Pre-Processing Censored Survival Data Using Inverse Covariance Matrix Based Calibration

Bhanukiran Vinzamuri, Yan Li, and Chandan K. Reddy, *Senior Member, IEEE*

Abstract—Censoring is a common phenomenon that arises in many longitudinal studies where an event of interest could not be recorded within the given time frame. Censoring causes missing time-to-event labels, and this effect is compounded when dealing with datasets which have high amounts of censored instances. In addition, dependent censoring in the data, where censoring is dependent on the covariates in the data leads to bias in standard survival estimators. This motivates us to develop an approach for pre-processing censored data which calibrates the right censored (RC) times in an attempt to reduce the bias in the survival estimators. This calibration is done using an imputation method which estimates the sparse inverse covariance matrix over the dataset in an iterative convergence framework. During estimation, we apply row and column-based regularization to account for both row and column-wise correlations between different instances while imputing them. This is followed by comparing these imputed censored times with the original RC times to obtain the final calibrated RC times. These calibrated RC times can now be used in the survival dataset in place of the original RC times for more effective prediction. One of the major benefits of our calibration approach is that it is a pre-processing method for censored data which can be used in conjunction with any survival prediction algorithm and improve its performance. We evaluate the goodness of our approach using a wide array of survival prediction algorithms which are applied over crowdfunding data, electronic health records (EHRs), and synthetic censored datasets. Experimental results indicate that our calibration method improves the AUC values of survival prediction algorithms, compared to applying them directly on the original survival data.

Index Terms—Survival analysis, pre-processing, right censoring, imputation, healthcare, crowdfunding

1 Introduction

Censoring is a common phenomenon that appears in many real-world application domains such as healthcare, engineering, social sciences, etc. In longitudinal studies, observations are called censored when the information about their event time is incomplete. For example, let us consider a healthcare application where a set of patients are being monitored over a period of time for a the occurrence of a particular event of interest (such as risk for readmission or death). A patient who does not experience the event of interest within the duration of the study is said to be right censored [1]. The survival time of such a patient is considered to be atleast as long as the duration of the study. Another important example of right censoring is when a subject drops out of the study before the end of the study and did not experience the event until that time. These characteristics make censoring an important issue in survival analysis representing a particular type of missing data. This problem is also called the missing time-to-events problem and has a significant practical value [2], [3], [4].

- *B. Vinzamuri is with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598. E-mail: bhanu.vinzamuri@ibm.com.*
- *Y. Li is with the Department of Computer Science, University of Michigan, Ann Arbor, MI 48109. E-mail: yanliwl@umich.edu.*
- *C. K. Reddy is with the Department of Computer Science, Virginia Tech, Arlington, VA 22203. E-mail: reddy@cs.vt.edu.*

Manuscript received 31 Dec. 2015; revised 3 Feb. 2017; accepted 13 June 2017. Date of publication 23 June 2017; date of current version 8 Sept. 2017.
(Corresponding author: Bhanukiran Vinzamuri.)
Recommended for acceptance by R. Cheng.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2017.2719028

In traditional survival analysis, the model is built on such data by labeling the censored instances with a value such as the duration of the study or last known follow up time [5], [6], [7]. This phenomenon is called right censoring and is observed in several real-world datasets. This can become a significant problem, especially when the data has many right censored instances ($\geq 40\%$ of overall number of instances). To overcome this problem, we propose to solve it by developing an approach called calibrated survival analysis which can learn an appropriate probable time-to-event label value for the right censored instances. We now explain our motivation for developing this framework for calibrating time-to-event values for censored instances by explaining the inherent problem associated with censored data, and we also explain the two-dimensional correlation-based structure in censored data using an example from the crowdfunding domain.

1.1 Motivation

Censoring in data can be divided into two kinds of methods, namely, independent and dependent censoring. Independent censoring is a phenomenon where the covariates and censoring are assumed to be independent [6], [8]. Only under this assumption alone, traditional estimators such as Kaplan-Meier (KM) remain unbiased yielding true estimates. However, most datasets violate independent censoring and exhibit a phenomenon called dependent censoring where the covariates in the data and censoring are correlated with each other. In this scenario, the KM estimator is biased which affects the performance of several existing survival analysis methods.

To address this issue, in this paper, we present an approach called calibrated survival analysis which employs

a novel form of censoring called imputed censoring. The goal of imputed censoring is to reduce the bias in standard survival estimators, and this is accomplished by using a regularized inverse covariance-based imputation algorithm. We use covariance-based imputation methods as they are well equipped to capture correlations between censored instances while performing imputation which other methods such as matrix factorization do not capture.

The correlation structure in censored data exhibits a unique phenomenon which can be explained by considering a typical crowdfunding scenario. In this scenario, we define an event of interest as the time taken by a project to reach its pre-defined goal amount and succeed. Considering two projects which got censored, one can notice that to impute the time-to-event labels for these instances two factors need to be considered which are (i) time taken by instances similar to both of them to reach the goal amount (row-wise correlation) (ii) importance of similar features for both censored instances in determining the time-to-event (column-wise correlation). To account for both these phenomena, we use a row and column based regularization approach within an inverse covariance estimation procedure to appropriately estimate the time-to-event label. Our proposed calibrated survival analysis approach imputes the time-to-event labels for censored instances using a regularized inverse covariance matrix approach. In this paper, we present both the column-based (*REC*) and the row, column based (*TREC*) algorithms in Section 3.

Another important motivation for proposing a calibrated survival analysis framework for such real-world censored datasets can be obtained from the theory of representation learning [9]. Representation learning attempts to learn a novel representation of the data which captures the inherent structure, so that any predictive algorithm can perform better on the learned new representation. In calibrated survival analysis, through imputed censoring, we are effectively learning a new represenation of the original survival data by solving the bias problem explained earlier. We also state that imputed censoring preserves the original censored nature of the problem, and does not output a predictive model directly. Hence, our proposed approach can be used in conjunction with other existing predictive survival analysis methods. In other words, our method can be treated as an important pre-processing step that incorporates correlation structures and accordingly imputes the censored values.

1.2 Our Contributions

The major contributions of this paper are as follows:

- Propose a calibrated survival analysis framework which uses a novel imputed censoring approach to model the time-to-event variable. This imputed censoring approach uses a row and column regularization based inverse covariance estimation algorithm to impute the censored instances. The goal of this approach is to impute the labels for the censored instances by estimating their probable time-to-event labels in order to build a more effective representation of the survival data which an algorithm can leverage upon.
- Study the formulation of our row and column based regularized inverse covariance method which is used in imputed censoring exhaustively. We discuss the properties of this algorithm using the L_1 and L_2 regularizers, but the framework can work with any regularizer with a defined L_p norm where ($p \geq 1$).
- Evaluate the effectiveness of our calibration method by comparing the survival AUC (concordance index) [10] values obtained using standard survival regression algorithms on the data with and without our time-to-event calibration. We also conduct experiments to assess the convergence and the impact of regularizers and regularization parameters on the performance of our algorithm.

This paper is organized as follows, In Section 2, we present the related work on the topic of using censoring with machine learning methods. In particular, we explain the advancements in the field of Cox regression and also discuss other approaches which integrate censoring with Bayesian methods. In Section 3, we introduce important notations and definitions. In Section 4, we explain the formulation of the proposed algorithms which integrate censoring with regularized inverse covariance models. In Section 5, we present the experimental results obtained using our methods and present the tables comparing the results before and after applying calibration and we also present runtime and convergence results. Finally, in Section 6, we present the conclusions derived at the end of our study and discuss the practical implications of the proposed work.

2 Related Work

In this section, we present the related work in the area of using machine learning methods for survival analysis, and also describe imputation methods for censored data. In the survival analysis domain, Cox regression has garnered significant interest from researchers in the biostatistics and machine learning communities [2], [11].

- *Cox regression and its extensions*: Cox regression is a semi-parametric method which uses a proportionality hazards (PH) assumption. It is widely used because of its effective performance and ease of availability. Some of the major extensions to Cox regression include using the lasso, elastic net and kernel elastic net regularizers [12], [13]. Graph regularization has also been used with Cox regression, where the graph laplacian is used as a penalty [14]. Finally, structured regularizers have also been used with Cox regression to integrate group based information into the optimization problem [13]. Other extensions include integrating active learning with Cox regression which can help an expert build an interactive Cox regression framework [15].
- *Bayesian methods for censored data*: Censored Naive Bayes (CensNB) is an approach which applies the standard Naive Bayes algorithm for censored data [16]. In this algorithm, the conditional survivor function is learned by initializing the functions using non-parametric densities, which are then subsequently smoothed using a weighted loess smoother. These models use an approach called inverse probability of censoring weighting (IPCW) for each of the records in the dataset. Bayesian Networks-based

TABLE 1
Notations Used in This Paper

Name	Description
n	number of instances
p	number of columns
X	$\Re^{n\times p}$ data matrix
δ	censored indicator variable
Σ	$n \times n$ Row covariance matrix
Δ	$p \times p$ column covariance matrix
υ_i	mean of i^{th} row
μ_j	mean of j^{th} column
q_r	row regularizer
ρ_r	row penalty
q_c	column regularizer
ρ_c	column penalty

methods have also been used to enhance the performance of survival trees [17], [18]. The imputation on missing instances is done using the bayesian network computed on complete instances and the model has shown to perform well in clinical trials.

- *Estimation of missing time-to-events*: Multiple imputation for censored data is a method where the failure times are imputed using an asymptotic data augmentation scheme based on the current estimates and the baseline survival curve [19]. Once this is done a standard procedure such as Cox regression is applied to the imputed data to update the estimates. A similar problem has been dealt within the crowdsourcing domain which predicts the time-to-event directly using the survival function [20]. Misglasso is an extension to the approach for imputing missing values by using the graphical lasso algorithm [23], [24]. Other popular approaches include the SoftImpute algorithm which uses a nuclear norm minimization subject to constraints to fill the missing entries [25]. Risk stratified imputation in survival analysis is another approach which performs stratified imputation of missing time-to-events based on groups of patients who are similar to each other. The stratification is done to ensure that not too many samples are imputed, and the imputation is done among censored instances which are similar to each other. An auxiliary variable approach to multiple imputation in survival analysis is proposed in this paper with the goal to improve efficiency using Monte Carlo methods [26]. Finally, Elastic net Buckley James (EN-BJ) [27] is a method which directly models the response for events using the least squares method, and for the censored instances the response variable is imputed using the conditional expectation values given the corresponding censoring times and covariates. This algorithm uses the elastic-net regularization term with this AFT model and was applied on high-dimensional genomic data obtaining good performance.

Our approach is different from the methods mentioned above as we aim at *calibrating* the time-to-event value and build a more effective representation of the survival data. Our approach is unique as it does not build a learner, and it can be used as a pre-processing step along with any base survival prediction algorithm to enhance its performance.

3 Preliminaries

In this section, we explain an overview of our proposed method for converting a censored dataset into a calibrated censored dataset. We begin by presenting the table of notations used in this paper in Table 1.

In this section, we present an overview of our pre-processing calibration method which can convert any given dataset with right censoring into a calibrated right censored dataset. In this approach, we build a framework that uses both single and composite regularization by imposing regularizers and user provided penalty parameters on both the rows (single) and rows, columns (composite) of the feature matrix.

Before presenting the algorithmic details, we review the notations used throughout the paper. In this paper, we will often refer to right censoring as censoring and vice versa. 1_n represents a unit vector of n entries. We use i to denote the row index and j to denote the column index.

X here represents the concatenated matrix of the features and time-to-event label values. We assume that X originally is not centered w.r.t. row and column means. The last column of X corresponds to the time attribute (T). The remaining features correspond to the survival covariates. The time-to-event labels for those instances which are right censored originally are represented using T_{orig}. The labels finally learned after using our approach are referred to as T_{calib}. We will also use the abbreviation (RC) for right censored instances frequently through the remainder of this section.

The observed and missing parts of row i are o_i and m_i, respectively, and o_j and m_j are the analogous parts of column index j. Let m and o denote the complete set of missing and observed elements, respectively. X_{i,o_i} denotes the observed components of an uncensored observation i and X_{i,m_i} denote the missing components of a censored instance. This is defined this way inorder to include the notion of both missing feature values and the missing time-to-event label information for the censored instance. However, this is simplified later on when we confine the missingness to the last column of the matrix alone which corresponds to the time attribute. For each instance in X, we partition the mean and covariance to correspond to the observed parts of instance i and denote them by μ_{o_i} and Δ_{o_i,o_i}, respectively.

We now look at the probability distribution used for estimating time-to-event labels in this paper. This distribution is called a *mean-restricted matrix variate normal distribution* [28] which has certain desirable properties such as non-negativity and dual covariance parameters which are needed for modeling censored event times. We provide the formulation of the probability distribution below

$$\begin{aligned} p(\upsilon,\mu,\Sigma,\Delta) = {} & (2\pi)^{-np/2}|\Sigma|^{-p/2}|\Delta|^{-n/2} \\ & \times etr\Big(-\frac{1}{2}(X-\upsilon 1_{(p)}^T - 1_{(n)}\mu^T)\Delta^{-1} \\ & (X-\upsilon 1_{(p)}^T - 1_{(n)}\mu^T)^T\Sigma^{-1}\Big). \end{aligned} \tag{1}$$

In Eq. (1), $etr(\cdot)$ represents the exponential of the trace term here. It can be clearly seen here that the row and column means are subtracted from X to center the concatenated feature time matrix. This distribution implies that the time-to-event labels are modeled with a mean υ_i +

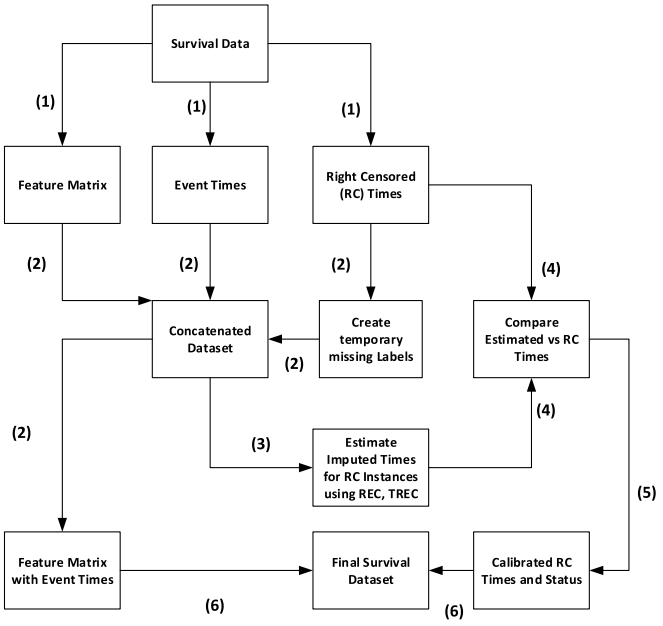


Fig. 1. Flow diagram of our approach.

μ_j along with variance $\Sigma_{ii}\Delta_{jj}$. Due to this formulation, the estimated time-to-event labels are non-negative which cannot be guaranteed with the simple normal distribution. This makes the mean-restricted matrix variate normal distribution ideally suited for modeling probable non-negative event times for censored instances.

This modeling can also be viewed as a random effects model where the estimated time-to-event label value can be expressed as $\upsilon_i + \mu_j + \epsilon_{ij}$ where $\epsilon_{ij} \sim N(0, \Sigma_{ii}\Delta_{jj})$ which has two additive fixed effects depending on the row and column means and a random effect whose variance depends on the product of the corresponding row and column covariances.

The goal of this method is to impute the time-to-event for RC events, and in this process, calibrating it to a more optimal value. This is called the *calibration* step of our method where we impute the time-to-event labels for the right censored instances. We emphasize that the imputed censoring employed here preserves the censoring in this dataset.

4 Pre-Processing Using Inverse Covariance Matrix Based Calibration

In this section, we present the flow diagram of our approach which is followed by presenting the two methods for calibrated survival analysis. We begin by explaining the REgularized inverse covariance based Calibration (*REC*) method, and then present the Transposable REgularized covariance based Calibration (*TREC*) method. Before exploring the inner details of these algorithms, we state explicitly that both these approaches are only meant to build a more effective representation of the original survival data. The time values in the calibrated censored dataset are not the predicted values, but are only estimated by our iterative convergence framework in an effort to facilitate the process.

4.1 Overview of Our Approach

We provide a flow diagram in Fig. 1 which explains how our approach works on right censored survival data. We labeled the flow diagram with numbers which indicate the steps and the direction being taken in the process. Initially, in (1), we identify the set of right censored instances in the dataset and extract the features, events and the original RC times and store them. In (2), we replace these times using temporary missing labels in order to facilitate our method for identifying and imputing these selected instances. These are combined with the features and events extracted from (1) to create a concatenated dataset. Before creating this concatenated dataset, the features and events are also stored separately to be used in the final step.

In (3), we apply the two main algorithms which will be discussed in this section, namely, *REC* and *TREC*. This gives us the imputed times for RC instances which are now compared with the original RC times in (4). The procedure used to compare these times is discussed in this section after the *REC* algorithm. After this step, in (5), we obtain the calibrated RC times and status variables. Finally, in (6), these calibrated outputs are combined with the feature and event matrix from (2) to obtain the final survival dataset. With this overview of our approach, we now look into the details of the two main algorithms proposed here, namely, *REC* and *TREC*.

4.2 REC Algorithm

In this section, we begin by explaining the *REC* method which receives the censored dataset as the input and outputs the calibrated times and status, which are used for learning the final model. This algorithm is designed using an *iterative convergence* style optimization procedure where we initialize the missing time-to-event values and update our estimates iteratively until convergence is observed.

We now present the regularized likelihood equation used in *REC* algorithm in Eq. (2) which uses a single column based regularization term. One can notice that an important difference between this and the EM algorithm term is the regularization term used. Imputation is a part of the E step of the algorithm in which the conditional expectation of the complete data log-likelihood is taken given the current parameter estimates. The computation in *REC* can be divided into two parts which are (i) imputation-based calibration and (ii) covariance correction steps respectively. We outline both these steps in Eqs. (3) and (4)

$$\ell_{obs}(\mu, \Delta) = \frac{1}{2}\sum_{i=1}^{n}[\log|\Delta^{-1}_{o_i,o_i}| - (X_{o_i} - \mu_{o_i})^T\Delta^{-1}_{o_i,o_i}(X_{o_i} - \mu_{o_i})] - \rho_c \parallel \Delta^{-1} \parallel_{q_c}. \tag{2}$$

The first step, imputation-based calibration, is given in Eq. (3). This step also involves the covariance-based correction term and the next step is given in Eq. (4). The covariance-based correction term is defined so because it is added to the cross products forming the covariance matrix

$$\hat{X}_{i,j} = E(X_{i,j} \mid X_{i,o_i}, \mu', \Delta') = \begin{cases} \mu'_{m_i} + \Delta'_{m_i,o_i}\Delta'^{-1}_{o_i,o_i}(x_{i,o_i} - \mu'_{o_i}), & \text{if} \quad j \in m_i \\ X_{i,j}, & \text{if} \quad j \in o_i \end{cases} \tag{3}$$

$$c_{i,jj'} = \begin{cases} \Delta'_{m_i,m_i} - \Delta'_{m_i,o_i}\Delta'^{-1}_{o_i,o_i}\Delta'_{o_i,m_i}, & \text{if } j,j' \in m_i \\ 0, & \text{otherwise.} \end{cases}$$

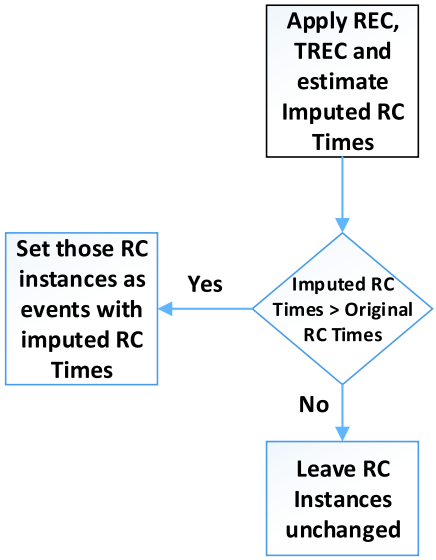


Fig. 2. Flow diagram for the *CompareTimes* procedure.

We can notice that in the covariance correction term $c_{i,jj'}$ is non-zero only when both j and j' are missing (censored in our context). The second step of our *REC* algorithm is the maximization step which is given in Eq. (5). In this maximization step, $\hat{\Delta}'$ is computed which is done by replacing μ with $\hat{\mu}$ in $\hat{\Delta}'$

$$E(X_{i,j}X_{i,j'} \mid X_{i,o_i}, \mu', \Delta') = \hat{X}_{i,j}\hat{X}_{i,j'} + c_{i,j'j}. \tag{4}$$

In Algorithm 1, we follow an iterative convergence routine similar to the traditional EM algorithm with the only difference being the introduction of a row-based regularization term and the corresponding covariance correction term. We set q_r=1 using the L_1 regularizer due to its formulation as the graphical lasso which can be solved using coordinate-descent techniques efficiently

$$\begin{aligned} Q(\theta|\theta^k) &= \frac{n}{2} log|\Delta^{-1}| - \frac{1}{2}\mathrm{tr}(\hat{\Delta}'\Delta^{-1}) - \rho_c \parallel \Delta^{-1} \parallel_{q_c} \\ \hat{\Delta}'_{jj'} &= \Sigma_{i=1}^{n}[(\hat{x}_{ij} - \mu_j)(\hat{x}_{ij} - \mu_j) + c_{i,jj'}]. \end{aligned} \tag{5}$$

In Algorithm 1, in Line 3, we do segregate the survival data as outlined in (1) and (2) in Fig. 1. In Lines 4-6, we do the EM style optimization as explained above to iteratively estimate the values for RC instances until convergence is observed. In Lines 7-8, we employ the *CompareTimes* procedure as given in Fig. 2, where we compare the imputed RC times obtained from Line 6 and compare those to the original RC times. The comparison is done using the following rule. We know that for right censored instances there is a chance for the event to occur in the future which is not captured due to censoring. After applying REC, if the imputed RC time for an instance exceeds its original RC time, we conclude that this can be considered as an event and relabel it accordingly. We also modify the censored status (δ) from 0 to 1 to indicate this is an event. This modified vector is stored in the calibrated status vector (δ_{calib}). However, if the imputed RC time is less than or equal to the original RC time, we cannot make any conclusion whether this is an event or not. So instances in this category are left unchanged and their censored time and status are the same as their original RC times and status, respectively.

Algorithm 1. REC Algorithm

1: **Input**: features and time matrix X, status δ
2: **Output**: calibrated RC times T_{calib}, calibrated status δ_{calib}
3: Initialization:
   (a) Store original RC times T_{orig} and replace them with temporary missing labels.
   (b) Set the missing labels as: $\hat{X}_{i,m_i} = \Sigma_{i \in o_i} X_{ij}/n_i$
   (c) Set $\mu^{(0)}, \Delta^{(0)}$ as the empirical mean, covariance.
4: E Step:
   (a) Compute $\mathrm{E}(X_{i,j}|x_{i,o_i}, \mu^{(k)}, \Delta^{(k)})$ as in Eq. (3)
   (b) Compute $\mathrm{E}(X_{i,j}X_{i,j'}|X_{i,o_i}, \mu^{(k)}, \Delta^{(k)})$ as in Eq. (4)
5: M Step:
   (a) Update Estimates: $\hat{\mu}_j \& \hat{\Delta}'_{jj'}$
   (b) Maximize penalized log-likelihood w.r.t. Δ^{-1} to obtain the new estimate $\hat{\Delta}$
6: Repeat Steps 3-5 until convergence.
7: Estimate imputed RC times (T_{imp}) and compare with original RC times (T_{orig}) using the *CompareTimes* procedure given in Fig. 2.
8: Output calibrated time-to-event variable T_{calib} and calibrated status δ_{calib}.

This column-based regularization captures one aspect of imputing censored instances by considering the feature importance among different censored instances in determining their corresponding time-to-event labels while imputing them. This approximation of the probable event time for right censored instances as done by our approach is one of its hallmarks which eventually leads to generating a better representation of the survival data. We now present the *TREC* algorithm which improves over *REC* by considering the two-dimensional correlation structure in contrast to the uni-dimensional correlation approach employed by the former method.

4.3 TREC Algorithm

In this section, we present the *TREC* algorithm which tries to learn the inverse covariance matrix from censored data by imposing row and column-based regularization on the likelihood function. This is called the Transposable REgularized covariance based Calibration (*TREC*) method for censored data. The novelty of this framework lies in interpreting censoring as an imputation problem on the time-to-event variable by modeling its dependence on both row and column based features [29]. The formulation for the log-likelihood function in *TREC* is given by Eq. (6)

$$\begin{aligned} \ell(\upsilon, \mu, \Sigma, \Delta) = &\frac{p}{2}\log|\Sigma^{-1}| + \frac{n}{2}\log|\Delta^{-1}| \\ &- \frac{1}{2}\mathrm{Tr}\Big(\Sigma^{-1}(X - \upsilon 1_{(p)}^T - 1_{(n)}\mu^T)\Delta^{-1}(X - \upsilon 1_{(p)}^T - 1_{(n)}\mu^T)^T\Big) \\ &- \rho_r \parallel \Sigma^{-1} \parallel_{q_r} - \rho_c \parallel \Delta^{-1} \parallel_{q_c}. \end{aligned} \tag{6}$$

In Eq. (6), $\|.\|_{q_r} = \Sigma_{i=1}^{m^2} |.|_{q_r}$ and q_r and q_c are either 1 or 2, which corresponds to either L_1 or L_2 regularizer. We

consider these two choices as they are the most popular regularizers employed. Considering the L_1 norm when q_r are q_c are set to 1 it is observed that solution obtained reaches a stationary point, but it not guaranteed to be the global maximum.

This happens because of the higher number of stationary points when using the L_1 penalty. However, maximization with the L_1 penalties can be achieved by applying the graphical lasso algorithm. This coordinate-wise maximization method used in the graphical lasso leads to a simple iterative algorithm, but it does not necessarily converge to a global maximum.

While considering the L_2 penalty, on the other hand, the problem can be solved by taking the eigenvalue decomposition and a global maximum can be found. This leads to a global maximum, but the solution does not have a simple iterative form as in the case of the L_1 norm. However, in both the cases, we observe that better initialization of the row and column estimates can result in a faster convergence rate.

The optimal way of beginning such assignment is through initializing them with their corresponding MLE estimates for faster convergence. In this regard, we now give the proof for the maximum likelihood estimate (MLE) of the mean parameters.

Theorem 1. *The MLE estimates for υ and μ are*

$$\begin{aligned}\hat{\upsilon} &= \Sigma_{j=1}^{p} \frac{(X_{cj} - \hat{\mu_j})}{p} \\ \hat{\mu} &= \Sigma_{i=1}^{n} \frac{(X_{ir} - \hat{\upsilon_i})}{n}.\end{aligned} \tag{7}$$

Proof. Expanding the trace term of $\ell(M, \Sigma, \Delta)$ w.r.t. μ and υ and then taking partial derivatives, we get

$$\begin{aligned}\frac{\partial \ell}{\partial \upsilon} &= 2\Sigma^{-1}\upsilon 1^T\Delta^{-1} - 2\Sigma^{-1}(X - 1\mu^T)\Delta^{-1} = 0 \\ &\Rightarrow \hat{\upsilon}1^T = X - 1\mu^T \\ &\Rightarrow \hat{\upsilon} = \frac{1^T(X - 1\mu^T)}{p} = \Sigma_{j=1}^{p}\frac{X_{cj} - \mu_j}{p}.\end{aligned}$$

This can be extended in a similar manner to obtain $\hat{\mu}$ as well which ends the proof. □

With these MLE initial estimates derived, we now propose the *TREC* algorithm with the L_1 and L_2 norms as regularizers. The algorithm uses a strategy similar to block coordinate descent by maximizing on one block of coordinates at a given time, thus saving considerable computational time [30]. Conditional maximization (CM) is done with respect to one block of coordinates either Σ^{-1} or Δ^{-1}.

We now put these steps together and present all the details in Algorithm 2. In this Algorithm, we begin by initializing $\hat{\upsilon}$ and $\hat{\mu}$ from the observed uncensored instances using the MLE estimates given in Eq. (7). We then use these values to initialize the time-to-event label and begin the computation as given in Eq. (10).

After convergence, the final values of $\hat{\upsilon}$ and $\hat{\mu}$ are calculated, subsequently T_{imp} is computed through our imputation step. Subsequently, we use the *CompareTimes* procedure to obtain T_{calib} and δ_{calib} which are the final outputs.

We now provide the details of the convergence and complexity of our *TREC* algorithm. The novelty of our framework lies in estimating both the row and column sparse inverse covariance matrices. The complexity associated with each column wise computation is O(np) and this computation over p columns amounts to a O(np^2) time complexity. The resulting optimization problem is convex with respect to each term and it can be efficiently solved using blockwise descent methods.

Algorithm 2. TREC Algorithm

1: **Input**: Features and time matrix X, status δ, regularization parameters ρ_r, ρ_c, q_r, q_c
2: **Output**: Calibrated RC Times T_{calib}, calibrated status δ_{calib}
3: Initialization:
   (a) Estimate $\hat{\upsilon}$ and $\hat{\mu}$ from observed uncensored instances using Eq. (7).
   (b) Initialize time for censored instances as $\hat{\upsilon_i} + \hat{\mu_j}$
   (c) Start with nonsingular estimates $\hat{\Sigma}$ and $\hat{\Delta}$.
   (d) Initalize matrices G, C, F, D.
4: E Step(Δ): Calculate $\hat{X}^T\hat{\Sigma}^{-1}\hat{X} + G(\hat{\Sigma}^{-1})$ as in Eq. (10)
5: M Step(Δ):
   (a) Update estimates of $\hat{\upsilon}$ and $\hat{\mu}$.
   (b) Maximize Q with respect to Δ^{-1} to obtain $\hat{\Delta}$ using gradient as given in Eqs. (11) and (12).
6: E Step(Σ): Calculate $\hat{X}\hat{\Delta}^{-1}\hat{X}^T + F(\hat{\Delta}^{-1})$ as in Eq. (10)
7: M Step(Σ):
   (a) Update estimates of $\hat{\upsilon}$ and $\hat{\mu}$.
   (b) Maximize Q with respect to Σ^{-1} to obtain $\hat{\Sigma}$ using gradient as in Eqs. (11) and (12).
8: Repeat Steps 3-7 until convergence.
9: Estimate imputed censored times (T_{imp}) and compare with original RC times (T_{orig}) using the *CompareTimes* procedure given in Fig. 2.
10: Output calibrated time-to-event variable T_{calib} and calibrated status δ_{calib}.

4.4 Algorithm Analysis

We now develop the steps involved in the blockwise optimization algorithm mathematically, beginning with the observed data log-likelihood which we seek to maximize. We use this term $X^*_{oj,j}$ to condense the likelihood equation to express it in a simpler form as given in Eq. (8)

$$\begin{aligned}X^*_{oj,j} &= \Sigma^{-1/2}_{oj,oj}(X_{oj,j} - \upsilon_{oj}) \\ \ell(\upsilon, \mu, \Sigma, \Delta) &= \frac{1}{2}\left[\Sigma_{j=1}^{p}\log|\Sigma^{-1}_{oj,oj}| + \Sigma_{i=1}^{n}|\Delta^{-1}_{oi,oi}|\right] \\ &\quad - \frac{1}{2}\mathrm{Tr}\left(\Sigma_{i=1}^{n}(X^*_{i,o_i} - \mu_{o_i})^T(X^*_{i,o_i} - \mu_{o_i})\Delta^{-1}_{oi,oi}\right) \\ &\quad - \rho_r \parallel \Sigma^{-1} \parallel_{q_r} - \rho_c \parallel \Delta^{-1} \parallel_{q_c}.\end{aligned} \tag{8}$$

We now derive a simple form to express each of our blockwise steps. One is expressed with respect to Σ^{-1} and the other with respect to Δ^{-1} as in Eq. (10). This is possible

because of the structure of the matrix-variate model, specifically the trace term. The model parameters are represented using $\theta = \{\upsilon, \mu, \Sigma, \Delta\}$. The E step, denoted by $Q(\theta \,|\, \theta^{'}, X_o)$, is expressed in Eq. (9)

$$\begin{aligned} Q(\theta \,|\, \theta^{'}, X_o) &= E(\ell(\upsilon, \mu, \Sigma, \Delta) \,|\, X_o, \theta^{'}) \\ &\propto E[\mathrm{Tr}(X^T\Sigma^{-1}\Delta^{-1}X)|X_o, \theta^{'}] \\ &\propto \mathrm{Tr}[E(X^T\Sigma^{-1}X \,|\, X_o, \theta^{'})\Delta^{-1}] \\ &\propto \mathrm{Tr}[E(X\Delta^{-1}X^T \,|\, X_o, \theta^{'})\Sigma^{-1}]. \end{aligned} \tag{9}$$

We now provide the proof for our proposition for obtaining the simple forms of the conditional maximization step which will be used in our blockwise algorithm.

Proposition 2. *The E step is proportional to the following form*

$$\begin{aligned} &E[\mathrm{Tr}(X^T\Sigma^{-1}X\Delta^{-1}) \,|\, X_o, \theta^{'}] \\ &= \mathrm{Tr}[(\hat{X}^T\Sigma^{-1}\hat{X} + G(\Sigma^{-1}))\Delta^{-1}] \\ &= \mathrm{Tr}[(\hat{X}\Delta^{-1}\hat{X}^T + F(\Delta^{-1}))\Sigma^{-1}] \\ &\text{where } \hat{X} = E(X \,|\, X_o, \theta^{'}) \text{ and} \end{aligned} \tag{10}$$

$$G(\Sigma^{-1}) = \begin{pmatrix} \mathrm{Tr}(C^{(11)}\Sigma^{-1}) & \dots & \mathrm{Tr}(C^{(1p)}\Sigma^{-1}) \\ \vdots & \ddots & \vdots \\ \mathrm{Tr}(C^{(p1)}\Sigma^{-1}) & \dots & \mathrm{Tr}(C^{(pp)}\Sigma^{-1}) \end{pmatrix}$$

$$F(\Delta^{-1}) = \begin{pmatrix} \mathrm{Tr}(D^{(11)}\Delta^{-1}) & \dots & \mathrm{Tr}(D^{(1n)}\Delta^{-1}) \\ \vdots & \ddots & \vdots \\ \mathrm{Tr}(D^{(n1)}\Delta^{-1}) & \dots & \mathrm{Tr}(D^{(nn)}\Delta^{-1}) \end{pmatrix}$$

$$C^{(jj^{'})} = Cov(X_{cj}, X_{cj^{'}} \,|\, X_o, \theta^{'})$$

$$D^{(ii^{'})} = Cov(X_{ir}, X_{i^{'}r} \,|\, X_o, \theta^{'}).$$

We now present the proof for this proposition

Theorem 2. *We first show that*

$$\begin{aligned} E[\mathrm{Tr}(X^T\Sigma^{-1}X\Delta^{-1})|X_o, \theta^{'}] = \mathrm{Tr}[(\hat{X}^T\Sigma^{-1}\hat{X} \\ + G(\Sigma^{-1}))\Delta^{-1}]. \end{aligned}$$

Proof. Let $A = X^T\Sigma^{-1}X$, then,

$$E[\mathrm{Tr}(X^T\Sigma^{-1}X\Delta^{-1}) \,|\, X_o, \theta^{'}] = \mathrm{Tr}[E(A|X_o, \theta^{'}), \Delta^{-1}]$$

$$\begin{aligned} E(A_{jj^{'}}|X_o, \theta^{'}) &= E(X_{cj}^T\Sigma^{-1}X_{cj^{'}}|X_o, \theta^{'}) \\ &= E\left[\sum_{k=1}^{n}\sum_{t=1}^{n} X_{tj}X_{kj^{'}}\sigma_{tk}^{-1}|X_o, \theta^{'}\right] \\ &= \sum_{k=1}^{n}\sum_{t=1}^{n} \hat{X}_{tj}\hat{X}_{kj^{'}}\sigma_{tk}^{-1} + \sum_{k=1}^{n}\sum_{t=1}^{n} C_{tk}^{(jj^{'})}\sigma_{tk}^{-1} \\ &= \hat{X}_{cj}^T\Sigma^{-1}\hat{X}_{cj^{'}} + \mathrm{Tr}(C^{(jj^{'})}\Sigma^{-1}). \end{aligned}$$

Thus, $E(A|X_o, \theta^{'}) = \hat{X}^T\Sigma^{-1}\hat{X} + G(\Sigma^{-1})$
The proof showing

$$\begin{aligned} &E[\mathrm{Tr}(X^T\Sigma^{-1}X\Delta^{-1}) \,|\, X_o, \theta^{'}] \\ &= \mathrm{Tr}[(\hat{X}^T\Sigma^{-1}\hat{X} + F(\Delta^{-1}))\Sigma^{-1}], \end{aligned}$$

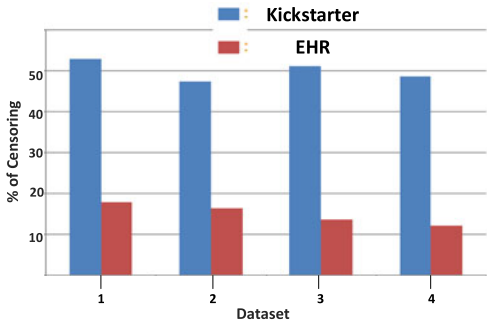


Fig. 3. Percentage of right censored instances in EHR and Kickstarter datasets.

is similar to the calculation above with $B = X\Delta^{-1}X^T$

$$E(B_{ii}|X_o, \theta^{'}) = \hat{X}_{ir}\Delta^{-1}\hat{X}_{i^{'}r}^T + \mathrm{Tr}(D^{(ii^{'})}\Delta^{-1}).$$

□

We now present the gradient equations which are being used in *TREC* with the L_1 and L_2 norms in Eqs. (11) and (12)

$$\begin{aligned} \frac{\partial Q}{\partial \Delta^{-1}} &= \Delta - \frac{\hat{X}^T\Sigma^{-1}\hat{X} + G(\Sigma^{-1})}{n} - \frac{2\rho_c}{n}sgn(\Delta^{-1}) \\ \frac{\partial Q}{\partial \Sigma^{-1}} &= \Sigma - \frac{\hat{X}\Delta^{-1}\hat{X}^T + F(\Delta^{-1})}{p} - \frac{2\rho_r}{p}sgn(\Sigma^{-1}). \end{aligned} \tag{11}$$

We use a notation now through the remainder of this paper to represent the regularizer being used in *TREC*. L_1-*TREC* represents using the L_1 norm in *TREC*. The same notation can be extended to the L_2-*TREC* algorithm

$$\begin{aligned} \frac{\partial Q}{\partial \Delta^{-1}} &= \Delta - \frac{\hat{X}^T\Sigma^{-1}\hat{X} + G(\Sigma^{-1})}{n} - \frac{4\rho_c}{n}\Delta^{-1} \\ \frac{\partial Q}{\partial \Sigma^{-1}} &= \Sigma - \frac{\hat{X}\Delta^{-1}\hat{X}^T + F(\Delta^{-1})}{p} - \frac{4\rho_r}{p}\Sigma^{-1}. \end{aligned} \tag{12}$$

5 Experimental Results

In this section, we present the experimental results obtained using the proposed *REC*, *TREC* methods for calibrated survival analysis on EHRs, Crowdfunding (Kickstarter) [31] and synthetic datasets. In Fig. 3, we present a bar graph which plots the censored statistics for the kickstarter and EHRs. One can clearly observe that the distribution of right censored instances is higher for the kickstarter data compared to the EHRs, which is an important characteristic of the data collected from the crowdfunding domain.

In this section, we will discuss the data collection and pre-processing steps for the EHRs and kickstarter datasets. We conduct various experiments to study the importance of imputing censored instances using our methods. We provide plots which illustrate the improvements obtained in survival regression algorithms after applying our approach. Finally, we also study the effect of both the regularizers and regularization parameters on the runtime performance of our algorithms.

5.1 Dataset Description

We will first describe the various kinds of datasets used in our experiments. This includes the Kickstarter data, EHRs

TABLE 2
Kickstarter Data Statistics for 18,143 Projects

Attr	Mean	Min	Max	StdDev
Goal	26,531	100	100,000,000	758,366
Pledged	11,023	100	6,224,955	78,550
backers	138	1	35,383	633
Days	31	1	60	10.5

TABLE 3
Description of Censored Statistics in the Kickstarter Projects

Name	Startdate	Enddate	# Projects	Censored (%)
Kick 1	1/12/2013	1/1/2013	4175	52.99
Kick 2	1/1/2014	15/3/2014	5229	47.36
Kick 3	16/3/2014	31/4/2014	5720	51.25
Kick 4	1/5/2014	30/6/2014	2969	48.58

and synthetic datasets. We explain the data collection and pre-processing steps involved with each of these datasets.

5.1.1 Crowdfunding Datasets

For the experiments in this paper, we obtained six months of Kickstarter (a popular crowdfunding platform) data from www.kickspy.com. This dataset spans from 12/15/13 to 06/15/14, which consists of projects characterized by 30 project-based attributes. The attributes in the kickstarter datasets include a number of static features such as project goal amount, duration, textual content, etc., and two dynamic features: per-day increase in number of backers and pledged amount as given in Table 2. In this manner, a total of 18,143 projects with over 1 million backers were obtained and processed using the procedures followed in [32]. The attribute used to determine the censoring in the kickstarter datasets is the duration of the project.

Each project in our kickstarter database is tracked over a period of time until either its goal date is reached or it obtains the goal amount. If a project reaches its goal amount (event in this scenario) in a specified duration (time-to-event) this is measured as a success. However, failure to reach the specified goal amount by the end of the study would imply that the instance has been censored (possibly attains the goal amount at a later time). With this notion of censoring, we present the percentage of censored instances in kickstarter data in Table 3.

5.1.2 Electronic Health Records (EHRs)

We now provide the description for the EHRs considered. These datasets are longitudinal EHRs for patients admitted at the Henry Ford Health System, Detroit, Michigan over a period of 10 years. The event here is heart failure readmission and the duration is measured after the patient has been discharged from primary index hospitalization.

The statistics with the right censored percentages are provided for 5 of our sample datasets in Table 4. Readm-index represents the index of readmission for the patients. EHR0 is for data for the index hospitalization. Similarly, EHRn represents the dataset for the n^{th} rehospitalization for the patient set considered. It should be noted that as n increases the number of patients will be reduced.

5.1.3 Synthetic Datasets

We generate synthetic datasets by setting the pairwise correlation between any pair of covariates to vary from -0.5 to 0.5. Feature vectors of different dimensionality are generated to construct three synthetic datasets. For each of these synthetic datasets, the generated failure times T are generated through a Weibull AFT model.

We compare the effect of calibrated survival analysis on any given dataset before and after applying it, by evaluating the performance using a standard survival learner. In our experiments, for each dataset, we create a new one after applying *TREC* based calibration and this is labeled as *With*, and the version before applying *TREC* based calibration is labeled as *Without*. We use this notation throughout this section.

TABLE 4
Basic Statistics for EHRs

Readm	Rows	Columns	Censored (%)
EHR 0	4417	77	22.20
EHR 1	3410	77	17.98
EHR 2	2749	77	16.44
EHR 3	2209	77	13.63
EHR 4	1801	76	12.05

5.2 Performance Evaluation

We will now describe the evaluation metrics used in this work along with some of the implementation details for both the algorithms proposed in this paper as well as details pertaining to baseline comparison algorithms.

5.2.1 Evaluation Metrics

In this section, we explain the evaluation metrics used for our experimental results. Popular metrics used in survival analysis, such as time-based AUC and survival AUC aim at evaluating the relative risk of a event for two instances, than predicting the absolute survival times for these instances. These metrics are introduced below

$$\begin{aligned}\mathrm{AUC}(T_c) &= P(\hat{Y}_i < \hat{Y}_j | Y_i < T_c, Y_j > T_c) \\ &= \frac{1}{num(T_c)} \sum_{Y_i < T_c} \sum_{Y_j > T_c} \mathrm{I}(\hat{Y}_i < \hat{Y}_j). \end{aligned} \tag{13}$$

In Eq. (13), we define the time-based AUC estimated at any given time T_c. $num(T_c)$ denotes the number of comparable pairs at time T_c and I is an indicator function. AUC(T_c) can be used to define the Survival AUC metric which measures the weighted average of the time-based AUC as given in Eq. (14). In this equation, T_e represents the set of all possible event times in the dataset, and num represents the cumulative number of comparable pairs calculated over all event times

$$\text{Survival AUC} = \frac{1}{num} \sum_{T_c \in T_e} AUC(T_c) \cdot num(T_c). \tag{14}$$

5.2.2 Implementation Details

We implemented both *REC* and *TREC* in the R programming language. As mentioned earlier we implemented the

versions corresponding to both the L_1 and L_2 norms in *TREC*. The glasso R package was used for solving the graphical lasso problem for solving the corresponding subproblems in *REC* and *TREC*. The iterative blockwise gradient descent algorithm was implemented as the main optimization routine for solving *TREC*. The corresponding parameters for regularization in *REC* and *TREC* were determined through five-fold cross validation.

In this section, we will refer to L_2-*TREC* as *TREC*, and this has been used for obtaining the results in this section. As mentioned earlier, we prefer the L_2 norm as it gives us a global maximum compared to the L_1 norm. So all the calibrated datasets have been generated using the L_2-*TREC* and *REC* algorithms. We reiterate explicitly that if a regularizer is not mentioned with *TREC*, then it is assumed to be the L_2-*TREC* algorithm itself. As *REC* uses the L_1 norm alone, we do not specify the norm explicitly, and it assumed that *REC* refers to using the L_1 norm formulation only.

We now briefly discuss the implementation details pertaining to baseline comparison algorithms. The software for CensNB is available at[1] and we used our code for KEN-COX and OSCAR-COX [13]. The randomForestSRC and CoxNet R packages are used for running random survival forests and EN-COX, respectively.

We now briefly explain the baseline imputation algorithms used for comparing the performance of *REC* and *TREC*. The first baseline algorithm is SoftImpute which is a method which uses the nuclear norm regularizer and iteratively replaces the missing elements with those obtained from a soft thresholded singular value decomposition (SVD). It tries to minimize the nuclear norm subject to certain constraints [25].

The other baseline method is Misglasso which is a method that replaces the missing values using the standard graphical lasso by modifying the update step in the EM iteration [24]. We implement the misglasso algorithm by using the graphical lasso R package (glasso). The softImpute R package is used for the SoftImpute algorithm. The code for *REC* and *TREC* algorithms is available here[2].

5.3 Integrating TREC with Survival Regression Algorithms

In this section, we present results which demonstrate the robustness of *TREC* algorithm on several datasets. We do not report the results obtained after applying *REC* as this is simply a part of the *TREC* framework, and we do not want to highlight this as two different contributions while presenting the results. *REC* is more simpler in terms of formulation and obtaining a solution compared to *TREC*. However, *TREC* is more robust in terms of performance which will be shown in this section. The baseline algorithms used in for comparison in this section are

(1) *Elastic net Cox (EN-COX)* [12]: EN-COX integrates the elastic net penalty with the Cox partial log-likelihood loss function to deal with correlated features in survival data.
(2) *Kernel elastic net Cox (KEN-COX)* [13]: KEN-COX supplements EN-COX with an additional feature kernel term to capture more feature correlation among the survival covariates.
(3) *Oscar Cox (OSCAR-COX)* [13]: This method uses the Octagonal Shrinkage Clustering Algorithm for Regression (OSCAR) [33] as a regularizer along with the Cox partial log-likelihood loss function to capture feature grouping among survival covariates.
(4) *CoxBoost* [34]: This is an extension of Cox regression which uses the boosting method to create an ensemble of learners.
(5) *Censored Naive Bayes (CensNB)* [16]: This is a bayesian based approach which uses inverse probability weighted censoring mechanism to obtaining probability estimates for prediction.
(6) *Random Survival Forests (RSF)* [35]: RSF and CoxBoost are ensemble based methods which use survival trees and boosting for prediction, respectively.
(7) *Boosted Concordance Index (BoostCI)* [36]: This approach optimizes the concordance index directly to build an effective regression model in contrast to other maximum likelihood based approaches such as Cox regression.
(8) *Elastic net Buckley James (EN-BJ)* [27]: EN-BJ uses a semi-parametric accelerated failure time (AFT) model with elastic net regularization.

The results from Tables 5 and 6 indicate that when *TREC* is applied on the censored dataset (*With*), the survival regression algorithm is able to yield a better performance in comparison to using the original right censored dataset (*Without*). We attribute this better performance to the fact that *TREC* models the censored missing time-to-event values using a row and column regularization method which infers the correlation patterns among censored instances which is needed to impute the time-to-event labels for RC instances correctly. The improvements in survival AUC values are prominent with both Cox regression-based algorithms as given in Table 5, and other survival algorithms as given in Table 6. These improvements also confirm that the performance of our approach does not depend on using any specific kind of survival regression algorithm.

In addition, in this experiment, we also report the p-values measuring whether the difference between the model built on the calibrated data using *TREC* differs significantly from the model built without calibration. We report these p-values in brackets next to the concordance index. These are reported for two algorithms here, namely, CoxBoost and EN-BJ only as the performance of our pre-processing approach does not depend the algorithm being used for prediction. This value is calculated by comparing two concordance indices using this method [37] and estimating if their difference is statistically significant by calculating the corresponding p-values. Using these metrics the better performing model among the *With* and *Without* calibrated datasets is marked in bold in Tables 5 and 6 for each algorithm. The p-values in Tables 5 and 6 indicate that survival models built on the calibrated data are more robust as indicated by the overall low p-values. p-values ≤ 0.05 are considered to be good enough to show the statistical significance of the results obtained using our methods. A good survival

1. https://sites.google.com/a/umn.edu/jwolfson/software
2. https://github.com/MLSurvival/survutils

TABLE 5
Comparison of Survival AUC Values with Standard Deviation (std) and p-Values for Different Cox Regression Algorithms without and with (*TREC*) Applied on Kickstarter, EHR, and Synthetic Censored Datasets

Dataset	EN-COX		KEN-COX		OSCAR-COX		CoxBoost	
	Without	With	Without	With	Without	With	Without	With (p-value)
Kick 1	0.812 (0.071)	**0.835 (0.044)**	0.794 (0.011)	**0.803 (0.027)**	0.811 (0.093)	**0.843 (0.064)**	0.831 (0.022)	**0.866 (0.039) (0.075)**
Kick 2	0.811 (0.132)	**0.865 (0.048)**	0.819 (0.031)	**0.841 (0.073)**	0.832 (0.045)	**0.874 (0.031)**	0.803 (0.045)	**0.825 (7e-9) (0.071)**
Kick 3	0.807 (0.049)	**0.820 (0.033)**	0.793 (0.104)	**0.833 (0.079)**	0.814 (0.022)	**0.833 (0.067)**	0.793 (0.097)	**0.827 (6e-15) (0.019)**
Kick 4	0.773 (0.088)	**0.817 (0.045)**	0.782 (0.091)	**0.811 (0.037)**	0.821 (0.102)	**0.853 (0.076)**	0.811 (0.050)	**0.833 (4e-7) (0.061)**
EHR 0	0.585 (0.094)	**0.606 (0.113)**	**0.618 (0.021)**	0.605 (0.065)	0.632 (0.016)	**0.643 (0.025)**	0.631 (0.033)	**0.642 (0.041) (0.089)**
EHR 1	0.592 0.106)	**0.609 (0.041)**	0.611 (0.087)	**0.637 (0.055)**	0.629 (0.031)	**0.655 (0.074)**	0.622 (0.011)	**0.625 (3e-14) (0.061)**
EHR 2	0.598 (0.082)	**0.605 (0.041)**	**0.624 (0.038)**	0.611 (0.091)	0.618 (0.028)	**0.599 (0.116)**	0.637 (0.081)	**0.665 (5e-16) (0.064)**
EHR 3	0.581 0.016)	**0.595 (0.019)**	0.611 (0.058)	**0.639 (0.022)**	0.607 (0.030)	**0.626 (0.084)**	0.631 (0.011)	**0.648 (9e-21) (0.082)**
EHR 4	0.618 (0.043)	**0.633 (0.096)**	0.641 (0.088)	**0.655 (0.101)**	0.644 (0.041)	**0.661 (0.037)**	0.689 (0.021)	**0.675 (1e-8) (0.066)**
Syn 1	0.668 (0.027)	**0.673 (0.074)**	0.681 (0.065)	**0.699 (0.032)**	0.677 (0.072)	**0.664 (0.121)**	0.693 (0.084)	**0.715 (2e-7) (0.033)**
Syn 2	0.872 (0.039)	**0.902 (0.104)**	0.890 (0.088)	**0.910 (0.057)**	0.927 (0.018)	**0.943 (0.041)**	0.872 (0.032)	**0.933 (0.009) (0.017)**
Syn 3	0.727 (0.096)	**0.719 (0.041)**	0.785 (0.016)	**0.854 (0.041)**	0.887 (0.043)	**0.931 (0.109)**	0.856 (0.037)	**0.922 (4e-10) (0.029)**

TABLE 6
Comparison of Survival AUC Values with Standard Deviation (std) and p-Values for CensNB, RSF, BoostCI, and EN-BJ Algorithms without and with (*TREC*) Applied on Kickstarter, EHR, and Synthetic Censored Datasets

Dataset	CensNB		RSF		BoostCI		EN-BJ	
	Without	With	Without	With	Without	With	Without	With (p-value}
Kick 1	0.771 (0.066)	**0.804 (0.089)**	**0.802 (0.069)**	0.799 (0.088)	0.785 (0.113)	**0.803 (0.045)**	0.818 (0.029)	**0.824 (0.017) (0.054)**
Kick 2	0.783 (0.145)	**0.809 (0.071)**	0.811 (0.044)	**0.833 (0.077)**	**0.743 (0.082)**	0.733 (0.103)	0.853 (0.077)	**0.879 (0.031) (0.141)**
Kick 3	**0.761 (0.002)**	0.753 (0.087)	0.732 (0.043)	**0.758 (0.021)**	**0.763 (0.089)**	0.725 (0.088)	0.835 (0.022)	**0.841 (1e-4) (0.071)**
Kick 4	0.722 (0.013)	**0.773 (0.091)**	0.796 (0.057)	**0.812 (0.028)**	0.722 (0.097)	**0.744 (0.114)**	0.839 (0.102)	**0.851 (0.019) (0.064)**
EHR 0	0.572 (0.037)	**0.591 (0.078)**	0.599 (0.069)	**0.611 (0.071)**	0.517 (0.093)	**0.552 (0.064)**	0.594 (0.033)	**0.601 (1e-26) (0.081)**
EHR 1	0.575 (0.092)	**0.588 (0.044)**	0.583 (0.117)	**0.609 (0.081)**	0.543 (0.025)	**0.565 (0.071)**	0.571 (0.062)	**0.604 (4e-10) (0.110)**
EHR 2	0.573 (0.044)	**0.606 (0.097)**	0.581 (0.118)	**0.591 (0.039)**	0.575 (0.068)	**0.591 (0.013)**	0.566 (0.034)	**0.601 (7e-12) (0.045)**
EHR 3	0.609 (0.177)	**0.631 (0.085)**	0.611 (0.035)	**0.636 (0.087)**	0.626 (0.041)	**0.639 (0.045)**	0.593 (0.025)	**0.614 (3e-16) (0.088)**
EHR 4	0.621 (0.034)	**0.659 (0.082)**	**0.633 (0.069)**	0.627 (0.076)	0.665 (0.901)	**0.693 (0.119)**	0.638 (0.081)	**0.644 (1e-9) (0.067)**
Syn 1	0.654 (0.099)	**0.661 (0.133)**	0.633 (0.078)	**0.642 (0.091)**	0.643 (0.125)	**0.679 (0.188)**	0.641 (0.105)	**0.669 (2e-10) (0.087)**
Syn 2	0.847 (0.109)	**0.867 (0.086)**	0.852 (0.122)	**0.905 (0.076)**	0.836 (0.065)	**0.896 (0.018)**	0.866 (0.026)	**0.875 (3e-8) (0.004)**
Syn 3	0.714 (0.096)	**0.764 (0.155)**	0.834 (0.071)	**0.841 (0.033)**	0.740 (0.027)	**0.748 (0.164)**	0.780 (0.019)	**0.799 (1e-4) (0.031)**

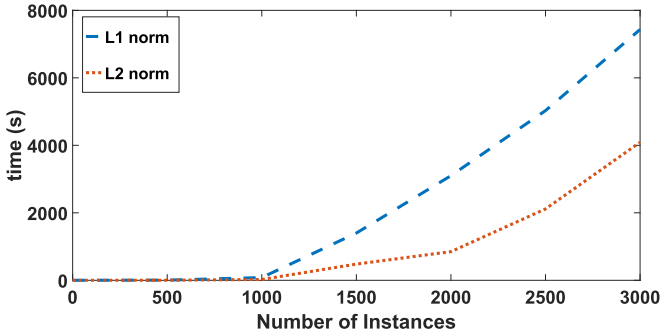


Fig. 4. Runtime on kickstarter dataset using L_1, L_2-norms in *TREC*.

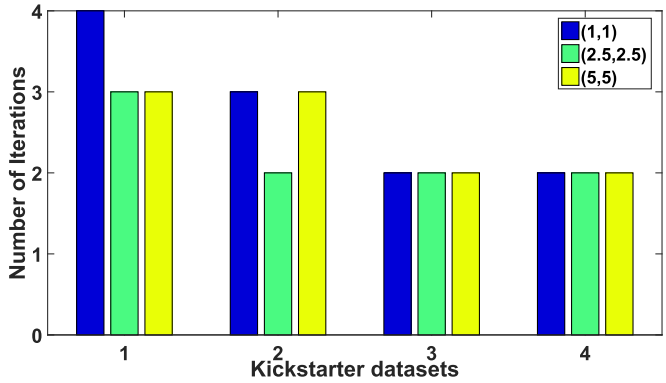


Fig. 5. Iterations for convergence using L_2-norm based *TREC*.

model will have high concordance index and low p-values as illustrated in the results from Tables 5 and 6. We attribute the good performance of these algorithms due to our effective calibration technique which enhances the predictive power of survival algorithms.

5.4 Impact of Regularizers and Regularization Parameters on TREC Algorithm

In this section, we study the influence of the row and column regularizers and parameters on the convergence and runtime of the *TREC* algorithm. We study the runtime using both L_1 and L_2 regularizers in *TREC* to assess their time efficiency. We use one of the kickstarter datasets (Kick 1) for this experiment. The values of the row and column regularization parameters were obtained using cross validation for this dataset.

In Fig. 4, we plot the runtime in seconds on the *Y*-axis, and the number of instances sampled from Kick 1 dataset are labeled on the *X*-axis. We run both our L_1 and L_2 norm based *TREC* algorithms separately to measure their runtime. We can observe that among the two norms L_2-norm is more time efficient compared to the L_1-norm. The L_1 norm uses the graphical lasso solver and the higher number of stationary points observed in this formulation results in higher runtime to obtain convergence. This makes the L_2 norm the more efficient regularizer due to better scalability. However, the L_2 norm does not provide sparse solutions with respect to the inverse covariance matrix estimated which affects the interpretability of the solution when dealing with high-dimensional datasets. So there is a trade-off between choosing the L_1 and L_2-norms.

In another experiment, we also study the impact of the choice of the regularization parameters on the convergence of *TREC*. In Fig. 5, the *X*-axis represents the indices of the four kickstarter datasets used in this paper. The *Y*-axis represents the number of iterations needed for *TREC* to converge for each dataset using three sets of regularization parameter values. The legend indicates the values chosen for the regularization parameters ρ_r and ρ_c. We observe that the choice of regularization parameters does not affect the convergence, as there is no uniform pattern observed. So these experiments allow us to conclude that the choice of regularizer is important, but the value of these regularization parameters does not affect the convergence of *TREC* significantly. We conducted this analysis also to evaluate the impact of the regularization parameters on the survival AUC values. We observed that the survival AUC values did not change much which indicates that our framework is not sensitive to the row and column regularization parameters in terms of both runtimes and performance.

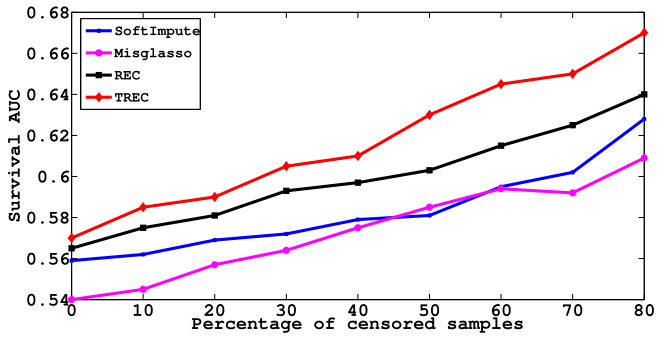
5.5 Improvement in AUC with Imputed Censoring

In this experiment, we plot the survival AUC values of the learning algorithm when we gradually sample instances from the calibrated data (*With*) using different methods for imputing the missing time-to-event values in survival data. This experiment helps us interpret how the calibrated samples are contributing towards building a more efficient model as they are sampled iteratively. The approaches used for imputation in this experiment include SoftImpute [25], Misglasso [24], *REC* and *TREC*. In this experiment, we present the results for the synthetic datasets, kickstarter datasets and EHRs.

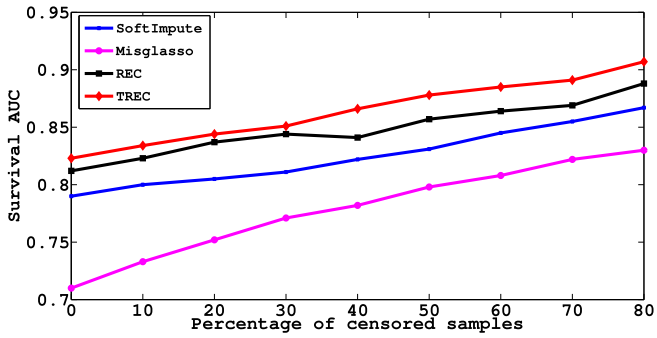
The learning algorithm considered for this experiment was the (EN-COX) algorithm. As determined from the previous experiment, the choice of the learning algorithm was not a part of our approach, so we can choose any arbitrary survival learner. We train the initial survival model using all the uncensored instances, and we continuously sample instances from a pool of censored instances and add them to retrain a survival model. These censored instances have been imputed using *REC* and *TREC*. Simultaneously, we also impute these instances iteratively using SoftImpute and Misglasso before training a new survival model.

As imputed censored instances are added to the training data from the censored pool, we retrain the model and plot the survival AUC values on this combined dataset of the initial set of uncensored instances and the sampled censored instances. From the plots in Fig. 6, we observe that the survival AUC values improve for most of the cases, with the improvements being prominent for *TREC* compared to other competing methods and *REC* stands as the second best method.

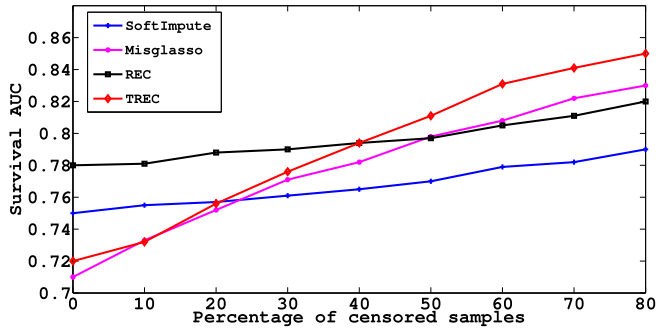
The better performance of *TREC* is because it is effective in interpreting the missing values in the time-to-event labels for censored instances, as it imputes these values considering the two-dimensional correlation structure within the covariance matrix in its formulation. Calibrated time-to-event labels tend to provide the survival model with more discriminative information which is evident from the improvement in the survival AUC values.



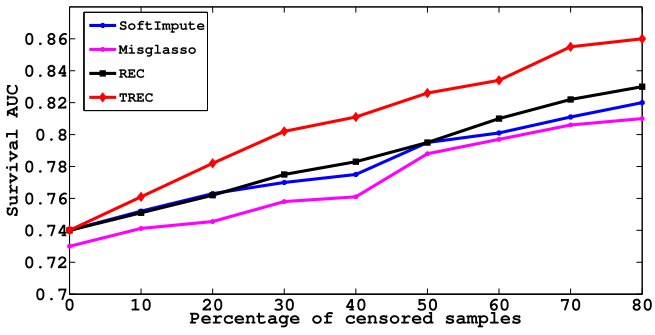
(a) AUC on Syn1



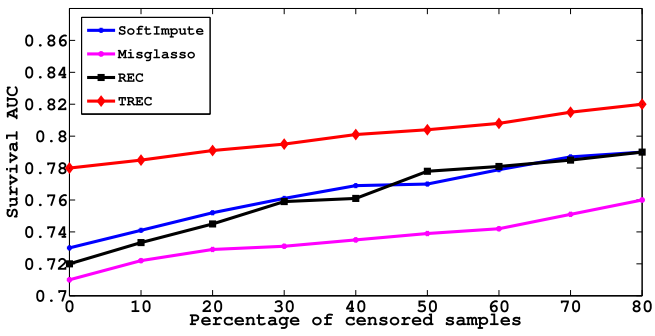
(b) AUC on Syn2



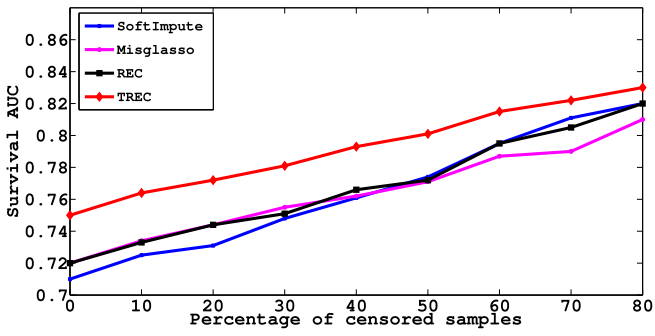
(c) AUC on Kick 1



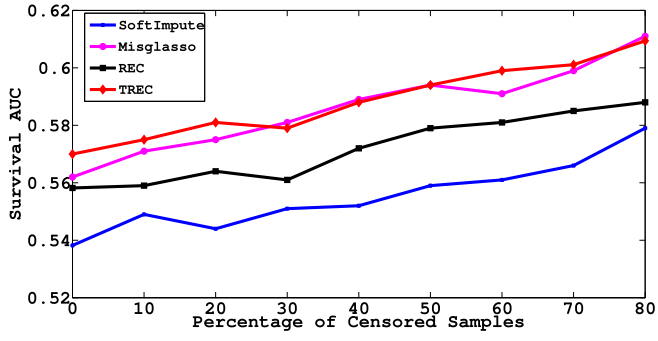
(d) AUC on Kick 2



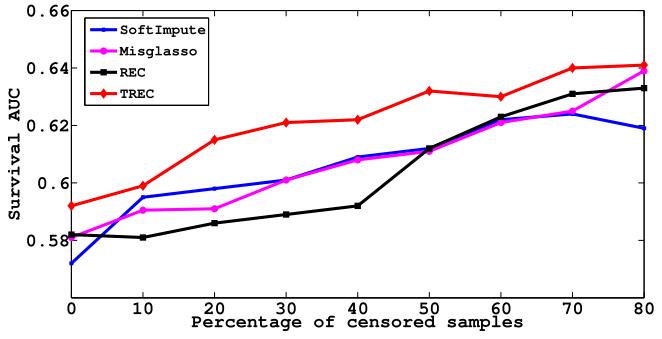
(e) AUC on Kick 3



(f) AUC on Kick 4



(g) AUC on EHR 0



(h) AUC on EHR 1

Fig. 6. Survival AUC values at varying percentages of censored samples obtained for calibrated synthetic, kickstarter, and EHR datasets using REC, TREC, SoftImpute, and Misglasso methods.

6 Conclusion

In this paper, we presented a framework for pre-processing survival data by calibrating the time-to-event labels for right censored instances in the dataset. We motivate the necessity for this application by considering the two-dimensional correlation structure in censored data which needs to be inferred by a method before labeling these censored instances. These methods are very useful in several real-world application problems such as (i) mining clinical data to identify hospital readmissions (ii) following projects in crowdfunding to determine their success. Traditional survival learners cannot be used directly for such data, since the time-to-event label information that is used for censored

instances is incomplete. Erroneous time-to-event labels in such instances could misguide the learning algorithm which is undesirable.

To overcome this problem, we introduce a pre-processing method which makes it easy for a domain expert to convert right censored data to calibrated right censored data which is a more effective representation of the dataset. We studied two methods in this paper, namely, *REC* and *TREC*. *REC* uses the column-based regularization and *TREC* uses a composite row and column-based regularization. The experimental results reveal that both these methods help in improving the survival AUC of algorithms in comparison to other methods. This work can be extended to interval based censoring to identify methods to calibrate censored instances in that domain.

Acknowledgments

This work was supported in part by the National Science Foundation grants IIS-1707498, IIS-1619028, and IIS-1646881. The authors would like to thank Dr. David E. Lanfear from the Henry Ford Health System, Detroit, MI, for providing us with electronic health records for our analysis. They also acknowledge the contributions of Dr. Vineeth Rakesh from Wayne State University for providing us with the crowdfunding datasets.

References

[1] H. Koul, V. Susarla, and J. Van Ryzin, "Regression analysis with randomly right-censored data," *Anna. Statist.*, vol. 9, no. 6, pp. 1276–1288, 1981.
[2] D. R. Cox, "Regression models and life-tables," *J. Roy. Statistical Soc.*, vol. 34, no. 2, pp. 187–220, 1972.
[3] A. Coolen and L. Holmberg, *'Principles of Survival Analysis*. London, U.K.: Oxford Univ. Press, 2013.
[4] P. Sasieni, "Cox regression model," *Encyclopedia Biostatistics*, vol. 1, pp. 1006–1020, 1999.
[5] P. Wang, Y. Li, and C. K. Reddy, "Machine learning for survival analysis: A survey," *ACM Comput. Surveys*, vol. 1, no. 1, pp. 1–38, 2017.
[6] J. P. Klein and M. L. Moeschberger, *Survival Analysis: Techniques for Censored and Truncated Data*. Berlin, Germany: Springer Science & Business Media, 2005.
[7] D. W. Hosmer Jr., S. Lemeshow, and S. May, *Applied Survival Analysis: Regression Modelling of Time to Event Data*. Hoboken, NJ, USA: Wiley, 2008.
[8] C.-F. Chung, P. Schmidt, and A. D. Witte, "Survival analysis: A survey," *J. Quantitative Criminology*, vol. 7, no. 1, pp. 59–98, 1991.
[9] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[10] M. Gönen and G. Heller, "Concordance probability and discriminatory power in proportional hazards regression," *Biometrika*, vol. 92, no. 4, pp. 965–970, 2005.
[11] R. Tibshirani, "The lasso method for variable selection in the cox model," *Statist. Med.*, vol. 16, no. 4, pp. 385–395, 1997.
[12] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for Cox's proportional hazards model via coordinate descent," *J. Statistical Softw.*, vol. 39, no. 5, 2011, Art. no. 1.
[13] B. Vinzamuri and C. K. Reddy, "Cox regression with correlation based regularization for electronic health records," in *Proc. IEEE 13th Int. Conf. Data Mining*, 2013, pp. 757–766.
[14] W. Zhang, T. Ota, V. Shridhar, J. Chien, B. Wu, and R. Kuang, "Network-based survival analysis reveals subnetwork signatures for predicting outcomes of ovarian cancer treatment," *PLoS Comput. Biol*, vol. 9, no. 3, pp. 1–16, 2013.
[15] B. Vinzamuri, Y. Li, and C. K. Reddy, "Active learning based survival regression for censored data," in *Proc. 23rd ACM Int. Conf. Inf. Knowl. Manag.*, 2014, pp. 241–250.
[16] J. Wolfson, et al., "A naive bayes machine learning approach to risk prediction using censored, time-to-event data," *Statisti. Med.*, vol. 34, no. 21, pp. 2941–2957, 2015.
[17] P. M. Rancoita, M. Zaffalon, E. Zucca, F. Bertoni, and C. P. De Campos, "Bayesian network data imputation with application to survival tree analysis," *Comput. Statist. Data Anal.*, vol. 93, pp. 373–387, 2016.
[18] M. J. Fard, P. Wang, S. Chawla, and C. K. Reddy, "A bayesian perspective on early stage event prediction in longitudinal data," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 12, pp. 3126–3138, Dec. 2016.
[19] W. Pan, "A multiple imputation approach to cox regression with interval-censored data," *Biometrics*, vol. 56, no. 1, pp. 199–203, 2000.
[20] J. Wang, S. Faridani, and P. Ipeirotis, "Estimating the completion time of crowdsourced tasks using survival analysis models," *Crowdsourcing Search Data Mining*, vol. 31, pp. 31–34, 2011.
[21] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.
[22] P. J. Green, "On use of the EM for penalized likelihood estimation," *J. Royal Statistical Soc. Series B (Methodological)*, vol. 52, no. 3, pp. 443–452, 1990.
[23] X.-L. Meng and D. B. Rubin, "Maximum likelihood estimation via the ECM algorithm: A general framework," *Biometrika*, vol. 80, no. 2, pp. 267–278, 1993.
[24] N. Städler and P. Bühlmann, "Missing values: Sparse inverse covariance estimation and an extension to sparse regression," *Statist. Comput.*, vol. 22, no. 1, pp. 219–235, 2012.
[25] R. Mazumder, T. Hastie, and R. Tibshirani, "Spectral regularization algorithms for learning large incomplete matrices," *J. Mach. Learn. Res.*, vol. 11, no. 10, pp. 2287–2322, 2010.
[26] C. L. Faucett, N. Schenker, and J. M. Taylor, "Survival analysis using auxiliary variables via multiple imputation, with application to aids clinical trial data," *Biometrics*, vol. 58, no. 1, pp. 37–47, 2002.
[27] S. Wang, B. Nan, J. Zhu, and D. G. Beer, "Doubly penalized buckley–james method for survival data with high-dimensional covariates," *Biometrics*, vol. 64, no. 1, pp. 132–140, 2008.
[28] B. Efron, "Are a set of microarrays independent of each other?" *Annal. Appl. Statist.*, vol. 3, no. 3, pp. 922–942, 2009.
[29] G. I. Allen and R. Tibshirani, "Transposable regularized covariance models with an application to missing data imputation," *Annal. Appl. Statist.*, vol. 4, no. 2, pp. 764–790, 2010.
[30] D. M. Witten, R. Tibshirani, and T. Hastie, "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis," *Biostatistics*, vol. 10, no. 3, pp. 515–534, 2009.
[31] Y. Li, V. Rakesh, and C. K. Reddy, "Project success prediction in crowdfunding environments," in *Proc. 9th ACM Int. Conf. Web Search Data Mining*, pp. 247–256. 2016.
[32] V. Rakesh, J. Choo, and C. K. Reddy, "Project recommendation using heterogeneous traits in crowdfunding," in *Proc. 9th Int. AAAI Conf. Web Social Media*, 2015, pp. 337–346.
[33] H. D. Bondell and B. J. Reich, "Simultaneous regression shrinkage, variable selection, and supervised clustering of predictors with oscar," *Biometrics*, vol. 64, no. 1, pp. 115–123, 2008.
[34] H. Binder, "CoxBoost: Cox models by likelihood based boosting for a single survival endpoint or competing risks," *R Package Version*, vol. 1, pp. 1778–1791, 2013.
[35] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer, "Random survival forests," *Annal. Appl. Statist.*, vol. 2, no. 3, pp. 841–860, 2008.
[36] A. Mayr and M. Schmid, "Boosting the concordance index for survival data–a unified framework to derive and evaluate biomarker combinations," *PloS ONE*, vol. 9, no. 1, pp. 834–843, 2014.
[37] L. Kang, W. Chen, N. A. Petrick, and B. D. Gallas, "Comparing two correlated C-indices with right-censored survival outcome: A one-shot nonparametric approach," *Statist. Med.*, vol. 34, no. 4, pp. 685–703, 2015.

Bhanukiran Vinzamuri received the BTech and MS degrees in computer science from the International Institute of Information Technology, Hyderabad (IIIT-H) and the PhD degree in computer science and engineering from Wayne State University. He is a postdoctoral research scientist in the Data Science Department, IBM Thomas J. Watson Research Center, Yorktown Heights, New York. His research interests include machine learning, biostatistics, and healthcare analytics. His research has been published in leading conferences and journals such as ICDM, CIKM, SDM, and *Data Mining and Knowledge Discovery*.

Yan Li received the BS degree from Xidian University, and the MS and PhD degrees from Wayne State University. He is a Postdoc fellow in the Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor. His primary research interests include data mining and machine learning with applications to healthcare analytics and bioinformatics. His research works have been published in leading conferences and journals including Special Interest Group on Knowledge Discovery in Data, ICDM, WSDM, SDM, CIKM, *Data Mining and Knowledge Discovery*, and *Information Science*.

Chandan K. Reddy received the MS degree from Michigan State University and the PhD degree from Cornell University. He is an associate professor in the Department of Computer Science, Virginia Tech. His primary research interests include the areas of data mining, machine learning, and big data with applications to healthcare, social network analysis, and bioinformatics. His research is funded by the US National Science Foundation, NIH, DOT, and Blue Cross Blue Shield. He has published more than 90 peer-reviewed articles in leading conferences and journals. He received several awards for his research work including the Best Application Paper Award at the ACM SIGKDD conference, in 2010, Best Poster Award at the IEEE VAST conference, in 2014, Best Student Paper Award at the IEEE ICDM conference, in 2016, and was a finalist of the INFORMS Franz Edelman Award Competition in 2011. He is a senior member of the IEEE and a life member of the ACM.