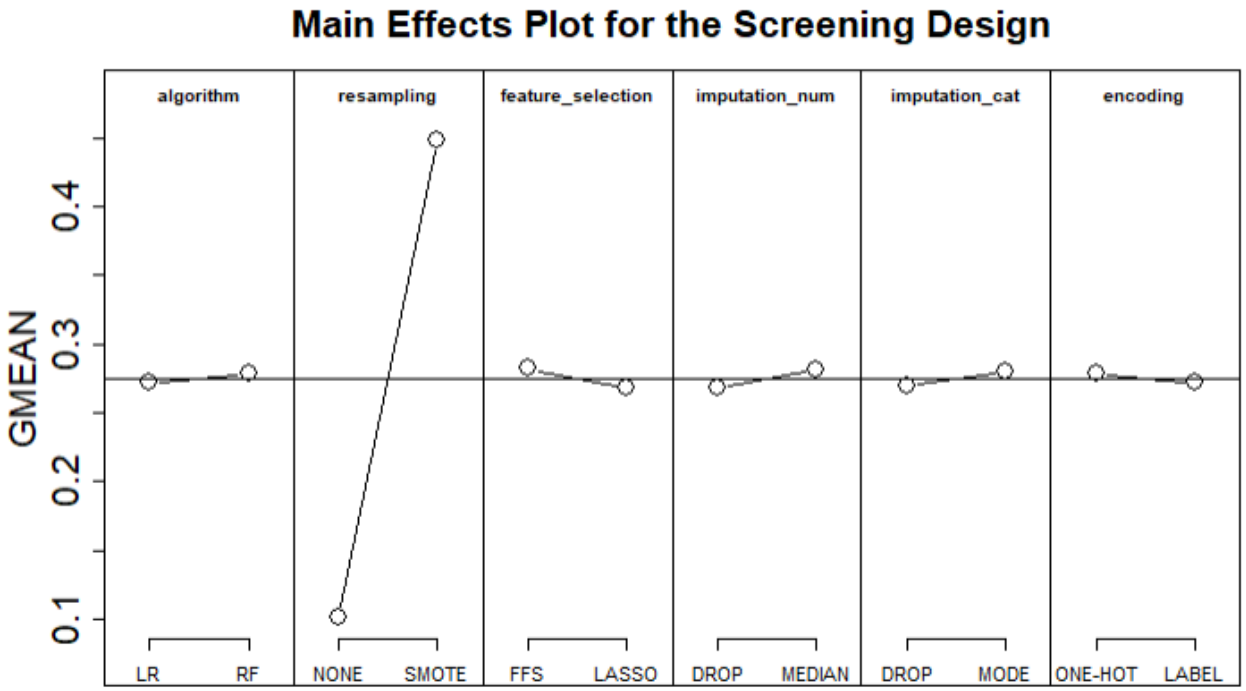


Figure 5: Main Factors' Effects on GMEAN in the Screening Design

The “Main Effects Plot for the Screening Design” is demonstrated in Figure 5 Also, Table 12 presents the details of DoE. The adjusted R-Squared is fairly high and there is no significant pattern in the residual plot.

Table 12: Screening Design's ANOVA

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.275	0.002	165.798	0
algorithm	0.003	0.002	1.895	0.059
resampling	0.173	0.002	104.607	0
feature selection	-0.007	0.002	-4.301	0
numerical imputation	0.006	0.002	3.779	0
categorical imputation	0.005	0.002	3.251	0.001
encoding	0.003	0.002	2.052	0.041
algorithm : resampling	-0.047	0.002	-28.321	0
algorithm : feature selection	-0.038	0.002	-22.714	0
algorithm : numerical imputation	-0.003	0.002	-2.004	0.046
resampling : feature selection	0.009	0.002	5.157	0
resampling : numerical imputation	-0.008	0.002	-4.528	0
resampling : encoding	0.004	0.002	2.337	0.02
feature selection : numerical imputation	0.006	0.002	3.513	0.001
feature selection : categorical imputation	0.005	0.002	3.018	0.003
feature selection : encoding	0.004	0.002	2.664	0.008
numerical imputation : categorical imputation	-0.013	0.002	-7.986	0
categorical imputation : encoding	0.003	0.002	1.757	0.08

R-Sqaure = 0.976

Adjusted R-Sqaure = 0.975

Table 12 shows that all the main factors are significant with more than 90% confidence. Then in the full factorial design, all the levels of the contributed factors are used for developing all the possible combinations of a data-mining study. However, SVM and GBM could not merge for a significant amount of the observations. However, KPLS and SVM both are kernel algorithms and there are other three ensemble algorithms (RF, XGB, and CART).

No non-random pattern is discover in the residual probability plot that is presented in Figure 6.

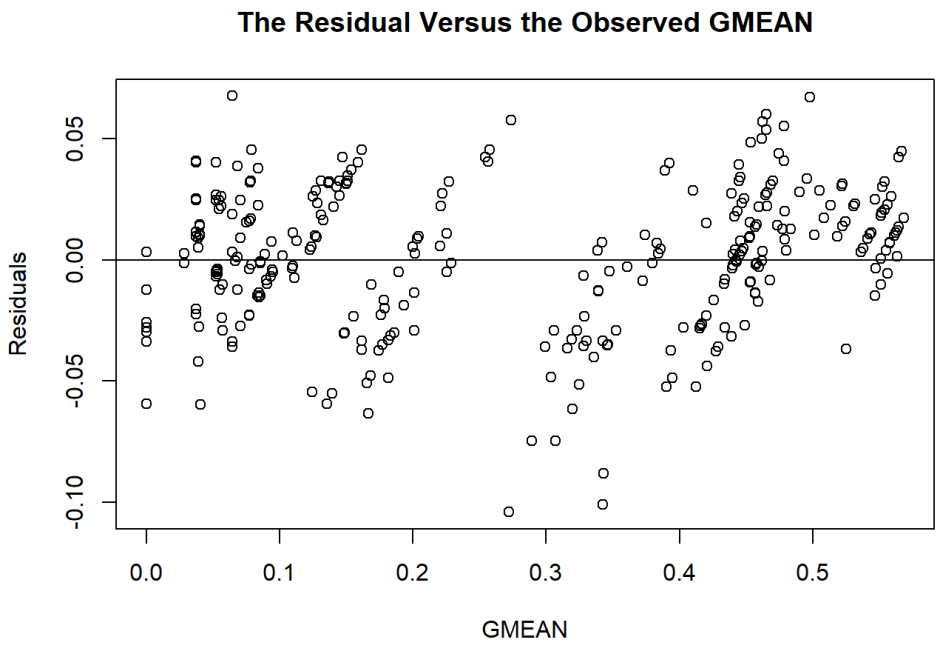


Figure 6: Residual versus Observed GMEAN Plot

The best design combination for every training algorithm based on highest GMEAN is demonstrated in Table 13:

Table 13: The Best GMEAN performance of the Algorithms

training algorithm	encoding	feature selection	numerical treatment	categorical treatment	resampling	AUC	sensitivity	specificity	accuracy	GMEAN
LR	One-Hot	LASSO	MEDIAN	UNKNOWN	up	65.47%	59.27%	62.88%	62.39%	61.04%
LDA	One-Hot	LASSO	MEDIAN	MISSING	up	64.65%	59.24%	62.16%	61.76%	60.68%
GLMNET	One-Hot	LASSO	MEDIAN	UNKNOWN	down	65.18%	60.18%	61.79%	61.57%	60.98%
KPLS	One-Hot	LASSO	MEDIAN	MISSING	up	64.94%	59.26%	62.49%	62.05%	60.85%
NNET	One-Hot	LASSO	DROP	UNKNOWN	up	61.56%	61.86%	56.19%	56.92%	58.65%
NB	LABEL	FFS	MEDIAN	UNKNOWN	down	61.34%	56.76%	59.50%	59.12%	58.02%
RF	One-Hot	LASSO	MEDIAN	MISSING	down	64.92%	60.33%	61.29%	61.16%	60.81%
XGB	One-Hot	LASSO	MEDIAN	MISSING	down	65.03%	59.24%	62.52%	62.07%	60.85%
CART	LABEL	LASSO	MEDIAN	MISSING	up	59.21%	53.67%	62.43%	61.23%	57.82%

The LR demonstrates the best performance among all of the algorithms. Interestingly, it was the fastest training algorithm. Also, in all of the presented combinations, the training algorithms had

similar performance in predicting both survival and death cases. Table 14 presents the variability of the predictive performance based on the training algorithms.

Table 14: Variability of Performance for each Training Algorithms

	Standard Deviation				
training algorithm	**AUC**	**sensitivity**	**specificity**	**accuracy**	**GMEAN**
LR	9.92%	21.16%	21.21%	16.45%	20.06%
LDA	1.66%	22.06%	15.81%	11.00%	19.78%
GLMNET	9.78%	22.51%	16.14%	11.23%	21.04%
KPLS	2.21%	21.24%	20.58%	15.77%	20.22%
NNET	6.69%	20.24%	16.26%	11.69%	20.26%
NB	1.90%	15.08%	12.31%	8.93%	10.74%
RF	2.41%	21.20%	15.69%	11.00%	17.60%
XGB	2.45%	20.36%	14.19%	9.81%	15.21%
CART	1.65%	17.70%	14.27%	10.21%	11.45%

The high degree of variability of performance measures is more evidence of the importance of selecting the best combination of the factors. In addition, the AUC measure has the lowest variability. In other words, selecting this measure shows less sensitivity in terms of selecting the factors' levels. The best design combination for each training algorithms based on highest AUC is demonstrated in Table 15:

Table 15: The Best AUC performance of the Algorithms

training algorithm	**encoding**	**feature selection**	**numerical treatment**	**categorical treatment**	**resampling**	**AUC**	**sensitivity**	**specificity**	**accuracy**	**GMEAN**
LR	One-Hot	LASSO	MEDIAN	UNKNOWN	up	65.47%	59.27%	62.88%	62.39%	61.04%
LDA	One-Hot	LASSO	MEDIAN	UNKNOWN	none	65.02%	1.36%	99.69%	86.24%	11.51%
GLMNET	One-Hot	LASSO	MEDIAN	UNKNOWN	up	65.22%	59.61%	62.24%	61.88%	60.91%
KPLS	One-Hot	LASSO	MEDIAN	UNKNOWN	up	65.09%	59.22%	62.38%	61.95%	60.77%
NNET	One-Hot	LASSO	DROP	MODE	none	64.04%	0.18%	99.94%	87.95%	1.89%
NB	LABEL	LASSO	MEDIAN	MODE	down	63.56%	54.05%	64.46%	63.08%	57.71%
RF	One-Hot	LASSO	MEDIAN	UNKNOWN	down	64.93%	59.84%	61.44%	61.22%	60.63%
XGB	One-Hot	LASSO	MEDIAN	UNKNOWN	up	65.26%	56.27%	65.50%	64.23%	60.67%
CART	LABEL	LASSO	MEDIAN	UNKNOWN	up	59.86%	54.57%	61.13%	60.23%	57.72%

The results reveal that if AUC is selected as the performance measure for the scenarios, in a couple of cases, the best model does not have the same capability in predicting survival and death cases.

3.5 Discussion and Future Studies

This research highlights the importance of adopting a systematic approach for selecting the best combinations of data preparation and model training sections of a data mining project. This approach includes two steps of a screening design and a full factorial DoE. Through a screen design, the significance of all the factors are presented and then a full factorial DoE is developed to make a surface response that contains all possible combinations of factors. The high variability in the performances is evidence for the significance of the approach. Also, the results demonstrate that if GMEAN is selected as the performance measure for selecting the best combination, the best training models are capable of a balance prediction for both death and survival cases. However, when AUC is selected for searching the best combination of the factors, in a couple of cases an imbalanced prediction capability is observed.

Chapter 4

Abstract

Accurate prediction of graft survival after a heart transplant is an important, yet challenging problem because: (a) it is the only treatment option for patients with end-stage heart failure; (b) the availability of hearts from deceased donors is scarce; (c) it requires an estimation of the matching suitability of patient-donor based on their medical information; and (d) its success is affected by the patient's adherence to strict medical instructions after transplant. Here, we propose a two-stage approach for estimating the graft survival probabilities at 1-10 years post surgery. First, we estimate the survival probability at different time points using machine learning methods. Then, we calibrate these probabilities using isotonic regression. Using a national registry of U.S. heart transplants from 1987-2016, we showed that our first stage produces an area under the receiver operating curve, AUC, between 0.60 and 0.71 for years 1-10. More importantly, the application of isotonic regression to smooth/calibrate the survival probabilities for each patient over the 10-year period guarantees monotonicity, while capitalizing on the data-driven and individualized nature of machine learning models. The monotonic, data-driven and individualized outcome prediction presented in this study is a novel contribution to both the transplantation and machine learning literatures. A web-based application (app), titled H-TOP: Heart Transplantation Outcome Predictor (see http://dataviz.miamioh.edu/Heart-Transplant/monotonic/), is provided as a decision support system for medical practitioners and policymakers. To encourage future work, we have made both our code and detailed analysis available online at: https://github.com/Ying-Ju/Heart_Transplant and https://ying-ju.github.io/heart_transplant.github.io/, respectively.

Keywords: App, Data mining, Health-care analytics, Longitudinal data analysis, Unbalanced data, United Network for Organ Sharing (UNOS)

4.1 Introduction

Heart failure is a serious medical condition, which can be characterized by the heart not being able to pump enough blood and oxygen to support other organs [168]. It is considered to be a global pandemic [169], affecting an estimated 26 million patients worldwide [170] and costing an estimated $108 billion per year [171]. The pandemic is exacerbated as the prevalence of heart failure has continued to increase. For example, in the United States, there were 5.7 million heart failure patients in 2012 [172], which has increased to 6.5 million in 2016 [173] and is expected to increase to 8.5 million by 2030 [174]. The outlook for these patients is poor despite the advancements in treatment protocols since: (a) ``heart failure has no cure'' [175], (b) the majority of patients who are admitted to a hospital with heart failure die within five years of admission [170], which is worse than the five-year survival of several cancers including bowl, breast, colon and prostate [176].

Heart transplantation is the most effective treatment for patients with end-stage heart failure [177-180]. For adult recipients, the median survival time after a heart transplant is 10.7 years [110]. To help understand the survival probability over time, the current 1-year survival rate is 84.5\% [110], with an attrition rate of 3-4% per year thereafter [180]. While these survival rates are excellent, transplantation is only available for a limited number of patients due to the scarcity of donors. In the U.S., there were 3,408 heart transplants performed in 2018 [181], with an additional 3,803 wait-listed candidates as of July 8, 2019 [182]. Therefore, risk stratification is essential to identify patients who are most likely to benefit [183, 184].

Risk stratification is centered around the ability to accurately predict graft survival over time (or at a given time point) at the time of transplantation by combining information from both

the donor and recipient. Due to the scarcity of donor hearts and the consequent importance of risk stratification, there is a growing body of literature on survival prediction post transplantation. The literature can be divided into three main streams: [3] (1) statistical-based methods, where the mean survival time or the combined effect of several predictor variables of interest on transplantation outcomes are examined (e.g., see [185, 186]); (2) machine-learning methods, which are used to predict survival outcome/probability at a predetermined time post transplant [8, 187]; and (3) studies focusing on survival prediction at multiple time points, often using machine learning methodologies (e.g., see [1, 3, 22]). The third stream provides the most complete picture of a patient's risk since: (a) survival probability is a more informative measure of risk than survival time [8]; and (b) the third stream can present an individualized survival probability curve for each patient (i.e. an individualized and data-driven counterpart of the population-based Kaplan-Meier survival curves presented in [110, 188-190]).

The utility of existing survival studies at multiple time points is limited. There are two major reasons that limit the utility of the existing literature. First, some applications (such as [3, 78]) develop independent machine learning models for each time period. The advantage of such an approach is that the prediction performance of each time period is optimized. However, due to the data-driven nature of machine learning models, such an approach does not guarantee monotonicity of survival probabilities over time (which means that, for a given patient, long-term survival probabilities can be higher than those of the short-term). To account for this limitation, some studies performed the prediction sequentially [5, 6]}, where the survival probability from years 1, 2, 3, ..., i are inputs to predicting the survival at time $i + 1$. While this can be an improvement for independently predicting survival probabilities, it does not guarantee monotonicity. An alternative approach is to utilize the population's survival statistics to calibrate

later years' survival probabilities. This approach was utilized in [1, 61, 187] and guarantees the monotonicity of outcomes. However, such an approach is not truly individualized because the probabilities obtained from this analysis are calibrated with the population's survival statistics. Second, the code used for analysis is not made available, which hinders the ability to reproduce the results [191-193]. For example, it is often unclear how authors: (a) handle missing data, i.e. imputation approach versus rules for removing observation and/or variable; (b) check and correct for outliers, such as conflicting information (e.g., a patient's age $\geq$ 18 and a value for the educational level indicating that the patient's age is an infant); (c) convert a categorical variable into multiple indicator variables, and if any grouping of categories is performed; and/ or (d) tune the parameters of the utilized machine learning models.

The overarching goal of this paper is to develop a modeling framework that can be used to obtain data-driven/individualized and monotonically decreasing survival probability curves. In pursuit of this goal, we propose a novel two-stage machine learning framework for heart transplantation outcome prediction using preoperative medical information. The first stage utilizes the standard approach of using independent machine learning models to predict transplantation outcomes for each time period of interest (e.g., see :[3, 78]). The primary objective in this stage is to select the most predictive/efficient machine learning model based on a predefined performance measure (e.g., area under the receiving operating characteristics curve, AUC). However, as mentioned earlier, this approach suffers from that the obtained survival probabilities are not guaranteed to be monotonically decreasing. Therefore, in the second stage, we calibrate the survival probabilities over time using isotonic regression [7]. The use of isotonic regression ensures that the survival probabilities are monotonically decreasing and that the results are individualized to a given patient.

The remainder of this paper is organized as follows. In Section 4.2, we present our two-stage methodology for obtaining a monotonically decreasing survival probability curve for each heart transplantation patient (given a set of donor characteristics). We provide our results in Section 4.3. In Section 4.4, we discuss their implications in practice, highlight the limitations in our analysis and present some ideas for future work. We offer our concluding remarks in Section 4.5. We present links for our code and analysis in the Appendix to allow researchers to replicate and build on this study.

4.2 Methods

Figure 7 provides an overview of our two-stage framework for obtaining monotonic survival probabilities over time. Stage I is comprised of the following steps: (a) detailed data preparation that explains how missing data is handled and how indicator variables are generated; (b) use re-sampling techniques to handle the imbalance between graft survivals and failures at a given time point; (c) use of a structured variable selection technique to improve the performance of machine learning models [194]; and (d) use of popular machine learning models [195] to predict survival probabilities for the 11 time periods. Our goals for the first stage are two-fold: select an appropriate model based on its predictive performance (through the use of AUC or G-mean) and examine how the selected variables importance change over time. On the other hand, Stage II has two steps: (a) combining outputs from all time points and forming a probability matrix, and (b)

application of isotonic regression for smoothing/calibrating the probability function. In the following subsections, we describe each stage and its associated steps in detail.

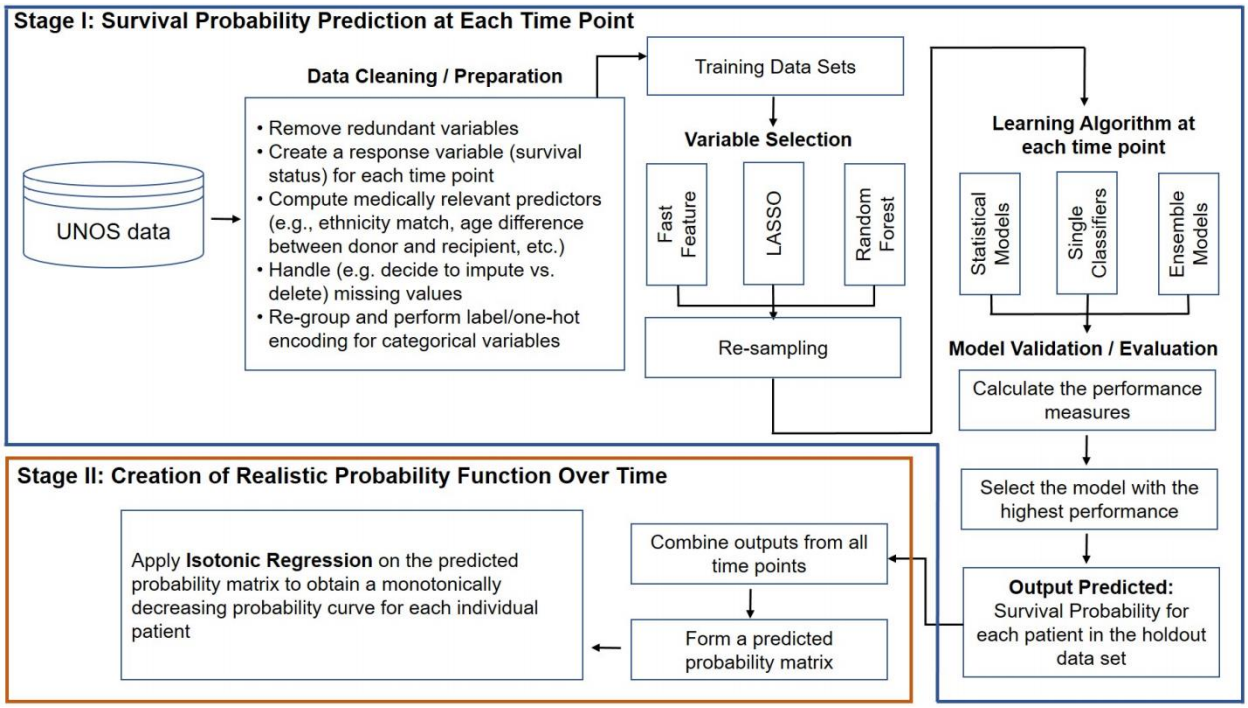


Figure 7: The proposed two-stage framework for obtaining monotonically decreasing heart transplantation survival probabilities

4.2.1 Stage I: Using Machine Learning Methods to Obtain a Survival Probability at Each Time Point

4.2.1.1 Data Description

The United Network for Organ Sharing (UNOS) maintains the national database, containing all heart (as well as other organs) transplant data for every transplant event that occurs in the United States since October 1, 1987 [196]. In this paper, our UNOS dataset covered 103,570 heart transplant events up to and including September 30, 2016. The dataset has been anonymized by UNOS and contains 494 columns/variables, capturing pre-, intra-, and post-transplant

information. Note that variables/records within the UNOS dataset have a large number of missing data (in our case, more than 30% of the "data-table cells" had missing values) since: (a) the start date of data collection for 327 of the 494 variables (66.19%) had a start date of 1990 or after, and additionally (b) some of the variables had missing-at-random values due to possible data entry and collection problems. Thus, a prerequisite to analyzing the UNOS dataset is a detailed (and ideally easily reproducible) data cleaning/preparation procedure.

4.2.1.2 Data Cleaning / Preparation

Data Cleaning/ Preparation

Our data cleaning procedure is comprised of six steps. The first step focuses on reducing the dimensionality of the data by removing the following sets of variables:

(a) *Intra- and post-transplant variables:* We recommend removing all such variables with the exception of G-TIME (which is a continuous variable capturing the time in days from transplant to graft failure/last follow-up time) and G-STATUS (a dichotomous variable, denoting if the graft has failed or not at the last follow-up time). We recommend retaining these two variables as they are used in constructing the dependent survival outcome at a given time point. The other remaining variables should be removed since they are unavailable at the time of transplant and thus, should not be used in modeling survival prediction based on pre-transplant information.

(b) *Variables with an end data collection date:* In our view, such variables should not be used in studies attempting to develop a decision support system for medical practitioners since they will not be collected for future cases by UNOS.

(c) *Variables with a recent data collection date:* While these variables will be collected in the future, most records will have no values for such variables. The selection of an appropriate ``recent data collection date" will depend on the time point of interest and whether researchers are interested in evaluating records going back to 1987. For the purpose of this analysis, we only included variables that were added before 2000.

(d) *Other variables with a large amount of missing data:* We removed any variable that had ≥ 90 % missing. Other rules of thumb can be used (see e.g. [45]).

For more details on the order and implementation of our variable exclusion procedure, we refer the reader to **R** Markdown document provided in the appendix of this manuscript.

The second step involves determining whether a record should be used for analysis (for a given time point). Certain observations/records should be censored due to the lack of information pertaining to a patient's survival. Our censoring procedure follows the approach of [3], capitalizing on both G-TIME and G-STATUS. Specifically, the data is censored IF AND only IF the G-TIME is less than the number of days required for the time-point analysis AND the G-STATUS indicates that the patient is still alive. Accordingly, the dichotomous outcome for the i^{th} can be coded as follows:

$$y_i = \begin{cases} 1, \text{ if the } i^{th} \text{ subject was alive at time t} \\ 0, \text{ if the } i^{th} \text{ subject was dead at time t} \end{cases} \quad (49)$$

The third step in our recommended data preparation procedure is to generate medically-relevant features that have been shown as important/predictive in the literature. These features can be computed by combining information from two or more independent variables. In our analysis, we have created the following features based on [1, 3, 8]

(a) pulmonary vascular resistance (PVR), (b) ischemia time in minutes, (c) mismatch in Extracorporeal membrane oxygenation (ECMO) for recipient between registration and transplant, (d) percent change in the recipient's body mass index (BMI) between registration and transplant, (e) percent change in the recipient's weight between registration and transplant, (f) percent change in the recipient's height between registration and transplant, (g) the absolute difference in age between donor and recipient, and (h) the absolute difference in BMI between donor and recipient. For additional detail on how each feature is generated, we refer the reader to **R** Markdown document provided in the appendix of this manuscript.

Fourth, one has to decide how to handle missing data. In the first step of our data cleaning process, we removed any independent variable that has $\geq 90\%$ missing. In our framework, we suggest the following imputation strategy: (a) correctly define independent variable types in the analytical software [197], and (b) compare the predictive performance of your models when no imputation is used versus when imputation (median/mean and mode imputation for numeric and categorical variables, respectively) is performed. Based on the performance comparison, researchers can decide if data imputation can lead to improving the prediction results and whether a more computationally intensive imputation technique is warranted (see [198] for an overview and taxonomy of imputation techniques).

In the fifth step, the levels of categorical variables should be regrouped. We attempted to reduce the number of levels, by removing any level which has a limited number of observations and re-coding the values such that they are added to the level corresponding to ``Other''. The purpose of this step was three-fold: (a) avoiding over-fitting, (b) reducing the possibility of errors due to factor levels not observed in the training of the model, and (c) reducing the dimensionality/complexity of the developed model (especially if a tree-based approach such as

Random Forests is used). Additionally, many implementations of machine learning algorithms require the categorical variables to be encoded [195] either using label or one-hot encoding [199]. We evaluated both approaches in our analysis.

The sixth and final step of our data cleaning/preparation procedure involved splitting the data randomly into a training (80%) and a hold-out (20%) set. We use a 5-fold cross validation approach for selecting an appropriate model from the training set since: (a) it is more computationally efficient than 10-fold cross validation, and (b) both k=5 and k=10 "have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance" [200].

4.2.1.3 Variable Selection

From the data cleaning step, it can be easily seen that the computational complexity for model training will increase significantly when one-hot encoding is examined. Variable selection approaches attempt to: (a) reduce the computational burden by reducing the dimensionality of the variable/feature space, (b) improve the prediction performance by focusing only on important/predictive variables, and (c) improve the generalizability of the developed prediction models. Variable selection approaches can be categorized into three main groups [109]: (a) *filter methods*, where univariate statistical approaches are typically used to select features based on their relationship to the response, (b) *wrapper methods*, where the important features are kept based on their prediction performance, and (c) *embedded methods*, which involve the use of methods such as LASSO and random forests for selecting the most predictive features. In our analysis, we compared the performance of fast feature selection, LASSO, and random forests for feature selection. Alternatively, we could have obtained a variable set through combining features from

the three approaches. Our preliminary analysis showed that there is no significant difference in performance between the best singular variable selection approach and the combined method. Thus, we used the singular approach to reduce the number of selected variables and consequently, the computational complexity associated with the analysis.

4.2.1.4 Re-sampling Approaches to Handle the Data Imbalance Problem

As stated in Section 4.1, based on the current state of heart transplantation, the expected survival probability decreases by about 3-4% per post-transplant year. Moreover, the censoring approach of Section 4.2.1.2, should result in removing those patients who are currently surviving, but has not reached the time-point of interest from the analysis. Consequently, we expect that the short time intervals (e.g., 1 month, 1 year, etc.) will have a much larger number of survivals than deaths, and the longer time-periods (e.g., 9- and 10- years post transplant) will have an opposite ratio of deaths to survivals. In the data mining literature, re-sampling methodologies are typically deployed to improve the prediction performance when the underlying training dataset is imbalanced [73, 201]. In our analysis, we considered the following five re-sampling strategies:

(a) No Re-sampling (None), which can be considered as the baseline for comparison;

(b) Random under-sampling (DOWN), where the majority class is sampled without replacement such that the number of observations in both classes (i.e. 0 and 1) are equal;

(c) Random over-sampling (UP),} where the minority class is sampled with replacement such that the number of observations in both classes (i.e. 0 and 1) are equal;

(d) Synthetic minority oversampling technique (SMOTE), where the minority class is over-sampled by “taking each minority class data point and introducing synthetic examples along the line segments joining any or all of the k-minority class nearest neighbors” [66],

and repeating the process until the number of observations within each class is approximately the same; and

(e) ROSE, where a statistical approach based on the smoothed bootstrap re-sampling technique of [202] is used for handling the class imbalance problem.

Note that these approaches were selected since they are widely used in machine learning practice due to their implementation within the popular *caret* package in **R** [146].

4.2.1.5 Application of Machine Learning Algorithms

As mentioned in Section 4.1, the primary goal of this study is to develop a framework that can be used to calibrate the survival probabilities obtained from deploying a machine learning (ML) algorithm over multiple time points. From this point of view, our approach (in Stage II) is designed to work with any machine learning methodology that can be used for two-class problems. In our experience with the UNOS dataset, the utility of a given ML algorithm can differ significantly based on how the data preparation procedure (and its underlying assumptions about how the model will be implemented in practice). Generally speaking, ML methodologies for two-class prediction problems can be classified into [159, 195]:

(a) *Statistical models*, which are not assumption free. The most commonly implemented statistical model in the transplantation literature is logistic regression (LR) (e.g., see: [3, 22, 78, 79, 82, 84]). Linear discriminant analysis (LDA) also showed good predictive performance in the literature (e.g., see: [203]);

(b) *Single (data-driven) classifiers*, which include a large group of assumption-free machine learning approaches. Commonly used algorithms in the transplantation literature include:

artificial neural networks (ANNs) [3, 5, 6, 8, 10, 78, 79, 82], decision trees (CART) [3, 84], and support vector machines (SVM) [3, 8, 10, 79]; and

(c) *Ensemble approaches*, where prediction outcomes from single classifiers are combined through a "voting" based approach. Random forests is the most commonly implemented approach in the literature [78, 82, 88]. Extensions to the standard random forest (RF) implementation include eXtreme Gradient Boosting (XGB) methods, which are growing in popularity due to their improved prediction performance in several applications [195]. It should be noted, however, they are not widely used in the transplantation literature.

4.2.1.6 Evaluation and Model Selection

Based on our Stage I description, our proposed framework allows for the following analytical decisions:

- categorical imputation strategies (no imputation and impute missing as ``unknown'');
- numerical imputation strategies (no imputation and median imputation);
- 2 categorical variable encoding approaches (label and one-hot);
- variable selection approaches (fast feature selection, LASSO, and random forests);
- re-sampling methods (none, DOWN, UP, SMOTE and ROSE); and
- machine learning models (LR, LDA, ANN, CART, SVM, RF and XGB).

We are also recommending the use of 5-fold cross validation in model selection. Moreover, we are interested in 11 prediction time periods to generate the survival probability curve. Consequently, the enumeration of these combinations would result in: $2 \times 2 \times 2 \times 3 \times 5 \times 7 \times 5 \times 11 = 46{,}200$ experimental runs. These runs correspond to running a general factorial experiment

to compute the main factor effects and all high order interactions for the settings associated with each step of our framework.

Given that the primary goal of this paper is calibrating the attained survival probabilities, we can reduce the computational burden to $4,200$ runs if we focus on only one time period (i.e., $\approx$ 91% reduction in computational load). While this can be reduced further through a fractional factorial design (since only the main effects and first order interactions would be of interest), we could run the 4,200 runs since we had access to 224 compute nodes on a supercomputer. For future implementations, we do recommend the use of a fractional factorial design to optimize the settings if computational complexity presents a resource constraint. For a detailed introduction on fractional factorial designs, the reader is referred to [4].

We will compare the predictive performance of the modeling approaches, with different data preparation procedures, for the 1-year prediction time frame. We have chosen 1-year since it: (i) provides us with the largest sample size (with the exception of 1-month which only focuses on accute organ rejection prediction), and (ii) it is a commonly used time frame in the literature (e.g. see: [1, 3, 78]). The selected modeling approach will then be trained for all other time frames for the sake of demonstrating the need and utility of our isotonic regression calibration approach in Stage II.

For model selection and evaluation, we report five common performance metrics that are suited for two class classification problems. Prior to presenting these metrics, consider the confusion matrix provided in Table 16. Based on this confusion matrix, we can define our five metrics as follows:

Table 16: Confusion matrix for heart transplantation prediction problems

		Prediction	
		0	1
Actual Class	0	True Negative (TN)	False Positive (FP)
	1	False Negative (FN)	True Positive (TP)

$$accuracy = \frac{TN+TP}{(TN+TP+FP+FN)}, \tag{50}$$

$$sensitivity = \frac{TP}{(TP+FN)}, \tag{51}$$

$$specificity = \frac{TN}{(TN+FP)}, \text{ and} \tag{52}$$

$$GMEAN = \sqrt{sensitivity \times specificity}. \tag{53}$$

The receiver operating characteristics (ROC) curve can be plotted with the sensitivity on the y-axis versus 1- specificity on the x-axis. The area under the curve (AUC) captures how well the model predicts actual survival cases as survivals and actual deaths as deaths. AUC is our fifth performance measure.

In this paper, we use G-mean as the primary metric for model selection since it: (a) is suitable for unbalanced classification problems, and (b) penalizes models where there is a large discrepancy between sensitivity and specificity (which makes it more suitable than AUC when similar values for sensitivity and specificity performances are preferred). Thus, we will select the modeling approach (i.e. the combination of categorical imputation, numerical imputation, categorical variable encoding, re-sampling and machine learning model) with the highest average G-mean across the five cross-validation models.

4.2.1.7 Applying the Selected Modeling Approach for the Remaining 10 Time Periods

From our framework's perspective, isotonic regression can be applied to survival probabilities obtained from different models. However, for the sake of convenience, in this paper we are applying it to probabilities obtained from the selected modeling approach (based on the criterion and description in Section 4.2.1.6). This means that, we will re-train our machine learning approach for each time period, while maintaining the "appropriate/optimal" data preparation procedure obtained from the comparison of the 1,400 runs for the first year post transplant analysis. Based on this step, for a given transplant event (that has not been censored), we will compute a survival probability for each of the 11 time periods.

4.2.2 Stage II: Calibrating the Stage I Prediction Probabilities

For a given transplantation event, let y = $(y_0, y_1, y_2, \ldots, y_{10})$ to correspond to the obtained probabilities from Stage 1, where 0 reflects the first month and 1→10 denote the number of years post transplant. Isotonic regression can be used to calibrate and ensure that the elements within **y** are non-increasing. This can be achieved through solving: [7, 204]

$$min \sum_{i=1}^{10} (y_i - \hat{y}_i)^2 \qquad (54)$$

$$s.t. \quad \hat{y}_{max} = \hat{y}_0 \geq \hat{y}_1 \geq \hat{y}_2 \ldots \geq \hat{y}_{10} = \hat{y}_{min},$$

where w_i is a strictly positive weight. In our analysis, we used w_i = 1, $\forall\ i$ since this corresponds to the squared Euclidean distance (i.e., it has a physical interpretation). This optimization problem can be solved in δ(n) time using the pool adjacent violators algorithm (PAVA)[205, 206]. The

solution computes the values for the decision variables $\hat{y} = (\hat{y}_0, \hat{y}_1, \hat{y}_2 \ldots, \hat{y}_{10})$. Based on Eq. 54, the values of the decision variables are monotonically decreasing. Each decision variable $\hat{y}_i$ represents the calibrated probability value for the corresponding y_i obtained from Stage I.

4.3 Results

4.3.1 Stage I Results: Using Machine Learning Methods to Obtain a Survival Probability at Each Time Point

4.3.1 Data Cleaning Results

Per our data cleaning procedure described in Section 4.2.1, we have a total of 8 scenarios (2 categorical imputations × 2 numerical imputations × 2 categorical encoding methods). Based on these scenarios, the number of observations and independent variables varied between 26,829-45,089 and 90-269, respectively. The interested reader is referred to our **R** Markdown code/results in https://ying-ju.github.io/heart_transplant.github.io to obtain the following details: (a) number of preoperative variables in the dataset; (b) distribution/frequency of percent missing in the pre-operative variables; (c) variables which were not included in the analysis; (d) percent cells changed by imputing unknown; (e) total number of indicator variables after applying one hot encoding; etc.

4.3.2 Predictor Variable Importance over the 11 Time Periods

Table 17 summarizes the important variables according to the number of times they were selected (i.e. each selection corresponds to the model for a given time-period). There were 12 variables that were selected for all time points: (a) the donor's age; (b) recipient's body mass index;

(c)deceased donor's cause of death; (d) recipient's serum creatinine level at the time of transplant; (e) recipient's primary diagnosis; (f) recipient's functional status at transplant; (g) the recipient's calculated height at listing; (h) ischemic time in hours; (i) recipient's medical condition pre-transplant at transplant; (j) UNOS region where transplanted/listed; (k) most recent serum total bilirubin at transplant time; and (l) candidate diagnosis at listing. On the other hand, there were 11 variables that were selected once (mostly for the 1-month prediction).

Table 17: Important Variables Selected with the corresponding frequency

Times Selected	**Variables**
11	AGE_DON, BMI_CALC, COD_CAD_DON, CREAT_TRR, DIAG, FUNC_STAT_TRR, INIT_HGT_CM_CALC, ISCHTIME, MED_COND_TRR, REGION, TBILI, TCR_DGN
10	CMV_DON, DAYS_STAT1A, DIAB, DRMIS, ETH_MAT, ETHCAT, PRI_PAYMENT_TCR, PRI_PAYMENT_TRR, THORACIC_DGN, TRANSFUSIONS, VENT_SUPPORT_AFTER_LIST
9	CARD_SURG, GENDER_MAT, HCV_SEROSTATUS, LIFE_SUP_TRR, PROC_TY_HR
8	PRIOR_CARD_SURG_TCR, SUD_DEATH
7	DOPAMINE, FUNC_STAT_TCR, HBV_CORE, HGT_CM_TCR, INFECT_IV_DRUG_TRR, PT_T4_DON, PULM_CATH_DON
6	AMIS, DAYS_STAT1, EDUCATION, HEPARIN, HGT_CM_CALC, INOTROP_VASO_CO_TRR, LIFE_SUP_TCR, PRIOR_CARD_SURG_TYPE_TCR, TBILI_DON
5	BUN_DON, CHEST_XRAY_DON, HIV_SEROSTATUS, IMPL_DEFIBRIL, INOTROP_VASO_PCW_TRR, INOTROP_VASO_SYS_TCR, OTHER_INF_DON, PT_DIURETICS_DON, STERNOTOMY_TRR
4	BRONCHO_LT_DON, CONTIN_CIG_DON, EBV_SEROSTATUS, ETHCAT_DON, HEMO_SYS_TCR, HEMO_SYS_TRR, HLAMIS, HTLV2_OLD_DON, LAST_INACT_REASON, SHARE_TY
3	CORONARY_ANGIO, DAYS_STAT1B, GENDER_DON, HBV_SUR_ANTIGEN, INIT_AGE, INOTROP_VASO_MN_TRR, PULM_INF_DON, TATTOOS, WGT_KG_DON_CALC
2	AGE, ANCEF, ANTIHYPE_DON, BMIS, CLIN_INFECT_DON, CRSMATCH_DONE, DAYSWAIT_CHRON, HGT_CM_DON_CALC, HIST_OTH_DRUG_DON, INIT_BMI_CALC, INIT_STAT, INOTROPIC, PRIOR_CARD_SURG_TRR, PVR, SGOT_DON, CONTIN_OTH_DRUG_DON
1	ABO_DON, AGE_MAT, BRONCHO_RT_DON, END_STAT, GENDER, HEMO_PA_DIA_TRR, INOTROP_VASO_SYS_TRR, INOTROPES_TRR, PROTEIN_URINE, VASODIL_DON

4.3.3 Comparisons of Machine Learning Methodologies

Table 18 presents an overview of the prediction performance of the seven machine learning algorithms (with the best data preparation/preprocessing approach for a given algorithm). We have arranged the table in descending order based on G-mean since we prefer to have a model that penalizes large discrepancies between sensitivity and specificity. Note that, irrespective of our pre-algorithm preparation procedure, SVM did not converge. Therefore, we do not report any results for SVM. Based on Table 18, logistic regression presented the best G-mean performance with: (a) median imputation for missing numeric values, (b) imputing missing categories as "unknown", (c) one-hot encoding for categorical variables, (d) LASSO for feature selection, and (e) UP sampling. While the results are not statistically significant when compared to other approaches, we believe LR is the best approach since it is the quickest to run and most understandable by health-professionals. We only discuss the logistic regression results, with (a)-(e) preprocessing, hereafter.

Table 18: Before Isotonic Regression (Same Patients in all the timestamps)

Algorithm	Numerical Imputation	Categorical Imputation	Encoding	Feature Selection	Resampling	G-Mean	AUC	Sensitivity	Specificity
LR	Median	Unknown	One-Hot	LASSO	UP	0.6104	0.6547	0.5927	0.6288
XGB	Median	Drop	One-Hot	LASSO	DOWN	0.6073	0.6493	0.5845	0.6309
LDA	Median	Unknown	One-Hot	LASSO	UP	0.6065	0.6484	0.5927	0.6207
RF	Median	Unkown	One-Hot	LASSO	DOWN	0.6063	0.6493	0.5984	0.6144
ANNs	Drop	Unkown	One-Hot	LASSO	UP	0.5865	0.6156	0.6186	0.5619
CART	Median	Unkown	Label	LASSO	UP	0.5772	0.5986	0.5457	0.6113
SVM	None of the SVM implementations converged in our allotted time frame of 20 hours.								

For any machine learning model, there is a trade-off in terms of the classifier's performance between sensitivity and false positive rates (i.e. 1 - specificity). In logistic regression (and the other machine learning models), the probability of graft failure is translated to a dichotomous decision

(0 for graft failure and 1 for survival) based on a cut-off value (where the default for computer programs is typically 0.5). The receiver's operating characteristic (ROC) curve is a standard approach for assessing the performance of a continuous variable for binary classification [206]. The ROC curve plots sensitivity (ability to predict survivals) versus 1 - specificity (false survival rate) for all the possible cutoff values of the continuous variable. We present the ROC curves for Years 1-9 in Figure 8 Our model yields an AUC value between 0.634-0.700 and 0.596-0.705 for the training and hold-out (testing) sets respectively. To present a complete picture of the model's hold-out performance, we provide the values for accuracy, specificity, sensitivity, AUC, and G-mean in Table 19. These values represent the performance of the logistic regression modeling approach, with (a)-(e) settings, for the hold-out dataset.

Table 19: Hold-out performance of the UP-LASSO-logistic regression implementation.

	Month1	**year1**	**year2**	**year3**	**year4**	**year5**	**year6**	**year7**	**year8**	**year9**	**year10**
Accuracy	0.588	0.575	0.569	0.571	0.591	0.596	0.611	0.625	0.634	0.636	0.627
Specificity	0.586	0.569	0.568	0.567	0.603	0.619	0.648	0.672	0.715	0.743	0.752
Sensitivity	0.604	0.599	0.572	0.581	0.567	0.558	0.562	0.571	0.556	0.545	0.533
AUC	0.641	0.614	0.596	0.61	0.623	0.63	0.649	0.665	0.692	0.705	0.702
G-Mean	0.595	0.584	0.57	0.574	0.585	0.587	0.603	0.619	0.63	0.636	0.633

In standard machine learning applications, the results shown in Figure 8 and Table 19 are sufficient to describe the performance of the model. However, in our case, we are interested in presenting an individualized survival probability curve for each patient. As a first step toward this goal, we have to evaluate the precision of our predicted probabilities, which we depict in Figure 9. To ensure that we have a sufficiently large sample size for inference, we have only plotted probabilities when the number of observed cases within that probability range was ≥ 100. The plot shows that our forecast survival probabilities underestimate the observed averages for the first 5 year holdout data sets. For the later years, the predicted probabilities are relatively close to the

observed ones. The finding from this plot is consistent with the performance metrics reported in Table 19.

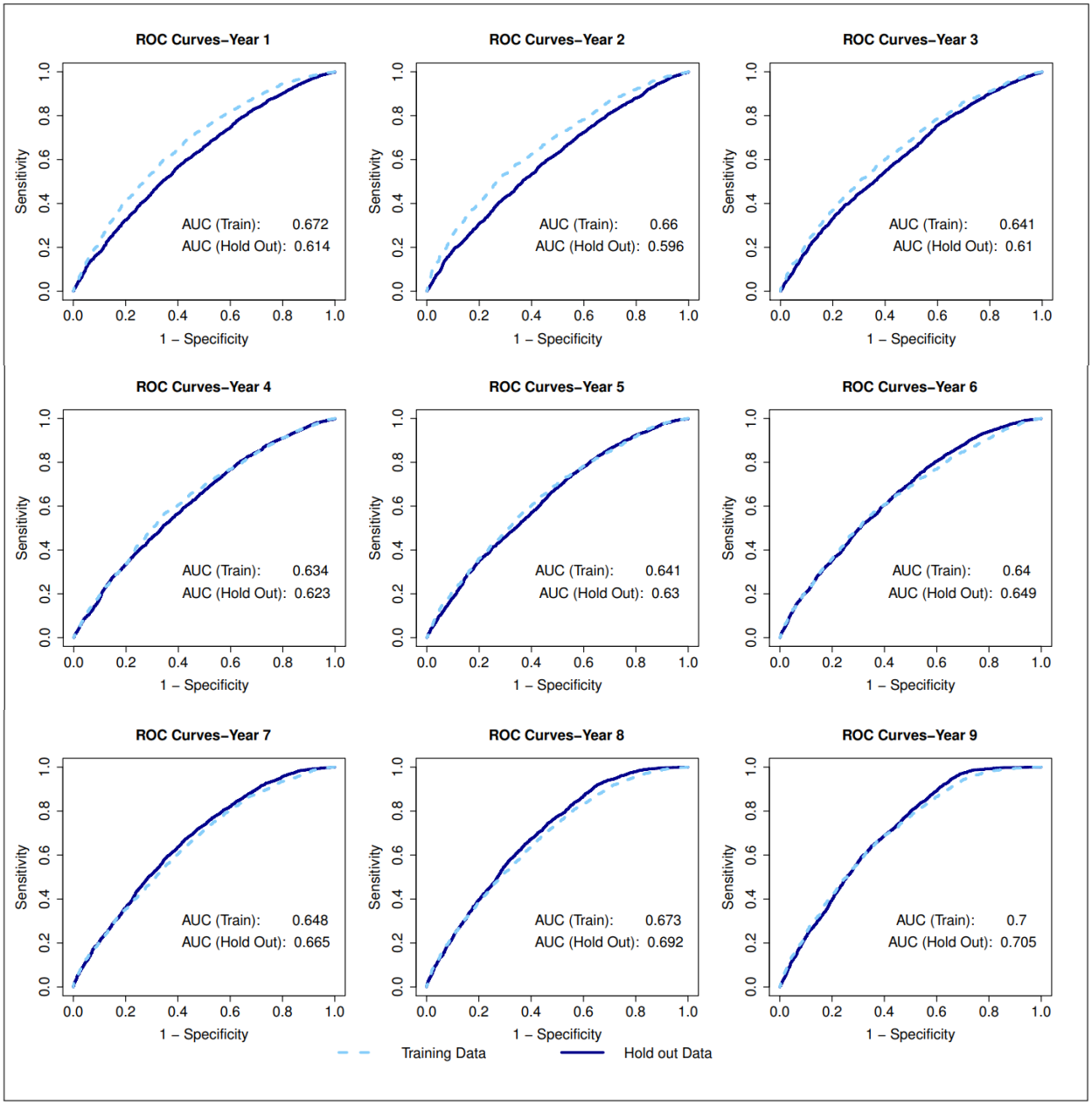


Figure 8: ROC plots for our logistic regression's training and hold-out datasets for Years $1-9$. The light blue and dark blue correspond to the training and hold-out performances, respectively.

4.3.2 Stage II Results: Calibrating the Stage I Prediction Probabilities

A closer examination of the individualized survival probabilities over time revealed a non-monotonic behavior of the curves. To alleviate this problem, we used isotonic regression, the second stage of our machine learning methodology, to smooth/calibrate the individualized curves while ensuring monotonicity for all patients' curves. Figure 10 shows how the survival probabilities before and after the calibration through the use of isotonic regression. In addition, we evaluate the precision of our calibrated probabilities in Figure 11. Similar to Figure 9, the figure shows the observed averages against the forecast survival probability in Year 1, 2, …, 9 after isotonic regression is applied. The overall pattern in each plot is consistent with the corresponding plot in Figure 8 which means that the application of isotonic regression did not lead to a deterioration of prediction performance. However, and more importantly, the application of isotonic regression leads to results that are: (a) consistent with the medical expectation of decreasing survival probabilities over time; and (b) data-driven and individualized to each patient.

4.4 Discussion

In this study, we presented a two-stage machine learning model that can be used to predict the probability of graft failure over time solely based on patient and donor data at the time of transplantation. Our results are medically compelling for several reasons. First, our machine learning based framework was trained and validated on at least 29,143 transplant events, capturing all heart transplantation events in the U.S. from 1987 - 09/2016 who had a known graft failure outcome for 10 years (i.e., we can assess whether they lived or died at 10-years post transplant). For shorter time periods, the number of events used for training and evaluation increased (due to

a reduced number of censored observations). Second, our data preparation process: (a) explicitly excluded any variables that UNOS has stopped collecting data for; (b) allows for having missing data in any of the categorical predictors; and (c) presents a medically informative approach for imputing missing categorical data through the use of an existing factor level corresponding to "Unknown". Third, our results can be easily explained since they are not based on black-box models: (a) the Stage I results from logistic regression are understandable to any medical professional with basic statistical training (e.g., logistic regression is covered in most masters of public health training); and (b) isotonic regression in, Stage II, ensures that the aggregation of results for multiple time points is monotonic. Fourth, our survival probabilities are individualized; the obtained probabilities are data-driven and generated by combining important predictors, which are identified and selected using LASSO, and calibrating the probabilities over time using isotonic regression. If we focus on the results of Stage I, our model is comparable to the prediction outcomes reported in the literature (where the UNOS dataset was used for the analysis). For example, Dag, Oztekin [3] reported AUC values of 0.624, 0.676 and 0.838 at 1, 5, and 9 years post transplant. Our results are similar for the first year and are more informative for years 5 and 9 (since they did not consider the consistency of predictions over time for a given patient).

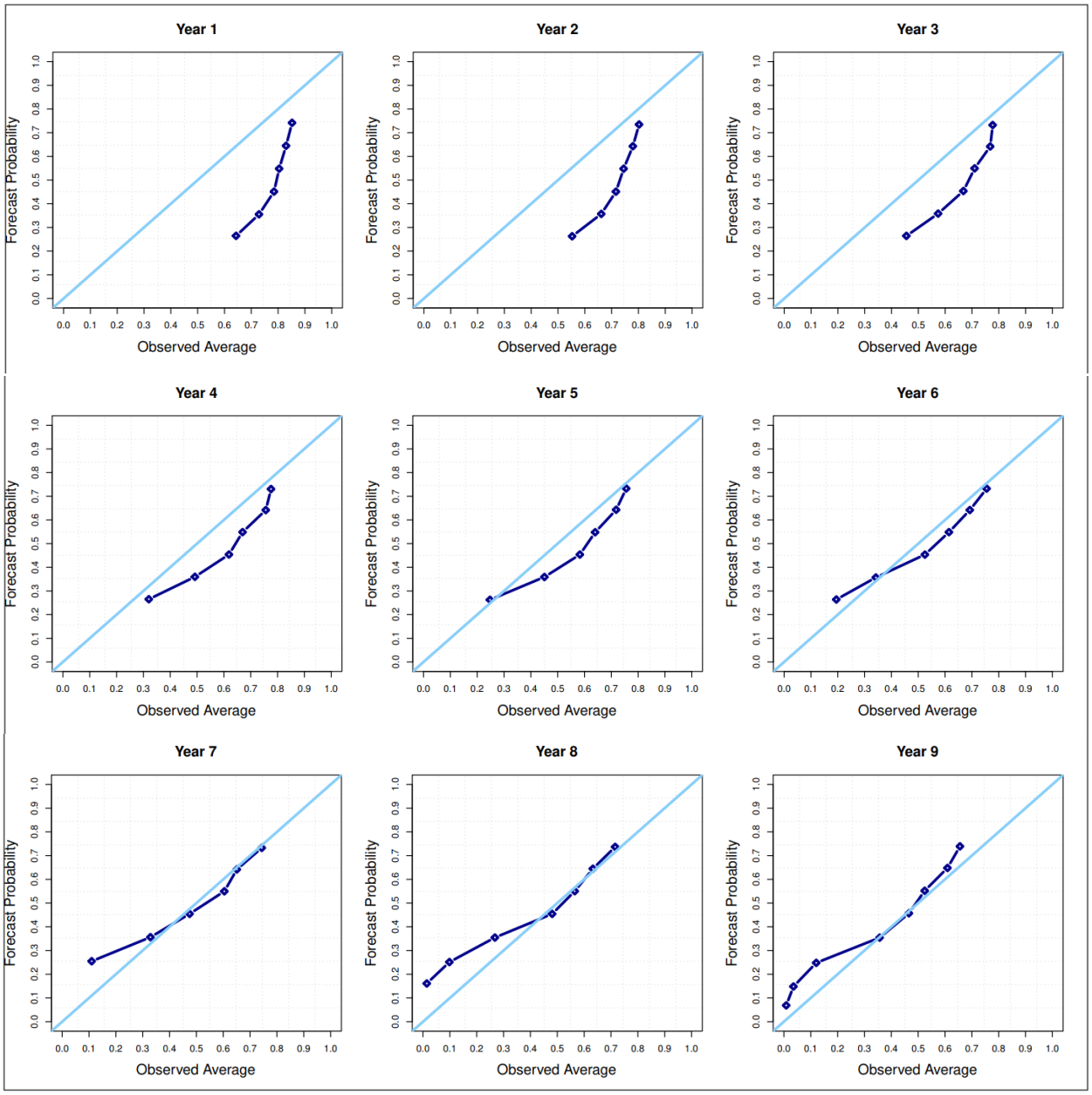


Figure 9: Plots of the observed versus forecast survival probabilities for Years $1-9$ (before isotonic regression). The light blue line correspond to the baseline case of a perfect match between both probabilities. The dark blue line correspond to the results obtained from our machine learning model. To create each plot, we grouped the patients in our holdout set based on their forecast probability (in 0.1 increments) and we then calculated the observed average as: (# Survivals for year from group)/(Total # patients for year from group).

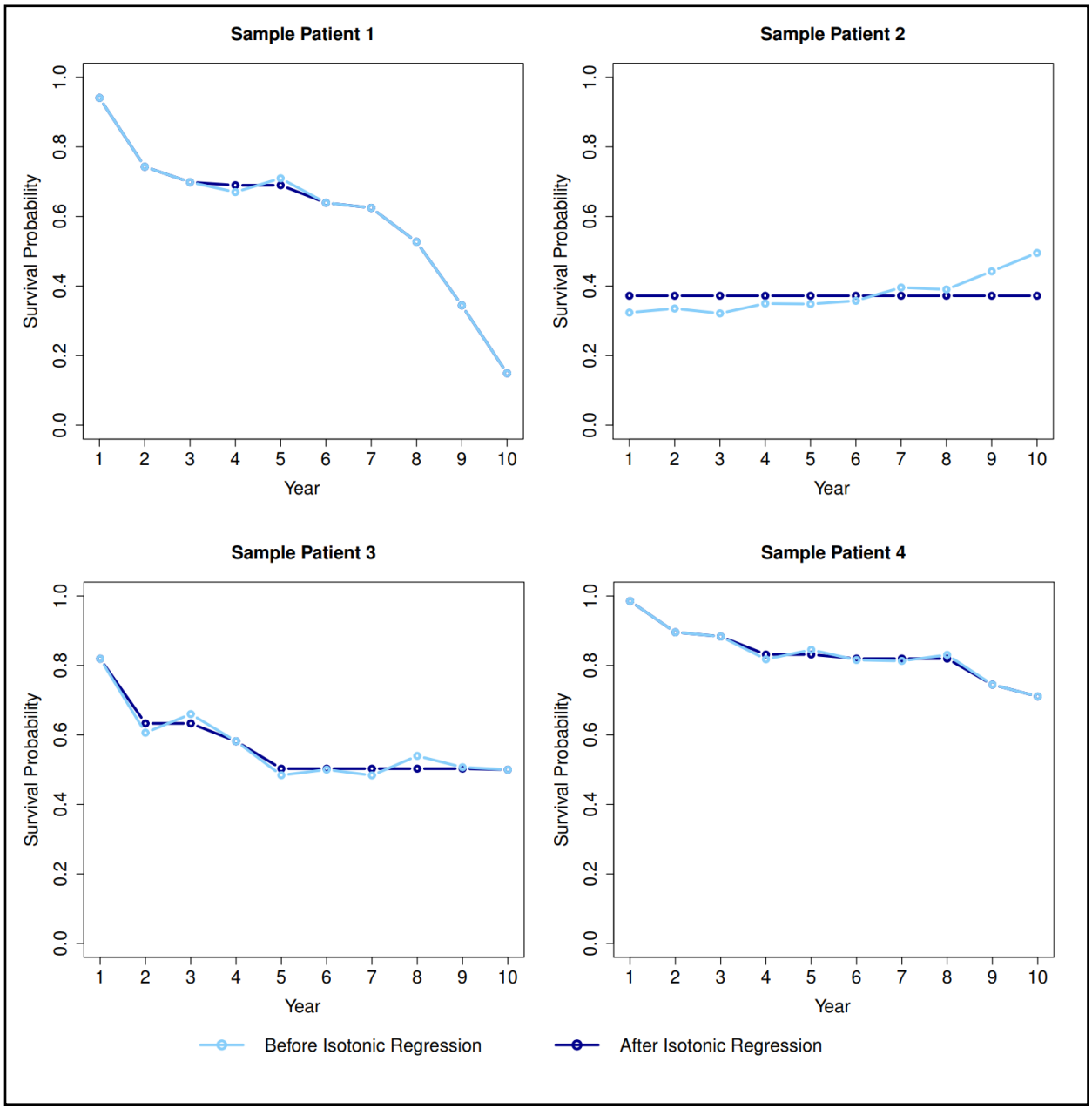


Figure 10: Application of Isotonic Regression on Sample Patients. The light and dark blue lines are associated with predicted probabilities before and after application of isotonic regression, respectively. Note when the dark blue line is not shown, this equates to a perfect overlap with the light blue line. We do not show month 1 here, for aesthetic purposes, since it is on a different time scale when compared to the yearly data.

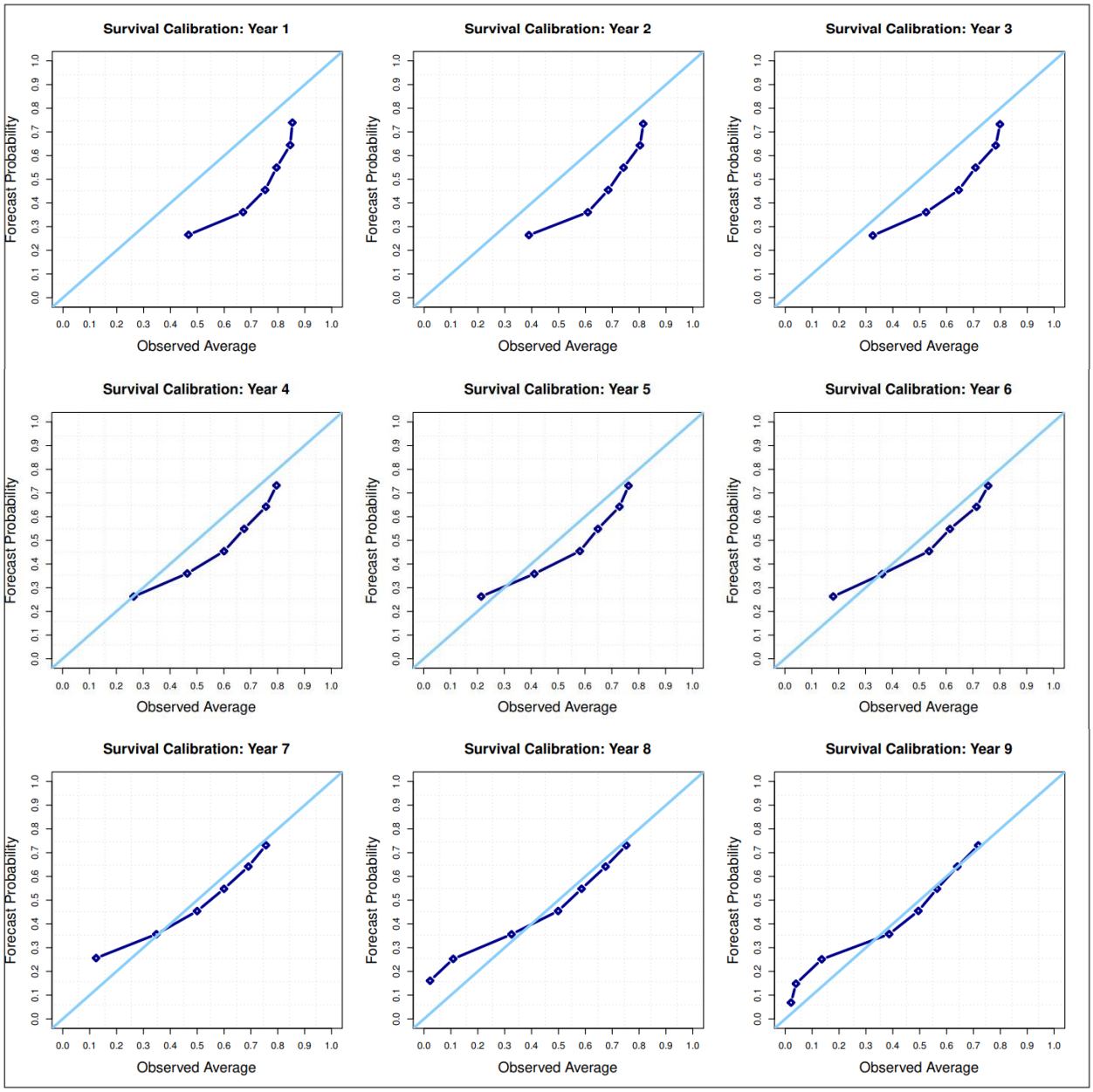


Figure 11: Plots of the observed versus forecast survival probabilities for Years $1-9$ (after isotonic regression is applied). The light blue line correspond to the baseline case of a perfect match between both probabilities. The dark blue line correspond to the results obtained from our machine learning model. To create each plot, we grouped the patients in our holdout set based on their forecast probability (in 0.1 increments) and we then calculated the observed average as: (# Survivals for year from group)/(Total # patients for year from group).

Moreover, Yoon, Zame [78] compared the performance of 13 machine learning models in predicting one-, three-, and ten-year survival post transplant. Their best model had AUC values of 0.641, 0.623, and 0.631 for the one-, three-, and ten-year survival, respectively. These results are close to our obtained results. We consider our similar prediction performance as a "bonus" outcome, as our main goal was to overcome the non-monotonic behavior in the aggregation of prediction outcomes from the models of [3, 78], while obtaining an individualized survival probability curve for each patient.

From the Stage I discussion above, there are three interesting observations to be made. First, based on the machine learning models that we investigated for the first year survival prediction, logistic regression is our recommended model. While good performance of logistic regression has been noted in [8, 62, 78], the literature typically recommends more complicated machine learning models (e.g., see [1, 22, 62, 78, 187]). Second, the logistic regression results reported in this study are comparable with approaches that did not consider monotonicity [3, 78] and those that did not result in an individualized prediction [1]. The small differences in our obtained performances are possibly due to: (a) the different starting/ending times utilized in the analysis of the UNOS data; (b) our data cleaning procedure, where we imputed empty cells with the factor level corresponding to unknown in the case of categorical variables; (c) our computed variables were selected in several of the final models for different time points (e.g., **ANCEF**, **CARD_SURG**, **DOPAMINE** and **ETH_MAT** were selected 2, 9, 7 and 10 times, respectively); and/or (d) our approach to group factor levels in order to generate indicator variables. Note that our decisions for (b)-(d) were informed by medical information that was reflected in our data cleaning procedure. This is not a typical case in the literature, where observations corresponding to missing variables are either removed or imputed using statistical methods, and the predictor variables are limited to those

provided in the UNOS dataset [3, 10]. Third, the sensitivity results (see Table 19) deteriorate over time, while the specificity improves as the time point for prediction increases. This means that our ability to predict survivals (since we coded them as "1") deteriorates for larger time frames, and the opposite for predicting deaths. We believe that this is reasonable since most people survive in the earlier years, and more deaths are expected as time increases.

In the second stage of our machine learning approach, we constructed a monotonically decreasing survival probability curve for each patient through the use of isotonic regression. Overall, the application of isotonic regression did not affect the overall shape of the curve when one compares Figure 11 and Figure 9. However, on an individual patient/transplantation-event, the results are now more medically compelling/explainable due to the monotonicity constraint. Additionally, the individualized and data-driven nature of the models should make it more appealing to medical practitioners as it brings us a step closer to personalized medicine [207].

The limitations of our study warrant further discussion. First, the goal of this study was to showcase how the use of isotonic regression can be used to calibrate and guarantee the monotonicity of survival probabilities over time. While we examined a large number of modeling approaches, we did not attempt to optimize the prediction performance (e.g., through a detailed investigation of parameter tuning). We examined these scenarios to show that even with the "best" modeling approach, non-monotone survival curves can be obtained. Second, the secondary and retrospective nature of our analysis from a registry database means that we cannot account for "the quality of the source data, the number of missing data, and the lack of standardization associated with multicenter studies (such as different immunosuppressive regimens and different matching criteria)" [1]. In addition, we have no control whether the data for the variables included in our model will continue to be collected in the future. By sharing our detailed code for data cleaning,

we explicitly show how we handled missing data, observations where we identified issues in data quality, and how we removed all variables that had an ending data per the time of our data acquisition. While our efforts cannot guarantee suitability for future changes in UNOS's data collection protocol, it allows researchers to easily build on our analysis if needed. Third, in our Stage I analysis, we examined only a few machine learning models (and their parameterizations). Thus, we cannot guarantee that logistic regression will be superior when compared to ensemble and hybrid models that were not considered in our analysis. However, we estimate that by considering speed, interpret-ability and prediction performance of our logistic regression model (where our AUC results are similar to the more advanced models found in \cite{yoon2018personalized}), our approach will be of practical value. Fourth, we did not consider how to optimize the smoothing/ calibration of the survival curve obtained from Stage I. We have only considered the use of isotonic regression since it will guarantee monotonicity. Future work can examine the use of other approaches to achieve monotonicity, while having a smoother function (e.g., the use of an exponential curve). The fifth limitation can be seen based on the results depicted Figs. 3 and 5, where our approach consistently underestimates the survival probabilities (when compared to observed rates) for the first four years. There are two important considerations that need to be emphasized: (a) it is unclear whether the previous methods in the literature would have similar performance characteristics since this type of analysis has not been done before (only metrics for the dichotomous classification are typically reported); and (b) the utility of our approach in practice is not diminished from this limitation. Specifically, from a medical professional's perspective knowing with a high degree of certainty that the prediction probabilities at 5+ years from transplant is accurate, is sufficient to obtain a lower bound on the survival probabilities for the earlier time periods.

4.5 Concluding Remarks

In conclusion, through our two-stage machine learning framework, we obtained data-driven and individualized survival probability curves for each transplantation event. This represents a significant contribution when compared to previous literature that: (a) had non-monotonic predictions over time [3, 78]; (b) attempted to account for the non-monotone predictions through features in their machine learning model [5, 6], which is an improvement over the methods in (a), however, such an approach does not guarantee monotonicity; and (c) achieved monotonic predictions through the use of a population adjustment approach [1, 61, 187], which makes such an approach no longer truly individualized. By providing our detailed code, we facilitate the reproducibility of our analysis and encourage future work in this important domain. Additionally, we have developed a web-based application (app) that facilitates the adoption of our approach. The app can be found at http://dataviz.miamioh.edu/Heart-Transplant/monotonic/. Our motivation for presenting this app stems from the work of [1] who noted that a practitioner-tool for transplantation can: (a) augment the confidence in the expected results post-transplant; (b) allow for an optimal allocation of organs; and (c) present a data-driven risk score/probability for graft failure over time. Note that, with the exception of [1, 61], presenting a tool for practitioners is not a common outcome in the literature.

Chapter 5

Summary Conclusion and Contributions

The main goal of this dissertation is providing an objective and systematic approach to improving longterm heart transplantation survival studies. Chapters 2-4 articulate the feasibility and success of the proposed approach. Systematic development of data mining study in heart transplantation field could improve predictive performance, optimize organ allocation, and provide consistent results in the consecutive timestamps. My approach utilizes the design of experiments' concepts to highlight the importance of factors and their combinations in the development of a data mining study. In the last chapter, I proposed the application of isotonic regression as a post calibration method for delivering a consistent and smooth predicted survival pattern for the individuals. This approach improved the quality of the survival predictions, too. The smooth monotonically decreasing survival pattern could facilities interpretation and application of data mining studies for organ allocation. In addition, an open source tool with a Graphical User Interface is developed to deliver long-term survival predictions based on the results of this study. This tool could extend the application of data mining techniques for longterm survival analysis.

References

1. Medved, D., et al., *Improving prediction of heart transplantation outcome using deep learning techniques.* Scientific reports, 2018. **8**(1): p. 3613.
2. Dag, A., *A Data Driven Framework to Identify the Critical Variables, Visualize Their Conditional Relations and Predict the Outcomes of US Heart Transplants.* 2016.
3. Dag, A., et al., *Predicting heart transplantation outcomes through data analytics.* Decision Support Systems, 2017. **94**: p. 42-52.
4. Montgomery, D.C., *Design and analysis of experiments*. 2017: John wiley & sons.
5. Ohno-Machado, L. and M.A. Musen, *Sequential versus standard neural networks for pattern recognition: an example using the domain of coronary heart disease.* Computers in biology and medicine, 1997. **27**(4): p. 267-281.
6. Ohno-Machado, L. and M.A. Musen, *Modular neural networks for medical prognosis: quantifying the benefits of combining neural networks for survival prediction.* Connection Science, 1997. **9**(1): p. 71-86.
7. Barlow, R.E. and H.D. Brunk, *The isotonic regression problem and its dual.* Journal of the American Statistical Association, 1972. **67**(337): p. 140-147.
8. Dag, A., et al., *A probabilistic data-driven framework for scoring the preoperative recipient-donor heart transplant survival.* Decision Support Systems, 2016. **86**: p. 1-12.
9. Nakayama, N., et al., *Algorithm to determine the outcome of patients with acute liver failure: a data-mining analysis using decision trees.* 2012. **47**(6): p. 664-677.
10. Oztekin, A., D. Delen, and Z.J.J.i.j.o.m.i. Kong, *Predicting the graft survival for heart–lung transplantation patients: An integrated data mining methodology.* 2009. **78**(12): p. e84-e96.
11. Kusiak, A., et al., *Predicting survival time for kidney dialysis patients: a data mining approach.* 2005. **35**(4): p. 311-327.
12. Lin, T.-h., N. Kaminski, and Z.J.B. Bar-Joseph, *Alignment and classification of time series gene expression in clinical studies.* 2008. **24**(13): p. i147-i155.
13. Cruz-Ramirez, M., et al., *Predicting patient survival after liver transplantation using evolutionary multi-objective artificial neural networks.* 2013. **58**(1): p. 37-49.
14. Tang, H., et al., *Validating prediction models of kidney transplant outcome using single center data.* Asaio Journal, 2011. **57**(3): p. 206-212.
15. Nilsaz-Dezfouli, H., et al., *Improving Gastric Cancer Outcome Prediction Using Single Time-Point Artificial Neural Network Models.* 2017. **16**: p. 1176935116686062.
16. Krakauer, H., et al., *Projected survival benefit as criterion for listing and organ allocation in heart transplantation.* 2005. **24**(6): p. 680-689.
17. Casadevall, A. and F.C. Fang, *Reproducible science*. 2010, Am Soc Microbiol.
18. Ellison, A.M., *Repeatability and transparency in ecological research.* Ecology, 2010. **91**(9): p. 2536-2539.
19. Medved, D., et al., *Improving prediction of heart transplantation outcome using deep learning techniques.* 2018. **8**(1): p. 3613.
20. Chi, C.-L., W.N. Street, and W.H. Wolberg. *Application of artificial neural network-based survival analysis on two breast cancer datasets*. in *AMIA Annual Symposium Proceedings*. 2007. American Medical Informatics Association.
21. Brown, T.S., et al., *Bayesian modeling of pretransplant variables accurately predicts kidney graft survival.* American journal of nephrology, 2012. **36**(6): p. 561-569.

22. Lin, R.S., et al., *Single and multiple time-point prediction models in kidney transplant outcomes.* Journal of biomedical informatics, 2008. **41**(6): p. 944-952.
23. Wirth, R. and J. Hipp. *CRISP-DM: Towards a standard process model for data mining*. in *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*. 2000. Citeseer.
24. HFSA. *Heart Failure Facts & Information*. [cited 2019 6/8/2019]; Available from: https://www.hfsa.org/patient/learn/.
25. Becker, G.S. and J.J. Elias, *Introducing incentives in the market for live and cadaveric organ donations.* Journal of economic perspectives, 2007. **21**(3): p. 3-24.
26. Iii, A.F.A., A.H. Barnett, and D.L. Kaserman, *Markets for organs: the question of supply.* Contemporary economic policy, 1999. **17**(2): p. 147-155.
27. Chapman, P., et al., *CRISP-DM 1.0 Step-by-step data mining guide.* 2000.
28. Cossé, T.J. and T.M. Weisenberger, *Words versus actions about organ donation: A four-year tracking study of attitudes and self-reported behavior.* Journal of Business Research, 2000. **50**(3): p. 297-303.
29. Chu, X., et al. *Data cleaning: Overview and emerging challenges*. in *Proceedings of the 2016 International Conference on Management of Data*. 2016. ACM.
30. Krishnan, S., et al. *A methodology for learning, analyzing, and mitigating social influence bias in recommender systems*. in *Proceedings of the 8th ACM Conference on Recommender systems*. 2014. ACM.
31. Krishnan, S., et al., *Activeclean: Interactive data cleaning while learning convex loss models.* 2016.
32. Xiao, H., et al. *Is feature selection secure against training data poisoning?* in *International Conference on Machine Learning*. 2015.
33. Russo, M.J., et al., *The effect of ischemic time on survival after heart transplantation varies by donor age: an analysis of the United Network for Organ Sharing database.* The Journal of thoracic and cardiovascular surgery, 2007. **133**(2): p. 554-559.
34. Krista Lentine, R.S., *OPTN/UNOS Thoracic Organ Transplantation Committee, Meeting Summary* 2017, U.S. Department of Health & Human Services.
35. Davies, R.R., et al., *Standard versus bicaval techniques for orthotopic heart transplantation: an analysis of the United Network for Organ Sharing database.* The Journal of thoracic and cardiovascular surgery, 2010. **140**(3): p. 700-708. e2.
36. Perito, E.R., et al., *Overweight and obesity in pediatric liver transplant recipients: Prevalence and predictors before and after transplant, United Network for Organ Sharing Data, 1987–2010.* Pediatric transplantation, 2012. **16**(1): p. 41-49.
37. Bohannon, P., et al. *A cost-based model and effective heuristic for repairing constraints by value modification*. in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. 2005. ACM.
38. Allison, P.D., *Missing data*. Vol. 136. 2001: Sage publications.
39. Cismondi, F., et al., *Missing data in medical databases: Impute, delete or classify?* 2013. **58**(1): p. 63-72.
40. Jerez, J.M., et al., *Missing data imputation using statistical and machine learning methods in a real breast cancer problem.* 2010. **50**(2): p. 105-115.
41. Musil, C.M., et al., *A comparison of imputation techniques for handling missing data.* 2002. **24**(7): p. 815-829.

42. Somasundaram, R. and R.J.I.J.o.C.A. Nedunchezhian, *Evaluation of three simple imputation methods for enhancing preprocessing of data with missing values.* 2011. **21**(10): p. 14-19.
43. Jakobsen, J.C., et al., *When and how should multiple imputation be used for handling missing data in randomised clinical trials–a practical guide with flowcharts.* 2017. **17**(1): p. 162.
44. Dong, Y. and C.-Y.J.J.S. Peng, *Principled missing data methods for researchers.* 2013. **2**(1): p. 222.
45. Bennett, D.A.J.A. and N.Z.j.o.p. health, *How can I deal with missing data in my study?* 2001. **25**(5): p. 464-469.
46. Van Buuren, S., H.C. Boshuizen, and D.L.J.S.i.m. Knook, *Multiple imputation of missing blood pressure covariates in survival analysis.* 1999. **18**(6): p. 681-694.
47. Xia, J. and D.S.J.N.p. Wishart, *Web-based inference of biological patterns, functions and pathways from metabolomic data using MetaboAnalyst.* 2011. **6**(6): p. 743.
48. Brzustowicz, M.R., *Data Science with Java: Practical Methods for Scientists and Engineers*. 2017: " O'Reilly Media, Inc.".
49. Dong, D. and T.J. McAvoy, *Nonlinear principal component analysis—based on principal curves and neural networks.* Computers & Chemical Engineering, 1996. **20**(1): p. 65-78.
50. Gheyas, I.A. and L.S.J.P.r. Smith, *Feature subset selection in large dimensionality domains.* 2010. **43**(1): p. 5-13.
51. Guyon, I. and A. Elisseeff, *An introduction to variable and feature selection.* Journal of machine learning research, 2003. **3**(Mar): p. 1157-1182.
52. Saeys, Y., I. Inza, and P. Larrañaga, *A review of feature selection techniques in bioinformatics.* bioinformatics, 2007. **23**(19): p. 2507-2517.
53. Das, S. *Filters, wrappers and a boosting-based hybrid for feature selection*. in *Icml*. 2001.
54. Kohavi, R. and G.H.J.A.i. John, *Wrappers for feature subset selection.* 1997. **97**(1-2): p. 273-324.
55. Tang, J., et al., *Feature selection for classification: A review.* 2014: p. 37.
56. Yu, L. and H. Liu. *Feature selection for high-dimensional data: A fast correlation-based filter solution*. in *Proceedings of the 20th international conference on machine learning (ICML-03)*. 2003.
57. Langley, P. *Selection of relevant features in machine learning*. in *Proceedings of the AAAI Fall symposium on relevance*. 1994.
58. Alpaydin, E., *Introduction to machine learning*. 2009: MIT press.
59. Kaplan, E.L. and P. Meier, *Nonparametric estimation from incomplete observations.* Journal of the American statistical association, 1958. **53**(282): p. 457-481.
60. Lunn, M. and D. McNeil, *Applying Cox regression to competing risks.* Biometrics, 1995: p. 524-532.
61. Nilsson, J., et al., *The International Heart Transplant Survival Algorithm (IHTSA): a new model to improve organ sharing and survival.* PloS one, 2015. **10**(3): p. e0118644.
62. Delen, D., A. Oztekin, and Z.J. Kong, *A machine learning-based approach to prognostic analysis of thoracic transplantations.* Artificial Intelligence in Medicine, 2010. **49**(1): p. 33-42.
63. Bollschweiler, E., *Benefits and limitations of Kaplan–Meier calculations of survival chance in cancer surgery.* Langenbeck's archives of surgery, 2003. **388**(4): p. 239-244.

64. Stel, V.S., et al., *Survival analysis I: the Kaplan-Meier method.* Nephron Clinical Practice, 2011. **119**(1): p. c83-c88.
65. Babińska, M., et al., *Limitations of cox proportional hazards analysis in mortality prediction of patients with acute coronary syndrome.* Studies in Logic, Grammar and Rhetoric, 2015. **43**(1): p. 33-48.
66. Chawla, N.V., et al., *SMOTE: synthetic minority over-sampling technique.* Journal of artificial intelligence research, 2002. **16**: p. 321-357.
67. Kamei, Y., et al. *The effects of over and under sampling on fault-prone module detection*. in *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*. 2007. IEEE.
68. Lunardon, N., G. Menardi, and N.J.R.j. Torelli, *ROSE: A Package for Binary Imbalanced Learning.* 2014. **6**(1).
69. McCarthy, K., B. Zabar, and G. Weiss. *Does cost-sensitive learning beat sampling for classifying rare classes?* in *Proceedings of the 1st international workshop on Utility-based data mining*. 2005. ACM.
70. Wu, Z., et al. *An ensemble random forest algorithm for insurance big data analysis*. in *Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC), 2017 IEEE International Conference on*. 2017. IEEE.
71. Massie, A.B., L. Kuricka, and D.L. Segev, *Big data in organ transplantation: registries and administrative claims.* American Journal of Transplantation, 2014. **14**(8): p. 1723-1730.
72. Du, J., et al., *Online Sequential Extreme Learning Machine with Under-Sampling and Over-Sampling for Imbalanced Big Data Classification*, in *Proceedings of ELM-2016*. 2018, Springer. p. 229-239.
73. He, H. and E.A. Garcia, *Learning from imbalanced data.* IEEE Transactions on Knowledge & Data Engineering, 2008(9): p. 1263-1284.
74. He, H., et al. *ADASYN: Adaptive synthetic sampling approach for imbalanced learning*. in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. 2008. IEEE.
75. Verbiest, N., et al., *Preprocessing noisy imbalanced datasets using SMOTE enhanced with fuzzy rough prototype selection.* Applied Soft Computing, 2014. **22**: p. 511-517.
76. Dreiseitl, S. and L. Ohno-Machado, *Logistic regression and artificial neural network classification models: a methodology review.* Journal of biomedical informatics, 2002. **35**(5-6): p. 352-359.
77. Krakauer, H., M.J.-Y. Lin, and R.C. Bailey, *Projected survival benefit as criterion for listing and organ allocation in heart transplantation.* The Journal of heart and lung transplantation, 2005. **24**(6): p. 680-689.
78. Yoon, J., et al., *Personalized survival predictions via Trees of Predictors: An application to cardiac transplantation.* PloS one, 2018. **13**(3): p. e0194985.
79. Oztekin, A., et al., *A decision analytic approach to predicting quality of life for lung transplant recipients: A hybrid genetic algorithms-based methodology.* European Journal of Operational Research, 2018. **266**(2): p. 639-651.
80. Li, J., et al., *Bayes net classifiers for prediction of renal graft status and survival period.* World Academy of Science, Engineering and Technology, 2010. **39**.

81. Akl, A., A.M. Ismail, and M. Ghoneim, *Prediction of graft survival of living-donor kidney transplantation: nomograms or artificial neural networks?* Transplantation, 2008. **86**(10): p. 1401-1406.
82. Lasserre, J., et al., *Predicting the outcome of renal transplantation.* Journal of the American Medical Informatics Association, 2011. **19**(2): p. 255-262.
83. Khan, M.E., et al. *Decision Support System for Renal Transplantation*. in *Avishek Choudhury, Ehsan khan, Decision Support System for Renal Transplantation, Proceedings of the 2018 IISE Annual Conference, Orlando*. 2018.
84. Goldfarb-Rumyantzev, A.S., et al., *Prediction of 3-yr cadaveric graft survival based on pre-transplant variables in a large national dataset.* Clinical transplantation, 2003. **17**(6): p. 485-497.
85. Tang, H., et al., *Predicting three-year kidney graft survival in recipients with systemic lupus erythematosus.* Asaio Journal, 2011. **57**(4): p. 300-309.
86. Krikov, S., et al., *Predicting kidney transplant survival using tree-based modeling.* Asaio Journal, 2007. **53**(5): p. 592-600.
87. Oztekin, A., *An analytical approach to predict the performance of thoracic transplantations.* 2012.
88. Agrawal, A., et al. *Lung transplant outcome prediction using unos data*. in *Big Data, 2013 IEEE International Conference on*. 2013. IEEE.
89. Misiunas, N., et al., *DEANN: A healthcare analytic methodology of data envelopment analysis and artificial neural networks for the prediction of organ recipient functional status.* Omega, 2016. **58**: p. 46-54.
90. Medved, D., P. Nugues, and J. Nilsson. *Predicting the outcome for patients in a heart transplantation queue using deep learning*. in *Engineering in Medicine and Biology Society (EMBC), 2017 39th Annual International Conference of the IEEE*. 2017. IEEE.
91. Raji, C. and S.V. Chandra, *Graft survival prediction in liver transplantation using artificial neural network models.* Journal of computational science, 2016. **16**: p. 72-78.
92. Cucchetti, A., et al., *Artificial neural network is superior to MELD in predicting mortality of patients with end-stage liver disease.* Gut, 2007. **56**(2): p. 253-258.
93. Bradley, A.P., *The use of the area under the ROC curve in the evaluation of machine learning algorithms.* Pattern recognition, 1997. **30**(7): p. 1145-1159.
94. Kubat, M., R.C. Holte, and S.J.M.l. Matwin, *Machine learning for the detection of oil spills in satellite radar images.* 1998. **30**(2-3): p. 195-215.
95. Kass, G.V.J.J.o.t.R.S.S.S.C., *An exploratory technique for investigating large quantities of categorical data.* 1980. **29**(2): p. 119-127.
96. Hastie, T. and J.J.R.f.h.w.w.s.e.h.P.G.V.p.A.S. Qian, *Glmnet vignette.* 2014. **20**: p. 2016.
97. Genuer, R., J.-M. Poggi, and C.J.P.R.L. Tuleau-Malot, *Variable selection using random forests.* 2010. **31**(14): p. 2225-2236.
98. Kursa, M.B. and W.R. Rudnicki, *Feature selection with the Boruta package.* J Stat Softw, 2010. **36**(11): p. 1-13.
99. Komarek, P. and A.W. Moore. *Fast Robust Logistic Regression for Large Sparse Datasets with Binary Outputs*. in *AISTATS*. 2003.
100. Maher, J.J., T.K.J.I.S.i.A. Sen, Finance, and Management, *Predicting bond ratings using neural networks: a comparison with logistic regression.* 1997. **6**(1): p. 59-72.
101. Shaikhina, T., et al., *Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation.* 2017.

102. Breiman, L.J.M.l., *Random forests.* 2001. **45**(1): p. 5-32.
103. Zweig, M.H. and G.J.C.c. Campbell, *Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine.* 1993. **39**(4): p. 561-577.
104. Angelis, L., I. Stamelos, and M. Morisio. *Building a software cost estimation model based on categorical data*. in *Proceedings Seventh International Software Metrics Symposium*. 2001. IEEE.
105. Niculescu-Mizil, A., et al. *Winning the KDD cup orange challenge with ensemble selection*. in *KDD-Cup 2009 Competition*. 2009.
106. Fu, X., L.J.I.T.o.S. Wang, Man,, and P.B. Cybernetics, *Data dimensionality reduction with application to simplifying RBF network structure and improving classification performance.* 2003. **33**(3): p. 399-409.
107. Kambhatla, N. and T.K.J.N.c. Leen, *Dimension reduction by local principal component analysis.* 1997. **9**(7): p. 1493-1516.
108. Tümer, M.B. and M.C. Demir. *A genetic approach to data dimensionality reduction using a special initial population*. in *international Work-Conference on the Interplay between natural and artificial computation*. 2005. Springer.
109. Blum, A.L. and P.J.A.i. Langley, *Selection of relevant features and examples in machine learning.* 1997. **97**(1-2): p. 245-271.
110. Lund, L.H., et al., *The Registry of the International Society for Heart and Lung Transplantation: thirty-fourth adult heart transplantation report—2017; focus theme: allograft ischemic time.* 2017. **36**(10): p. 1037-1046.
111. Seiffert, C., et al. *RUSBoost: Improving classification performance when training data is skewed*. in *2008 19th International Conference on Pattern Recognition*. 2008. IEEE.
112. Bao-Liang, L., et al., *Learning from imbalanced data sets with a Min-Max modular support vector machine.* 2011. **6**(1): p. 56-71.
113. Elrahman, S.M.A., A.J.J.o.N. Abraham, and I. Computing, *A review of class imbalance problem.* 2013. **1**(2013): p. 332-340.
114. Allias, N., et al. *A hybrid gini pso-svm feature selection based on taguchi method: an evaluation on email filtering*. in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*. 2014. ACM.
115. Chuang, L.-Y., et al., *Correlation-based gene selection and classification using Taguchi-BPSO.* 2010. **49**(03): p. 254-268.
116. Kwak, N. and C.-H.J.I.t.o.n.n. Choi, *Input feature selection for classification problems.* 2002. **13**(1): p. 143-159.
117. Yang, C.-H., et al. *A novel ga-taguchi-based feature selection method*. in *International Conference on Intelligent Data Engineering and Automated Learning*. 2008. Springer.
118. Suzuki, A. and K.J.I.T.o.I.I. Ryu, *Feature selection method for estimating systolic blood pressure using the taguchi method.* 2013. **10**(2): p. 1077-1085.
119. Maji, U., M. Mitra, and S.J.E.S.w.A. Pal, *Imposed target based modification of Taguchi method for feature optimisation with application in arrhythmia beat detection.* 2016. **56**: p. 268-281.
120. Packianather, M., et al., *Optimizing the parameters of multilayered feedforward neural networks through Taguchi design of experiments.* 2000. **16**(6): p. 461-473.
121. Wang, Q., D. Stockton, and P.J.I.j.o.p.r. Baguley, *Process cost modelling using neural networks.* 2000. **38**(16): p. 3811-3821.

122. Yang, S.-M., G.J.J.o.D.S. Lee, Measurement,, and Control, *Neural network design by using Taguchi method.* 1999. **121**(3): p. 560-563.
123. Khaw, J.F., B. Lim, and L.E.J.N. Lim, *Optimal design of neural networks using the Taguchi method.* 1995. **7**(3): p. 225-245.
124. Peterson, G.E., et al., *Using Taguchi's method of experimental design to control errors in layered perceptrons.* 1995. **6**(4): p. 949-961.
125. Tortum, A., et al., *The investigation of model selection criteria in artificial neural networks by the Taguchi method.* 2007. **386**(1): p. 446-468.
126. Bashiri, M. and A.F.J.S.I. Geranmayeh, *Tuning the parameters of an artificial neural network using central composite design and genetic algorithm.* 2011. **18**(6): p. 1600-1608.
127. Otok, B.W., B.S. Ulama, and A.J. Endharta. *Design of Experiment to Optimize the Architecture of Wavelet Neural Network for Forecasting the Tourist Arrivals in Indonesia*. in *International Conference on Informatics Engineering and Information Science*. 2011. Springer.
128. Ortiz-Rodríguez, J., M. Martínez-Blanco, and H. Vega-Carrillo. *Robust design of artificial neural networks applying the Taguchi methodology and DoE*. in *Electronics, Robotics and Automotive Mechanics Conference (CERMA'06)*. 2006. IEEE.
129. Pontes, F.J., et al., *Design of experiments and focused grid search for neural network parameter optimization.* 2016. **186**: p. 22-34.
130. Tyasnurita, R., E. Özcan, and R. John. *Learning heuristic selection using a time delay neural network for open vehicle routing*. in *2017 IEEE Congress on Evolutionary Computation (CEC)*. 2017. IEEE.
131. Ardalani-Farsa, M., S.J.C. Zolfaghari, and Systems, *Taguchi's design of experiment in combination selection for a chaotic time series forecasting method using ensemble artificial neural networks.* 2013. **44**(4): p. 351-377.
132. Kumar, D., et al., *Optimization of neural network parameters using Grey–Taguchi methodology for manufacturing process applications.* 2015. **229**(14): p. 2651-2664.
133. Balestrassi, P.P., et al., *Design of experiments on neural network's training for nonlinear time series forecasting.* 2009. **72**(4-6): p. 1160-1178.
134. Chan, K.Y., et al., *Selection of significant on-road sensor data for short-term traffic flow forecasting using the Taguchi method.* 2011. **8**(2): p. 255-266.
135. Lin, T., et al. *A systematic approach to the optimization of artificial neural networks*. in *2011 IEEE 3rd International Conference on Communication Software and Networks*. 2011. IEEE.
136. Sukthomya, W., J.J.N.C. Tannock, and Applications, *The optimisation of neural network parameters using Taguchi's design of experiments approach: an application in manufacturing process modelling.* 2005. **14**(4): p. 337-344.
137. Kim, Y.-S. and B.-J.J.E.A.o.A.I. Yum, *Robust design of multilayer feedforward neural networks: an experimental approach.* 2004. **17**(3): p. 249-263.
138. Hsu, W.-C. and T.-Y.J.J.o.C.I.T. Yu, *E-mail spam filtering based on support vector machines with Taguchi method for parameter selection.* 2010. **5**(8): p. 78-88.
139. Huang, M.-L., et al., *SVM-RFE based feature selection and Taguchi parameters optimization for multiclass SVM classifier.* 2014. **2014**.

140. Huang, M.L., Y.H. Hung, and E.J.J.I.J.o.A.I.T. Lin, *Effects of SVM parameter optimization based on the parameter design of Taguchi method.* 2011. **20**(03): p. 563-575.
141. Erfanifard, Y., et al., *Tree crown delineation on UltraCam-D aerial imagery with SVM classification technique optimised by Taguchi method in Zagros woodlands.* 2014. **5**(4): p. 300-314.
142. Zare, M., N. Behnia, and D.J.J.o.t.I.S.o.R.S. Gabriels, *Assessment of Land Cover Changes Using Taguchi-Based Optimized SVM Classification Approach.* 2019. **47**(1): p. 45-52.
143. Kleinbaum, D.G., et al., *Logistic regression*. 2002: Springer.
144. Balakrishnama, S., A.J.I.f.S. Ganapathiraju, and i. Processing, *Linear discriminant analysis-a brief tutorial.* 1998. **18**: p. 1-8.
145. Friedman, J., et al., *Lasso and Elastic-Net Regularized Generalized Linear Models. R-package version 2.0-5. 2016*. 2016.
146. Kuhn, M., et al., *caret: Classification and regression training. R package version 6.0–81*. 2018.
147. Shalev-Shwartz, S. and S. Ben-David, *Understanding machine learning: From theory to algorithms*. 2014: Cambridge university press.
148. Farid, D.M., et al., *Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks.* 2014. **41**(4): p. 1937-1946.
149. Wehrens, R. and B.-H. Mevik, *The pls package: principal component and partial least squares regression in R.* 2007.
150. Wright, M.N. and A.J.a.p.a. Ziegler, *ranger: A fast implementation of random forests for high dimensional data in C++ and R.* 2015.
151. Friedman, J.H.J.C.s. and d. analysis, *Stochastic gradient boosting.* 2002. **38**(4): p. 367-378.
152. Habyarimana, E., et al., *Towards Predictive Modeling of Sorghum Biomass Yields Using Fraction of Absorbed Photosynthetically Active Radiation Derived from Sentinel-2 Satellite Imagery and Supervised Machine Learning Techniques.* 2019. **9**(4): p. 203.
153. Rowlands, H. and J.J.A.A. Antony, *Application of design of experiments to a spot welding process.* 2003. **23**(3): p. 273-279.
154. Anderson, M.J. and P.J. Whitcomb, *RSM simplified: optimizing processes using response surface methods for design of experiments*. 2016: Productivity press.
155. Subra, P., P.J.I. Jestin, and e.c. research, *Screening design of experiment (DOE) applied to supercritical antisolvent process.* 2000. **39**(11): p. 4178-4184.
156. Ng, W.W., et al., *Diversified sensitivity-based undersampling for imbalance classification problems.* 2014. **45**(11): p. 2402-2412.
157. Chawla, N.V., et al., *SMOTE: synthetic minority over-sampling technique.* 2002. **16**: p. 321-357.
158. Knox, S.W., *Machine learning: a concise introduction*. Vol. 285. 2018: John Wiley & Sons.
159. Friedman, J., T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Vol. 1. 2001: Springer series in statistics New York.
160. Attoh-Okine, N.O., et al., *Multivariate adaptive regression (MARS) and hinged hyperplanes (HHP) for doweled pavement performance modeling.* 2009. **23**(9): p. 3020-3023.

161. Hastie, T., R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction, Springer Series in Statistics*. 2009, Springer New York.
162. Nguyen, T.T. and Y.J.S.P. Tsoy, *A kernel PLS based classification method with missing data handling.* 2017. **58**(1): p. 211-225.
163. Wee, L.J., et al. *SVM-based prediction of linear B-cell epitopes using Bayes Feature Extraction*. in *BMC genomics*. 2010. BioMed Central.
164. Babajide Mustapha, I. and F.J.M. Saeed, *Bioactive molecule prediction using extreme gradient boosting.* 2016. **21**(8): p. 983.
165. Timofeev, R.J.H.U., Berlin, *Classification and regression trees (CART) theory and applications.* 2004.
166. Moore, A.W.J.S.o.C.S.C.M.U., *Cross-validation for detecting and preventing overfitting.* 2001.
167. Center, O.S., *Ohio supercomputer center*. 1987.
168. Centers for Disease Control and Prevention. Division for Heart Disease and Stroke Prevention. *Heart failure fact sheet,*. 2019, January 8; Available from: https://www.cdc.gov/dhdsp/data_statistics/fact_sheets/fs_heart_failure.htm.
169. Savarese, G. and L.H.J.C.f.r. Lund, *Global public health burden of heart failure.* 2017. **3**(1): p. 7.
170. Ponikowski, P., et al., *Heart failure: preventing disease and death worldwide.* 2014. **1**(1): p. 4-25.
171. Cook, C., et al., *The annual global economic burden of heart failure.* 2014. **171**(3): p. 368-376.
172. Mozaffarian, D., et al., *Heart disease and stroke statistics-2016 update a report from the American Heart Association.* 2016. **133**(4): p. e38-e48.
173. Benjamin, E.J., et al., *Heart disease and stroke statistics-2017 update: a report from the American Heart Association.* 2017. **135**(10): p. e146-e603.
174. Heidenreich, P.A., et al., *Forecasting the impact of heart failure in the United States: a policy statement from the American Heart Association.* 2013. **6**(3): p. 606-619.
175. The national heart, l., and blood institute, U.S. Department of Health & Human Service. *Heart failure*. 2017, September 22; Available from: https://www.nhlbi.nih.gov/health-topics/heart-failure.
176. Allemani, C., et al., *Global surveillance of trends in cancer survival 2000–14 (CONCORD-3): analysis of individual records for 37 513 025 patients diagnosed with one of 18 cancers from 322 population-based registries in 71 countries.* 2018. **391**(10125): p. 1023-1075.
177. Ansari, D., et al., *CODUSA-Customize Optimal Donor Using Simulated Annealing In Heart Transplantation.* 2013. **3**: p. 1922.
178. Parissis, J., et al., *Advanced End-Stage Heart Failure: Epidemiology and Management*, in *The Pathology of Cardiac Transplantation*. 2016, Springer. p. 13-20.
179. Ponikowski, P., et al., *2016 ESC Guidelines for the diagnosis and treatment of acute and chronic heart failure: The Task Force for the diagnosis and treatment of acute and chronic heart failure of the European Society of Cardiology (ESC). Developed with the special contribution of the Heart Failure Association (HFA) of the ESC.* 2016. **18**(8): p. 891-975.

180. Wilhelm, M.J.J.J.o.t.d., *Long-term outcome following heart transplantation: current perspective.* 2015. **7**(3): p. 549.
181. Sharing, U.-.-U.N.f.O. *Transplant trends.* 2019, July 9]; Available from: https://unos.org/data/transplant-trends/.
182. Sharing, U.-U.N.f.O., *National data.*
183. Alraies, M.C. and P.J.J.o.t.d. Eckman, *Adult heart transplant: indications and outcomes.* 2014. **6**(8): p. 1120.
184. FRoM, P.J.C.C.j.o.m., *Advanced heart failure: Transplantation, LVADs, and beyond.* 2013. **80**(1): p. 33.
185. Hong, K.N., et al., *Who is the high-risk recipient? Predicting mortality after heart transplant using pretransplant donor and recipient risk factors.* The Annals of thoracic surgery, 2011. **92**(2): p. 520-527.
186. Kilic, A., et al., *What predicts long-term survival after heart transplantation? An analysis of 9,400 ten-year survivors.* 2012. **93**(3): p. 699-704.
187. Yoo, K.D., et al., *A Machine Learning Approach Using Survival Statistics to Predict Graft Survival in Kidney Transplant Recipients: A Multicenter Cohort Study.* Scientific Reports, 2017. **7**(1): p. 8904.
188. Lund, L.H., et al., *The registry of the International Society for Heart and Lung Transplantation: thirty-third adult heart transplantation report—2016; focus theme: primary diagnostic indications for transplant.* 2016. **35**(10): p. 1158-1169.
189. Lund, L.H., et al., *The registry of the International Society for Heart and Lung Transplantation: thirty-first official adult heart transplant report—2014; focus theme: retransplantation.* 2014. **33**(10): p. 996-1008.
190. Lund, L.H., et al., *The Registry of the International Society for Heart and Lung Transplantation: Thirty-second Official Adult Heart Transplantation Report--2015; Focus Theme: Early Graft Failure.* 2015. **34**(10): p. 1244.
191. Goodman, A., et al., *Ten simple rules for the care and feeding of scientific data*. 2014, Public Library of Science.
192. Peng, R.D.J.S., *Reproducible research in computational science.* 2011. **334**(6060): p. 1226-1227.
193. Stodden, V., P. Guo, and Z.J.P.o. Ma, *Toward reproducible computational research: an empirical analysis of data and code policy adoption by journals.* 2013. **8**(6): p. e67111.
194. Wu, L., et al., *Feature Ranking in Predictive Models for Hospital-Acquired Acute Kidney Injury.* 2018. **8**(1): p. 17298.
195. Han, J., J. Pei, and M. Kamber, *Data mining: concepts and techniques*. 2011: Elsevier.
196. UNOS. *About UNOS.* 2019, July 15]; Available from: https://unos.org/about/.
197. De Jonge, E. and M. Van Der Loo, *An introduction to data cleaning with R*. 2013: Statistics Netherlands Heerlen.
198. García-Laencina, P.J., et al., *Pattern classification with missing data: a review*. 2010. **19**(2): p. 263-282.
199. Long, J.S. and J. Freese, *Regression models for categorical dependent variables using Stata*. 2006: Stata press.
200. James, G., et al., *An introduction to statistical learning*. Vol. 112. 2013: Springer.
201. Kotsiantis, S., et al., *Handling imbalanced datasets: A review.* 2006. **30**(1): p. 25-36.
202. Menardi, G., N.J.D.M. Torelli, and K. Discovery, *Training and assessing classification rules with imbalanced data.* 2014. **28**(1): p. 92-122.

203. Decruyenaere, A., et al., *Prediction of delayed graft function after kidney transplantation: comparison between logistic regression and machine learning methods.* 2015. **15**(1): p. 83.
204. Barlow, R.E., et al., *Statistical inference under order restrictions: The theory and application of isotonic regression*. 1972, Wiley New York.
205. Mair, P., K. Hornik, and J.J.J.o.s.s. de Leeuw, *Isotone optimization in R: pool-adjacent-violators algorithm (PAVA) and active set methods.* 2009. **32**(5): p. 1-24.
206. Swets, J.A.J.S., *Measuring the accuracy of diagnostic systems.* 1988. **240**(4857): p. 1285-1293.
207. Ansari, D., et al., *CODUSA - Customize Optimal Donor Using Simulated Annealing In Heart Transplantation.* Scientific Reports, 2013. **3**: p. 1922.

Appendix

This Appendix documents the data cleaning and preparation procedure, variable selection, statistical modeling, and survival probability calibration procedures used in these research articles: 1) A Design of Experiments Approach for Improving Performance of Machine Learning Algorithms in predicting Survival of Transplant Surgeries. & 2) A Two-Stage Machine Learning Approach to Predict Heart Transplantation Survival Probabilities over Time with a Monotonic Probability Constraint.

Details of the codes and the functions are available over Github of the study: https://github.com/transplantation/unos-ht

Data was provided from the **UNOS** registry by staff at the US, United Network for Organ Sharing.

The reader can **show** any code chunk by clicking on the *code* button. We chose to make the default for the code hidden since we: (a) wanted to improve the readability of this document; and (b) assumed that the readers will not be interested in reading every code chunk.

Section: Loading Data & Data Clearing and Preparation

The snippet below documents the list of **R** packages and functions that were used in this research. For convenience, we used the pacman package since it allows for installing/loading the needed packages in one-step.

```
rm(list = ls()) # clear global environment
graphics.off() # close all graphics
library(pacman) # needs to be installed first
# p_load is equivalent to combining both install.packages() and library()
p_load(haven,dplyr,caret,foreign,glmnet,
    lubridate,dataPreparation,httr, DT, stringr, AUC, snow, testit,caretEnsemble,
    C50, Biocomb, varImp, party,
    randomForest,
```

```
      kernlab,gower,
      e1071,Boruta,ROSE,DMwR)
# UNOS data are loaded from this local drive, data could be obtained from www.unos.org
data.path <- "C:\\Users\\hza0020\\OneDrive - Auburn University\\Transplant\\BUAL-LAB\\DoE_10years
\\"

source("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/functions.R")
```

Loading Data and Assigning IDs for each case

In this snippet below, we load the data file and the form that records the information regarding variables in the data. Both files are provided by UNOS but we added one column: **INTERPRETATION_TYPE** to the information form. It records the variable type for each variable in the data. This is a varaible we created based on the code book provided by UNOS. The information can be found here. The data dictionary can be found here.

```
# load data set
heart.df <- read_sas(paste0(data.path,"thoracic_data.sas7bdat"))

# add ids for each row
heart.df$ID <- row.names(heart.df)
# heart.form contains the variable definitions of each variable in data
heart.form <- read.csv("C:/Users/hza0020/OneDrive - Auburn University/Transplant/BUAL-LAB/DoE_
10years/Interpretation_type.csv")
```

Here is the information included in the form.

Dropping un-used variables and observations

Since our focus is to study the long term survival prediction for an adult patient after a heart transplant is performed, we drop some variables and observations, based on the following criteria:

- variables that did not end & those added before 2000
- patients who were under 18 years old
- patients whose weights were less than the 0.01-th percentile of all patients' weights
- patients whose heights were less that the 0.01-th percentile of all patients' heights

- patient who did not have information of transplanted organ
- patients who has lung transplant, too.
- variables that are not interesting/relevant for survival analysis (e.g. follow up number, Donor ID, Dates, and …)

```
## In the following lines, we want to find variables that are not used after 2000 or they were added after 2000.
# Frist, we removes whitespace from start and end of each element in the variable VAR.END.DATE
var.end.dates <- trimws(heart.form$VAR.END.DATE) %>% str_trim()

# Second, identify the varaibles that are still used.
names.of.var.did.not.end <- heart.form[which(var.end.dates==""), 1]

# Make sure ID is included.
names.of.var.did.not.end <- c(as.character(names.of.var.did.not.end), "ID")

# Third, identify the variables that are not used now.
vars_ended <- heart.form[which(heart.form$VARIABLE.NAME %in% names(heart.df)[!(names(heart.df) %in% names.of.var.did.not.end)]),(1:2)]

# Figure out which date each variable was added
vars.added.dates <- heart.form$VAR.START.DATE %>% as.character.Date()

# Fix an error from the information, there were two dates corresponding to one variable. We chose the later date.
vars.added.dates[which(vars.added.dates=="01-Oct-87, 01-Oct-90")] <- "01-Oct-90"

# Figure out which year each variable was added and add this information to the heart form
heart.form$YR_ADDED <- sapply(vars.added.dates, function(x) str_extract_all(x,"[0-9]{1,2}")[[1]][2]) %>% as.integer()

# available variables added before 2000 and they are still used now
vars.added.before.2000 <- subset(heart.form, YR_ADDED>=87, select = c(1))
vars.added.NA <- subset(heart.form,is.na(YR_ADDED),select = c(1))
vars.added.all <- rbind(vars.added.before.2000,vars.added.NA)
vars.added.all <- vars.added.all[["VARIABLE.NAME"]] %>%
  as.character()

vars.added.all <- c(as.character(vars.added.all), "ID")

# Based on the criteria we have to find a subset of data: getting rid of variables that are ended and patients who were under 18 or too light or too short or didn't have a heart transplant.

heart.df.cleaned <- subset(heart.df, WL_ORG=="HR") %>% # Heart
  subset(AGE>=18) %>% # Adults only
  # we excluded too light or too short people
  subset(WGT_KG_DON_CALC >= quantile(WGT_KG_DON_CALC, 0.0001, na.rm = TRUE)) %>%
```

```
  subset(WGT_KG_TCR >= quantile(WGT_KG_TCR, 0.0001, na.rm = TRUE)) %>%
  subset(HGT_CM_DON_CALC >= quantile(HGT_CM_DON_CALC, 0.0001, na.rm = TRUE)) %>%
  subset(HGT_CM_TCR >= quantile(HGT_CM_TCR, 0.0001, na.rm = TRUE)) %>%
  subset(select=intersect(names.of.var.did.not.end,vars.added.all))

# Find variables that are related to dates
vars_discarded <- heart.form %>%
  subset(INTERPRETATION_TYPE=="D", select=c(1,2))
heart.discard <- vars_discarded$VARIABLE.NAME %>% as.character()

# Identify the variables that are related to dates in the data and remove them
heart.discard <- intersect(heart.discard, colnames(heart.df.cleaned))

heart.df.cleaned <- select(heart.df.cleaned, -heart.discard)
```

Remove variables that are related to post transplant

In the snippet below, we identify and remove the variables that are post transplant based on the information from the heart form and form Section Descriptors.

```
vars.post.trans.index1 <-  sapply(heart.form$FORM.SECTION,
                function(x) str_detect(x, "POST TRANSPLANT CLINICAL INFORMATION"))
vars.post.trans.index2 <-  sapply(heart.form$FORM,
                function(x) str_detect(x, "TRF/TRR|TRR/TRF-CALCULATED|TRR/TRF|TRF"))
vars_post<- heart.form[as.logical(vars.post.trans.index1+vars.post.trans.index2),][,1] %>%
  as.character()

# Identify the post transplant variables in the heart form
vars_posttrans <- heart.form[which(heart.form$VARIABLE.NAME %in% vars_post),(1:2)]
# Identify the post transplant variables in the data
vars.post.trans <- intersect(colnames(heart.df.cleaned),
                     vars_post)
# Remove the post transplant variables in the data
heart.df.cleaned <- select(heart.df.cleaned,-vars.post.trans)
```

Below are two reports. The first one reports the organ distribution of patients in the dataset and the

second one is about the remaining variables (discarded means irrelevant/not interesting variables).

We already got rid of discarded variables so the corresponding frequency is zero

```
rem_type <- as.data.frame(table(heart.form[which(heart.form$VARIABLE.NAME %in% names(heart
.df.cleaned)), "INTERPRETATION_TYPE"]))
names(rem_type) <- c("Variable Type","Frequency")
rem_type$`Variable Type` <- c("Categorical","Initially Discarded","Date","Numerical")
org_type <- as.data.frame(table(heart.df$WL_ORG))
names(org_type)<-c("Organ Type","No. of Patients")
org_type$`Variable Type` <- c("UNKNOWN","Heart & Lung","Heart","Lung")
DT::datatable(org_type)
```

```
DT::datatable(rem_type)

cat("Number of patients after dropping irrelevant patients: ",nrow(heart.df.cleaned))
```

```
org_type <- read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/rmarkdown/two_stage/org_type.csv")
rem_type <- read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/rmarkdown/two_stage/rem_type.csv")
DT::datatable(org_type)
```

Show 10 entries Search:

	Organ.Type	No..of.Patients	Variable.Type
1		428	UNKNOWN
2	HL	3148	Heart & Lung
3	HR	103570	Heart
4	LU	52172	Lung

Showing 1 to 4 of 4 entries Previous 1 Next

```
DT::datatable(rem_type)
```

Show 10 entries Search:

	Variable.Type	Frequency
1	Categorical	249
2	Initially Discarded	0
3	Date	2
4	Numerical	57

Showing 1 to 4 of 4 entries Previous 1 Next

Create several variables based on literature

In this section, we create several variables based on several references. 1. Medved, Dennis, et al. “Improving prediction of heart transplantation outcome using deep learning techniques.” Scientific reports 8.1 (2018): 3613. 2. Dag, Ali, et al. “Predicting heart transplantation outcomes through data analytics.” Decision Support Systems 94 (2017): 42-52.

```
# ref1: Medved, Dennis, et al. "Improving prediction of heart transplantation outcome using deep learnin
g techniques." Scientific reports 8.1 (2018): 3613.
  # refer to a tool provided in the: http://ihtsa.cs.lth.se/ , which is product of this paper:
  # https://www.nature.com/articles/s41598-018-21417-7.pdf

# ref2: Dag, Ali, et al. "Predicting heart transplantation outcomes through data analytics." Decision Sup
port Systems 94 (2017): 42-52.

# create several variables based on references and obtain the subset of the data based on the criteria

heart.df.cleaned <- subset(heart.df.cleaned) %>%
  #ref1
  mutate(PVR = (HEMO_PA_MN_TRR- HEMO_PCW_TRR)*79.72/HEMO_CO_TRR) %>%
  #ref1
  mutate(ISCHTIME = ISCHTIME*60) %>%
  #ref1
  mutate(ECMO = ifelse(ECMO_TCR + ECMO_TRR == 0, 0, 1)) %>%
  # PVR, pulmonary vascular resistance / its calculation is based on the below mentioned links:
  # https://en.wikipedia.org/wiki/Vascular_resistance
  # http://www.scymed.com/en/smnxph/phkhr013.htm
  # https://radiopaedia.org/articles/mean-pulmonary-arterial-pressure  (calculation of Mean Pulmonary
Arterial Pressure)
  # PVR= (Mean Pulmonary Arterial Pressure (mmHg) - Pulmonary Capillary Wedge Pressure (mmHg))
* 79.72 / Cardiac Output (L/min)
  # PVR = (HEMO_PA_MN_TRR - HEMO_PCW_TRR)* 79.72 / HEMO_CO_TRR

  # ECMO / merge of (ECMO_TCR, ECMO_TRR)

  # The following variables are mutated by the authors
  mutate(BMI_CHNG = 100*(BMI_CALC- INIT_BMI_CALC)/INIT_BMI_CALC) %>%
  # mutate(WAITING_TIME = TX_DATE - INIT_DATE)  %>%  #no need, because it is already in there a
s "DAYSWAIT_CHRON"
  mutate(WGT_CHNG = 100*(WGT_KG_CALC - INIT_WGT_KG_CALC)/INIT_WGT_KG_CALC)
%>%
  mutate(HGT_CHNG = 100*(HGT_CM_CALC - INIT_HGT_CM_CALC)/INIT_HGT_CM_CALC) %
>%
  mutate(AGE_MAT = abs(AGE - AGE_DON)) %>%
  mutate(BMI_MAT = abs(BMI_CALC - BMI_DON_CALC))
```

Re-group some categorical variables based on literature

In this section, we re-group levels in some categorical variables and develop some categorical variables based on literature considering the pool of patients.

The function we use to re-group levels in a categorical variable is cat_changer() and it has four input values. The usage of this function can be found in **functions.R** over the Github of the

study. In the end of this snippet, we remove two variables (DEATH_CIRCUM_DON" and DEATH_MECH_DON) due to discrepancy.

```
# We regroup Diagnosis variables (three variables: DIAG, TCR_DGN, THORACIC_DGN) as follows

val_old <- c(1000,1001,1002,1003,1004,1005,1006,1049,1007,1200,999,1999)
val_new <- c("DILATED_MYOPATHY_IDI","DILATED_MYOPATHY_OTH","DILATED_MYOPATHY_OTH","DILATED_MYOPATHY_OTH","DILATED_MYOPATHY_OTH","DILATED_MYOPATHY_OTH","DILATED_MYOPATHY_OTH","DILATED_MYOPATHY_OTH","DILATED_MYOPATHY_ISC","CORONARY","OTHER","UNKNOWN")

heart.df.cleaned <- cat_changer(heart.df.cleaned,var="DIAG",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="TCR_DGN",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="THORACIC_DGN",val_old,val_new)


# The variable: COD_CAD_DON is re-grouped
val_old <- c(1,2,3,4,999,"Unknown")
val_new <- c("ANOXIA","CEREBROVASCULAR_STROKE","HEAD_TRAUMA","OTHER","OTHER","UNKNOWN")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="COD_CAD_DON",val_old,val_new)

# The variables regarding blood types are re-grouped.
val_old <- c("A","A1","A2","B","O","AB","A1B","A2B")
val_new <- c("A","A","A","B","O","AB","AB","AB")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="ABO",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="ABO_DON",val_old,val_new)

# The variable: DIAB is re-grouped.
val_old <- c(1,2,3,4,5,998)
val_new<-c("N","ONE","TWO","OTHER","OTHER","UNKNOWN")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="DIAB",val_old,val_new)

# previous cardiac surgery /merge of (PRIOR_CARD_SURG_TCR, PRIOR_CARD_SURG_TRR)

heart.df.cleaned$CARD_SURG <- NA
for(i in 1:nrow(heart.df.cleaned)){
  if(!is.na(heart.df.cleaned$PRIOR_CARD_SURG_TCR[i])){
    if(heart.df.cleaned$PRIOR_CARD_SURG_TCR[i]=="Y"){
      heart.df.cleaned$CARD_SURG[i] <- "Y"
    }
    if(heart.df.cleaned$PRIOR_CARD_SURG_TCR[i]=="N"){
      if(!is.na(heart.df.cleaned$PRIOR_CARD_SURG_TRR[i])){
        if(heart.df.cleaned$PRIOR_CARD_SURG_TRR[i]=="N"){
          heart.df.cleaned$CARD_SURG[i] <- "N"
          }
      }
    }
  }
```

```
 if(!is.na(heart.df.cleaned$PRIOR_CARD_SURG_TRR[i])){
   if(heart.df.cleaned$PRIOR_CARD_SURG_TRR[i]=="Y"){heart.df.cleaned$CARD_SURG[i] <- "Y"
   }
  }
}

# In the data set, several variables are related to different types of antigen alleles. Each antigen has two a
llele types, so these variables recoreded: 0: no mismatch, 1: one matched, 2: both mismatched. Instead of
using these variables, we use the variables that recored summaries of matches in these antigen alleles: H
LAMIS, AMIS, BMIS , DRMIS and remove the following variables:

heart.df.cleaned[c("DA1","DA2","RA1","RA2","DB1","DB2","RB1","RB2","RDR1","RDR2","DDR1","
DDR2")] <- NULL

# refrence for HLAMIS
# Weisdorf, Daniel, et al. "Classification of HLA-matching for retrospective analysis of unrelated donor t
ransplantation: revised definitions to predict survival." Biology of Blood and Marrow Transplantation 14
.7 (2008): 748-758.
# refrences for HLAMIS, AMIS, BMIS, DRMIS:
# Parham, Peter. The immune system. Garland Science, 2014 (PAGE 446).

val_old <- 0:6
val_new <- c("LOWEST","LOWEST","LOWEST","LOW","MEDIUM","HIGH","HIGHEST")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="HLAMIS",val_old,val_new)

# the following block is our variable manipulation based on the other literature as specified in Ali Dag's p
aper.

heart.df.cleaned$ETH_MAT <- NA
for(i in 1:nrow(heart.df.cleaned)){
 if(!is.na(heart.df.cleaned$ETHCAT[i])){
   if(!is.na(heart.df.cleaned$ETHCAT_DON[i])){
     if(heart.df.cleaned$ETHCAT_DON[i]==heart.df.cleaned$ETHCAT[i]){
       heart.df.cleaned$ETH_MAT[i] <- "Y"
     }else{
        heart.df.cleaned$ETH_MAT[i]<-"N"
        }
   }
  }
}

heart.df.cleaned$GENDER_MAT <- NA
for(i in 1:nrow(heart.df.cleaned)){
 if(!is.na(heart.df.cleaned$GENDER[i])){
   if(!is.na(heart.df.cleaned$GENDER_DON[i])){
     if(heart.df.cleaned$GENDER[i]==heart.df.cleaned$GENDER_DON[i]){
       heart.df.cleaned$GENDER_MAT[i]<-"Y"
     }else{
        heart.df.cleaned$GENDER_MAT[i]<-"N"
        }
   }
```

```
  }
}

# PROC_TY_HR, from literature
val_old <- c(1,2)
val_new <- c("BICAVAL","TRADITIONAL")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="PROC_TY_HR",val_old,val_new)

# The variable: SHARE_TY is regrouped.
# ALLOCATION TYPE-LOCAL/REGIONAL/NATIONAL - 3=LOCAL/4=REGIONAL/5=NATIONAL/6=FOREIGN
val_old <- c(3,4,5)
val_new <- c("LOCAL","REGIONAL","NATIONAL")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="SHARE_TY",val_old,val_new)

# The variable: EDUCATION is regrouped.
val_old <- c(1,2,3,4,5,6,996,998)
val_new <- c("OTHER","GRADE","HIGH","COLLEGE","UNIVERSITY","UNIVERSITY","OTHER",
"UNKNOWN")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="EDUCATION",val_old,val_new)

# The variables: ETHCAT, ethnicity of recepients are re-grouped
val_old <- c(1,2,4,5,6,7,9,998)
val_new <- c("WHITE","BLACK","HISPANIC","OTHER","OTHER","OTHER","OTHER","UNKNOW
N")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="ETHCAT",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="ETHCAT_DON",val_old,val_new)

# We dropped PRI_PAYMENT_CTRY_TRR and PRI_PAYMENT_CTRY_TRR because of too many NAs
heart.df.cleaned[c("PRI_PAYMENT_CTRY_TCR","PRI_PAYMENT_CTRY_TRR")] <- NULL


# Two variables: PRI_PAYMENT_TCR and PRI_PAYMENT_TRR are re-grouped.
val_old <- seq(1,15)
val_new<-c("PRIVATE","MEDICAID","MEDICARE_FREE","PUBLIC_OTHER","PUBLIC_OTHER"
,"PUBLIC_OTHER","PUBLIC_OTHER","OTHER","OTHER","OTHER","OTHER","OTHER","PUBLI
C_OTHER","OTHER","UNKNOWN")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="PRI_PAYMENT_TCR",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="PRI_PAYMENT_TRR",val_old,val_new)

# The variable: REGION is re-grouped.
val_old <- seq(1,11)
val_new <- c("NOTH_EAST","NOTH_EAST","SOUTH_EAST","SOUTH_EAST","WEST","WEST","
MIDWEST","MIDWEST","NOTH_EAST","MIDWEST","SOUTH_EAST")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="REGION",val_old,val_new)

# here we re-group two variables: FUNC_STAT_TRR and FUNC_STAT_TCR, based on their activity lev
el and hospitalization status
# https://www.communitycarenc.org/media/tool-resource-files/what-does-it-take-qualify-personal-care-se
rvices-d.pdf
# http://www.npcrc.org/files/news/karnofsky_performance_scale.pdf
```

```
val_old <- c(1,2,3,996,998,2010,2020,2030,2040,2050,2060,2070,2080,2090,2100)
val_new <- c("ABLE","ASSISTED","DISABLE","OTHER","UNKNOWN","DISABLE","DISABLE","
DISABLE","DISABLE","ASSISTED","ASSISTED","ASSISTED","ABLE","ABLE","ABLE")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="FUNC_STAT_TRR",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="FUNC_STAT_TCR",val_old,val_new)

#Due to discrepency, we drop these 2 variables especially the frequencies of natural cause for death are
not consistent
heart.df.cleaned[c("DEATH_CIRCUM_DON","DEATH_MECH_DON")] <- NULL
```

Re-group some categorical variables based on their definitions and distributions

In this section, we re-group levels in some categroical variables and develop some categorical variables based on their definitions from the code book provided by UNOS and their corresponding distributions. In addition, we drop categorical variables that have more than 90% of observations are NAs, variables that have more than 90% of observations are in the same level (category), and the numerical variables that have more than 30% of observations are NAs.

```
# We drop variables that are related to identifiers or dates or not related to the goal of the study.
# We drop maligancy (MALIG_TY,MALIG_TY_TCR) variables because the levels of categories are not di
stinguishable well and there are too many NAs.
heart.df.cleaned[ c("WL_ID_CODE", "WL_ORG","INIT_DATE","TX_DATE","CTR_CODE","DATA_
TRANSPLANT",
                   "DATA_WAITLIST","DISTANCE", "DON_RETYP","ECD_DONOR","END_OPO_CTR_
CODE","HOME_STATE_DON",
                   "INIT_OPO_CTR_CODE", "LISTING_CTR_CODE","LOS",
                   "MALIG_TY","MALIG_TY_TCR","OPO_CTR_CODE","ORGAN","OTH_LIFE_SUP_OS
TXT_TCR","OTH_LIFE_SUP_OSTXT_TRR",
                   "PERM_STATE","PRIOR_CARD_SURG_TYPE_OSTXT_TCR","PRIOR_CARD_SURG_
TYPE_OSTXT_TRR","PT_CODE",
                   "TRR_ID_CODE")] <- NULL

# Nine variables: CMV_DON, EBV_SEROSTATUS, HBV_CORE, HBV_CORE_DON, HBV_SUR_ANTIG
EN, HCV_SEROSTATUS, HEP_C_ANTI_DON, HTLV2_OLD_DON, HIV_SEROSTATUS are re-grouped
.
val_old <- c("C","I","N","ND","P","PD","U")
val_new <- c("UNKNOWN","UNKNOWN","NEGATIVE","UNKNOWN","POSITIVE","UNKNOWN"
,"UNKNOWN")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="CMV_DON",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="EBV_SEROSTATUS",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="HBV_CORE",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="HBV_CORE_DON",val_old,val_new)
```

```
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="HBV_SUR_ANTIGEN",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="HCV_SEROSTATUS",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="HEP_C_ANTI_DON",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="HTLV2_OLD_DON",val_old,val_new)

# In the code book, the variable HIV_SEROSTATUS didn't specify the SAS ANALYSIS FORMAT, however
, after checking it's values we believe it also uses SERSTAT.
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="HIV_SEROSTATUS",val_old,val_new)

# Here the NA equivalent characters are changed to NA
{
  NA_cells <- c("","  ","U")

  for(i in 1:length(NA_cells)){
    heart.df.cleaned[heart.df.cleaned == NA_cells[i]] <- NA
    gc()}
}

# Two variables: BRONCHO_LT_DON & BRONCHO_RT_DON are re-grouped.
val_old <- c(1,2,3,4,5,6,7,998)
val_new <- c("N","NORMAL","ABNORMAL","ABNORMAL","ABNORMAL","ABNORMAL","OTH
ER","UNKNOWN")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="BRONCHO_LT_DON",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="BRONCHO_RT_DON",val_old,val_new)


# The variable: CHEST_XRAY_DON is re-grouped.
val_old <- c(0,1,2,3,4,5,998,999)
val_new <- c("UNKNOWN","N","NORMAL","ABNORMAL_SINGLE","ABNORMAL_SINGLE","A
BNORMAL_BOTH","UNKNOWN","UNKNOWN")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="CHEST_XRAY_DON",val_old,val_new)


# The variable: CORONARY_ANGIO is re-grouped.
val_old <- c(1,2,3)
val_new <- c("N", "Y_NORMAL","Y_ABNORMAL")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="CORONARY_ANGIO",val_old,val_new)

# Two variable: HIST_DIABETES_DON and HYPERTENS_DUR_DON are re-grouped.
val_old <- c(1,2,3,4,5,998)
val_new <- c("N","Y","Y","Y","Y","UNKNOWN")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="HIST_DIABETES_DON",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="HYPERTENS_DUR_DON",val_old,val_new)

# Two variables: END_STAT, INIT_STAT are re-grouped.
# although 2099 has low frequeency we do not merge based on the literature: Huang, Edmund, et al. "Inc
idence of conversion to active waitlist status among temporarily inactive obese renal transplant candidate
s, Transplantation 98.2 (2014): 177-186."
val_old <- c(1010, 1020, 1030, 1090, 1999,2010,2020,2030,2090,2999,7010,7999)
val_new <- c("ONE","ONE","TWO","ONE","TEMPORARILY_INACTIVE","ONE","ONE","TWO","O
NE","TEMPORARILY_INACTIVE","ACTIVE","TEMPORARILY_INACTIVE")
```

```
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="END_STAT",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="INIT_STAT",val_old,val_new)

# The variable: LAST_INACT_REASON definition is re-grouped
# We got its definition from: https://www.srtr.org/requesting-srtr-data/saf-data-dictionary/
# 1: Candidate cannot be contacted
# 2: Candidate choice
# 3: Candidate work-up incomplete
# 4: Insurance issues
# 5: Medical non-compliance
# 6: Inappropriate substance use
# 7: Temporarily too sick
# 8: Temporarily too well
# 9: Weight currently inappropriate for transplant
# 10: TX'ed - removal pending UNET data correction
# 11: Inactivation due to VAD implantation and/or VAD complication
# 12: TX Pending
# 13: Physician/Surgeon unavailable
# 14: Candidate for living donor transplant only
# We regroup this variable based on if it's inactive due to health issues

val_old <- seq(1,14)
val_new <- c("OTHER", "OTHER","OTHER","OTHER","OTHER","OTHER","HEALTH","OTHER","
HEALTH","OTHER","HEALTH","OTHER","OTHER","OTHER")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="LAST_INACT_REASON",val_old,val_new)

# The variable: NUM_PREV_TX is re-grouped.
# later we regroup this variable.
val_old <- seq(0,10)
val_new <- c("ZERO", "MORE", "MORE", "MORE", "MORE", "MORE", "MORE", "MORE", "MORE
", "MORE", "MORE")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="NUM_PREV_TX",val_old,val_new)

# Two variables: PRIOR_CARD_SURG_TYPE_TCR and PRIOR_CARD_SURG_TYPE_TRR are re-grou
ped.
#search for procurement in this form: All_Forms_eg_RH.pdf
# CABG: Coronary artery bypass graft

val_old <- seq(1,31)
val_new <- c("CABG_ONLY","VALVE_ONLY", "MULTIPLE","OTHER", "MULTIPLE","MULTIPL
E", "MULTIPLE","OTHER", "MULTIPLE","MULTIPLE",
          "MULTIPLE","MULTIPLE", "MULTIPLE","MULTIPLE", "MULTIPLE","OTHER", "MULTI
PLE","MULTIPLE", "MULTIPLE","MULTIPLE",
       "MULTIPLE","MULTIPLE", "MULTIPLE","MULTIPLE", "MULTIPLE","MULTIPLE", "MULT
IPLE","MULTIPLE", "MULTIPLE","MULTIPLE", "MULTIPLE")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="PRIOR_CARD_SURG_TYPE_TCR",val_old,val
_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="PRIOR_CARD_SURG_TYPE_TRR",val_old,val
_new)

# We changed the following medicines to "HEPARIN", "ANCEF", "DOPAMINE", "ZOSYN" since these a
```

```
re the most repeted medicines.

temp <- heart.df.cleaned[c("PT_OTH1_OSTXT_DON", "PT_OTH2_OSTXT_DON","PT_OTH3_OSTX
T_DON", "PT_OTH4_OSTXT_DON")]
new_vars <- data.frame(matrix(NA, ncol=4, nrow=nrow(heart.df.cleaned)))
colnames(new_vars) <- c("HEPARIN","ANCEF","DOPAMINE","ZOSYN")

temp <- as.data.frame(apply(temp, 2, function(x) gsub("^$| ^", NA, x)), stringsAsFactors=FALSE)

for (i in 1:ncol(new_vars)){
new_vars[,i] <- apply(temp, 1, function(x) detect_terms(x, colnames(new_vars)[i]))
}

# we remove PT_OTH1_OSTXT_DON,PT_OTH2_OSTXT_DON,PT_OTH3_OSTXT_DON,PT_OTH4_O
STXT_DON and use the four variables: HEPARIN, ANCEF, DOPAMINE, ZOSYN we just create.
heart.df.cleaned[c("PT_OTH1_OSTXT_DON","PT_OTH2_OSTXT_DON","PT_OTH3_OSTXT_DON",
"PT_OTH4_OSTXT_DON")]<-NULL
heart.df.cleaned<-cbind(heart.df.cleaned,new_vars)

rm(list=c("new_vars")) #remove the unnecessary object

# The variable: STERNOTOMY_TRR is re-grouped.The definition is based on pediatric TRR
val_old <- c(1,2,3,998)
val_new <- c("ONE", "MORE","MORE", "UNKNOWN")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="STERNOTOMY_TRR",val_old,val_new)

# We drop the variable: TX_YEAR from dataset since we want to have a more versatile model
heart.df.cleaned$TX_YEAR<-NULL


# The variable: ABO_MAT is re-grouped.
val_old <- c(1,2,3)
val_new <- c("IDENTICAL", "NOT_IDENTICAL", "NOT_IDENTICAL")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="ABO_MAT",val_old,val_new)

# The variable: AMIS, BMIS, AND DRMIS are re-grouped.
val_old <- c(0,1,2)
val_new <- c("NO_MISMATCH", "ONE_MISMATCHED", "TWO_MISMATCHED")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="AMIS",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="BMIS",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="DRMIS",val_old,val_new)

# The variable: INOTROPES_TCR, INOTROPES_TRR, OTHER_INF_DON, AND PULM_INF_DON are
re-grouped.
val_old <- c(0,1)
val_new <- c("N", "Y")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="INOTROPES_TCR",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="INOTROPES_TRR",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="OTHER_INF_DON",val_old,val_new)
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="PULM_INF_DON",val_old,val_new)
```

```
# The variable: MED_COND_TRR are re-grouped.
val_old <- c(1,2,3)
val_new <- c("ICU_HOSPITALIZED", "HOSPITALIZED","NOT_HOSPITALIZED")
heart.df.cleaned <- cat_changer(heart.df.cleaned,var="MED_COND_TRR",val_old,val_new)


#=============================================================
=====
# Here we drop columns that 90% of their data is NA.
NA_Col_Rate <- col_missing_function(heart.df.cleaned)
NA_Col_Rate$varname <- rownames(NA_Col_Rate)
NA_Col_Rate <- NA_Col_Rate[which(NA_Col_Rate$na_count_col>0.9),]
NA_Col_Rate <- NA_Col_Rate$varname
heart.df.cleaned[NA_Col_Rate] <- NULL

#=============================================================
=====
# Here we drop variables that have more than 90% of the observations in one level / category.
cat_dis <- vector(mode="numeric", length=ncol(heart.df.cleaned))
for(i in 1:ncol(heart.df.cleaned)){
struct <- as.data.frame(table(heart.df.cleaned[i]))
if(nrow(struct)>0){
max_cat <- max(struct$Freq)
all_freq <- sum(struct$Freq)
if((max_cat/all_freq)>0.90) cat_dis[i] <- 1}
}

heart.df.cleaned[names(heart.df.cleaned[(cat_dis==1)])]<-NULL
#=============================================================
=====

# Here we define categorical and numerical variables
initial_num <- heart.form$VARIABLE.NAME[which(heart.form$INTERPRETATION_TYPE=="NUM
")]
mutated_num <- c("BMI_CHNG" ,"WGT_CHNG" ,"HGT_CHNG" ,"AGE_MAT","BMI_MAT","PVR",
"ECMO")
pool_num <- c(as.character(initial_num),mutated_num)

# the numerical variables in the cleaned dataset are saved in pool_num_clean
pool_num_clean <- pool_num[which(pool_num %in% names(heart.df.cleaned))]

initial_char <- heart.form$VARIABLE.NAME[which(heart.form$INTERPRETATION_TYPE=="CHA
R")]
mutated_char <- c("GENDER_MAT","ETH_MAT" ,"CARD_SURG" ,"HEPARIN" ,"ANCEF" ,"DOPA
MINE","ZOSYN")
pool_char <- c(as.character(initial_char),mutated_char)

# the categorical variables in the cleaned dataset are saved in pool_char_clean
pool_char_clean <- pool_char[which(pool_char %in% names(heart.df.cleaned))]
```

```
#=============================================================
=====
# Here we drop the numerical variables that more than 30% of the observations are NA.
# we decided not to impute numerical values, so a more conservative approach is adopted.

NA_Col_Rate <- col_missing_function(heart.df.cleaned[pool_num_clean])
NA_Col_Rate$varname <- rownames(NA_Col_Rate)
NA_Col_Rate <- NA_Col_Rate[which(NA_Col_Rate$na_count_col>0.3),]
NA_Col_Rate <- NA_Col_Rate$varname
heart.df.cleaned[NA_Col_Rate] <- NULL

# changing category of "NUM_PREV_TX" from numerical to categorical
pool_num<-pool_num[!pool_num %in% "NUM_PREV_TX"]
pool_char<-c(pool_char,"NUM_PREV_TX")


######################

# We updated numerical and categorical variables used here:
pool_num_clean <- pool_num[which(pool_num %in% names(heart.df.cleaned))]
pool_char_clean <- pool_char[which(pool_char %in% names(heart.df.cleaned))]
pool_num_clean_base<-pool_num_clean
pool_char_clean_base<-pool_char_clean
#removing the extra created data
rm(list = "temp")
```

Treating missing values for numerical and categorical variables

One-hot encoding algorithm is applied

for numerical variable we either drop numerical observations or impute by median (2 conditions), for categorical variables we could impute missing values by mode, dropping the observation, impute by UNKNOWN level, or impute as missing level (4 conditions) then in total we have 2*4 = 8 different conditions

Creating scenarios for Design of Experiments

###1st scenario: NA values of categorical variables are unknown and for numerical variables we drop the observation.

{r , message=FALSE, cache=TRUE, eval=FALSE, error=FALSE, eval=FALSE}

We dropped the numerical NAs and replaced the NA (missing values) in categorical variables to “UNKNOWN”

heart.df.cleaned.copy<-readRDS(paste0(data.path,“heart.df.cleaned.copy.rds”))
heart.df.cleaned<-heart.df.cleaned.copy #dropping observations with numerical NAs
heart.df.cleaned <- heart.df.cleaned[complete.cases(heart.df.cleaned[pool_num_clean]),]

heart.df.cleaned_char <- heart.df.cleaned[pool_char_clean]
heart.df.cleaned_char[is.na(heart.df.cleaned_char)] <- “UNKNOWN” heart.df.cleaned_num <- heart.df.cleaned[pool_num_clean]

df.un.dr<-cbind(heart.df.cleaned_char,heart.df.cleaned_num,heart.df.cleaned[“ID”])

#Make sure the type of each variable is recorded correctly

for(i in names(df.un.dr)){ if(i %in% pool_char_clean){df.un.dr[i] <- as.character(df.un.dr[,i])} if(i %in% pool_num_clean){df.un.dr[i] <- as.numeric(df.un.dr[,i])} }

dfs[[1]]<-df.un.dr

Remove unnecessary datasets

rm(list=c(“heart.df.cleaned_char”,“heart.df.cleaned_num”,“df.un.dr”))

#scenario1

```
###2nd scenario: NA values of categorical variables are missing and for numrical variables we drop the o
bservation**.
{r , message=FALSE, cache=TRUE, eval=FALSE, error=FALSE, eval=FALSE}
heart.df.cleaned.copy<-readRDS(paste0(data.path,"heart.df.cleaned.copy.rds"))
heart.df.cleaned<-heart.df.cleaned.copy
#dropping observations with numerical NAs
heart.df.cleaned <- heart.df.cleaned[complete.cases(heart.df.cleaned[pool_num_clean]),]

heart.df.cleaned_char <- heart.df.cleaned[pool_char_clean]
heart.df.cleaned_char[is.na(heart.df.cleaned_char)] <- "MISSING"
heart.df.cleaned_num <- heart.df.cleaned[pool_num_clean]

df.mi.dr<-cbind(heart.df.cleaned_char,heart.df.cleaned_num,heart.df.cleaned["ID"])


#Make sure the type of each variable is recorded corrctly
for(i in names(df.mi.dr)){
  if(i %in% pool_char_clean){df.mi.dr[i] <- as.character(df.mi.dr[,i])}
  if(i %in% pool_num_clean){df.mi.dr[i] <- as.numeric(df.mi.dr[,i])}
}

dfs[[2]]<-df.mi.dr

# remove unnecessary datasets
```

```
rm(list=c("heart.df.cleaned_char","heart.df.cleaned_num","df.mi.dr"))

#scenario2
```

###3rd scenario: NA values of categorical variables are imputed by mode and for numrical variables we drop the observation**.

```
heart.df.cleaned.copy<-readRDS(paste0(data.path,"heart.df.cleaned.copy.rds"))
heart.df.cleaned<-heart.df.cleaned.copy
#dropping observations with numerical NAs
heart.df.cleaned <- heart.df.cleaned[complete.cases(heart.df.cleaned[pool_num_clean]),]

heart.df.cleaned_char <- heart.df.cleaned[pool_char_clean]


char_mode<-as.data.frame(apply(heart.df.cleaned_char,2,cahrmode) )

for(j in pool_char_clean){
  heart.df.cleaned_char[is.na(heart.df.cleaned_char[j]),j]<-as.character(char_mode[j,1])
}

heart.df.cleaned_num <- heart.df.cleaned[pool_num_clean]

df.mo.dr<-cbind(heart.df.cleaned_char,heart.df.cleaned_num,heart.df.cleaned["ID"])


#Make sure the type of each variable is recorded corrctly
for(i in names(df.mo.dr)){
  if(i %in% pool_char_clean){df.mo.dr[i] <- as.character(df.mo.dr[,i])}
  if(i %in% pool_num_clean){df.mo.dr[i] <- as.numeric(df.mo.dr[,i])}
}

dfs[[3]]<-df.mo.dr

# remove unnecessary datasets
rm(list=c("heart.df.cleaned_char","heart.df.cleaned_num","df.mo.dr"))


#scenario3
```

###4th scenario: NA values of categorical variables are dropped and for numrical variables we drop the observation, too**. If we drop simply, this is not practical because we lost all the observations. We try to maximize remaining cells through a heuristic algorithm which through a

loop, we drop rows and columns based on the maximum emptiness until just non-empty cells are remained

```
heart.df.cleaned.copy<-readRDS(paste0(data.path,"heart.df.cleaned.copy.rds"))
heart.df.cleaned<-heart.df.cleaned.copy
#dropping observations with  NAs
# "FUNC_STAT_TRR" and "FUNC_STAT_TCR" are widely cited in the literature, then we want to keep t
hen through the
# heuristic approach of dropping rows and column
obj_data <- table_cleaner(heart.df.cleaned,1,"ID","year1",c("KEEP_ALL"))



df.dr.dr<-obj_data$data


#Make sure the type of each variable is recorded corrctly
for(i in names(df.dr.dr)){
  if(i %in% pool_char_clean){df.dr.dr[i] <- as.character(df.dr.dr[,i])}
  if(i %in% pool_num_clean){df.dr.dr[i] <- as.numeric(df.dr.dr[,i])}
}

dfs[[4]]<-df.dr.dr

# remove unnecessary datasets
rm(list=c("df.dr.dr"))

#scenario4
```

###5th scenario: NA values of categorical variables are unknown and for numrical variables are imputed by median**.

```
# The unavailable numerical are imputed by median and the NA (missing values) in categorical variables
by "UNKNOWN"
heart.df.cleaned.copy<-readRDS(paste0(data.path,"heart.df.cleaned.copy.rds"))
heart.df.cleaned<-heart.df.cleaned.copy

heart.df.cleaned_num <- heart.df.cleaned[pool_num_clean]

num_med<-as.data.frame(apply(heart.df.cleaned_num,2,function(x) median(x,na.rm = TRUE) ))


for(j in pool_num_clean){
  heart.df.cleaned_num[is.na(heart.df.cleaned_num[j]),j]<-num_med[j,1]
}
```

```
heart.df.cleaned_char <- heart.df.cleaned[pool_char_clean]
heart.df.cleaned_char[is.na(heart.df.cleaned_char)] <- "UNKNOWN"


df.un.med<-cbind(heart.df.cleaned_char,heart.df.cleaned_num,heart.df.cleaned["ID"])



#Make sure the type of each variable is recorded corrctly
for(i in names(df.un.med)){
  if(i %in% pool_char_clean){df.un.med[i] <- as.character(df.un.med[,i])}
  if(i %in% pool_num_clean){df.un.med[i] <- as.numeric(df.un.med[,i])}
}

dfs[[5]]<-df.un.med
# remove unnecessary datasets
rm(list=c("heart.df.cleaned_char","heart.df.cleaned_num","df.un.med"))

#scenario5
```

##6th scenario: NA values of categorical variables are imputed by missing and for numerical variables are imputed by median**.

```
# The unavailable numerical are imputed by median and the NA (missing values) in categorical variables to "UNKNOWN"
heart.df.cleaned.copy<-readRDS(paste0(data.path,"heart.df.cleaned.copy.rds"))
heart.df.cleaned<-heart.df.cleaned.copy

heart.df.cleaned_num <- heart.df.cleaned[pool_num_clean]

num_med<-as.data.frame(apply(heart.df.cleaned_num,2,function(x) median(x,na.rm = TRUE) ))


for(j in pool_num_clean){
  heart.df.cleaned_num[is.na(heart.df.cleaned_num[j]),j]<-num_med[j,1]
}



heart.df.cleaned_char <- heart.df.cleaned[pool_char_clean]
heart.df.cleaned_char[is.na(heart.df.cleaned_char)] <- "MISSING"


df.mi.med<-cbind(heart.df.cleaned_char,heart.df.cleaned_num,heart.df.cleaned["ID"])


#Make sure the type of each variable is recorded corrctly
for(i in names(df.mi.med)){
```

```
  if(i %in% pool_char_clean){df.mi.med[i] <- as.character(df.mi.med[,i])}
  if(i %in% pool_num_clean){df.mi.med[i] <- as.numeric(df.mi.med[,i])}
}

dfs[[6]]<-df.mi.med

# remove unnecessary datasets
rm(list=c("heart.df.cleaned_char","heart.df.cleaned_num","df.mi.med"))

#scenario6
```

###7th scenario: NA values of categorical variables are imputed by mode and for numrical variables are imputed by median**.

```
# The unavailable numerical are imputed by median and the NA (missing values) in categorical variables by "UNKNOWN"
heart.df.cleaned.copy<-readRDS(paste0(data.path,"heart.df.cleaned.copy.rds"))
heart.df.cleaned<-heart.df.cleaned.copy

heart.df.cleaned_num <- heart.df.cleaned[pool_num_clean]

num_med<-as.data.frame(apply(heart.df.cleaned_num,2,function(x) median(x,na.rm = TRUE) ))


for(j in pool_num_clean){
  heart.df.cleaned_num[is.na(heart.df.cleaned_num[j]),j]<-num_med[j,1]
}

heart.df.cleaned_char <- heart.df.cleaned[pool_char_clean]
char_mode<-as.data.frame(apply(heart.df.cleaned_char,2,cahrmode) )

for(j in pool_char_clean){
  heart.df.cleaned_char[is.na(heart.df.cleaned_char[j]),j]<-as.character(char_mode[j,1])
}


df.mo.med<-cbind(heart.df.cleaned_char,heart.df.cleaned_num,heart.df.cleaned["ID"])


#Make sure the type of each variable is recorded corrctly
for(i in names(df.mo.med)){
  if(i %in% pool_char_clean){df.mo.med[i] <- as.character(df.mo.med[,i])}
  if(i %in% pool_num_clean){df.mo.med[i] <- as.numeric(df.mo.med[,i])}
}

dfs[[7]]<-df.mo.med

# remove unnecessary datasets
rm(list=c("heart.df.cleaned_char","heart.df.cleaned_num","df.mo.med"))
```

```
#scenario7
```

###8th imputation scenario: NA values of categorical variables are dropped and for numerical variables are imputed by median**. If we drop observations that their categorical variables are NA, we lose all the observations. We try to maximize remaining cells through a heuristic algorithm which through a loop, we drop rows and columns based on the maximum emptiness until just non-empty cells are remained

```
# The unavailable numerical are imputed by median and the NA (missing values) in categorical variables by "UNKNOWN"

heart.df.cleaned.copy<-readRDS(paste0(data.path,"heart.df.cleaned.copy.rds"))
heart.df.cleaned<-heart.df.cleaned.copy


heart.df.cleaned_num <- heart.df.cleaned[pool_num_clean]

num_med<-as.data.frame(apply(heart.df.cleaned_num,2,function(x) median(x,na.rm = TRUE) ))


for(j in pool_num_clean){
  heart.df.cleaned_num[is.na(heart.df.cleaned_num[j]),j]<-num_med[j,1]
}

heart.df.cleaned_char <- heart.df.cleaned[pool_char_clean]



df.dr.med<-cbind(heart.df.cleaned_char,heart.df.cleaned_num,heart.df.cleaned["ID"])


#dropping observations with  NAs through the huristic algorithm
# "FUNC_STAT_TRR" and "FUNC_STAT_TCR" are widely cited in the literature, then we want to keep t hen through the
# heuristic approach of dropping rows and column
obj_data2 <- table_cleaner(df.dr.med,1,"ID","year1",c("KEEP_ALL"))


df.dr.med<-obj_data2$data



#Make sure the type of each variable is recorded corrctly
for(i in names(df.dr.med)){
```

```
  if(i %in% pool_char_clean){df.dr.med[i] <- as.character(df.dr.med[,i])}
  if(i %in% pool_num_clean){df.dr.med[i] <- as.numeric(df.dr.med[,i])}
}

dfs[[8]]<-df.dr.med

# remove unnecessary datasets
rm(list=c("df.dr.med"))

#scenario8
rm(list=c("heart.df.cleaned_char","heart.df.cleaned_num",
          "heart.df.cleaned.copy"))
```

Encoding the variables and identifying train and test sets

In this snippet, we encode the driven data from each scenario in two ways of hard encoding and label encoding (8scenarios x 2 encodings =16 scenarios). Also, holdout IDs for distinguishing between train and holdout sets are identified.

```
# We keep IDs
## identifying holdout IDs from cleaned data
## holdout would be separated and for rest we have 5-fold cross validation

idenx_holdout <- holdout_index(heart.df.cleaned$ID, 2019)

cat_vars <- setdiff(pool_char_clean, "year1")

## encoding (method = c("numeric", "factor"))
df_num <- lapply(dfs, function(x) encode_cat(x,cat_vars,"numeric"))
df_cat <- lapply(dfs, function(x) encode_cat(x,cat_vars,"factor"))
All_data <- c(df_num, df_cat)

## assign names for each component in the list
names(All_data) <- c(paste0("NUM_",1:8), paste0("CAT_",1:8))
saveRDS(All_data,paste0(data.path,"All_data.rds"))
## remove unused objects
rm(dfs,df_num,df_cat)

#end of part two
```

Here the distribution of the dependent variable in each timestamp is demonstrated

```
target_dist<-as.data.frame(matrix(NA, ncol = 11, nrow = 2))
names(target_dist)<-paste0("year", seq(0,10))
for(i in 1:11){
  target_dist[i]<-as.numeric(table(Responses[i]))
}
```

```
row.names(target_dist)<-c("Died","Survived")

DT::datatable(target_dist)
```

Section 2: Variable Selection

In our study, three feature selection methods are adopted: Random Forest, Fast correlation based feature selection, and LASSO. This section summarizes the results from these variable selection algorithms. In order to use the code we provide here, you will need to have some experience in parallel computation using parSapply() function in the R package: snow. The Random Forest selection algorithm takes lots of time to find important features in the data. The part is conducted on Ohio Supercomputer Center.

```
# The data is loaded from local drive
All_data<-readRDS(paste0(data.path,"All_data.rds"))

# Here, we use the function parSapply() in snow package to perform the parallel computation for three va
riable selection algorithms

# for checking the definition of the feature selection algorithms, check this paper:
# Fonti, Valeria, and Eduard Belitser. "Feature Selection using LASSO."
# for increasing ram that is dedicated to the next package you may use this command and then
# specify the dedicated ram memory: options( java.parameters = "-Xmx26g")

features <- rep(list(NA), 4)
names(features) <- c("FFS", "LASSO", "RF", "all")
impute_no <- 16  #from 1 to 16

#============================================================
#==============Fast Feature selection
#============================================================

features_FFS <- vector(mode="list", impute_no)
for (i in 1:impute_no){
  cl <- makeCluster(4, type="SOCK")
  features_FFS[[i]] <- parSapply(cl, 1:5, select_vars, All_data[[i]], "year1", "FFS", idenx_holdout, 2090)
  stopCluster(cl)
}

#============================================================
#==============Lasso Feature selection for Binomial TARGETS
#============================================================
```

```
features_LASSO <- vector(mode="list", impute_no)
for (i in 1:impute_no){
  cl <- makeCluster(4, type="SOCK")
  features_LASSO[[i]] <- parSapply(cl, 1:5, select_vars, All_data[[i]], "year1", "LASSO", idenx_holdout
, 2090)
  stopCluster(cl)
}
save.image("C:/Users/hza0020/OneDrive - Auburn University/Transplant/BUAL-LAB/DoE/save_files/n
ewsave4.RData")
#save4
#==============================================
#==============Random Forest Feature Selection
#==============================================

features_RF <- vector(mode="list", impute_no)
for (i in 1:impute_no){
  cl <- makeCluster(4, type="SOCK")
  features_RF[[i]] <- parSapply(cl, 1:5, select_vars, All_data[[i]], "year1", "RF", idenx_holdout, 2090)
  stopCluster(cl)
}


features<-list()
features$RF<-features_RF
features$LASSO<-features_LASSO
features$FFS<-features_FFS
saveRDS(features,paste0(data.path,"features.rds"))
```

Section 3: screening design

The following block of codes is performed over Ohio Super Computer. In this section of the study we define our screening design in order to understand the contributing factors that may affect performance of the predictive algorithms.

```
All_data<-readRDS(paste0(data.path,"All_data.rds"))
features<-readRDS(paste0(data.path,"features.rds"))
# here you can find how we setup data preparation and feature selection steps in the earlier section
# for each style of factorial design we could find the ID and then use associated data and
# features that we used
data_design<-as.data.frame(matrix(0, ncol = 3, nrow = 16))
names(data_design)<-c("numerical_imputation","categorical_imputation","encoding")
data_design$numerical_imputation<-rep(rep(c("DROP", "MEDIAN"),each=4),2)
data_design$categorical_imputation<-rep(c("UNKNOWN", "MISSING", "MODE", "DROP"),4)
data_design$encoding<-rep(c("LABEL", "HARD"),each=8)
data_design$ID<-rownames(data_design)
```

```
feature_selection<-c("FFS", "LASSO","RF")
data_design_features<-as.data.frame(matrix(0, ncol = 1, nrow = nrow(data_design)*length(feature_selection)))
names(data_design_features)<-c("features")
data_design_features$features<-rep(c("FFS", "LASSO","RF"),each=nrow(data_design))
data_design_features<-cbind(data_design_features, data_design[rep(seq_len(nrow(data_design)), length(feature_selection)),])

resampling<-c("none", "down", "up", "rose", "smote")
DDFB<-as.data.frame(matrix(0, ncol = 1, nrow = nrow(data_design_features)*length(resampling)))
names(DDFB)<-c("resampling")
DDFB$resampling<-rep(resampling,each=nrow(data_design_features))
DDFB<-cbind(DDFB, data_design_features[rep(seq_len(nrow(data_design_features)), length(resampling)),])

algorithm<-c("glm" , "lda","glmnet", "nnet" , "naive_bayes" , "kernelpls" , "xgbDART",  "rpart" , "ranger")
DDFBA<-as.data.frame(matrix(0, ncol = 1, nrow = nrow(DDFB)*length(algorithm)))
names(DDFBA)<-c("algorithm")
DDFBA$algorithm<-rep(algorithm,each=nrow(DDFB))
DDFBA<-cbind(DDFBA, DDFB[rep(seq_len(nrow(DDFB)), length(algorithm)),])

folds<-5
DDFBAF<-as.data.frame(matrix(0, ncol = 1, nrow = nrow(DDFBA)*folds))
names(DDFBAF)<-c("fold")
DDFBAF$fold<-rep(c(1:folds),each=nrow(DDFBA))
DDFBAF<-cbind(DDFBAF, DDFBA[rep(seq_len(nrow(DDFBA)), folds),])



# here is the screening design formation
screen_design<-DDFBAF[which( (DDFBAF$algorithm %in% c("glm", "ranger"))
                & (DDFBAF$resampling %in% c("none", "smote"))
                & (DDFBAF$encoding %in% c("LABEL", "HARD"))
                & (DDFBAF$numerical_imputation %in% c("DROP", "MEDIAN"))
                & (DDFBAF$features %in% c("FFS", "LASSO"))
                & (DDFBAF$categorical_imputation %in% c("MODE", "DROP"))  ) ,]

# since models have different learning times, we do parallel processing for each model seperately
screen_design_glm<-screen_design[which( (screen_design$algorithm %in% c("glm")) ),]
screen_design_ranger<-screen_design[which( (screen_design$algorithm %in% c("ranger")) ),]

# now two parallel processing for screening design is performed here
# identifying number of created clusters for parallel processing
if(as.character(as.data.frame(t(Sys.info()))$sysname)=="Linux"){

 library(Rmpi)
 library(snow)

 # assign cores used in the parallel computing
 workers <- as.numeric(Sys.getenv(c("PBS_NP"))) - 1
```

```
 cl <- makeCluster(workers,"MPI")


}else{if(as.character(as.data.frame(t(Sys.info()))$sysname)=="Windows"){
 cl <- makeCluster(parallel::detectCores(), type="SOCK")
 }}

time.begin <- proc.time()[3]

screeing_glm <- parSapply(cl, row.names(screen_design_glm), train_para,design_table0=screen_design
_glm                                , All_data0=All_data,TARGET0="year1"
                    , Index_test0=idenx_holdout, features0=features,seed0=2019)
time.window <- proc.time()[3] - time.begin



time.begin <- proc.time()[3]

screeing_ranger <- parSapply(cl, row.names(screen_design_ranger), train_para,design_table0=screen_d
esign_ranger                                , All_data0=All_data,TARGET0="year1"
                    , Index_test0=idenx_holdout, features0=features,seed0=2019)
time.window <- proc.time()[3] - time.begin


stopCluster(cl)


if(as.character(as.data.frame(t(Sys.info()))$sysname)=="Linux"){mpi.quit()}

saveRDS(screeing_glm,paste0(data.path,"screeing_glm.rds"))
saveRDS(screeing_ranger,paste0(data.path,"screeing_ranger.rds"))

##############################################################################################
######
```

The following block of codes summarizes the performance all of the predictions related to DoE into a table.

```
modeling_result_rf<-readRDS(paste0(data.path,"_screen_ranger_gmean.rds"))


performance_rf<-as.data.frame(matrix(NA, ncol=13, nrow=dim(modeling_result_rf)[2]))
names(performance_rf)<-c("AUC","Sensitivity","Specificity","Accuracy","data_scenario" ,"algorithm" ,
                 "resampling", "feature_selection", "imputation_num", "imputation_cat", "encoding", "fold",
"TARGET")

for(i in 1:dim(modeling_result_rf)[2]){
 performance_rf[i,1:4]<-modeling_result_rf[[2,i]][,]
 performance_rf[i,5:13]<-modeling_result_rf[[1,i]][,]
```

```
}

write.csv(performance_rf,paste0(data.path,"_screen_rf_results_gmean.csv"), row.names=FALSE)


modeling_result_log<-readRDS(paste0(data.path,"_screen_glm_gmean.rds"))


performance_log<-as.data.frame(matrix(NA, ncol=13, nrow=dim(modeling_result_log)[2]))
names(performance_log)<-c("AUC","Sensitivity","Specificity","Accuracy","data_scenario" ,"algorithm"
,
                  "resampling", "feature_selection", "imputation_num", "imputation_cat", "encoding", "fold",
"TARGET")

for(i in 1:dim(modeling_result_log)[2]){
  performance_log[i,1:4]<-modeling_result_log[[2,i]][,]
  performance_log[i,5:13]<-modeling_result_log[[1,i]][,]
}

write.csv(performance_log,paste0(data.path,"_screen_glm_results_gmean.csv"), row.names=FALSE)
```

In this section, screening DoE is performed. The models are executed over Ohio Super Computer.

Then the results are imported from Github of the project. Here you see ANOVA of DoE ## Section:

Loading Data, installing the required libraries.

```
# gc()
if(!"pacman" %in% rownames(installed.packages())){
  install.packages(pkgs = "pacman",repos = "http://cran.us.r-project.org")
}
library(pacman)
p_load(DoE.base,FrF2,DoE.wrapper,RcmdrPlugin.Export,
       RcmdrPlugin.FactoMineR,RcmdrPlugin.HH
       ,RcmdrPlugin.TeachingDemos,RcmdrPlugin.orloca,Rcmdr,conf.design
       ,lhs,AlgDesign,DiceDesign,rsm,RcmdrPlugin.DoE,pivottabler,data.table,DT , htmltools, nortest)

# The pivottabler and data.table libraries are for creating PIVOT plot at the end of this file
# The DT and htmltools libraries are for visualizing tables

# this library helps to produce codes required in DoE
# library(RcmdrPlugin.DoE)

# install.packages("RcmdrPlugin.Export") # Graphically export objects to LaTeX or HTML
# install.packages("RcmdrPlugin.FactoMineR") # Graphical User Interface for FactoMineR
# install.packages("RcmdrPlugin.HH") # Rcmdr support for the HH package
# install.packages("RcmdrPlugin.IPSUR") # Introduction to Probability and Statistics Using R
# install.packages("RcmdrPlugin.SurvivalT") # Rcmdr Survival Plug-In
# install.packages("RcmdrPlugin.TeachingDemos") # Rcmdr Teaching Demos Plug-In
# install.packages("RcmdrPlugin.epack") # Rcmdr plugin for time series
```

```
# install.packages("RcmdrPlugin.orloca") # orloca Rcmdr Plug-in
# install.packages("Rcmdr") # at the R prompt
# nortest is package for test of normality

# load the experiments results
experiments_glm<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/_screen_glm_results_gmean.csv")
experiments_ranger<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/_screen_ranger_results_gmean.csv")
```

Creating design, assigning IDs for each observation, and joining the experiments to the created design

```
# merging results of the experiments
experiments_screen<-rbind(experiments_glm,experiments_ranger)

# creating geometric mean as a performance measure
experiments_screen$GMEAN<-sqrt(experiments_screen$Sensitivity*experiments_screen$Specificity)

experiments_screen$ex.ID0<-paste0(experiments_screen[["algorithm"]],experiments_screen[["resampling"]],
                        experiments_screen[["feature_selection"]],experiments_screen[["imputation_num"]],
                        experiments_screen[["imputation_cat"]],experiments_screen[["encoding"]])
```

Running design of experiment for geometric mean as the response variable

I limit interactions to 2 level, and first I include all , then step by step drop the insignificant factors/interactions

Running DoE for all second level interactions for geometric mean

First I perform DoE for GMEAN as the response variable

```
results_GMEAN1<-summary(lm.default(formula = GMEAN ~ (algorithm + resampling + feature_selection +
                        imputation_num + imputation_cat + encoding+
                          algorithm*resampling+
                          algorithm*feature_selection+
                          algorithm*imputation_num+
                          algorithm*imputation_cat+
                          algorithm*encoding+
                          resampling*feature_selection+
                          resampling*imputation_num+
                          resampling*imputation_cat+
                          resampling*encoding+
                          feature_selection*imputation_num+
                          feature_selection*imputation_cat+
```

```
                        feature_selection*encoding+
                        imputation_num*imputation_cat+
                        imputation_num*encoding+
                        imputation_cat*encoding), data = experiments_screen))

round(results_GMEAN1$coefficients,3)

cat(paste("R-Squared is: ",results_GMEAN1$adj.r.squared))

## R-Squared is:  0.975077920929835
```

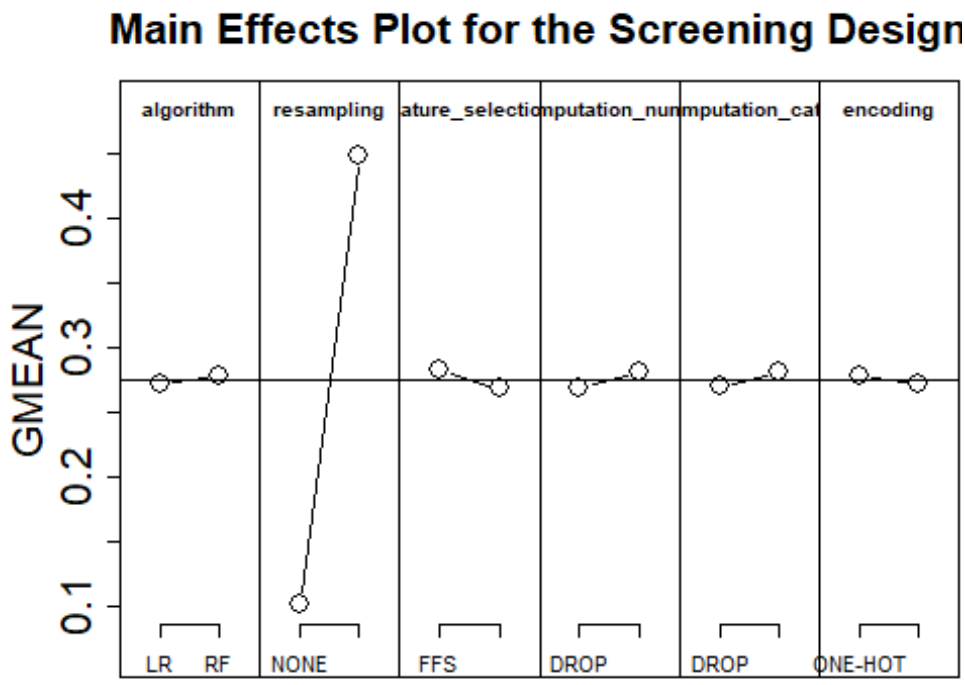
Exclude the interactions terms that are not significant for geometric mean

It returned that all factors are significantly affect the performance then I used them for a full factorial design

```
results_GMEAN2<-summary(lm.default(formula = GMEAN ~ (algorithm + resampling + feature_select
ion +
                       imputation_num + imputation_cat + encoding+
                        algorithm*resampling+
                        algorithm*feature_selection+
                        algorithm*imputation_num+
                        # algorithm*imputation_cat+
                        # algorithm*encoding+
                        resampling*feature_selection+
                        resampling*imputation_num+
                        # resampling*imputation_cat+
                        resampling*encoding+
                        feature_selection*imputation_num+
                        feature_selection*imputation_cat+
                        feature_selection*encoding+
                        imputation_num*imputation_cat
                        # +imputation_num*encoding
                        +imputation_cat*encoding
                        ), data = experiments_screen))

round(results_GMEAN2$coefficients,3)

cat(paste("R-Squared is: ",results_GMEAN2$r.squared))

## R-Squared is:  0.976343011856975

cat(paste("Adjusted R-Squared is: ",results_GMEAN2$adj.r.squared))

## Adjusted R-Squared is:  0.975011327093957
```

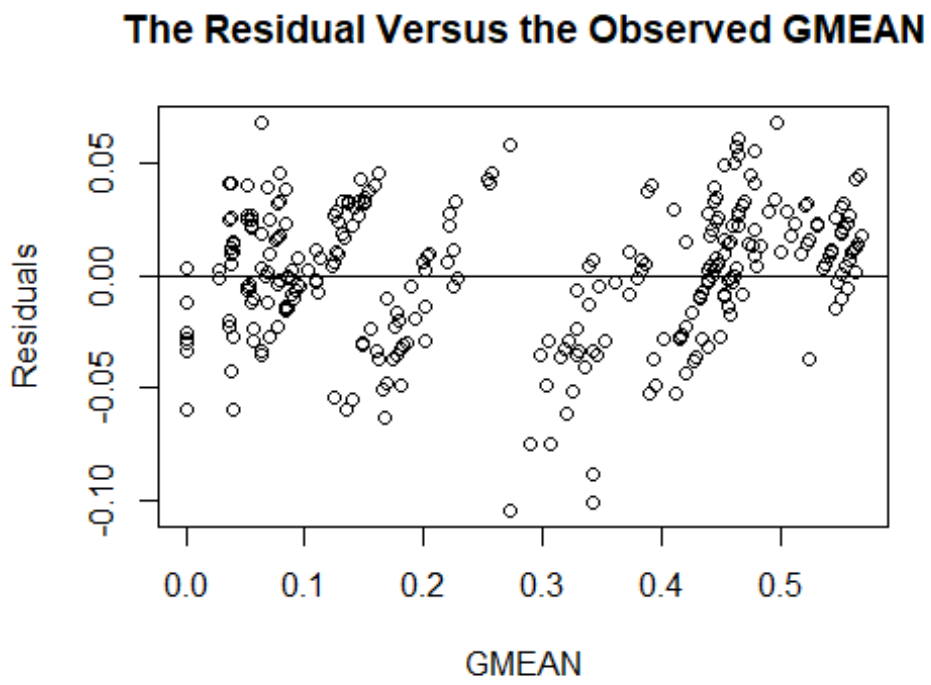
Plot association between the main effects that are significant and geometric mean

```
# renaming levels' names for visualization
levels(experiments_screen$algorithm)[levels(experiments_screen$algorithm)=="glm"] <- "LR"
levels(experiments_screen$algorithm)[levels(experiments_screen$algorithm)=="ranger"] <- "RF"
levels(experiments_screen$encoding)[levels(experiments_screen$encoding)=="HARD"] <- "ONE-HOT"
levels(experiments_screen$resampling)[levels(experiments_screen$resampling)=="none"] <- "NONE"
levels(experiments_screen$resampling)[levels(experiments_screen$resampling)=="smote"] <- "SMOTE"

GMEAN_screen_model<-lm.default(formula = GMEAN ~ (algorithm + resampling + feature_selection
+
                         imputation_num + imputation_cat + encoding+
                           algorithm*resampling+
                           algorithm*feature_selection+
                           algorithm*imputation_num+
                           # algorithm*imputation_cat+
                           # algorithm*encoding+
                           resampling*feature_selection+
                           resampling*imputation_num+
                           # resampling*imputation_cat+
                           resampling*encoding+
                           feature_selection*imputation_num+
                           feature_selection*imputation_cat+
                           feature_selection*encoding+
                           imputation_num*imputation_cat
                           # +imputation_num*encoding
                           +imputation_cat*encoding
                           ), data = experiments_screen)

MEPlot(GMEAN_screen_model,select = c(1,2,3,4,5,6),response="GMEAN",cex.xax=1,cex.yax=2,abbre
v=7,cex.main=.98,
     cex.title=2,pch=1,
     xlab="Levels of the Factors",ylab="GMEAN", main = "Main Effects Plot for the Screening Design")
```

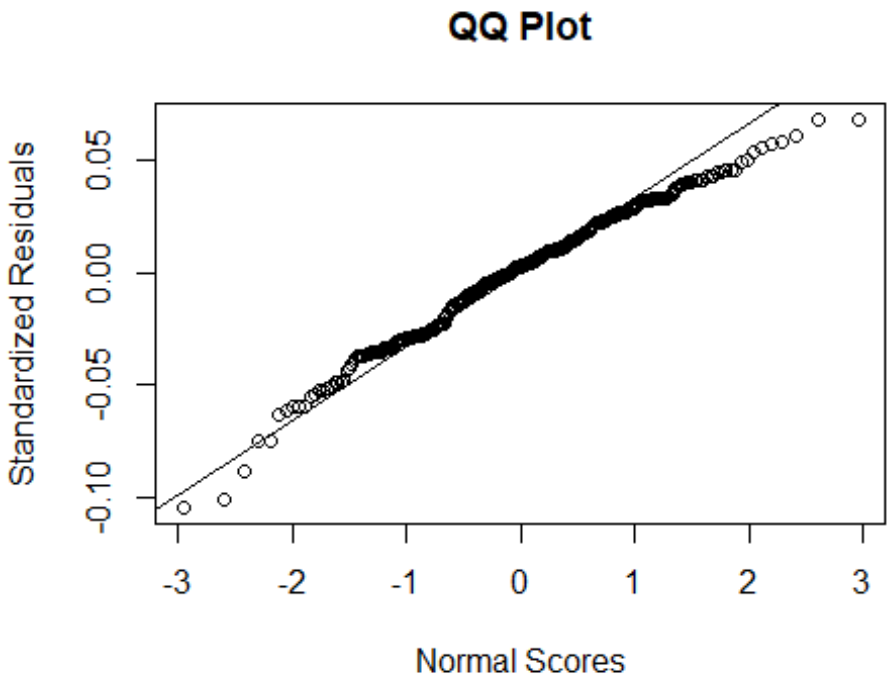


The Residual Versus the Observed Values Plot is Demonstrated Below

```
GMEAN_screen_model.res = resid(GMEAN_screen_model)

plot(experiments_screen$GMEAN, GMEAN_screen_model.res,
    ylab="Residuals", xlab="GMEAN",
    main="The Residual Versus the Observed GMEAN")
abline(0, 0)                # the horizon
```




```
qqnorm(GMEAN_screen_model.res,
      ylab="Standardized Residuals",
      xlab="Normal Scores",
      main="QQ Plot"
      ,cex=1
      )
qqline(GMEAN_screen_model.res, probs = c(0.25, 0.75))
```



Section 4: Full Factorial Design Based on Significant Variables

In all the earlier section, all the variables turned out to be significant. Then a full factorial design. The following block of codes are executed over Ohio Super Computer. Each scenarios are saved separately (e.g. “glm-1.1.1.5.1”) or all the results are merged (e.g. "_full_glm.rds").

```
if(!"pacman" %in% rownames(installed.packages())){
  install.packages(pkgs = "pacman",repos = "http://cran.us.r-project.org")
}
# p_load is equivalent to combining both install.packages() and library()
pacman::p_load(caret,AUC,MASS,ROSE,DMwR,snow,ranger,parallel,xgboost,gbm,naivebayes,e1071,k
ernlab,Rmpi)

# for parallel processing over Ohio Super Computer the Function should be called so it's redefigned here
# because it might be better to call it here instead of connecting to Github for each processes in parallel
mode.
############main function#################################################################
train_para<-function(iii, design_table0 ,All_data0,TARGET0="year1",
                     Index_test0=idenx_holdout, features0,seed0=2019){

  gc()
  if(!"pacman" %in% rownames(installed.packages())){
```

```
  install.packages(pkgs = "pacman",repos = "http://cran.us.r-project.org")
 }
 # p_load is equivalent to combining both install.packages() and library()
 pacman::p_load(caret,AUC,MASS,ROSE,DMwR,snow,ranger,parallel,xgboost,gbm,naivebayes,e1071,
kernlab,parallel)

 set.seed(seed0)
 fold_no<-design_table0[iii,"fold"]
 alg_fact<-design_table0[iii,"algorithm"]
 resample_fact<-design_table0[iii,"resampling"]
 feature_fact<-design_table0[iii,"features"]
 impute_num_fact<-design_table0[iii,"numerical_imputation"]
 impute_cat_fact<-design_table0[iii,"categorical_imputation"]
 encoding_fact<-design_table0[iii,"encoding"]
 data_fact<-as.numeric(design_table0[iii,"ID"])

 # putting summary of the experiment in here
 experiment.summary<-as.data.frame(matrix(0, ncol = 9, nrow = 1))
 names(experiment.summary)<-c("data_scenario","algorithm","resampling","feature_selection",
                       "imputation_num","imputation_cat","encoding","fold","TARGET")

 experiment.summary[1,]<-c(data_fact,alg_fact,resample_fact,feature_fact,
                     impute_num_fact,impute_cat_fact,encoding_fact,fold_no,TARGET0)


 df <- All_data0[[data_fact]]
 index_t<- which(df$ID%in%Index_test0[[fold_no]])
 vars<-features0[feature_fact][[1]][[data_fact]][[fold_no]]
 df <- df[,which(names(df)%in% c(vars,as.character(TARGET0[1])) )]
 hold_out <- df[index_t,]
 traindata <- df[-index_t,]
 traindata$ID <- NULL

 set.seed(seed0+fold_no)

 # 1: survival; 0: death
 traindata[,as.character(TARGET0)] <- as.factor(ifelse(traindata[,as.character(TARGET0)]=="0", "De
ath", "Survival"))
 hold_out[,as.character(TARGET0)] <- as.factor(ifelse(hold_out[,as.character(TARGET0)]=="0", "D
eath", "Survival"))


 formul<-as.formula(paste0(as.character(TARGET0[1]),"~."))

 run.code<-"YES"
 sampling.method<-resample_fact
 if(resample_fact=="none"){sampling.method<-NULL}
 # Reference for geometric mean
 # Kim, Myoung-Jong, Dae-Ki Kang, and Hong Bae Kim. "Geometric mean based boosting
 # algorithm with over-sampling to resolve data imbalance problem for bankruptcy prediction."
 # Expert Systems with Applications 42.3 (2015): 1074-1082.
```

```
 gmeanfunction <- function(data, lev = NULL, model = NULL) {
   sub_sens<-caret::sensitivity(data$pred,data$obs)
   sub_spec<-caret::specificity(data$pred,data$obs)
   c(gmean = sqrt(sub_sens*sub_spec))
 }



 # I used 5 fold cross validation
 control_setting <- caret::trainControl(method = "cv", number=5, sampling=sampling.method , verbose
Iter= TRUE,
                          #summaryFunction = twoClassSummary,
                          search="random", classProbs = TRUE, selectionFunction="best"
                          ,summaryFunction = gmeanfunction)
 #models:
 # KPLS: kernelpls
 # CART: rpart
 # XGB: xgbDART
 # random forest (RF): ranger
 # logistic regression (LR): glm


 if (alg_fact%in% c("glm", "nnet", "svmRadial","kernelpls")){
   if((class(try( result_model <- train(formul, data=traindata, method=alg_fact, family="binomial",
                  trControl = control_setting, metric="gmean"),silent = TRUE))=="try-error")[1]){run.code
<-"NO"}
 }else if (alg_fact%in%c("rf", "gbm", "earth", "rpart", "xgbTree", "naive_bayes","xgbDART" ,"ranger",
"glmnet")){
   #if(alg_fact=="earth"){alg_fact<-"glmnet"}
   if((class(try( result_model <- train(formul,  data=traindata, method=alg_fact,
                  trControl = control_setting, tuneLength=10, metric="gmean"),silent = TRUE))=="try-err
or")[1]){run.code<-"NO"}
 }else if(alg_fact=="lda"){
   if((class(try(  result_model <- train(formul, data=traindata, method=alg_fact, preProcess="pca", prePr
ocOptions = list(method="BoxCox"),
                  trControl = control_setting, metric="gmean"),silent = TRUE))=="try-error")[1]){run.code
<-"NO"}
 }else if(alg_fact=="treebag"){
   if((class(try(  result_model <- train(formul, data=traindata, method=alg_fact, family="binomial",
                  trControl = control_setting, tuneLength=10, metric="gmean"),silent = TRUE))=="try-err
or")[1]){run.code<-"NO"}
 }

 if(run.code=="YES"){
 resul_raw <- as.data.frame(matrix(NA, ncol = 3, nrow = nrow(hold_out)))
 colnames(resul_raw) <- c("TARGET", alg_fact, "Probability")
 resul_raw$TARGET <- hold_out[as.character(TARGET0)]

 train_raw <- as.data.frame(matrix(NA, ncol = 3, nrow = nrow(traindata)))
 colnames(train_raw) <- c("TARGET", alg_fact, "Probability")
```

```
  train_raw$TARGET <- traindata[as.character(TARGET0)]

  resul_pred_perf<-as.data.frame(matrix(NA, ncol = 1, nrow = 4))
  colnames(resul_pred_perf)<-c(alg_fact)
  rownames(resul_pred_perf)<-c("auc","sen","spec","accu")

  #if(class(try(varImp(result_model),silent = TRUE))!="try-error"){
  train_raw$Probability <- predict(result_model, newdata=traindata, type="prob")[,2]
  train_raw[alg_fact] <- predict(result_model, newdata=traindata, type="raw")
  #train_auc <- AUC::auc(roc(train_raw$Probability, traindata[,TARGET]))
  resul_raw[alg_fact] <- predict(result_model, newdata=hold_out, type="raw")
  resul_raw$Probability <- predict(result_model, newdata=hold_out, type="prob")[,2]
  resul_pred_perf[1,1] <- AUC::auc(roc(resul_raw$Probability,hold_out[,as.character(TARGET0)]))
  resul_pred_perf[2,1] <- caret::sensitivity(resul_raw[,alg_fact],hold_out[,as.character(TARGET0)])
  resul_pred_perf[3,1] <- caret::specificity(resul_raw[,alg_fact],hold_out[,as.character(TARGET0)])
  resul_pred_perf[4,1] <- (as.data.frame(confusionMatrix(resul_raw[,alg_fact], hold_out[,as.character(
TARGET0[1])])$overall))[1,]
  #}
  gmean_overal<-as.data.frame(result_model$results)
  gmean_folds<-as.data.frame(result_model$resample)
  equat<-as.data.frame(result_model$finalModel$coefficients)
  names(equat)<-"coef"
  equat$vars<-rownames(equat)
  equat$vars[which(equat$vars==c("(Intercept)"))]<-c("INTERCEPT")

  dir.create(paste0(getwd(),"/data_pile"))
  rm(list=setdiff(ls(), c("experiment.summary","resul_pred_perf","resul_raw","alg_fact","iii","gmean_ov
eral","gmean_folds","equat")))
  save.image(paste0(getwd(),"/data_pile/",alg_fact,"-",iii,".RData"))
  saveRDS(list(experiment.summary=experiment.summary, Performance=resul_pred_perf, Predicted=res
ul_raw,gmean_overal=gmean_overal,
              gmean_folds=gmean_folds,equat=equat),
          paste0(getwd(),"/data_pile/",alg_fact,"-",iii,".rds"))

  return(list(experiment.summary=experiment.summary, Performance=resul_pred_perf, Predicted=resul_
raw))
  }else{
    dir.create(paste0(getwd(),"/data_pile"))
    rm(list=setdiff(ls(), c("experiment.summary","alg_fact","iii")))
    save.image(paste0(getwd(),"/data_pile/NOT-",alg_fact,"-",iii,".RData"))
    saveRDS(list(experiment.summary=experiment.summary, Performance="NOT", Predicted="NOT",g
mean_overal="NOT",
              gmean_folds="NOT",equat="NOT"),
            paste0(getwd(),"/data_pile/NOT-",alg_fact,"-",iii,".rds"))
    return(list(experiment.summary=experiment.summary, Performance="NOT", Predicted="NOT",gmea
n_overal="NOT",
              gmean_folds="NOT",equat="NOT"))
  }
}

##############################################################################
```

```
# design_xxx.csv has information of the full factorial design
# instead of glm in below, for each algorithm the algorithm name should be subsituted
# an example is: "design_lda.csv". The other algorithms' names are:
# "lda","glmnet", "nnet" , "naive_bayes" , "kernelpls" , "xgbDART",  "rpart" , "ranger"
# server_base.rds contains the following data:
# DDFBAF: it has data of the full factorial design which has 10800 rows
# idenx_holdout: index of the holdout set
# features: features that attained after feature selection

# if Operating System is Windows means running over PC

if(Sys.info()[1]=="Windows"){
  server_base<-readRDS(paste0(data.path,"server_base.rds"))
  design_import<-read.csv(paste0(data.path,"design_glm.csv"))
}else{
  server_base<-readRDS("/users/PMIU0138/miu0150/Transplant/data/server_base.rds")
  design_import<-read.csv("/users/PMIU0138/miu0150/Transplant/data/design_glm.csv")
}

design<-list()
design$resampling<-as.character(design_import$resampling[!is.na(design_import$resampling)])
design$features<-as.character(design_import$features[!is.na(design_import$features)])
design$categorical_imputation<-as.character(design_import$categorical_imputation[!is.na(design_impo
rt$categorical_imputation)])
design$algorithm<-as.character(design_import$algorithm[!is.na(design_import$algorithm)])
design$Dname<-as.character(design_import$Dname[!is.na(design_import$Dname)])

DDFBAF<-server_base$DDFBAF
All_data<-server_base$All_data
idenx_holdout<-server_base$idenx_holdout
features<-server_base$features

# here is the screening design formation
screen_design<-DDFBAF[which( (DDFBAF$algorithm %in% design$algorithm) & (DDFBAF$resamp
ling %in% design$resampling)
                      & (DDFBAF$features %in% design$features)
                      & (DDFBAF$categorical_imputation %in% design$categorical_imputation) ) ,]


#identifying number of created clusters for parallel processing
if(Sys.info()[1]=="Windows"){
  library(snow)
  cl <- parallel::makeCluster(4)
}else{
  library(Rmpi)
  library(snow)
  # assign cores used in the parallel computing
  workers <- as.numeric(Sys.getenv(c("PBS_NP"))) - 1
  s.no<-nrow(screen_design)
  workers.no<-min(workers,s.no)
  cl <- makeCluster(workers.no,"MPI")
```

```
}


time.begin <- proc.time()[3]

results <- parSapply(cl, row.names(screen_design), train_para,design_table0=screen_design,
                     All_data0=All_data,TARGET0="year1",
                     Index_test0=idenx_holdout, features0=features,seed0=2019)

time.window <- proc.time()[3] - time.begin

stopCluster(cl)

rm(list=setdiff(ls(), c("results","design")))
# save.image(paste0("saver_glm.RData"))
saveRDS(results,paste0("/users/PMIU0138/miu0150/Transplant/data/_",design$Dname,".rds"))

mpi.quit()
```

Consolidating results of Full Factorial Designs

After running the designs over Ohio Super Computer the following block of codes merge performances of all the scenarios into a file.

```
server_base<-readRDS(paste0(data.path,"server_base.rds"))
DDFBAF<-server_base$DDFBAF

DDFBAF$auc<-NA
DDFBAF$sen<-NA
DDFBAF$spec <-NA
DDFBAF$accu<-NA
DDFBAF$data_scenario<-DDFBAF$ID
DDFBAF$ID<-row.names(DDFBAF)
DDFBAF$imputation_cat<-DDFBAF$categorical_imputation
DDFBAF$imputation_num<-DDFBAF$numerical_imputation
DDFBAF$TARGET<-"year1"
DDFBAF$feature_selection<-DDFBAF$features
DDFBAF[c("categorical_imputation","numerical_imputation","features")]<-NULL

# There is two ways to provide a report that has all the results

# First way: Here is the way if you alread have a large "*.rds" file that has results of all the scenarios tog
ether

# instead of naive_bayes in below, for each algorithm the algorithm name should be subsituted
# an example is: "glm.rds". The pther algorithms' names are:
# "lda","glmnet", "nnet" , "glm" , "kernelpls" , "xgbDART",  "rpart" , "ranger"
```

```
naive_bayes<-readRDS(paste0(data.path,"_full_naive_bayes.rds"))

naive_bayes.summary<-as.data.frame(matrix(0, ncol = 14, nrow = ncol(naive_bayes)))
names(naive_bayes.summary)<-c("data_scenario","algorithm","resampling","feature_selection",
                              "imputation_num","imputation_cat","encoding","fold","TARGET",
                              "auc","sen","spec","accu","ID")

for(i in 1:ncol(naive_bayes)){
  naive_bayes.summary[i,1:9]<-as.character(naive_bayes[[1,i]])
  naive_bayes.summary[i,10:13]<-as.numeric(t(naive_bayes[[2,i]]))
  naive_bayes.summary[i,14]<-colnames(naive_bayes)[i]
}

naive_bayes.summary<-naive_bayes.summary[complete.cases(naive_bayes.summary),]

naive_bayes.summary$gmean<-sqrt(naive_bayes.summary$sen*naive_bayes.summary$spec)
# rm(list=setdiff(ls(), c("nnet.summary","lda.summary","glm.summary","naive_bayes.summary","DDFB
AF","server_base")))

write.csv(naive_bayes.summary,"naive_bayes.summary.csv",row.names = FALSE)

# Second way: Here is the way if results of each scenario is reported seperately
# instead of rpart in below, for each algorithm the algorithm name should be subsituted
# an example is: "glm.rds". The pther algorithms' names are:
# "lda","glmnet", "nnet" , "glm" , "kernelpls" , "xgbDART",  "naive_bayes" , "ranger"

for(i in row.names(DDFBAF)){
  if((class(try(
    data.temp<-readRDS(paste0(data.path,"/all_scenarios/",DDFBAF[i,"algorithm"],"-",i,".rds")),silent =
TRUE))!="try-error")){
    data.temp<-as.data.frame(t(data.temp$Performance))
    DDFBAF[i,"auc"]<-data.temp$auc
    DDFBAF[i,"sen"]<-data.temp$sen
    DDFBAF[i,"spec"] <-data.temp$spec
    DDFBAF[i,"accu"]<-data.temp$accu
  }
}

naive_bayes<-DDFBAF[which(DDFBAF$algorithm=="naive_bayes" & !is.na(DDFBAF$auc)),]
naive_bayes$gmean<-sqrt(naive_bayes$sen*naive_bayes$spec)

write.csv(naive_bayes.summary,"naive_bayes.summary.csv",row.names = FALSE)
```

Here the Best Algorithms Based on their performance is selected for the case that we have repeated class validation

The dataset is fairly large and random selection is performed. The results of the best model in both repeated CV and CV was same. In all the scenario, the training models find the highest GMEAN. However, other outcomes of the scenarios are investigated and among them scenarios that highest "accuracy", "sensitivity", "specificity", and "AUC" are reported

```
# Here summary of all the scenarios are provided
glm_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/glm.summary.csv")
nnet_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/nnet.summary.csv")
ranger_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/ranger.summary.csv")
xgbDART_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/xgbDART.summary.csv")
naive_bayes_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/naive_bayes.summary.csv")
kernelpls_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/kernelpls.summary.csv")
glmnet_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/glmnet.summary.csv")
rpart_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/rpart.summary.csv")
lda_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/lda.summary.csv")

# showing performance of screening algorithms
screen_glm_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/_screen_glm_results_gmean.csv")
screen_ranger_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/_screen_ranger_results_gmean.csv")
DF.names<-c("auc" , "sen" , "spec", "accu" ,  "data_scenario","algorithm","resampling" , "feature_selection", "imputation_num","imputation_cat" ,"encoding","fold" ,"TARGET")
names(screen_glm_res)<-DF.names
names(screen_ranger_res)<-DF.names
screen_glm_res$gmean<-sqrt(screen_glm_res$sen*screen_glm_res$spec)
screen_ranger_res$gmean<-sqrt(screen_ranger_res$sen*screen_ranger_res$spec)

# defining the function that is used for making pivot figures
pvt_maker<-function(data_res,functio_pvt=c("mean"),
                    rows_names=c("encoding","feature_selection","imputation_num","imputation_cat","resampl
```

```
ing"),
                columns_names=c("auc","sen","spec","accu","gmean")){

 if(!"pacman" %in% rownames(installed.packages())){
  install.packages(pkgs = "pacman",repos = "http://cran.us.r-project.org")
 }
 # p_load is equivalent to combining both install.packages() and library()
 pacman::p_load(pivottabler,openxlsx)

pt <- PivotTable$new()
pt$addData(data_res)
#pt$addColumnDataGroups("auc")
for(i in rows_names){
 pt$addRowDataGroups(i, totalCaption="Total")
}
for(j in columns_names){
 pt$defineCalculation(calculationName=paste0(j,"_",functio_pvt),
                summariseExpression=paste0(functio_pvt,"(",j, ", na.rm=TRUE)"))
}

pt$renderPivot()
# library(openxlsx)
# wb <- createWorkbook(creator = Sys.getenv("USERNAME"))
# addWorksheet(wb, "Data")
# pt$writeToExcelWorksheet(wb=wb, wsName="Data",
#                          topRowNumber=1, leftMostColumnNumber=1, applyStyles=FALSE)
# saveWorkbook(wb, file="glm_pivot.xlsx", overwrite = TRUE)
# setwd("C:/Users/hza0020/OneDrive - Auburn University/Transplant/BUAL-LAB/DoE/temp/test RF/res
ults")
pt<-pt$asDataFrame()
return(pt)}

pvt_glm_mean<-pvt_maker(data_res=glm_res,functio_pvt=c("mean"))
pvt_nnet_mean<-pvt_maker(data_res=nnet_res,functio_pvt=c("mean"))
pvt_ranger_mean<-pvt_maker(data_res=ranger_res,functio_pvt=c("mean"))
pvt_xgbDART_mean<-pvt_maker(data_res=xgbDART_res,functio_pvt=c("mean"))
pvt_naive_bayes_mean<-pvt_maker(data_res=naive_bayes_res,functio_pvt=c("mean"))
pvt_kernelpls_mean<-pvt_maker(data_res=kernelpls_res,functio_pvt=c("mean"))
pvt_glmnet_mean<-pvt_maker(data_res=glmnet_res,functio_pvt=c("mean"))
pvt_rpart_mean<-pvt_maker(data_res=rpart_res,functio_pvt=c("mean"))
pvt_lda_mean<-pvt_maker(data_res=lda_res,functio_pvt=c("mean"))

pvt_glm_sd<-pvt_maker(data_res=glm_res,functio_pvt=c("sd"))
pvt_nnet_sd<-pvt_maker(data_res=nnet_res,functio_pvt=c("sd"))
pvt_ranger_sd<-pvt_maker(data_res=ranger_res,functio_pvt=c("sd"))
pvt_xgbDART_sd<-pvt_maker(data_res=xgbDART_res,functio_pvt=c("sd"))
pvt_naive_bayes_sd<-pvt_maker(data_res=naive_bayes_res,functio_pvt=c("sd"))
pvt_kernelpls_sd<-pvt_maker(data_res=kernelpls_res,functio_pvt=c("sd"))
pvt_glmnet_sd<-pvt_maker(data_res=glmnet_res,functio_pvt=c("sd"))
pvt_rpart_sd<-pvt_maker(data_res=rpart_res,functio_pvt=c("sd"))
pvt_lda_sd<-pvt_maker(data_res=lda_res,functio_pvt=c("sd"))
```

```
screen_pvt_glm_mean<-pvt_maker(data_res=screen_glm_res,functio_pvt=c("mean"))
screen_pvt_ranger_mean<-pvt_maker(data_res=screen_ranger_res,functio_pvt=c("mean"))
screen_pvt_glm_sd<-pvt_maker(data_res=screen_glm_res,functio_pvt=c("sd"))
screen_pvt_ranger_sd<-pvt_maker(data_res=screen_ranger_res,functio_pvt=c("sd"))

##############################screening step#################################################
##############

if(!"pacman" %in% rownames(installed.packages())){
  install.packages(pkgs = "pacman",repos = "http://cran.us.r-project.org")
}
library(pacman)
p_load(DoE.base,FrF2,DoE.wrapper,RcmdrPlugin.Export,
       RcmdrPlugin.FactoMineR,RcmdrPlugin.HH
       ,RcmdrPlugin.TeachingDemos,RcmdrPlugin.orloca,Rcmdr,conf.design
       ,lhs,AlgDesign,DiceDesign,rsm,RcmdrPlugin.DoE,pivottabler,data.table,DT , htmltools, nortest)

# excluding total rows for screening design
screen_DT_glm <- data.table(rownames(screen_pvt_glm_mean), result=grepl("Total", rownames(scre
en_pvt_glm_mean)))
screen_DT_glm<-as.data.frame(screen_DT_glm)
screen_pvt_glm_mean<-screen_pvt_glm_mean[which(screen_DT_glm$result==FALSE),]

screen_DT_ranger <- data.table(rownames(screen_pvt_ranger_mean), result=grepl("Total", rownames(
screen_pvt_ranger_mean)))
screen_DT_ranger<-as.data.frame(screen_DT_ranger)
screen_pvt_ranger_mean<-screen_pvt_ranger_mean[which(screen_DT_ranger$result==FALSE),]
##############################Full Design step#################################################
################


# excluding total rows for full factorial design
DT_glm <- data.table(rownames(pvt_glm_mean), result=grepl("Total", rownames(pvt_glm_mean)))
DT_glm<-as.data.frame(DT_glm)
pvt_glm_mean<-pvt_glm_mean[which(DT_glm$result==FALSE),]

DT_nnet <- data.table(rownames(pvt_nnet_mean), result=grepl("Total", rownames(pvt_nnet_mean)))
DT_nnet<-as.data.frame(DT_nnet)
pvt_nnet_mean<-pvt_nnet_mean[which(DT_nnet$result==FALSE),]

DT_ranger <- data.table(rownames(pvt_ranger_mean), result=grepl("Total", rownames(pvt_ranger_me
an)))
DT_ranger<-as.data.frame(DT_ranger)
pvt_ranger_mean<-pvt_ranger_mean[which(DT_ranger$result==FALSE),]

DT_xgbDART <- data.table(rownames(pvt_xgbDART_mean), result=grepl("Total", rownames(pvt_xg
bDART_mean)))
DT_xgbDART<-as.data.frame(DT_xgbDART)
pvt_xgbDART_mean<-pvt_xgbDART_mean[which(DT_xgbDART$result==FALSE),]
```

```
DT_naive_bayes <- data.table(rownames(pvt_naive_bayes_mean), result=grepl("Total", rownames(pvt
_naive_bayes_mean)))
DT_naive_bayes<-as.data.frame(DT_naive_bayes)
pvt_naive_bayes_mean<-pvt_naive_bayes_mean[which(DT_naive_bayes$result==FALSE),]

DT_kernelpls <- data.table(rownames(pvt_kernelpls_mean), result=grepl("Total", rownames(pvt_kern
elpls_mean)))
DT_kernelpls<-as.data.frame(DT_kernelpls)
pvt_kernelpls_mean<-pvt_kernelpls_mean[which(DT_kernelpls$result==FALSE),]

DT_glmnet <- data.table(rownames(pvt_glmnet_mean), result=grepl("Total", rownames(pvt_glmnet_
mean)))
DT_glmnet<-as.data.frame(DT_glmnet)
pvt_glmnet_mean<-pvt_glmnet_mean[which(DT_glmnet$result==FALSE),]

DT_rpart <- data.table(rownames(pvt_rpart_mean), result=grepl("Total", rownames(pvt_rpart_mean)))
DT_rpart<-as.data.frame(DT_rpart)
pvt_rpart_mean<-pvt_rpart_mean[which(DT_rpart$result==FALSE),]

DT_lda <- data.table(rownames(pvt_lda_mean), result=grepl("Total", rownames(pvt_lda_mean)))
DT_lda<-as.data.frame(DT_lda)
pvt_lda_mean<-pvt_lda_mean[which(DT_lda$result==FALSE),]




# finding the standard deviation of all algorithms, the algorithm names are taken from CARET library
SD_all<-as.data.frame(rbind(sapply(pvt_glm_mean, sd),sapply(pvt_lda_mean, sd),sapply(pvt_glmnet
_mean, sd),
                    sapply(pvt_kernelpls_mean, sd),sapply(pvt_nnet_mean, sd), sapply(pvt_naive_bayes_
mean, sd),
                    sapply(pvt_ranger_mean, sd),sapply(pvt_xgbDART_mean, sd),sapply(pvt_rpart_mean,
sd)  ))
names(SD_all)<-c("AUC_SD","Sen_SD","Spec_SD","Accu_SD","GMEAN_SD")
SD_all$algorithms<-c("glm","lda","glmnet",
                "kernelpls","nnet","naive_bayes",
                "ranger","xgbDART","rpart")

##############################screening step##################################################
##############
screen_SD_all<-as.data.frame(rbind(sapply(screen_pvt_glm_mean, sd),
                    sapply(screen_pvt_ranger_mean, sd)  ))
names(screen_SD_all)<-c("AUC_SD","Sen_SD","Spec_SD","Accu_SD","GMEAN_SD")
screen_SD_all$algorithms<-c("glm","ranger")
##############################screening step##################################################
##############

model_best <- as.data.frame(matrix(NA, ncol = 10, nrow = 5))
colnames(model_best) <- c("encoding","feature_selection","Num_treat","Cat_treat","resampling",
                    "auc_mean","sen_mean","spec_mean","accu_mean","gmean_mean")
```

```
row.names(model_best)<-c("best_accuracy","best_sensitivity","best_specificity","best_AUC","best_GMEAN")
glm_best<-model_best
ranger_best<-model_best
nnet_best<-model_best
xgbDART_best<-model_best
naive_bayes_best<-model_best
kernelpls_best<-model_best
glmnet_best<-model_best
rpart_best<-model_best
lda_best<-model_best

screen_glm_best<-model_best
screen_ranger_best<-model_best

##############################screening step##################################################
###############
screen_glm_best[1,1:5]<-unlist(strsplit(rownames(screen_pvt_glm_mean[which(screen_pvt_glm_mean$accu_mean==max(screen_pvt_glm_mean$accu_mean)),]),"  "))
screen_glm_best[1,6:10]<-screen_pvt_glm_mean[which(screen_pvt_glm_mean$accu_mean==max(screen_pvt_glm_mean$accu_mean)),]

screen_glm_best[2,1:5]<-unlist(strsplit(rownames(screen_pvt_glm_mean[which(screen_pvt_glm_mean$sen_mean==max(screen_pvt_glm_mean$sen_mean)),]),"  "))
screen_glm_best[2,6:10]<-screen_pvt_glm_mean[which(screen_pvt_glm_mean$sen_mean==max(screen_pvt_glm_mean$sen_mean)),]

screen_glm_best[3,1:5]<-unlist(strsplit(rownames(screen_pvt_glm_mean[which(screen_pvt_glm_mean$spec_mean==max(screen_pvt_glm_mean$spec_mean)),]),"  "))
screen_glm_best[3,6:10]<-screen_pvt_glm_mean[which(screen_pvt_glm_mean$spec_mean==max(screen_pvt_glm_mean$spec_mean)),]

screen_glm_best[4,1:5]<-unlist(strsplit(rownames(screen_pvt_glm_mean[which(screen_pvt_glm_mean$auc_mean==max(screen_pvt_glm_mean$auc_mean)),]),"  "))
screen_glm_best[4,6:10]<-screen_pvt_glm_mean[which(screen_pvt_glm_mean$auc_mean==max(screen_pvt_glm_mean$auc_mean)),]

screen_glm_best[5,1:5]<-unlist(strsplit(rownames(screen_pvt_glm_mean[which(screen_pvt_glm_mean$gmean_mean==max(screen_pvt_glm_mean$gmean_mean)),]),"  "))
screen_glm_best[5,6:10]<-screen_pvt_glm_mean[which(screen_pvt_glm_mean$gmean_mean==max(screen_pvt_glm_mean$gmean_mean)),]

#####

screen_ranger_best[1,1:5]<-unlist(strsplit(rownames(screen_pvt_ranger_mean[which(screen_pvt_ranger_mean$accu_mean==max(screen_pvt_ranger_mean$accu_mean)),]),"  "))
screen_ranger_best[1,6:10]<-screen_pvt_ranger_mean[which(screen_pvt_ranger_mean$accu_mean==max(screen_pvt_ranger_mean$accu_mean)),]

screen_ranger_best[2,1:5]<-unlist(strsplit(rownames(screen_pvt_ranger_mean[which(screen_pvt_ranger_mean$sen_mean==max(screen_pvt_ranger_mean$sen_mean)),]),"  "))
```

```
screen_ranger_best[2,6:10]<-screen_pvt_ranger_mean[which(screen_pvt_ranger_mean$sen_mean==max(screen_pvt_ranger_mean$sen_mean)),]

screen_ranger_best[3,1:5]<-unlist(strsplit(rownames(screen_pvt_ranger_mean[which(screen_pvt_ranger_mean$spec_mean==max(screen_pvt_ranger_mean$spec_mean)),])," "))
screen_ranger_best[3,6:10]<-screen_pvt_ranger_mean[which(screen_pvt_ranger_mean$spec_mean==max(screen_pvt_ranger_mean$spec_mean)),]

screen_ranger_best[4,1:5]<-unlist(strsplit(rownames(screen_pvt_ranger_mean[which(screen_pvt_ranger_mean$auc_mean==max(screen_pvt_ranger_mean$auc_mean)),])," "))
screen_ranger_best[4,6:10]<-screen_pvt_ranger_mean[which(screen_pvt_ranger_mean$auc_mean==max(screen_pvt_ranger_mean$auc_mean)),]

screen_ranger_best[5,1:5]<-unlist(strsplit(rownames(screen_pvt_ranger_mean[which(screen_pvt_ranger_mean$gmean_mean==max(screen_pvt_ranger_mean$gmean_mean)),])," "))
screen_ranger_best[5,6:10]<-screen_pvt_ranger_mean[which(screen_pvt_ranger_mean$gmean_mean==max(screen_pvt_ranger_mean$gmean_mean)),]
##############################screening step#################################################################

glm_best[1,1:5]<-unlist(strsplit(rownames(pvt_glm_mean[which(pvt_glm_mean$accu_mean==max(pvt_glm_mean$accu_mean)),])," "))
glm_best[1,6:10]<-pvt_glm_mean[which(pvt_glm_mean$accu_mean==max(pvt_glm_mean$accu_mean)),]

glm_best[2,1:5]<-unlist(strsplit(rownames(pvt_glm_mean[which(pvt_glm_mean$sen_mean==max(pvt_glm_mean$sen_mean)),])," "))
glm_best[2,6:10]<-pvt_glm_mean[which(pvt_glm_mean$sen_mean==max(pvt_glm_mean$sen_mean)),]

glm_best[3,1:5]<-unlist(strsplit(rownames(pvt_glm_mean[which(pvt_glm_mean$spec_mean==max(pvt_glm_mean$spec_mean)),])," "))
glm_best[3,6:10]<-pvt_glm_mean[which(pvt_glm_mean$spec_mean==max(pvt_glm_mean$spec_mean)),]

glm_best[4,1:5]<-unlist(strsplit(rownames(pvt_glm_mean[which(pvt_glm_mean$auc_mean==max(pvt_glm_mean$auc_mean)),])," "))
glm_best[4,6:10]<-pvt_glm_mean[which(pvt_glm_mean$auc_mean==max(pvt_glm_mean$auc_mean)),]

glm_best[5,1:5]<-unlist(strsplit(rownames(pvt_glm_mean[which(pvt_glm_mean$gmean_mean==max(pvt_glm_mean$gmean_mean)),])," "))
glm_best[5,6:10]<-pvt_glm_mean[which(pvt_glm_mean$gmean_mean==max(pvt_glm_mean$gmean_mean)),]


nnet_best[1,1:5]<-unlist(strsplit(rownames(pvt_nnet_mean[which(pvt_nnet_mean$accu_mean==max(pvt_nnet_mean$accu_mean)),])," "))
nnet_best[1,6:10]<-pvt_nnet_mean[which(pvt_nnet_mean$accu_mean==max(pvt_nnet_mean$accu_mean)),]
```

```
nnet_best[2,1:5]<-unlist(strsplit(rownames(pvt_nnet_mean[which(pvt_nnet_mean$sen_mean==max(pvt_nnet_mean$sen_mean)),]),"_"))
nnet_best[2,6:10]<-pvt_nnet_mean[which(pvt_nnet_mean$sen_mean==max(pvt_nnet_mean$sen_mean)),]

nnet_best[3,1:5]<-unlist(strsplit(rownames(pvt_nnet_mean[which(pvt_nnet_mean$spec_mean==max(pvt_nnet_mean$spec_mean)),]),"_"))
nnet_best[3,6:10]<-pvt_nnet_mean[which(pvt_nnet_mean$spec_mean==max(pvt_nnet_mean$spec_mean)),]

nnet_best[4,1:5]<-unlist(strsplit(rownames(pvt_nnet_mean[which(pvt_nnet_mean$auc_mean==max(pvt_nnet_mean$auc_mean)),]),"_"))
nnet_best[4,6:10]<-pvt_nnet_mean[which(pvt_nnet_mean$auc_mean==max(pvt_nnet_mean$auc_mean)),]

nnet_best[5,1:5]<-unlist(strsplit(rownames(pvt_nnet_mean[which(pvt_nnet_mean$gmean_mean==max(pvt_nnet_mean$gmean_mean)),]),"_"))
nnet_best[5,6:10]<-pvt_nnet_mean[which(pvt_nnet_mean$gmean_mean==max(pvt_nnet_mean$gmean_mean)),]


ranger_best[1,1:5]<-unlist(strsplit(rownames(pvt_ranger_mean[which(pvt_ranger_mean$accu_mean==max(pvt_ranger_mean$accu_mean)),]),"_"))
ranger_best[1,6:10]<-pvt_ranger_mean[which(pvt_ranger_mean$accu_mean==max(pvt_ranger_mean$accu_mean)),]

ranger_best[2,1:5]<-unlist(strsplit(rownames(pvt_ranger_mean[which(pvt_ranger_mean$sen_mean==max(pvt_ranger_mean$sen_mean)),]),"_"))
ranger_best[2,6:10]<-pvt_ranger_mean[which(pvt_ranger_mean$sen_mean==max(pvt_ranger_mean$sen_mean)),]

ranger_best[3,1:5]<-unlist(strsplit(rownames(pvt_ranger_mean[which(pvt_ranger_mean$spec_mean==max(pvt_ranger_mean$spec_mean)),]),"_"))
ranger_best[3,6:10]<-pvt_ranger_mean[which(pvt_ranger_mean$spec_mean==max(pvt_ranger_mean$spec_mean)),]

ranger_best[4,1:5]<-unlist(strsplit(rownames(pvt_ranger_mean[which(pvt_ranger_mean$auc_mean==max(pvt_ranger_mean$auc_mean)),]),"_"))
ranger_best[4,6:10]<-pvt_ranger_mean[which(pvt_ranger_mean$auc_mean==max(pvt_ranger_mean$auc_mean)),]

ranger_best[5,1:5]<-unlist(strsplit(rownames(pvt_ranger_mean[which(pvt_ranger_mean$gmean_mean==max(pvt_ranger_mean$gmean_mean)),]),"_"))
ranger_best[5,6:10]<-pvt_ranger_mean[which(pvt_ranger_mean$gmean_mean==max(pvt_ranger_mean$gmean_mean)),]


xgbDART_best[1,1:5]<-unlist(strsplit(rownames(pvt_xgbDART_mean[which(pvt_xgbDART_mean$accu_mean==max(pvt_xgbDART_mean$accu_mean)),]),"_"))
xgbDART_best[1,6:10]<-pvt_xgbDART_mean[which(pvt_xgbDART_mean$accu_mean==max(pvt_xgbDART_mean$accu_mean)),]
```

```
xgbDART_best[2,1:5]<-unlist(strsplit(rownames(pvt_xgbDART_mean[which(pvt_xgbDART_mean$sen_mean==max(pvt_xgbDART_mean$sen_mean)),]),"_"))
xgbDART_best[2,6:10]<-pvt_xgbDART_mean[which(pvt_xgbDART_mean$sen_mean==max(pvt_xgbDART_mean$sen_mean)),]

xgbDART_best[3,1:5]<-unlist(strsplit(rownames(pvt_xgbDART_mean[which(pvt_xgbDART_mean$spec_mean==max(pvt_xgbDART_mean$spec_mean)),]),"_"))
xgbDART_best[3,6:10]<-pvt_xgbDART_mean[which(pvt_xgbDART_mean$spec_mean==max(pvt_xgbDART_mean$spec_mean)),]

xgbDART_best[4,1:5]<-unlist(strsplit(rownames(pvt_xgbDART_mean[which(pvt_xgbDART_mean$auc_mean==max(pvt_xgbDART_mean$auc_mean)),]),"_"))
xgbDART_best[4,6:10]<-pvt_xgbDART_mean[which(pvt_xgbDART_mean$auc_mean==max(pvt_xgbDART_mean$auc_mean)),]

xgbDART_best[5,1:5]<-unlist(strsplit(rownames(pvt_xgbDART_mean[which(pvt_xgbDART_mean$gmean_mean==max(pvt_xgbDART_mean$gmean_mean)),]),"_"))
xgbDART_best[5,6:10]<-pvt_xgbDART_mean[which(pvt_xgbDART_mean$gmean_mean==max(pvt_xgbDART_mean$gmean_mean)),]


naive_bayes_best[1,1:5]<-unlist(strsplit(rownames(pvt_naive_bayes_mean[which(pvt_naive_bayes_mean$accu_mean==max(pvt_naive_bayes_mean$accu_mean)),]),"_"))
naive_bayes_best[1,6:10]<-pvt_naive_bayes_mean[which(pvt_naive_bayes_mean$accu_mean==max(pvt_naive_bayes_mean$accu_mean)),]

naive_bayes_best[2,1:5]<-unlist(strsplit(rownames(pvt_naive_bayes_mean[which(pvt_naive_bayes_mean$sen_mean==max(pvt_naive_bayes_mean$sen_mean)),]),"_"))
naive_bayes_best[2,6:10]<-pvt_naive_bayes_mean[which(pvt_naive_bayes_mean$sen_mean==max(pvt_naive_bayes_mean$sen_mean)),]

naive_bayes_best[3,1:5]<-unlist(strsplit(rownames(pvt_naive_bayes_mean[which(pvt_naive_bayes_mean$spec_mean==max(pvt_naive_bayes_mean$spec_mean)),]),"_"))
naive_bayes_best[3,6:10]<-pvt_naive_bayes_mean[which(pvt_naive_bayes_mean$spec_mean==max(pvt_naive_bayes_mean$spec_mean)),]

naive_bayes_best[4,1:5]<-unlist(strsplit(rownames(pvt_naive_bayes_mean[which(pvt_naive_bayes_mean$auc_mean==max(pvt_naive_bayes_mean$auc_mean)),]),"_"))
naive_bayes_best[4,6:10]<-pvt_naive_bayes_mean[which(pvt_naive_bayes_mean$auc_mean==max(pvt_naive_bayes_mean$auc_mean)),]

naive_bayes_best[5,1:5]<-unlist(strsplit(rownames(pvt_naive_bayes_mean[which(pvt_naive_bayes_mean$gmean_mean==max(pvt_naive_bayes_mean$gmean_mean)),]),"_"))
naive_bayes_best[5,6:10]<-pvt_naive_bayes_mean[which(pvt_naive_bayes_mean$gmean_mean==max(pvt_naive_bayes_mean$gmean_mean)),]


kernelpls_best[1,1:5]<-unlist(strsplit(rownames(pvt_kernelpls_mean[which(pvt_kernelpls_mean$accu_mean==max(pvt_kernelpls_mean$accu_mean)),]),"_"))
kernelpls_best[1,6:10]<-pvt_kernelpls_mean[which(pvt_kernelpls_mean$accu_mean==max(pvt_kernelp
```

```
ls_mean$accu_mean)),]

kernelpls_best[2,1:5]<-unlist(strsplit(rownames(pvt_kernelpls_mean[which(pvt_kernelpls_mean$sen_
mean==max(pvt_kernelpls_mean$sen_mean)),]),"_"))
kernelpls_best[2,6:10]<-pvt_kernelpls_mean[which(pvt_kernelpls_mean$sen_mean==max(pvt_kernelpl
s_mean$sen_mean)),]

kernelpls_best[3,1:5]<-unlist(strsplit(rownames(pvt_kernelpls_mean[which(pvt_kernelpls_mean$spec_
mean==max(pvt_kernelpls_mean$spec_mean)),]),"_"))
kernelpls_best[3,6:10]<-pvt_kernelpls_mean[which(pvt_kernelpls_mean$spec_mean==max(pvt_kernelp
ls_mean$spec_mean)),]

kernelpls_best[4,1:5]<-unlist(strsplit(rownames(pvt_kernelpls_mean[which(pvt_kernelpls_mean$auc_
mean==max(pvt_kernelpls_mean$auc_mean)),]),"_"))
kernelpls_best[4,6:10]<-pvt_kernelpls_mean[which(pvt_kernelpls_mean$auc_mean==max(pvt_kernelpl
s_mean$auc_mean)),]

kernelpls_best[5,1:5]<-unlist(strsplit(rownames(pvt_kernelpls_mean[which(pvt_kernelpls_mean$gmea
n_mean==max(pvt_kernelpls_mean$gmean_mean)),]),"_"))
kernelpls_best[5,6:10]<-pvt_kernelpls_mean[which(pvt_kernelpls_mean$gmean_mean==max(pvt_kerne
lpls_mean$gmean_mean)),]

glmnet_best[1,1:5]<-unlist(strsplit(rownames(pvt_glmnet_mean[which(pvt_glmnet_mean$accu_mean=
=max(pvt_glmnet_mean$accu_mean)),]),"_"))
glmnet_best[1,6:10]<-pvt_glmnet_mean[which(pvt_glmnet_mean$accu_mean==max(pvt_glmnet_mean
$accu_mean)),]

glmnet_best[2,1:5]<-unlist(strsplit(rownames(pvt_glmnet_mean[which(pvt_glmnet_mean$sen_mean=
=max(pvt_glmnet_mean$sen_mean)),]),"_"))
glmnet_best[2,6:10]<-pvt_glmnet_mean[which(pvt_glmnet_mean$sen_mean==max(pvt_glmnet_mean$
sen_mean)),]

glmnet_best[3,1:5]<-unlist(strsplit(rownames(pvt_glmnet_mean[which(pvt_glmnet_mean$spec_mean=
=max(pvt_glmnet_mean$spec_mean)),]),"_"))
glmnet_best[3,6:10]<-pvt_glmnet_mean[which(pvt_glmnet_mean$spec_mean==max(pvt_glmnet_mean
$spec_mean)),]

glmnet_best[4,1:5]<-unlist(strsplit(rownames(pvt_glmnet_mean[which(pvt_glmnet_mean$auc_mean=
=max(pvt_glmnet_mean$auc_mean)),]),"_"))
glmnet_best[4,6:10]<-pvt_glmnet_mean[which(pvt_glmnet_mean$auc_mean==max(pvt_glmnet_mean$
auc_mean)),]

glmnet_best[5,1:5]<-unlist(strsplit(rownames(pvt_glmnet_mean[which(pvt_glmnet_mean$gmean_mea
n==max(pvt_glmnet_mean$gmean_mean)),]),"_"))
glmnet_best[5,6:10]<-pvt_glmnet_mean[which(pvt_glmnet_mean$gmean_mean==max(pvt_glmnet_me
an$gmean_mean)),]


rpart_best[1,1:5]<-unlist(strsplit(rownames(pvt_rpart_mean[which(pvt_rpart_mean$accu_mean==max
(pvt_rpart_mean$accu_mean)),]),"_"))
rpart_best[1,6:10]<-pvt_rpart_mean[which(pvt_rpart_mean$accu_mean==max(pvt_rpart_mean$accu_m
```

```
ean)),]

rpart_best[2,1:5]<-unlist(strsplit(rownames(pvt_rpart_mean[which(pvt_rpart_mean$sen_mean==max(
pvt_rpart_mean$sen_mean)),]),"_"))
rpart_best[2,6:10]<-pvt_rpart_mean[which(pvt_rpart_mean$sen_mean==max(pvt_rpart_mean$sen_mea
n)),]

rpart_best[3,1:5]<-unlist(strsplit(rownames(pvt_rpart_mean[which(pvt_rpart_mean$spec_mean==max
(pvt_rpart_mean$spec_mean)),]),"_"))
rpart_best[3,6:10]<-pvt_rpart_mean[which(pvt_rpart_mean$spec_mean==max(pvt_rpart_mean$spec_m
ean)),]

rpart_best[4,1:5]<-unlist(strsplit(rownames(pvt_rpart_mean[which(pvt_rpart_mean$auc_mean==max(
pvt_rpart_mean$auc_mean)),]),"_"))
rpart_best[4,6:10]<-pvt_rpart_mean[which(pvt_rpart_mean$auc_mean==max(pvt_rpart_mean$auc_mea
n)),]

rpart_best[5,1:5]<-unlist(strsplit(rownames(pvt_rpart_mean[which(pvt_rpart_mean$gmean_mean==m
ax(pvt_rpart_mean$gmean_mean)),]),"_"))
rpart_best[5,6:10]<-pvt_rpart_mean[which(pvt_rpart_mean$gmean_mean==max(pvt_rpart_mean$gmea
n_mean)),]


lda_best[1,1:5]<-unlist(strsplit(rownames(pvt_lda_mean[which(pvt_lda_mean$accu_mean==max(pvt_
lda_mean$accu_mean)),]),"_"))
lda_best[1,6:10]<-pvt_lda_mean[which(pvt_lda_mean$accu_mean==max(pvt_lda_mean$accu_mean)),]

lda_best[2,1:5]<-unlist(strsplit(rownames(pvt_lda_mean[which(pvt_lda_mean$sen_mean==max(pvt_l
da_mean$sen_mean)),]),"_"))
lda_best[2,6:10]<-pvt_lda_mean[which(pvt_lda_mean$sen_mean==max(pvt_lda_mean$sen_mean)),]

lda_best[3,1:5]<-unlist(strsplit(rownames(pvt_lda_mean[which(pvt_lda_mean$spec_mean==max(pvt_
lda_mean$spec_mean)),]),"_"))
lda_best[3,6:10]<-pvt_lda_mean[which(pvt_lda_mean$spec_mean==max(pvt_lda_mean$spec_mean)),]

lda_best[4,1:5]<-unlist(strsplit(rownames(pvt_lda_mean[which(pvt_lda_mean$auc_mean==max(pvt_l
da_mean$auc_mean)),]),"_"))
lda_best[4,6:10]<-pvt_lda_mean[which(pvt_lda_mean$auc_mean==max(pvt_lda_mean$auc_mean)),]

lda_best[5,1:5]<-unlist(strsplit(rownames(pvt_lda_mean[which(pvt_lda_mean$gmean_mean==max(pv
t_lda_mean$gmean_mean)),]),"_"))
lda_best[5,6:10]<-pvt_lda_mean[which(pvt_lda_mean$gmean_mean==max(pvt_lda_mean$gmean_mea
n)),]


screen_MEAN_all<-as.data.frame(rbind( screen_glm_best[5,],screen_ranger_best[5,]  ))
row.names(screen_MEAN_all)<-NULL
screen_MEAN_all$algorithms<-c("glm","ranger")
cat("##########Screening Design#################")
cat("Standard Deviation of Algorithms:")
DT::datatable(screen_SD_all)
```

```
cat("Peformance of the Best Combinations:")
DT::datatable(screen_MEAN_all)

cat("##########Full Factorial Design#################")

MEAN_all_GMEAN<-as.data.frame(rbind( glm_best["best_GMEAN",],lda_best["best_GMEAN",],gl
mnet_best["best_GMEAN",],
                      kernelpls_best["best_GMEAN",],nnet_best["best_GMEAN",], naive_bayes_best["best
_GMEAN",],
                      ranger_best["best_GMEAN",],xgbDART_best["best_GMEAN",],rpart_best["best_G
MEAN",]  ))

MEAN_all_GMEAN$algorithms<-c("glm","lda","glmnet",
                  "kernelpls","nnet","naive_bayes",
                  "ranger","xgbDART","rpart")
row.names(MEAN_all_GMEAN)<-NULL
# average performance of algorithms in GMEAN
cat("Peformance of the Best Combinations in GMEAN:")
DT::datatable(MEAN_all_GMEAN)


MEAN_all_AUC<-as.data.frame(rbind( glm_best["best_AUC",],lda_best["best_AUC",],glmnet_best["
best_AUC",],
                        kernelpls_best["best_AUC",],nnet_best["best_AUC",], naive_bayes_best["best_A
UC",],
                        ranger_best["best_AUC",],xgbDART_best["best_AUC",],rpart_best["best_AUC",
] ))

MEAN_all_AUC$algorithms<-c("glm","lda","glmnet",
                    "kernelpls","nnet","naive_bayes",
                    "ranger","xgbDART","rpart")
row.names(MEAN_all_AUC)<-NULL
# average performance of algorithms in AUC
cat("Peformance of the Best Combinations in AUC:")
DT::datatable(MEAN_all_AUC)



# standard deviation of the algorithms's performance
cat("Standard Deviation of Algorithms:")
DT::datatable(SD_all)
```

Section: Here the Best Algorithms Based on their performance is selected for the case that we have just class validation

```
# Here summary of all the scenarios are provided
glm_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/glm.summary.
csv")
nnet_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/nnet.summar
```

```
y.csv")
ranger_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/ranger.summary.csv")
xgbDART_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/xgbDART.summary.csv")
naive_bayes_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/naive_bayes.summary.csv")
kernelpls_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/kernelpls.summary.csv")
glmnet_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/glmnet.summary.csv")
rpart_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/rpart.summary.csv")
lda_res<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/lda.summary.csv")

# Excluding repeated cv and just working on one of the repeats
glm_res<-glm_res[which(glm_res$fold==1),]
glm_res<-glm_res[which(glm_res$fold==1),]

nnet_res<-nnet_res[which(nnet_res$fold==1),]
nnet_res<-nnet_res[which(nnet_res$fold==1),]

ranger_res<-ranger_res[which(ranger_res$fold==1),]
ranger_res<-ranger_res[which(ranger_res$fold==1),]

xgbDART_res<-xgbDART_res[which(xgbDART_res$fold==1),]
xgbDART_res<-xgbDART_res[which(xgbDART_res$fold==1),]

naive_bayes_res<-naive_bayes_res[which(naive_bayes_res$fold==1),]
naive_bayes_res<-naive_bayes_res[which(naive_bayes_res$fold==1),]

kernelpls_res<-kernelpls_res[which(kernelpls_res$fold==1),]
kernelpls_res<-kernelpls_res[which(kernelpls_res$fold==1),]

glmnet_res<-glmnet_res[which(glmnet_res$fold==1),]
glmnet_res<-glmnet_res[which(glmnet_res$fold==1),]

rpart_res<-rpart_res[which(rpart_res$fold==1),]
rpart_res<-rpart_res[which(rpart_res$fold==1),]

lda_res<-lda_res[which(lda_res$fold==1),]
lda_res<-lda_res[which(lda_res$fold==1),]

best_settings_norepeat<-rbind(
  glm_res[which(glm_res$gmean==max(glm_res$gmean)),],
nnet_res[which(nnet_res$gmean==max(nnet_res$gmean)),] ,
ranger_res[which(ranger_res$gmean==max(ranger_res$gmean)),],
xgbDART_res[which(xgbDART_res$gmean==max(xgbDART_res$gmean)),],
naive_bayes_res[which(naive_bayes_res$gmean==max(naive_bayes_res$gmean)),],
kernelpls_res[which(kernelpls_res$gmean==max(kernelpls_res$gmean)),],
```

```
glmnet_res[which(glmnet_res$gmean==max(glmnet_res$gmean)),],
rpart_res[which(rpart_res$gmean==max(rpart_res$gmean)),] ,
lda_res[which(lda_res$gmean==max(lda_res$gmean)),]
)
```

Section 5: Multi time Points Prediction

Here the Best Algorithms Based on their performance is selected

Now, the best scenario is: Logistic Regression, One-Hot encoding, LASSO feature seceltion, Median for Numerical variables' imputation, "UNKNOWN" label for Categorical variables' imputation, and Up sampling. This combination is used for the rest of years.

```
# We dropped the numerical NAs and replaced the NA (missing values) in categorical variables to "UNK
NOWN"
heart.df.cleaned_all<-readRDS(paste0(data.path,"heart.df.cleaned_all.rds"))
pool_char_clean_all<-readRDS(paste0(data.path,"pool_char_clean_all.rds"))
pool_num_clean_all<-readRDS(paste0(data.path,"pool_num_clean_all.rds"))

# The unavailable data in numerical variables are imputed by median
heart.df.cleaned_all_num <- heart.df.cleaned_all[pool_num_clean_all]

num_med<-as.data.frame(apply(heart.df.cleaned_all_num,2,function(x) median(x,na.rm = TRUE) ))
for(j in pool_num_clean_all){
  heart.df.cleaned_all_num[is.na(heart.df.cleaned_all_num[j]),j]<-num_med[j,1]
}

# imputing the unavailable data for categorical variables by "UNKNOWN"
heart.df.cleaned_all_char <- heart.df.cleaned_all[pool_char_clean_all]
heart.df.cleaned_all_char[is.na(heart.df.cleaned_all_char)] <- "UNKNOWN"


heart.df.cleaned_all<- cbind(heart.df.cleaned_all_num,
                             heart.df.cleaned_all_char,
                             heart.df.cleaned_all[c("ID")])

pool_char_clean <- c(pool_char_clean,paste("year", seq(0,10), sep=""))

# remove unnecessary datasets
rm(list=c("heart.df.cleaned_all_char","heart.df.cleaned_all_num"))

#Make sure the type of each variable is recorded corrctly
for(i in names(heart.df.cleaned_all)){
  if(i %in% pool_char_clean_all){heart.df.cleaned_all[i] <- as.character(heart.df.cleaned_all[,i])}
  if(i %in% pool_num_clean_all){heart.df.cleaned_all[i] <- as.numeric(heart.df.cleaned_all[,i])}
}
```

Create the training and holdout datasets

In this snippet, we create the training and holdout datasets at each time point of interest. In order to both validate models on the holdout datasets and use isotonic regression to calibrate survival probabilities, we make sure all patients selected in the 10th year holdout dataset were included in previous time points. This is not a necessary procedure for either prediction purpose or calibration of survival probabilities.

We use a list object keep_NA to record the following IDs: 1. IDs for the dataset at each time point of interest 2. IDs for the holdout dataset at each time point of interest 3. IDs for the training dataset at each time point of interest

```
# We keep IDs
keep_NA <- rep(list(NA), 33)
for (i in 1:11){
  keep_NA[[i]] <- heart.df.cleaned_all[!is.na(heart.df.cleaned_all[paste0("year", (i-1))]),"ID"]
}

names(keep_NA)[1:11] <- paste0("ID",seq(0,10))
all_sizes <- unname(unlist(lapply(keep_NA, length)))
test_sizes <- round(all_sizes*0.2)
train_sizes <- all_sizes - test_sizes

# We select the holdout sets that contain the year 10 test data
# Then we exclude those IDs from each year to find IDs for other training sets
set.seed(2019)
keep_NA[[12]] <- sample(keep_NA$ID10,test_sizes[11])
names(keep_NA)[12] <- "ID_holdout10"

for (i in 2:11){
  keep_NA[[(11+i)]] <- c(keep_NA$ID_holdout10, sample(setdiff(keep_NA[[(12-i)]], keep_NA$ID_hol
dout10), (test_sizes[(12-i)]-length(keep_NA$ID_holdout10))))
}

names(keep_NA)[13:22] <- paste0("ID_holdout", seq(9,0))

for (i in 1:11){
 keep_NA[[(22+i)]] <- setdiff(keep_NA[[i]], keep_NA[[paste0("ID_holdout",(i-1))]])
}

names(keep_NA)[23:33] <- paste0("ID_train", seq(0,10))
```

```
# The one-hot encoding algorithm was applied to independent categorical variables, the function dummy
_maker() is used.
exclud <- c(paste0("year",seq(0,10)), "ID")
var_ind_char <- pool_char_clean[!pool_char_clean %in% exclud]
heart.df.cleaned_all.dum <- dummy_maker(heart.df.cleaned_all,var_ind_char)
saveRDS(heart.df.cleaned_all.dum,paste0(data.path,"heart.df.cleaned_all.dum.rds"))
saveRDS(keep_NA,paste0(data.path,"keep_NA.rds"))
# making the data ready for running feature selection over super computer
server_data<-list()
exclud <- c(paste0("year",seq(0,10)), "ID")
for (i in 1:11){
  temp<-heart.df.cleaned_all.dum[heart.df.cleaned_all.dum$ID %in% keep_NA[[paste0("ID_train", i-1)]
],]
  temp2<- temp[!names(temp) %in% exclud]
  temp2[[paste0("year",i-1)]]<-temp[[paste0("year",i-1)]]
  server_data[[i]]<-temp2
}
rm(list=c("temp","temp2"))

# end of part 3
```

Multi Time Points Feature Selection

The following code is what is performed over super computer to achive feature selection with LASSO. The LASSO feature selection algorithm is loaded

```
gc()
if(!"pacman" %in% rownames(installed.packages())){
  install.packages(pkgs = "pacman",repos = "http://cran.us.r-project.org")
}
# p_load is equivalent to combining both install.packages() and library()
pacman::p_load(caret,AUC,MASS,ROSE,DMwR,snow,ranger,parallel,xgboost,gbm,naivebayes,e1071,k
ernlab,pls,Rmpi)

# making the data ready for running feature selection over super computer
heart.df.cleaned_all.dum<-readRDS("/users/PMIU0138/miu0150/Transplant/data/heart.df.cleaned_all.du
m.rds")
keep_NA<-readRDS("/users/PMIU0138/miu0150/Transplant/data/keep_NA.rds")

server_data<-list()
exclud <- c(paste0("year",seq(0,10)), "ID")
for (i in 1:11){
  temp<-heart.df.cleaned_all.dum[heart.df.cleaned_all.dum$ID %in% keep_NA[[paste0("ID_train", i-1)]
],]
  temp2<- temp[!names(temp) %in% exclud]
```

```
  temp2[[paste0("year",i-1)]]<-temp[[paste0("year",i-1)]]
  server_data[[i]]<-temp2
}
rm(list=c("temp","temp2"))



##################################LASSO Feature Selection Function#######################
LASSO_features <- function(ii,data,exclud,folds=5,trace=F,alpha=1,seed=110){
  set.seed(seed)
  dataset<-data[[ii]]
  dff<-dataset[!names(dataset)%in%paste0("year",ii-1)]

  for(i in 1:ncol(dff)){
    dff[i] <- as.numeric(dff[,i])
  }
  x <- data.matrix(dff)
  glmnet1 <- glmnet::cv.glmnet(x=x,y=as.factor(dataset[,paste0("year",ii-1)]),type.measure='auc',nfolds
=folds,alpha=alpha, family="binomial")
  co <- coef(glmnet1,s = "lambda.1se")
  inds <- which(co[,1]!=0)
  variables <- row.names(co)[inds]
  variables <- as.data.frame(variables[!(variables %in% '(Intercept)')])
  colnames(variables) <- c("variables")
  newlist <- list(variables)
  names(newlist) <- paste("year", ii-1, sep="")
  return(newlist)
}
##########################################################################################
###

#identifying number of created clusters for parallel processing
if(Sys.info()[1]=="Windows"){
  library(snow)
  cl <- parallel::makeCluster(4)
}else{
  library(Rmpi)
  library(snow)
  # assign cores used in the parallel computing
  workers <- as.numeric(Sys.getenv(c("PBS_NP"))) - 1
  s.no<-nrow(screen_design)
  workers.no<-min(workers,s.no)
  cl <- makeCluster(workers.no,"MPI")
}


time.begin <- proc.time()[3]

LASSO <- parSapply(cl, 1:11, LASSO_features,data=server_data,
                   folds=5,trace=F, alpha=1,seed=110)
```

```
time.window <- proc.time()[3] - time.begin

stopCluster(cl)

saveRDS(LASSO,paste0("/users/PMIU0138/miu0150/Transplant/data/_Features_LASSO.rds"))

mpi.quit()
```

Multi Time Points training and prediction

The following code is what is performed over super computer to perform Logistic Regression

```
if(Sys.info()[1]=="Windows"){
data.path<-"C:/Users/hza0020/OneDrive - Auburn University/Transplant/BUAL-LAB/DoE_10years/"
LASSO<-readRDS(paste0(data.path,"_Features_LASSO.rds"))
keep_NA<-readRDS(paste0(data.path,"keep_NA.rds"))
heart.df.cleaned_all.dum<-readRDS(paste0(data.path,"heart.df.cleaned_all.dum.rds"))
}else{
# making the data ready for running feature selection over super computer
LASSO<-readRDS("/users/PMIU0138/miu0150/Transplant/data/_Features_LASSO.rds")
heart.df.cleaned_all.dum<-readRDS("/users/PMIU0138/miu0150/Transplant/data/heart.df.cleaned_all.dum.rds")
keep_NA<-readRDS("/users/PMIU0138/miu0150/Transplant/data/keep_NA.rds")
}



train_data<-list()
holdOut_data<-list()
exclud <- c(paste0("year",seq(0,10)), "ID")
exclud_h <- c(paste0("year",seq(0,10)))
for (i in 1:11){
  temp<-heart.df.cleaned_all.dum[heart.df.cleaned_all.dum$ID %in% keep_NA[[paste0("ID_train", i-1)]],]
  temp2<- temp[!names(temp) %in% exclud]
  temp2[[paste0("year",i-1)]]<-temp[[paste0("year",i-1)]]
  train_data[[i]]<-temp2

  temp0<-heart.df.cleaned_all.dum[heart.df.cleaned_all.dum$ID %in% keep_NA[[paste0("ID_holdout", i-1)]],]
  temp02<- temp0[!names(temp0) %in% exclud_h]
  temp02[[paste0("year",i-1)]]<-temp0[[paste0("year",i-1)]]
  holdOut_data[[i]]<-temp02
}

# saveRDS(holdOut_data,paste0(data.path,"_holdOut_data.rds"))

rm(list=c("temp","temp2","temp0","temp02"))
```

```
gc()
if(!"pacman" %in% rownames(installed.packages())){
  install.packages(pkgs = "pacman",repos = "http://cran.us.r-project.org")
}
# p_load is equivalent to combining both install.packages() and library()
pacman::p_load(caret,AUC,MASS,ROSE,DMwR,snow,ranger,parallel,xgboost,gbm,naivebayes,e1071,k
ernlab,pls,Rmpi)


###########main function#############################################################
train_para2<-function(iii, t_data,h_data,features,folds=5,resampl_meth="up",alg_used,seed0=2019){

  gc()
  if(!"pacman" %in% rownames(installed.packages())){
    install.packages(pkgs = "pacman",repos = "http://cran.us.r-project.org")
  }
  # p_load is equivalent to combining both install.packages() and library()
  pacman::p_load(caret,AUC,MASS,ROSE,DMwR,snow,ranger,parallel,xgboost,gbm,naivebayes,e1071,
kernlab,pls,parallel)

  set.seed(seed0)
  fold_no<-folds
  alg_fact<-alg_used
  resample_fact<-resampl_meth
  coef<-"NO"
  var_imp<-"NO"


  df <- t_data[[iii]]
  hold_out_ <- h_data[[iii]]
  traindata <- df[c(as.character(features[[paste0("year",iii-1)]][,]),paste0("year",iii-1) )]
   hold_out<-hold_out_[c(as.character(features[[paste0("year",iii-1)]][,]),paste0("year",iii-1),"ID" )]
  TARGET0<-paste0("year",iii-1)

  # 1: survival; 0: death
  traindata[,as.character(TARGET0)] <- as.factor(ifelse(traindata[,as.character(TARGET0)]=="0", "De
ath", "Survival"))
  hold_out[,as.character(TARGET0)] <- as.factor(ifelse(hold_out[,as.character(TARGET0)]=="0", "D
eath", "Survival"))


  formul<-as.formula(paste0(as.character(TARGET0),"~."))

  run.code<-"YES"
  sampling.method<-resample_fact
  if(resample_fact=="none"){sampling.method<-NULL}
  # Reference for geometric mean
  # Kim, Myoung-Jong, Dae-Ki Kang, and Hong Bae Kim. "Geometric mean based boosting
  # algorithm with over-sampling to resolve data imbalance problem for bankruptcy prediction."
  # Expert Systems with Applications 42.3 (2015): 1074-1082.
```

```
 gmeanfunction <- function(data, lev = NULL, model = NULL) {
   sub_sens<-caret::sensitivity(data$pred,data$obs)
   sub_spec<-caret::specificity(data$pred,data$obs)
   c(gmean = sqrt(sub_sens*sub_spec))
 }


 # I used 5 fold cross validation
 control_setting <- caret::trainControl(method = "cv", number=5, sampling=sampling.method ,
                                        #summaryFunction = twoClassSummary,
                                        # the next one is for saving the predictions
                                        savePredictions = TRUE,
                                        search="random", classProbs = TRUE, selectionFunction="best"
                                        ,summaryFunction = gmeanfunction)


 if (alg_fact%in% c("glm", "nnet", "svmRadial")){

   if((class(try( result_model <- train(formul, data=traindata, method=alg_fact, family="binomial",
                  trControl = control_setting, metric="gmean"),silent = TRUE))=="try-error")[1]){run.code
<-"NO"}



 }else if (alg_fact%in%c("rf", "gbm", "earth", "rpart", "xgbTree", "naive_bayes","xgbDART" ,"ranger",
"glmnet")){

   if((class(try( result_model <- train(formul,  data=traindata, method=alg_fact,
                  trControl = control_setting, tuneLength=10, metric="gmean"),silent = TRUE))=="try-err
or")[1]){ run.code<-"NO"}


 }else if(alg_fact=="lda"){
   if((class(try(  result_model <- train(formul, data=traindata, method=alg_fact, preProcess="pca", prePr
ocOptions = list(method="BoxCox"),
                  trControl = control_setting, metric="gmean"),silent = TRUE))=="try-error")[1]){run.code
<-"NO"}


 }else if(alg_fact=="treebag"){
   if((class(try(  result_model <- train(formul, data=traindata, method=alg_fact, family="binomial",
                  trControl = control_setting, tuneLength=10, metric="gmean"),silent = TRUE))=="try-err
or")[1]){run.code<-"NO"}

 }

 if(run.code=="YES"){

 coef<-as.data.frame(summary(result_model)$coefficients)
 coef$vars<-rownames(coef)
 coef$vars<-as.character(sapply(coef$vars,function(x) gsub("1","",x)))
```

```
coef$vars[which(coef$vars=="(Intercept)")]<-"INTERCEPT"

coef$vars[match("DAYS_STAT1",rownames(coef))]<-"DAYS_STAT1"
coef$vars[match("DAYS_STAT1A",rownames(coef))]<-"DAYS_STAT1A"
coef$vars[match("DAYS_STAT1B",rownames(coef))]<-"DAYS_STAT1B"

if(alg_fact=="glm"){
var_imp<-as.data.frame(caret::varImp(result_model)$importance)
var_imp$vars<-rownames(var_imp)
var_imp$vars<-as.character(sapply(var_imp$vars,function(x) gsub("1","",x)))
var_imp$vars[match("DAYS_STAT1",rownames(var_imp))]<-"DAYS_STAT1"
var_imp$vars[match("DAYS_STAT1A",rownames(var_imp))]<-"DAYS_STAT1A"
var_imp$vars[match("DAYS_STAT1B",rownames(var_imp))]<-"DAYS_STAT1B"
}


resul_raw <- as.data.frame(matrix(NA, ncol = 4, nrow = nrow(hold_out)))
colnames(resul_raw) <- c("TARGET_raw", alg_fact, "Probability","ID")
resul_raw$TARGET_raw <- hold_out[as.character(TARGET0)]
resul_raw$ID<-hold_out$ID

train_raw <- as.data.frame(matrix(NA, ncol = 3, nrow = nrow(traindata)))
colnames(train_raw) <- c("TARGET", alg_fact, "Probability")
train_raw$TARGET <- traindata[as.character(TARGET0)]

resul_pred_perf<-as.data.frame(matrix(NA, ncol = 1, nrow = 5))
colnames(resul_pred_perf)<-c(alg_fact)
rownames(resul_pred_perf)<-c("auc","sen","spec","accu","GMEAN")

#if(class(try(varImp(result_model),silent = TRUE))!="try-error"){
train_raw$Probability <- predict(result_model, newdata=traindata, type="prob")[,2]
train_raw[alg_fact] <- predict(result_model, newdata=traindata, type="raw")
#train_auc <- AUC::auc(roc(train_raw$Probability, traindata[,TARGET]))
resul_raw[alg_fact] <- predict(result_model, newdata=hold_out, type="raw")
resul_raw$Probability <- predict(result_model, newdata=hold_out, type="prob")[,2]
resul_pred_perf[1,1] <- AUC::auc(roc(resul_raw$Probability,hold_out[,as.character(TARGET0)]))
resul_pred_perf[2,1] <- caret::sensitivity(resul_raw[,alg_fact],hold_out[,as.character(TARGET0)])
resul_pred_perf[3,1] <- caret::specificity(resul_raw[,alg_fact],hold_out[,as.character(TARGET0)])
resul_pred_perf[4,1] <- (as.data.frame(confusionMatrix(resul_raw[,alg_fact], hold_out[,as.character(
TARGET0[1])])$overall))[1,]
resul_pred_perf[5,1] <- sqrt(resul_pred_perf[2,1]*resul_pred_perf[3,1])
#}
gmean_overal<-as.data.frame(result_model$results)
gmean_folds<-as.data.frame(result_model$resample)
equat<-as.data.frame(result_model$finalModel$coefficients)
names(equat)<-"coef"
equat$vars<-rownames(equat)
equat$vars[which(equat$vars==c("(Intercept)"))]<-c("INTERCEPT")

max_gmean<-max(result_model$resample$gmean)
fold_pick<-result_model$resample[which(result_model$resample$gmean==max_gmean),c("Resample"
```

```
)]
  foldbest<-result_model$pred[which(result_model$pred$Resample==fold_pick),c("obs","pred","Survival")]
  names(foldbest)<-c(paste0("year",iii-1),"log","Probability")


  dir.create(paste0(getwd(),"/data_pile"))
  rm(list=setdiff(ls(), c("resul_pred_perf","resul_raw","alg_fact","iii","coef","var_imp",
                          "gmean_overal","gmean_folds","equat","foldbest")))

  saveRDS(list(Performance=resul_pred_perf, Predicted=resul_raw,coef=coef,var_imp=var_imp,gmean_overal=gmean_overal,gmean_folds=gmean_folds,
               equat=equat,foldbest=foldbest ),
          paste0(getwd(),"/data_pile/",alg_fact,"-year-",iii-1,".rds"))

  return(list(Performance=resul_pred_perf, Predicted=resul_raw,coef=coef,var_imp=var_imp,gmean_overal=gmean_overal,gmean_folds=gmean_folds,
              equat=equat,foldbest=foldbest))
  }else{
    dir.create(paste0(getwd(),"/data_pile"))
    rm(list=setdiff(ls(), c("experiment.summary","alg_fact","iii","coef","var_imp")))
    save.image(paste0(getwd(),"/data_pile/NOT-",alg_fact,"-",iii,".RData"))
    saveRDS(list(Performance="NOT", Predicted="NOT",coef=coef,var_imp=var_imp,gmean_overal="NOT",gmean_folds="NOT",
              equat="NOT",foldbest="NOT"),
            paste0(getwd(),"/data_pile/NOT-",alg_fact,"-year-",iii-1,".rds"))
    return(list(Performance="NOT", Predicted="NOT",coef=coef,var_imp=var_imp,gmean_overal="NOT",gmean_folds="NOT",
              equat="NOT",foldbest="NOT"))
  }
}

##############################################################################


#identifying number of created clusters for parallel processing
if(Sys.info()[1]=="Windows"){
  library(snow)
  cl <- parallel::makeCluster(4)
}else{
  library(Rmpi)
  library(snow)
  # assign cores used in the parallel computing
  workers <- as.numeric(Sys.getenv(c("PBS_NP"))) - 1
  s.no<-nrow(screen_design)
  workers.no<-min(workers,s.no)
  cl <- makeCluster(workers.no,"MPI")
}

time.begin <- proc.time()[3]
```

```
pred_LR <- parSapply(cl, 1:11,train_para2 ,t_data=train_data,h_data=holdOut_data,features=LASSO,
                     folds=5,resampl_meth="up",alg_used="glm",seed0=2019)

time.window <- proc.time()[3] - time.begin

stopCluster(cl)

saveRDS(pred_LR,paste0("/users/PMIU0138/miu0150/Transplant/data/_pred_LR717.rds"))

mpi.quit()
saveRDS(pred_LR,paste0(data.path,"_pred_LR717.rds"))
```

Multi Time Points Reports

The following code demonstrates a report about importance of the features

```
data.path<-"C:/Users/hza0020/OneDrive - Auburn University/Transplant/BUAL-LAB/DoE_10years/"
LR_models<-readRDS(paste0(data.path,"_pred_LR717.rds"))

heart.df.cleaned_all.dum<-readRDS(paste0(data.path,"heart.df.cleaned_all.dum.rds"))
exclud <- c(paste0("year",seq(0,10)), "ID")
vars<-names(heart.df.cleaned_all.dum)[!names(heart.df.cleaned_all.dum) %in% exclud]

varimp_years<-as.data.frame(matrix(0, ncol = 12, nrow = length(vars)))
colnames(varimp_years)<-c("vars",paste0("year",seq(0,10)))
varimp_years$vars<-vars

perf_years<-as.data.frame(matrix(0, ncol = 13, nrow = 5))
colnames(perf_years)<-c("measure",paste0("year",seq(0,10)),"all_mean")
perf_years$measure<-c("AUC","Sensitivity","Specificity","Accuracy","Geometic_Mean")

for (i in 0:10){
  varimp_years[varimp_years$vars %in% LR_models[4,i+1]$var_imp$vars,paste0("year",i)]<-1
  perf_years[paste0("year",i)]<-LR_models[1,i+1]$Performance

}

perf_years$all_mean<-rowMeans(perf_years[paste0("year",seq(0,10))])


varimp_years$no_years<-rowSums(varimp_years[paste0("year",seq(0,10))])
varimp_years<-varimp_years[which(varimp_years$no_years!=0),]

saveRDS(varimp_years,paste0(data.path,"_holdOut_data.rds"))

cat("The next 4 figure represent Coefficients of Variables and their importance  for the First Month & Co
efficients of Variables and their importance  for the First Year, consecutively:")
# LR_models[3,1]$coef
# LR_models[4,1]$var_imp
```

```
# LR_models[3,2]$coef
# LR_models[4,2]$var_imp
```

The following code demonstrates a report about performance of predictions in multiple time

points.

```
library(pacman) # needs to be installed first
# p_load is equivalent to combining both install.packages() and library()
p_load(caret, AUC)
# data.path<-"C:/Users/hza0020/OneDrive - Auburn University/Transplant/BUAL-LAB/DoE_10years/"
# h_data<-readRDS(paste0(data.path,"_holdOut_data.rds"))
#
# data<-h_data[[iii]][coef$vars[2:length(coef$vars)]]
#
# data<-data[,coef$vars[2:length(coef$vars)]]
# for(i in 1:ncol(data)){
#   if(is.factor(data[,i])){data[,i]<-as.character(data[,i])}
# }
#
# sum((as.numeric(data[1,]) * coef$Estimate[2:length(coef$Estimate)]))+coef$Estimate[1]


logit2prob <- function(logit){
  odds <- exp(logit)
  prob <- odds / (1 + odds)
  return(prob)
}

# logit2prob(122.99)
#
# LR_models[1,8]
# str(LR_models[2,1]$Predicted)
# nrow(LR_models[2,1]$Predicted)
# nrow(LR_models[2,2]$Predicted)
# nrow(LR_models[2,3]$Predicted)
# nrow(LR_models[2,11]$Predicted)

# LR_models[4,1]$var_imp$vars
#
#  varimp_years$vars %in% LR_models[4,11]$var_imp$vars

 for (i in 0:10){
   varimp_years[varimp_years$vars %in% LR_models[4,i+1]$var_imp$vars,paste0("year",i)]<-1
 }
 # head(predictions[[2]])
 # head(LR_models[2,11]$Predicted)
 # nrow(LR_models[2,11]$Predicted)
 # LR_models[2,11]$Predicted$glm

 predictions<-list()
```

```
in_all_h<-LR_models[2,11]$Predicted$ID
all_preds<-as.data.frame(matrix(0, ncol = 12, nrow = length(in_all_h)))
all_TARGETS<-as.data.frame(matrix(0, ncol = 11, nrow = length(in_all_h)))
colnames(all_preds)<-c(paste0("year",seq(0,10)),"monotonic")
colnames(all_TARGETS)<-c(paste0("year",seq(0,10)))
all_TARGETS_pred<-all_TARGETS

for(i in 0:10){
  predictions[[i+1]]<-LR_models[2,i+1]$Predicted[which(LR_models[2,i+1]$Predicted$ID %in%
                                                 in_all_h),]
  if(sum(predictions[[i+1]]$ID==in_all_h)==length(in_all_h)){
    all_preds[paste0("year",i)]<-predictions[[i+1]]$Probability
    all_TARGETS[paste0("year",i)]<-predictions[[i+1]]$TARGET_raw[paste0("year",i)]
    all_TARGETS_pred[paste0("year",i)]<-predictions[[i+1]]$glm
  }
  #monotonic
}
all_preds2<-all_preds
all_TARGETS_pred2<-all_TARGETS_pred
for(i in 1:nrow(all_preds)){
  all_preds$monotonic[i]<-all(as.numeric(all_preds[i,paste0("year",seq(0,10))])==cummin(
    as.numeric(all_preds[i,paste0("year",seq(0,10))])))
  all_preds2[i,paste0("year",seq(0,10))]<-rev(isoreg(rev(as.numeric(all_preds[i,paste0("year",seq(0,10)
)]))))$yf)
}

for(i in 1:nrow(all_preds2)){
  for(j in 1:ncol(all_preds2[paste0("year",seq(0,10))])){
    if(all_preds2[i,j]>=0.5){all_TARGETS_pred2[i,j]<-"Survival"}else{all_TARGETS_pred2[i,j]<-"Deat
h"}
  }

}


perf_before_iso<-as.data.frame(matrix(0, ncol = 13, nrow = 5))
colnames(perf_before_iso)<-c("Measures",paste0("year",seq(0,10)),"Mean")
perf_before_iso$Measures<-c("AUC", "Sensitivity","Specificity","Accuracy","GMean" )

perf_after_iso<-perf_before_iso

for(i in 0:10){
  perf_before_iso[1,paste0("year",i)]<-AUC::auc(roc(all_preds[,paste0("year",i)],all_TARGETS[,paste
0("year",i)]))
  perf_before_iso[2,paste0("year",i)]<-caret::sensitivity(all_TARGETS_pred[,paste0("year",i)],all_TAR
GETS[,paste0("year",i)])
  perf_before_iso[3,paste0("year",i)]<-caret::specificity(all_TARGETS_pred[,paste0("year",i)],all_TAR
GETS[,paste0("year",i)])
  perf_before_iso[4,paste0("year",i)]<-(as.data.frame(confusionMatrix(all_TARGETS_pred[,paste0("
year",i)],
                                                 all_TARGETS[,paste0("year",i)])$overall))[1,]
```

```
  perf_before_iso[5,paste0("year",i)] <- sqrt(perf_before_iso[2,paste0("year",i)]*perf_before_iso[3,paste0("year",i)])


  perf_after_iso[1,paste0("year",i)]<-AUC::auc(roc(all_preds2[,paste0("year",i)],all_TARGETS[,paste0("year",i)]))
  perf_after_iso[2,paste0("year",i)]<-caret::sensitivity(all_TARGETS_pred2[,paste0("year",i)],all_TARGETS[,paste0("year",i)])
  perf_after_iso[3,paste0("year",i)]<-caret::specificity(all_TARGETS_pred2[,paste0("year",i)],all_TARGETS[,paste0("year",i)])
  perf_after_iso[4,paste0("year",i)]<-(as.data.frame(confusionMatrix(all_TARGETS_pred2[,paste0("year",i)],
                                                     all_TARGETS[,paste0("year",i)])$overall))[1,]
  perf_after_iso[5,paste0("year",i)] <- sqrt(perf_after_iso[2,paste0("year",i)]*perf_after_iso[3,paste0("year",i)])


 }

 perf_before_iso$Mean<-rowMeans(perf_before_iso[paste0("year",seq(0,10))])
 perf_after_iso$Mean<-rowMeans(perf_after_iso[paste0("year",seq(0,10))])
```

Results

Here you can find performance of predictions before/after isotonic regression for the patients that their data is available in all the timestamps ***Here the results before isotonic regression is presented***

```
before_iso<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/rmarkdown/two_stage/before-isotonicregression.csv")
DT::datatable(before_iso)
```

Show 10 entries Search:

	Measures.before.isotonic.regression	year0	year1	year2	year3	year4	year5	year6	year7	year8	year9	year10	Mean
1	AUC	0.6077467	0.5808928	0.5709949	0.5944147	0.6193418	0.6305694	0.6542878	0.6710463	0.6983645	0.703257	0.701616	0.6393211
2	Sensitivity	0.5891892	0.5896633	0.5650833	0.5791444	0.5621749	0.558937	0.5580945	0.5692749	0.5605245	0.5410981	0.5332734	0.5642234
3	Specificity	0.5422829	0.5217487	0.5323194	0.5483708	0.5990845	0.618135	0.6593779	0.6838646	0.7225919	0.7434653	0.7517048	0.6293587
4	Accuracy	0.546749	0.5366272	0.5414308	0.5582433	0.5856922	0.5944416	0.6145136	0.6290959	0.6399039	0.634071	0.6266941	0.5915875
5	GMean	0.5652497	0.5546675	0.5484567	0.5635476	0.5803364	0.5877912	0.6066261	0.6239447	0.6364201	0.6342615	0.6331383	0.59404

Showing 1 to 5 of 5 entries Previous 1 Next

Here the Results after isotonic regression is presented

```
after_iso<-read.csv("https://raw.githubusercontent.com/transplantation/unos-ht/master/data/rmarkdown/two_stage/after-isotonicregression.csv")
DT::datatable(after_iso)
```

Show 10 entries Search:

	Measures.after.isotonic.regression	year0	year1	year2	year3	year4	year5	year6	year7	year8	year9	year10	Mean
1	AUC	0.6153929	0.5980169	0.5999845	0.6154114	0.6307456	0.6372251	0.6556869	0.6723508	0.6983075	0.705457	0.7022439	0.6482566
2	Sensitivity	0.4756757	0.4847298	0.5015423	0.5320856	0.5527187	0.567081	0.5917893	0.6191673	0.6472764	0.653126	0.6675659	0.5720689
3	Specificity	0.6920743	0.6575132	0.6418726	0.6317252	0.6313947	0.624714	0.6227287	0.6224121	0.6206655	0.613891	0.5904533	0.6317677
4	Accuracy	0.6714702	0.6196603	0.6028478	0.5997598	0.6028478	0.6016469	0.6090238	0.6208612	0.6342426	0.6351004	0.6345857	0.6210952
5	GMean	0.5737621	0.5645496	0.5673854	0.5797688	0.5907484	0.5952003	0.6070619	0.6207876	0.6338313	0.6332047	0.6278268	0.5994661

Showing 1 to 5 of 5 entries Previous 1 Next

The Interactive App

We have created an interactive app that presents two modules for performing the analysis:

(1) Manual Entry, where users can insert the values of predictor variables using several text boxes.

(2) CSV Entry, where users can upload the values of predictor variables using a comma separated variable (CSV) file.

Visit the app: http://dataviz.miamioh.edu/Heart-Transplant/monotonic/