Imperial College London

MEng Mathemtatics and Computer Science (Computational Statistics) Individual Project

Imperial College London

Department of Computing

Apply Machine Learning Approaches to Survival Data

Author:
Jingya Wang

Supervisor:
Dr Matt Williams

Second Marker:
Dr Erisa Karafili

2017 - 2018

Abstract

A number of machine learning methods have been developed for survival analysis, aiming to predict cancer susceptibility, recurrence and survival. However, many do not outperform the most widely used statistical method - Cox proportional hazard model. This project first investigates three existing survival models - Cox proportional hazard model, random survival forest and Multi-Task Learning for Survival Analysis (MTLSA) , on five real-world low-dimensional datasets. Upon the failure of MTLSA, I proposed linear MTLSA and MTL-ANN in both non-linear and linear forms to resolve drawbacks in MTLSA. Based on the available data, experiments suggest that a linear model is likely to be sufficient for survival analysis.

Acknowledgements

I would like to express my gratitude to my supervisor, Matt, for his guidance, encouragement, valuable suggestions and time throughout the project.

Contents

Chapter 1

Introduction

Cancer is one of the most deadliest diseases in the world. "More than a third of people in the UK fear cancer more than other life-threatening conditions – such as Alzheimer's, stroke and heart disease according to a Cancer Research UK survey [1]". According to statistics by Cancer Research UK [2] in Figure 1.1, there were 359, 960 new cases of cancer in UK in 2015, cancer has caused 163, 444 deaths in UK in 2016, and its survival rate for ten or more years is about 50% in England and Wales. Cancer is a life threatening disease and we all want to beat it sooner.

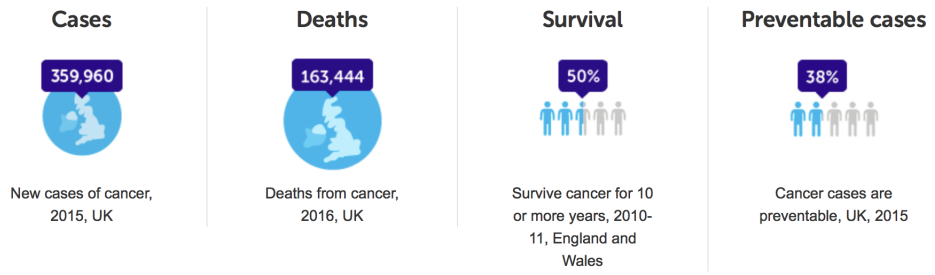

Figure 1.1: Cancer statistics for the UK [2]

But until we are close to curing cancer, 'how long do I have?' is probably the most important question for people who know they are dying. Henderson et al. in [3] mentioned three reasons why accurate survival prediction is important. Firstly, "prognostic judgement can influence choice of treatment". There are treatments which have significant side effects and patients may only accept them if they are likely to live long enough to experience subsequent benefit. Secondly, "accurate prediction can be important in the

effective use of limited health care resources". Finally, "accurate prediction may help patients and their families come to terms with impending death and make suitable plans for their remaining lifespan".

Survival analysis is widely applied not only in medical statistics, but also in other fields. For example, retail banks use survival analysis to predict when customers are likely to default on their credits, and website analysis uses it to predict when customers are likely to click on the advertisement. Time of death, default time and the time where customers click on the advertisement are known as events of interest in survival analysis. The most important problem to tackle in survival analysis is censored data, where the event of interest is not observed. In medical studies, a large number of patients are censored. For example, in our datasets, around 50% data is censored data. Simply excluding these censored observations create bias in the model.

In statistical methods in survival analysis, Cox proportional hazard model (Cox PH model) is the most widely used model, along with lots of modifications to deal with problems like tied survival times, time-dependent covariates, feature selection and so on. As machine learning becomes more and more popular these days, classification and regression trees (CART), artificial neural networks (ANN), support vector machine (SVM), to name a few, become mainstream predictive models. By feeding in a training dataset with predictor variables and actual outcomes, we are able to build a machine learning model and use it to predict outcomes on unseen data.

The idea of replacing statistical methods with machine learning methods in survival analysis is tempting. Because for example, in Cox PH model, we need to consider tied survival times and adopt specific methods to deal with the problem. Cox PH model also relies on the assumption that covariates are time-independent, but it is not the case most of the time. In contrast, machine learning methods are not restricted by these problems. They are able to provide great discriminative power or represent complicated non-linear relationship between inputs and outputs, and have proved to be successful in many different fields. Moreover, statistical methods tend to predict survival for a group of patients with similar risk, whereas machine learning methods can provide personalized and predictive medicine [4].

We believe that when censored data is handled in the correct way in machine learning methods, they can have more accurate survival predictions than Cox PH model. A number of machine learning methods have been developed for cancer survival analysis, aiming to predict cancer susceptibil-

ity, recurrence and survival [5]. However, many existing methods do not outperform Cox PH model, possibly because their algorithms still rely on statistical methods in survival analysis. This project aims to investigate existing machine learning methods for survival analysis, propose variants of existing survival models, and compare them with the Cox PH model. I am particularly interested in any methods which do not involve any statistical methods, because they are more likely to outperform the state-of-the-art statistical method in survival analysis.

This project makes the following contributions -

- Identify weaknesses in Multi-Task Learning for Survival Analysis (MTLSA) - a machine learning method for survival analysis which was proposed recently. (Section 4.8)
- Propose linear MTLSA and MTL-ANN to overcome weaknesses in MTLSA. (Chapter 5)
- Assess the performance of six survival models on five real-world low-dimensional datasets. Six survival models are Cox PH model, random survival forests (RSF), MTLSA, linear MTLSA and MTL-ANN in both non-linear and linear forms.
- Experimental results based on the available data suggest that a linear model is likely to be sufficient for survival analysis. (Chapter 6)

In addition, I also give a discussion of the limitations of using concordance index (C-index) in assessing the performance of survival models.

In general, it is believed that MTL-ANN has the potential to outperform state-of-the-art methods because it has demonstrated its predictive power when measuring by integrated Brier score (IBS), it is independent of any statistical methods, its structure is simple and has great flexibility for future extensions.

Chapter 2

Background

2.1 Mathematical Preliminaries

2.1.1 Censoring

A survival time is said to be censored if the exact event time is not observed. There are different types of censoring - right censoring, left censoring, Type I censoring, Type II censoring, Type III censoring and so on.

An event time is right censored if the censoring mechanism prematurely terminates observation of the individual, before the event has actually occurred. For example, a medical study follows a patient after operation for t years, and at the end of the study, the patient is alive. Then we only learn that the lifetime of this patient is greater than t years. Medical study coming to an end, hospital losing track of the patient, or patient dropping out of the study, all can cause right censoring.

An event time is left censored if the event occurred before observation of the individual began. For example, a medical study begins t months after the operation, but only finds that the patient has died. In this case, we only learn that the lifetime of this patient is less than t months.

Observations in medical statistics are right censored in most cases. Furthermore, there are three types of right censoring - Type I, II and III censoring.

Type I censoring occurs if we take n individuals and observe them for a pre-specified time t. Any non-event is right censored with censoring time t. For example, a medical study follows n patients after operation for t years,

and at the end of the study, c patients are alive and $c \leq n$.

Type II censoring occurs if we take n individuals and observe them until m events occur, where $m \leq n$ and m is pre-specified. The rest $(n-m)$ observations are right censored, with censoring time being equal to the maximum exact event time.

Type III censoring [6] occurs when individuals join the study at different times, right censored observations have different durations in the study, and they may withdraw before the end of the study. An example of Type III censoring is shown in Figure 2.1. The x-axis represents the calendar time of the study. The event of interest occurs in observations A, C and E. Observations B, D and F are right censored, where observation B are right censored because it is lost track during the study, and observations D and F are right censored because the study comes to the end.

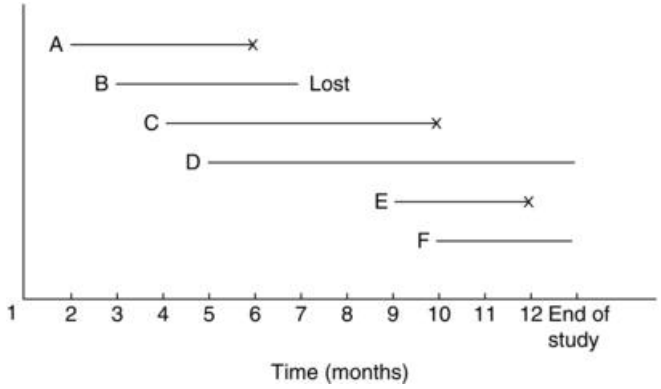

Figure 2.1: Type III censoring [6]. The x-axis denotes the calendar time of the study.

Type III censoring is the case in most clinical studies, and datasets used in this project are Type III censored. Ten observations are randomly sampled from the biggest dataset used in experiments and their survival times are plotted in Figure 2.2 below. It should be noticed that here the x-axis denotes the survival time in days, thus all lines start from zero.

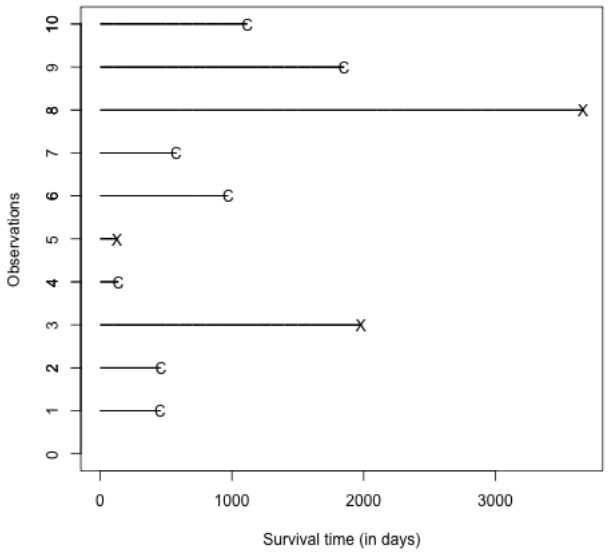

Figure 2.2: Survival times of ten observations from the biggest dataset used in experiments. Lines ending with 'C' mean that observations are right censored, lines ending with 'X' mean that the exact event time is observed.

2.1.2 Continuous Lifetime Survival Analysis

Let T denote the future lifetime of an individual aged 0. T is a continuous random variable which takes values on $\mathbb{R}^+ = [0, \infty)$. For human life calculation, the limiting age is typically 120 years, thus $T \in [0, 120]$.

Definition 1. Cumulative distribution function of T is the probability of death by age t and is defined as

$$F(t) = P(T \leq t) \tag{2.1}$$

Definition 2. Survival function of T is the probability of surviving beyond age t and is defined as

$$S(t) = P(T > t) = 1 - F(t) \tag{2.2}$$

Definition 3. Probability density function of T is defined as below and relationship between $f(t)$ and $S(t)$ is derived from Equation 2.2.

$$f(t) = \lim_{h \downarrow 0} \frac{F(t+h) - F(t)}{h} = \frac{d}{dt} F(t) = -\frac{d}{dt} S(t) \tag{2.3}$$

Definition 4. Hazard function of T, or the force of mortality at age t is defined as below and the relationship between $h(t)$ and $S(t)$ is derived from Equation 2.3.

$$h(t) = \lim_{h \downarrow 0} \frac{P(T \leq t + h | T > t)}{h} = \frac{f(t)}{S(t)} = -\frac{d}{dt} \log S(t) \tag{2.4}$$

Definition 5. The cumulative Hazard rate of T, denoted $H(t)$, is

$$H(t) = \int_0^t h(s) ds \tag{2.5}$$

This leads to another important relationship between $S(t)$ and $H(t)$, which is derived from Equation 2.4 and 2.5.

$$S(t) = \exp\{-H(t)\} \tag{2.6}$$

2.1.3 Discrete Lifetime Survival Analysis

Discrete lifetime survival analysis [7] is critical to understanding some existing machine learning methods for survival analysis and non-parametric statistical methods.

Let T be a discrete random variable with a probability mass function $\pi_j = P(T = a_j)$ for a countable set of values $\{a_1 < a_2 < \cdots\}$ and $\sum_j \pi_j = 1$. Discrete hazard rate at time a_j is

$$h_j = P(T = a_j | T \geq a_j) = \frac{\pi_j}{1 - \sum_{i<j} \pi_i} \tag{2.7}$$

Then by induction,

$$\pi_j = \begin{cases} h_1 & j = 1 \\ h_j \prod_{i<j} (1 - h_i) & j > 1 \end{cases} \tag{2.8}$$

Discrete survival probability can be expressed as

$$S(t) = P(T > a_j) = P(T \geq a_{j+1}) = \frac{\pi_{j+1}}{h_{j+1}} = \prod_{i=1}^{j} (1 - h_i) = \prod_{j: a_j \leq t} (1 - h_j) \tag{2.9}$$

2.2 Statistical Methods

Statistical models in survival analysis can be classified into three categories - parametric models, semi-parametric models, and non-parametric models. Figure 2.3 below provides a comprehensive summary of different statistical methods.

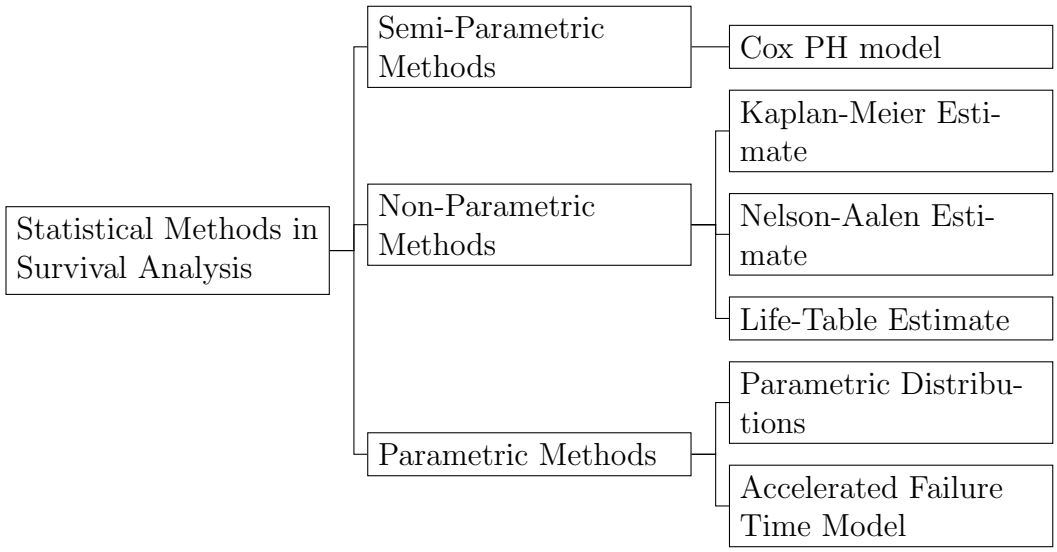

Figure 2.3: Statistical Methods in Survival Analysis

Among all statistical methods, non-parametric methods themselves do not deal with multivariate regression analysis, and parametric methods reply on distribution selection. In contrast, semi-parametric methods should be the best approach, because it is distribution free and can handle multivariate regression problems. Therefore, the most widely used semi-parametric statistical method, Cox PH model, was implemented in this project. But in many existing machine learning methods for survival analysis, non-parametric methods are integrated with standard machine learning methods to deal with censored data and predict survival.

2.2.1 Cox Proportional Hazards Model

Cox PH model is given by

$$h(t, \mathbf{x}(t)) = h_0(t) \exp(\boldsymbol{\beta} \cdot \mathbf{x}(t)) \tag{2.10}$$

where $\mathbf{x}(t)$ are expressed as time-dependent predictor variables, $\boldsymbol{\beta}$ denotes coefficients of predictor variables, and $h_0(t)$ is called the baseline hazard

function, and it is non-parametric. Both $\boldsymbol{\beta}$ and $h_0(t)$ are unknown. However, $h_0(t)$ can be cancelled out when time-dependent covariates are not included. Cox PH model can be rewritten as

$$h(t, \mathbf{x}) = h_0(t) \exp(\boldsymbol{\beta} \cdot \mathbf{x}) \tag{2.11}$$

where $\mathbf{x}$ is assumed to be independent of time. The hazard ratio of two individuals $\mathbf{x}_1$ and $\mathbf{x}_2$ is given in Equation 2.12. It is constant and independent of time.

$$\frac{h(t, \mathbf{x}_1)}{h(t, \mathbf{x}_2)} = \frac{\exp(\boldsymbol{\beta} \cdot \mathbf{x}_1)}{\exp(\boldsymbol{\beta} \cdot \mathbf{x}_2)} = \exp(\boldsymbol{\beta} \cdot (\mathbf{x}_1 - \mathbf{x}_2)) \tag{2.12}$$

$\boldsymbol{\beta}$ is estimated by partial likelihood and survival function $S_i(t)$ for individual i can be estimated from Equation 2.5, 2.6 and 2.11.

$$\hat{S}_i(t) = \exp\left(-\int_0^t \hat{h}_0(u) \exp(\hat{\beta} \cdot \mathbf{x}_i) du\right) = \exp\left(-\hat{H}_0(t) \exp(\hat{\beta} \cdot \mathbf{x}_i)\right) \tag{2.13}$$

where $\hat{H}_0(t)$ is the estimated cumulative baseline hazard. There are several approaches to approximate $\hat{\beta}$ and $\hat{H}_0(t)$. They are heavily statistics based and are beyond the scope of this project. Approximation method selection is discussed in Section 4.2.

2.2.2 Non-parametric Statistical Methods

As mentioned above, non-parametric statistical methods are critical to understanding some existing machine learning methods for survival analysis. Quite a few methods use non-parametric statistical methods to deal with censored data.

Suppose there are n independent identically distributed individuals in the population. Let $t_1 < t_2 < \cdots < t_k$ be the ordered death time with $k \leq n$. Let d_j denote the number of deaths at time t_j. $\sum_{j=1}^{k} d_j = n$, and n_j is the number of individuals who are still at risk at time t_j. By maximum likelihood estimation and Equation 2.9,

$$\hat{S}(t) = \prod_{j:t_j \leq t} (1 - \hat{h}_j) = \prod_{j:t_j \leq t} \left(1 - \frac{d_j}{n_j}\right) \tag{2.14}$$

This is known as Kaplan-Meier estimate. The cumulative hazard function in discrete case is known as Nelson-Aalen estimate and is given by

$$\hat{H}(t) = \sum_{j:t_j \leq t} \hat{h_j} = \sum_{j:t_j \leq t} \frac{d_j}{n_j} \tag{2.15}$$

2.3 Machine Learning Methods

Following I introduce ANN and MTL, which are fundamental to my proposed machine learning method for survival analysis.

2.3.1 Artificial Neural Networks

"The study of ANNs has been inspired in part by the observation that biological learning systems are built of very complex webs of interconnected neurons [8]."

The general structure of an ANN is illustrated in Figure 2.4. Each neuron in the input layer represents a feature in the dataset. Hidden layers represent the non-linear relationship between inputs and outputs. Both the number of hidden layers and the number of neurons in each hidden layer require experiments to optimize. Neurons in the output layer represent predicted outcomes.

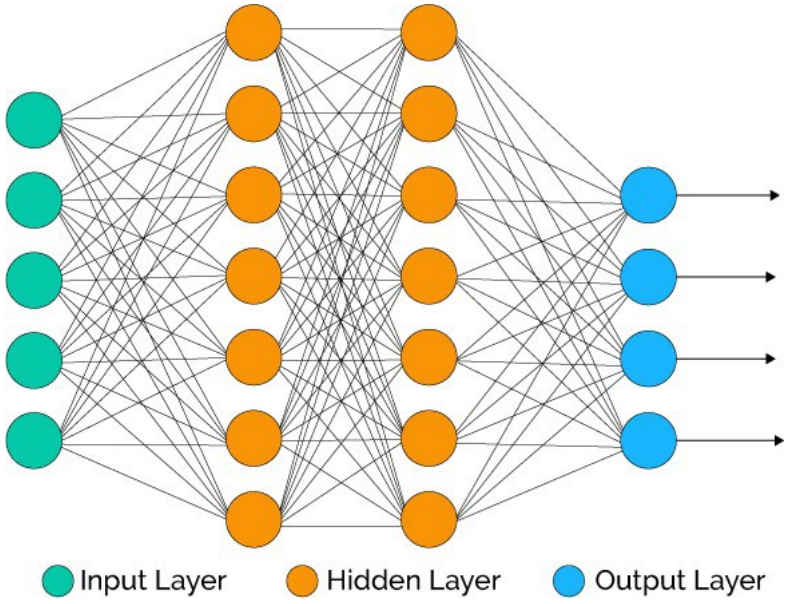

Figure 2.4: General Structure of an ANN [9]

Both hidden layers and output layer take activation functions as seen in Figure 2.5. Each neuron takes the weighted sum from previous outputs, transfers it through an activation function, and outputs this value to its downstream neuron. Sigmoid function is commonly used to predict probabilities because output are in $(0, 1)$ as illustrated in Figure 2.6. Other common

activation functions include `ReLu`, `softmax`, `tanh` etc., and experiments are required to find the best activation function for each hidden layer and output layer.

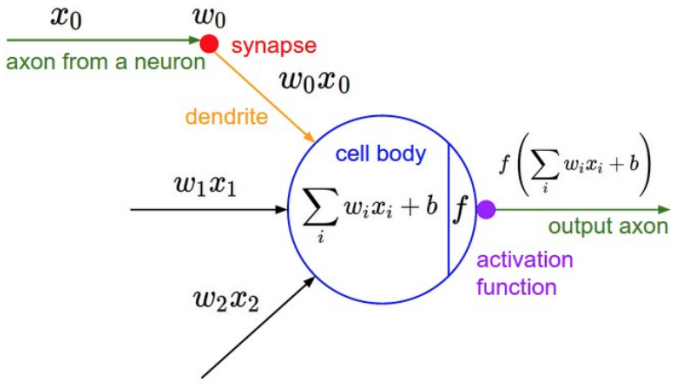

Figure 2.5: Activation Function [10]

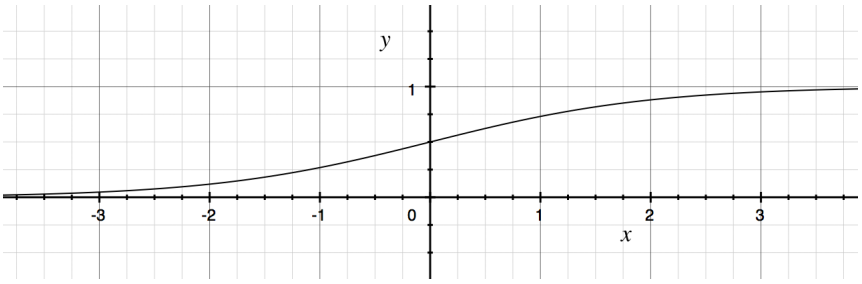

Figure 2.6: Sigmoid function $S(x) = \frac{1}{1+e^{-x}}$

Minimizing the loss function is the objective of an ANN. A common loss function in regression problems is the mean squared error (MSE), which minimizes the mean of the square difference between predicted and actual outcomes. Other common loss functions include mean absolute error, cross entropy function and so on. However, for survival analysis, these loss functions may require some asymmetric modifications.

All neurons are fully connected with weights. Back-propagation algorithm makes ANN adjust weights and learn the non-linear pattern between inputs and outputs. The algorithm works as follows,

1. Initialize all weights to some random numbers.

2. Repeat until certain termination condition is met. One forward and backward propagation are called one epoch.

   (a) Propagate inputs forward through the network and compute outputs.

   (b) Use optimizer to minimize the loss function with respect to each weight w_{ji} and update weight $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$ after each epoch, where Δw_{ji} is determined by the optimizer. Common optimizer include Stochastic gradient descent, Adam etc.

Again, the most suitable optimizer is determined by the experiment. Within the optimizer, hyperparameters like learning rate, decay rate, momentum and so on, can be tuned for specific tasks as well.

It is important to avoid overfitting in ANN, so that the model performs well not only on the training set, but also on unseen data. In ANN, the dataset can be split into three disjoint sets - training, test and validation sets to avoid overfitting. in this project, the ratio between training plus validation sets and test set is $9:1$ and the ratio between training set and validation set is $9:1$. This matches with the 10-fold cross-validation concept which is introduced later. The purpose of validation set is to test if ANN overfits the training data during the training process. As seen in Figure 2.7, if the loss on validation set starts to increase whereas the loss on training set still decreases, this is a sign of overfitting.

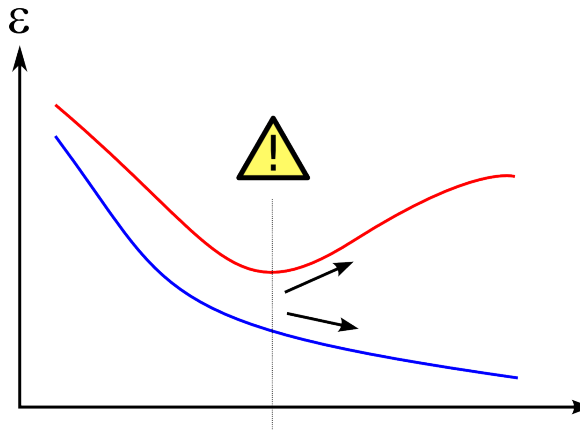

Figure 2.7: Loss on training and validation sets indicates overfitting. Blue line represents loss on training set and red line represents loss on validation set. [11]

Common techniques to prevent overfitting in ANN include regularizations, early stopping and dropout. Regularization is often applied on high-dimensional data and it prevents overfitting by selecting the most important predictor variables to build the model. Early stopping stops the training process of ANN when the error in the validation set increases for n consecutive times. Dropout randomly drops neurons in the network during the training process.

2.3.2 Multi-Task Learning

Multi-Task Learning (MTL) is the concept of "learning tasks in parallel while using a shared representation; what is learned for each task can help other tasks be learned better [12]." MTL can be adopted in common machine learning methods such as ANN, k-Nearest Neighbour, decision trees etc. The simplest way to use MTL for a time series prediction is to use a single ANN with multiple outputs, each output corresponds to the same type of task at a different time [12]. Hidden layer plays the role as the shared representation. The idea is illustrated in Figure 2.8.

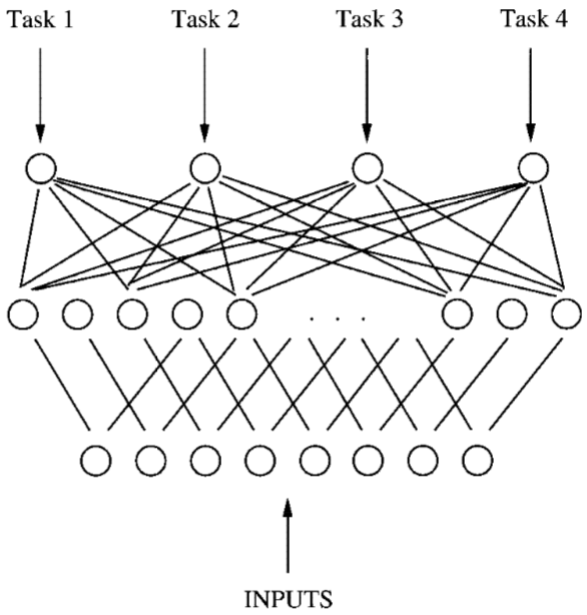

Figure 2.8: MTL of four tasks as outputs [12]

2.3.3 K-Fold Cross-validation and Bootstrapping

K-fold cross-validation can provide a general idea about the model performance on unseen data. As Figure 2.9 illustrates, it divides the entire dataset into k disjoint folds with equal number of observations in each fold. In each iteration, one fold is used as the test set, and the rest is treated as the training set. K iterations are performed and the mean of performance evaluation metrics is used to compare among models. In this project, 10-fold cross-validation was applied to all methods.

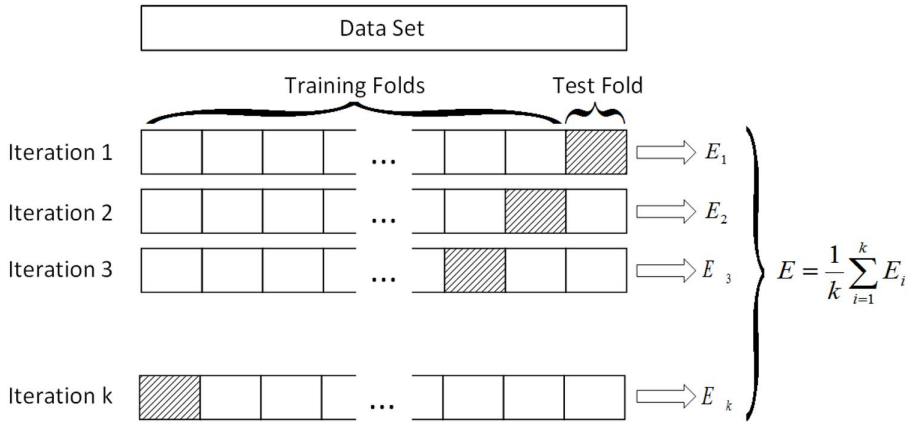

Figure 2.9: K-Fold Cross-Validation [13]

Bootstrapping randomly selects n instances with replacement to train the model. In other words, there are possibilities that one instance may appear in the training set multiple times and some instances may never be selected into the training set. By default, each bootstrapped sample excludes $\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^n \to \frac{1}{e} \approx \frac{1}{3}$ of the data, which is called out-of-bag (OOB) data.

Zhou in [14] stated that in ensemble learning, bootstrapping creates different training sets for individual learners, and learns the best from individual learners to prevent overfitting. However, bootstrapping changes the distribution of the original dataset. When there is sufficient data, k-fold cross-validation is recommended.

Chapter 3

Datasets

Five publicly available real-world low-dimensional medical datasets with censored observations were used for experiments. The nature dataset was obtained from [15], which is our largest dataset with originally more than three thousand observations across twelve cancer types. The colon cancer dataset was obtained from R `survival` package, which contains "data from the first successful trials of adjuvant chemotherapy for colon cancer [16]", and it is further segmented to two datasets based on its event type. Both breast cancer dataset and Primary biliary cirrhosis (PBC) dataset were obtained from [17] and were used for experiments in active learning based survival regression for censored data. According to the authors, "breast cancer data is from the German Breast Cancer Study Group" and "PBC data is from the Mayo Clinic trial in PBC of the liver conducted between 1974 and 1984".

Exploratory data analysis is an important step before model build and experiments on these datasets. It checks whether there is any unusual distributions in any covariates and prepares datasets to their best forms for model build. The nature dataset is the biggest dataset among five datasets and it is ill-formed, whereas the other datasets have been used in survival models. So here I provide a detailed exploration on the nature dataset. Problems in the other datasets were dealt in the same way, and a brief description is given at the end.

3.1 The Nature dataset

Sample observations from the original nature dataset are shown in Appendix A.1. The original dataset consists of 139 variables for each observation. These variables include

- `Sample.ID`;
- `Days.to.death`, `Days.to.last.followup`, `Vital.status`;
- 8 predictor variables including
  - `Years.to.birth`
  - `Gender`
  - `Date.of.initial.pathologic.diagnosis`
  - `TCGA.tumor.type`
  - `Somatic.mutations`
  - `Nonsilent.somatic.mutations`
  - `Tumor.stage`
  - `Tumor.grade`
- 127 binary predictor variables for gene expressions.

To prepare the dataset for survival analysis, `Days.to.death` and `Days.to.last.followup` are combined to one variable, representing the event time. `Vital.status = 1` if the event of interest is observed, `Vital.status = 0` if the observation is censored. For observations with unknown survival times, or censored observations with no censored times, no useful information can be retrieved from them, thus these observations were removed. There were lots of missing values in gene expressions in the original nature dataset, and high-dimensional data with sparse gene expressions can easily overfit the data. So in this project, we aim to build a robust model with low-dimensional datasets and gene expressions are excluded from predictor variables.

Figures are good for visualizing the characteristics of covariates intuitively and help with decision making. Depending on whether each predictor variable is numerical or categorical -

- For continuous variable like the survival time, both histograms and box plots are good at presenting characteristics of variables.
- For categorical variables like vital status, gender and TCGA tumour type, pie charts and bar charts are given.
- For numerical variables with handful numbers such as tumour stage, it is better to treat them as categorical variables and plot the corresponding bar charts.

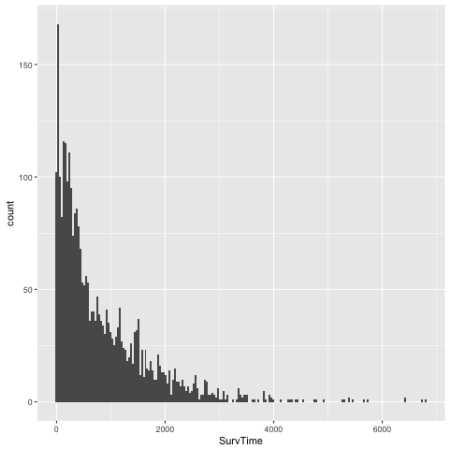

Figure 3.1: SurvTime

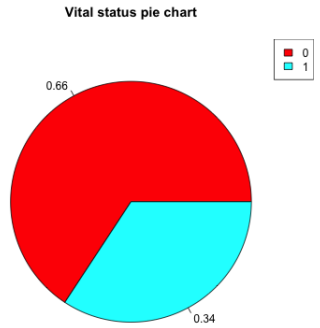

Figure 3.2: Vital.status

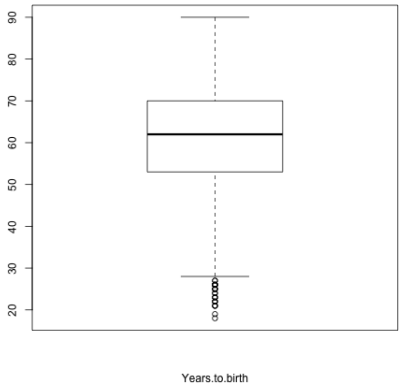

Figure 3.3: Years.to.birth

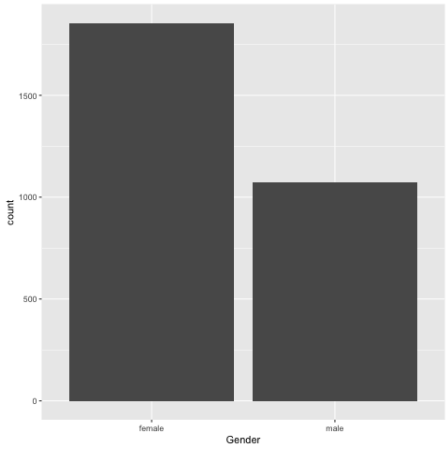

Figure 3.4: Gender

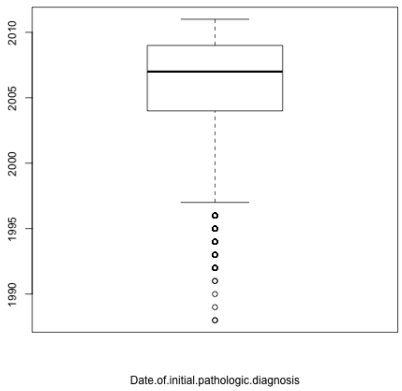

Figure 3.5: Date.of.initial.pathologic.diagnosis

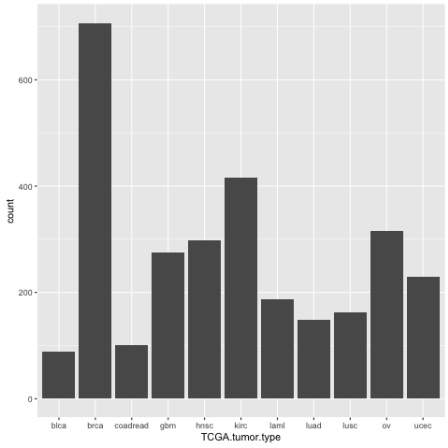

Figure 3.6: TCGA.tumor.type

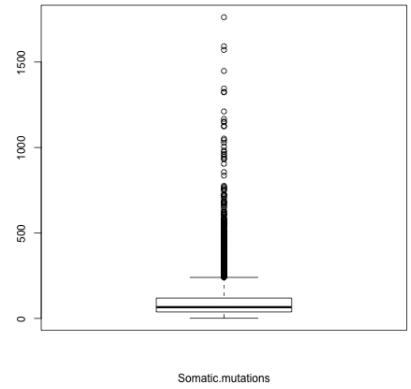

Figure 3.7: Somatic.mutations

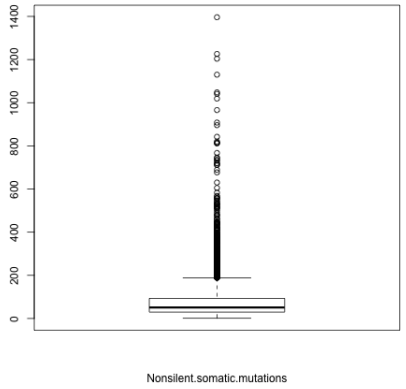

Figure 3.8: Nonsilent.somatic.mutations

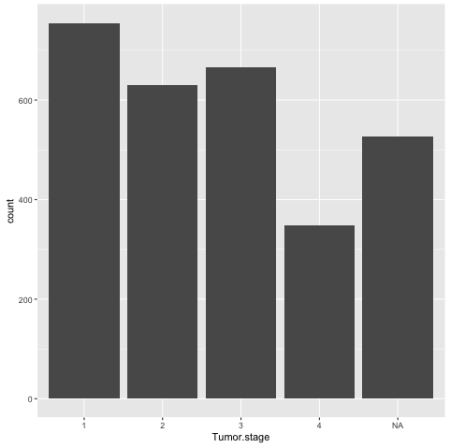

Figure 3.9: Tumor.stage

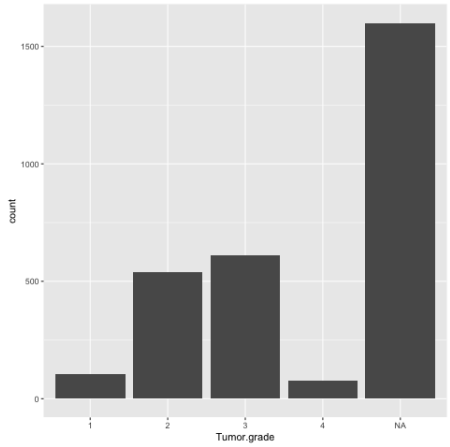

Figure 3.10: Tumor.grade

Figure 3.1 shows that `SurvTime` is right skewed, i.e. there are fewer observations with longer survival times. However, in order to predict the survival time as accurate as possible, survival times were still measured in days in experiments. Figure 3.2 indicates that there are more than 50% censored observations in the dataset. It highlights the importance of dealing with censored observations in survival analysis. There are sufficient observations in each `Gender` category as seen in Figure 3.4.

Units of measurements of all numerical variables are in hundreds except `Date.of.initial.pathologic.diagnosis`. I chose to rescale its unit of measurement from thousands to hundreds. Although it does not make any difference on the Cox PH model, it can help with a quick convergence in ANN. After rescaling, the predictor variable denotes the date of initial pathologic

diagnosis since 1988 and is measured in years.

3.1.1 Outliers

For numerical variables, the number of outliers in `Year.to.birth` and `Date.of.initial.pathologic.diagnosis` are acceptable as seen in Figure 3.3 and Figure 3.5. However, there were a lot of outliers beyond the upper whiskers in Figure 3.7 and Figure 3.8. We chose to discard observations with outliers in `Somatic.mutations` and `Nonsilent.somatic.mutations` to reduce the noise in the model. Consequently, there are 2572 observations left for experiments.

3.1.2 Missing Values

There were missing values in both `Tumor.stage` and `Tumor.grade` as shown in Figure 3.9 and Figure 3.10 respectively. However, the proportion of missing values in `Tumor.grade` dominated all other categories. So we had to remove `Tumor.grade` from predictor variables. Although the information of `Tumor.grade` was lost in predictive models, alternative solution such as imputing missing values is likely to alter the data distribution, thus introduce bias to the model.

For the remaining 7 predictor variables, Table 3.1 summarizes predictor variables along with numbers of missing values. It was believed that these values were missing at random, i.e. there is no connection between the occurrence of missing values and predictor variables. Simply discarding observations with missing values is likely to result in insufficient observations. An alternative solution is to impute missing values.

Table 3.1: Missing values in predictor variables

Predictor variables	`Years.to.birth`	`Tumor.stage`
Number of missing values	9	507

`mice()` in R `mice` package [18] provides the method to impute missing values. Available imputation methods are shown in Table B.1. All methods can deal with numerical variables, and experiments suggested that they did not make much difference on the nature dataset. Here I chose CART as the imputation method, because it has great discriminative power.

3.1.3 Including Categorical Variables

Categorical variables are usually included in the model as a series of binary dummy variables for each possible category. For categorical variable with only two categories - `Gender` in the nature dataset, `male` was encoded as 0 and `female` was encoded as 1. For categorical variable with more than two categories - `Tumor.type` in the nature dataset, `Tumor.type` was replaced with ten dummy variables. The tumor type `brca` was treated as the base category and excluded from the model to ensure a unique optimized solution [19].

3.2 Other datasets

Sample observations from original colon, breast and PBC datasets are included from Appendix A.2 to A.4. Figures for visualizing colon, breast and PBC datasets as part of the exploratory data analysis are included from Appendix B.2 to B.4.

For the colon dataset, it was noticed that categorical variables `perfor` and `extent` have highly imbalanced categories, where most observations have `perfor` as 1 and `extent` as 3. It may suggest other categories of these variables are under represented in the data, but the medical meanings of these variables suggest that perforation of colon and extent of local spread as serosa should be common cases for many patients. Hence these two variables were left unmodified. The categorical variable `rx` was encoded as dummy variables with the base category `Obs` being excluded. Missing values in `nodes` and `differ` were imputed by CART. The colon dataset records two event types for each individual - cancer recurrence and cancer survival respectively. It is natural to segment the dataset based on different purposes and built two models accordingly. They are referred to as colon recurrence and colon survival datasets respectively.

For the breast dataset, outliers in `progrec` and `estrec` as seen in Figure B.21 and B.22 are likely to introduce noise to the model. Therefore, observations with outliers in `progrec` and `estrec` were removed.

For the PBC dataset, categories in categorical variables `sex`, `ascites` and `edema` are highly imbalanced, but it is believed that there are medical reasons behind these unbalanced categories. For example, PBC "is more common in women, with a female to male ratio of approximately 9:1 [20]", this explains

the unbalanced ratio in `sex`. Consequently, these categorical variables were left unchanged. Outliers in `bili`, `chol`, `copper`, `alk` and `trig` were removed as they may introduce noise to predictive models, thus around a hundred observations were deleted. PBC data has the smallest sample size among all five datasets.

Table 3.2 summarizes general properties of five datasets, ready for experiments. All datasets are quite large in the content of survival analysis, because the normal sample size of a real-world survival dataset is about the size of the PBC dataset. All datasets are low-dimensional and have around 50% censored observations. For the purpose of applying MTL on these datasets, the maximum event time for each dataset is included as well.

Table 3.2: Dataset summary

Dataset	Number of observations	Number of predictor variables	Number of censored observations	Maximum event time
Nature	2572	16	1678(65.2%)	6795
Colon recurrence	929	12	461(49.6%)	3329
Colon survival	929	12	477(51.3%)	3329
Breast	565	8	304(53.8%)	2659
PBC	311	17	187(60.1%)	4556

Chapter 4

Performance Evaluation Metrics and Existing Survival Models

4.1 Performance Evaluation Metrics

4.1.1 Concordance Index

Before moving onto investigating existing survival models, it is important to have a uniform metric to evaluate the performance. A common performance evaluation metric in survival analysis is the C-index. It is defined as the probability of agreement for any two randomly chosen observations, where agreement means that the observation with the shorter survival time should have the larger risk score [16] and vice versa. Censored observation cannot be compared with any observations with event time after its censored time, since its exact event time is unknown [21]. Any other pairs of observations are called comparable. Among those comparable observations, the C-index of the Cox PH model can be expressed as

$$c = \frac{1}{\text{number of comparable pairs}} \sum_{i:\delta_i=1} \sum_{j:y_i<y_j} I[\mathbf{x}_i\hat{\boldsymbol{\beta}} > \mathbf{x}_j\hat{\boldsymbol{\beta}}] \tag{4.1}$$

where $I[\cdot]$ is the indicator function and risk is measured by the linear predictor $\mathbf{x} \cdot \boldsymbol{\beta}$. If predicted outcomes are survival times, ensemble mortalities etc., the C-index is given by

$$c = \frac{1}{\text{number of comparable pairs}} \sum_{i:\delta_i=1} \sum_{j:y_i<y_j} I[\hat{Y}_i < \hat{Y}_j] \tag{4.2}$$

where $\hat{Y}$ denotes the predicted outcome, and in this case, shorter survival time means smaller predicted outcome. Greater C-index means better agreement among comparable pairs.

The advantage of the C-index is, it works for any types of predicted outcomes. For example, Cox PH model uses linear predictor for comparison, and later in this chapter we see that, RSF uses ensemble mortality for comparison, and MTLSA uses the area under the survival curve for comparison.

The biggest disadvantages of the C-index are, firstly, it only measures the relative risk, instead of the actual difference between predicted and actual outcomes. Secondly, existing machine learning methods for survival analysis do not adopt the same predicted outcomes, thus it is hard to say comparisons are uniform among different models. A change in the type of the predicted outcome can easily result in a different C-index value.

4.1.2 Integrated Brier Score

Most existing methods are capable of estimating survival probabilities for each individual at times of interest. For example, survival probabilities are estimated by Equation 2.13 in Cox PH model, Equation 2.6 and Equation 2.15 in RSF, or as a direct prediction in MTLSA. Using probability calibration as the means of evaluation ensures all methods have the same type of predicted outcomes.

Due to limitations of C-index, I used IBS as another performance evaluation metric in the project. The idea is similar to MSE and it is asymmetrically modified to discard errors at time points after the censored time. The unweighted Brier score (BS) at time t is defined as [21]

$$BS(t) = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}(\hat{y}_i(t)-0)^2 & \text{if } t_i \le t, \delta_i = 1\\ (\hat{y}_i(t)-1)^2 & \text{if } t_i > t\\ 0 & \text{if } t_i \le t, \delta_i = 0\end{cases} \tag{4.3}$$

where N is the number of observations at time t, $\hat{y}_i(t)$ is the predicted survival probability of individual i at time t, t_i is the event time of i, and δ_i is the censored status of i at time t. It should be noticed that weighted Brier score is also available, but the project did not adopt the weighted form because the error is weighted by Kaplan-Meier estimate. I am interested in a machine learning method for survival analysis without any statistical methods

involving. The IBS in discrete case can be written as [22]

$$IBS = \frac{1}{\max(t_i)} \sum_{t=0}^{\max(t_i)} BS(t) \tag{4.4}$$

It was quickly found out that the Cox PH model and RSF implemented in R packages could only estimate survival probabilities at time of interest, i.e. when censoring or event of interest happens. Therefore, IBS is modified for the convenience of the Cox PH model and RSF as

$$IBS_{\text{modified}} = \frac{1}{\text{length of time of interest}} \sum_{t \in \text{time of interest}} BS(t) \tag{4.5}$$

Equation 4.4 and 4.5 can both be interpreted as the average of BS across a range of discrete time points, though the term 'integrated' is misnomer in Equation 4.5. IBS is always within $[0, 1]$ and smaller IBS indicates better probability calibration.

In this project, I concentrated more on minimizing IBS, but I would still like to keep a reasonably good C-index for several reasons. Firstly, there is no indication of random guessing in terms of the orders of predicted outcomes. Secondly, it ensures models do not overfit on IBS. Finally, C-index is still the most widely used performance evaluation metric in survival analysis, so it is good to record this metric for future reference.

4.2 Cox Proportional Hazards Model

Cox PH model was implemented by `coxph()` in R `survival` package [16]. The original Cox PH model assumed distinct failure times and presented partial likelihood to estimate covariate coefficients. However, tied survival times are more often the case in real-world datasets. Tied survival times mean either there is more than one death at any time, or some censored observations coincide with a death time [7].

To take tied survival times into consideration, Breslow approximation, Efron approximation, and exact partial likelihood are alternative ways to estimate coefficients of covariates, the baseline cumulative hazard rate, and corresponding survival probabilities for each individual. The exact partial likelihood is appropriate when the times are a small set of discrete values. Maximum survival times in our datasets are all in thousands, thus exact partial likelihood is inappropriate in this case. Comparing with Breslow method, Efron approximation is more accurate when dealing with a large number of tied death times, which is the case in our datasets as seen in Figure 3.1, Figure B.1, Figure B.13 and Figure B.23, and is computationally efficient. As a result, Efron approximation was applied to handle ties and estimate survival function.

Following, the nature dataset was used to demonstrate the difference between time-independent and time-dependent Cox PH models, and discuss why the time-independent Cox PH model was adopted in this project.

A Cox PH model was built on the entire dataset and the summary is included in Appendix C.1. The C-index as 0.798 indicates a good performance, as $0.6 - 0.7$ is a common result for survival data [16]. This Cox PH model is based on the assumption that predictor variables are time-independent. In the nature dataset, `Years.to.birth` and `Date.of.initial.pathologic.diagnosis` are two obvious predictor variables which could be treated as time-dependent covariates. Cox PH model can be generalized by stratifying time-dependent covariates, and assume proportional hazard in each strata [6]. The summary of Cox PH model with time-dependent covariates on the entire nature dataset is shown in Appendix C.2.

C-index improved from 0.798 to 0.806 when time-dependent covariates were included. Although there was a indication of better performance, stratification made testing and performance evaluation hard. Although the nature dataset is our largest dataset, its sample size still cannot guarantee to cover

all stratified levels, so there may be a new level obtained from the test set and it is not covered in the stratified model based on the training set. As a result, I implemented 10-fold cross-validation with time-independent Cox PH model and experimental results are shown in Chapter 6.

4.3 Existing Machine Learning Methods for Survival Analysis

The rest of this chapter introduces existing machine learning methods for survival analysis. Literatures were selected based on references in 'Machine Learning for Survival Analysis: A Survey' by Wang et al. [21] in 2017. Figure 4.1 below summarizes methods which were reviewed in this project. These methods were chosen for several reasons -

- They all proposed a new machine learning approach to deal with censored observations in survival analysis.
- "Almost 70% of all reported studies use neural networks as their primary (and sometimes only) predictor. Support vector machines are a distant second with 9% [4]". It is worth reviewing the most widely used machine learning methods for survival analysis.
- RSF is well implemented in R, and it is a common method used in many literatures to make comparisons with proposed approaches.
- Other machine learning methods like active learning based survival regression for censored data and MTLSA was proposed in recent years, and authors have stated that their proposed methods have outperformed the state-of-the-art methods.

For each machine learning method, the focus was on three aspects - what approaches were used to deal with censored observations; how did they evaluate their proposed approaches; and did proposed methods outperform the state-of-the-art methods. As mentioned before, C-index is the most widely used performance evaluation metric in survival analysis literature, so I justified the performance of existing methods based on it. Finally, I applied RSF and MTLSA to our datasets.

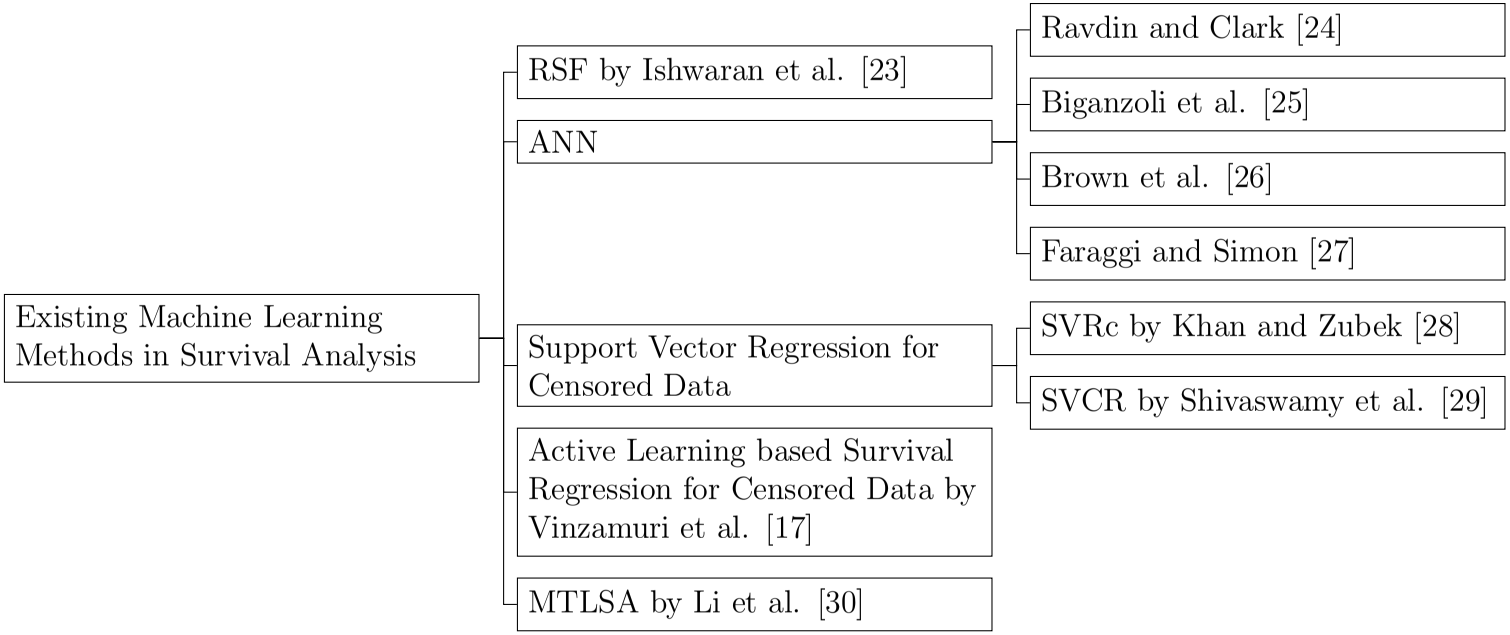

Figure 4.1: Existing Machine Learning Methods for Survival Analysis

4.4 Random Survival Forests

RSF developed by Ishwaran et al. [23] can be viewed as a combination of decision trees, bootstrapping and non-parametric statistical method to compute the predicted outcome - ensemble mortality. The algorithm of RSF is as followed -

1. Draw B bootstrapped samples from the dataset to train the model. Each bootstrapped sample consists of n randomly selected instances with replacement.

2. Grow a tree for each bootstrapped sample. At each node of the tree, randomly select p predictor variables as candidate variables. Then the predictor variable which provides the most discriminative power to the tree is selected for the node.

3. Grow the tree until certain termination condition is met. For example, the constraint can be, a terminal node should have no less than $d_0 > 0$ unique deaths.

4. Calculate the cumulative hazard function (CHF) for each terminal node in each tree using Nelson-Alaen estimate (Equation 2.15). CHF for i in each tree is represented by CHF for the terminal node it falls in. Then the OOB ensemble CHF for i is the average of CHF for i in each tree where i is OOB.

5. The predicted outcome is the OOB ensemble mortality for i, which is equal to the sum of OOB ensemble CHF for i at each time of interest.

Measured by C-index, the paper concluded that RSF is "consistently better than, or at least as good as, competing methods [23]". Similar result was reported by Omurlu et al. in [31]. Although RSF does not outperform the Cox PH model, it is one of the well developed machine learning methods for survival analysis in R. Hence, I chose it as one of the machine learning methods for comparison.

The R `randomSurvivalSRC` package [32] provides the nice function `rfsrc()` to implement RSF. Bootstrapping is part of the algorithm and was nested in 10-fold cross-validation, so different models were calibrated in the same way for comparison. Table D.1 in Appendix summarizes important hyperparameters which were considered in order to build the RSF model.

There are a number of ways to search for the best hyperparameters. For example, grid search uses brute force to search through the hyperparameter space and return the best result. It guarantees to find the best result but is computational expensive. Random search randomly select different points in hyperparameter space and evaluate the performance, its computational cost is user defined but it does not guarantee to find the best result.The selection of the search algorithm largely depends on how many possible combinations of hyperparameters need experimenting.

According to the algorithm described in the original paper [23], bootstrapping was performed on the entire data by sampling with replacement at the root node. As the number of predict variables is relatively small, there is no need to specify the maximum depth to which a tree should be grown. In theory, more trees in the forest certainly result in better performance, but it is also computationally expensive. 500 number of trees were grown in the forest, so that the algorithm was fast and relatively accurate. While fixing all other predictor variables, experiments showed that log-rank splitting rule resulted in 1% higher C-index than log-rank score splitting rule. Thus the splitting rule were set to log-rank before performing the search algorithm.

RSF API also provides options to associate observations with different weights. Weightings are usually performed when data is imbalanced. After data preparation, it is believed that the data is fairly balanced. Hence these parameters were left as default, i.e. uniform weights.

This left a handful of hyperparameters to search on. Hence grid search was adopted to search best values for - number of variables randomly selected as candidates for splitting a node (`nsplit`), and forest average number of unique cases in a terminal node (`nodesize`). The best hyperparameter combination on each dataset is shown in Table 4.1.

To compare with other predictive models, for each dataset, grid search was first performed based on the entire dataset and the best hyperparameter combination was selected based on C-index on OOB data. Then 10-fold cross-validation was performed with 10 different RSFs built and evaluated. Final C-index and IBS are shown in Chapter 6. The potential reason why RSF cannot outperform the Cox PH model are also discussed in Chapter 6.

Table 4.1: RSF grid search results

Dataset	`nsplit`	`nodesize`
Nature	9	25
Colon recurrence	4	60
Colon survival	2	35
Breast	7	45
PBC	14	5

4.5 Artificial Neural Networks

Most ANN methods for survival analysis were developed in the 1990s, and were under-researched in the past two decades, as evidenced by references in [33] in 2005 and [21] in 2017. Table 4.2 summarises four ANN methods for survival analysis.

Both Ravdin and Clark [24] and Biganzoli et al. [25] coded one observation as a series of observations with different time intervals and survival statuses. Censored data at later time intervals was excluded. They adopted similar ANN architecture - a time interval as an additional predictor variable and a survival probability or discrete hazard rate at this time interval as output. Ravdin and Clark stated that their predicted outcome is "roughly proportional to the survival probability", whereas Biganzoli et al. directly modelled the discrete hazard rate, which can be easily transferred to monotonically decreasing survival probability by Equation 2.9. Hence Partial logistic regression models with ANN (PLANN) proposed by Biganzoli et al. is preferable. However, Ravdin and Clark pointed out that due to censored observations, in later time intervals, data largely represents individuals who have died. To correct this bias, they first used Kaplan-Meier estimate to determine the survival probability at this time interval, and randomly selected censored observations to balance the proportion. This approach of dealing with selection bias is referable.

Both PLANN and Brown et al. adopted discrete hazard rate as predicted outcome. Unlike PLANN, Brown et al. trained one ANN to predict a series of discrete hazard rates at each time, which is the idea of MTL. PLANN used empirical estimates from Kaplan-Meier estimate (Equation 2.14) as actual outcome, whereas Brown et al. used binaries as actual outcomes depending on survival status at each time, and they can be transferred to monotonically decreasing survival probabilities by Equation 2.9. To cope with censored observations, errors at any undefined time points are set to zero, preventing unknown hazards from updating weights. The architecture of ANN by Brown et al. is preferable because only one ANN is required to train.

The idea of Faraggi and Simon [27] is different from the other three methods. They proposed non-linear PH model by replacing the linear predictor in Cox PH model with the non-linear output from ANN. However, experiments by Mariani et al. [34] showed that this approach does not outperform the Cox PH model.

Overall, none of the methods above for survival analysis outperformed the Cox PH model. However, I would still like to explore this approach for several reasons. Firstly, ANN has proved its strength in other fields in recent years. ANN for survival analysis was under-researched for almost two decades, so there are great possibilities to find a better ANN. Secondly, ANN is not restricted by the proportional assumption of the Cox PH model, except the methodology proposed by Faraggi and Simon, which still relies on the Cox PH model. Finally, most of the methods above did not use C-index or a uniform performance evaluation metric for comparison, so it is ideal to have a uniform comparison among Cox PH model, ANN and other existing machine learning methods for survival analysis. Further investigation of ANN is discussed in Section 5.2.

Table 4.2: ANN methods for survival analysis

Authors	Inputs	Loss function	Activation function	Single or multiple outputs?	Predicted outcome	Monotonically decreasing survival curve?	Performance measure matrices	Outperformed Cox PH model?
Ravdin and Clark in 1992 [24]	Predictor variables and time intervals	Unknown	Unknown	Single	Probability of death for individual over time	No	Chi-square goodness-of-fit test	Similar global chi-squared value
Biganzoli et al. in 1998 [25]	Predictor variables and time intervals	Cross-entropy	Sigmoid	Single	conditional failure probabilities	Yes	Graphs	No
Brown et al. in 1997 [26]	Predictor variables	Sum of squares error	Sigmoid	Multiple	hazard at each time	Yes	Root mean square	No
Faraggi and Simon in 1995 [27]	Predictor variables	Unknown	Sigmoid	Single	Replace linear predictor $\boldsymbol{\beta}\cdot\mathbf{x}$ in Cox PH model	Yes	Log partial likelihood and C-index	No

4.6 Support Vector Regression for Censored Data

Shivaswamy et al. in [29] and Khan and Zubek in [28] both proposed support vector regression for censored data, namely SVCR and SVRc respectively to deal with both left and right censored data in survival analysis. The core concept of both approaches is to asymmetrically modify the ε-insensitive loss function - regularization parameters γ and margin of errors δ in specific. A summary of the difference between SVCR and SVRc is illustrated by Van [35] in Figure 4.2 where (a) corresponds to SVCR and (b) corresponds to SVRc. There is no penalty when the predicted survival time is greater than the right censored time in SVCR. There are four hyperparameters in each case in SVRc to cope with both left and right censored events.

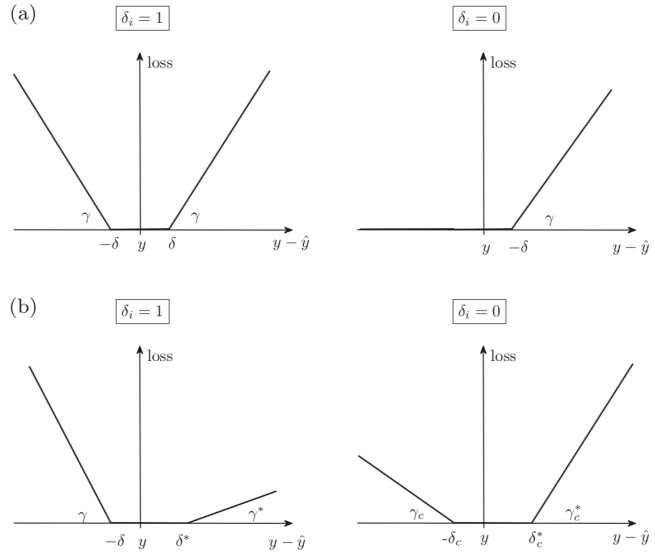

Figure 4.2: Difference between (a) SVCR and (b) SVRc [35]

Using linear kernel and C-index as one of the performance measures, experiments by Khan and Zubek [28] on SVRc and experiments by Belle et al. [35] on SVCR both showed that SVRc and SVCR can outperform the Cox PH model on most datasets. But the reason behind the choice of linear kernel for experiments was unclear. None of them make their source code publicly available, and existing SVM packages in R, Python or MATLAB do not provide the flexibility to modify the ε-insensitive loss function easily. Hence I did not test support vector regression for censored data on our datasets.

4.7 Active Learning based Survival Regreession for Censored Data

Vinzamuri et al. in 'Active Learning based Survival Regression for Censored Data' [17] integrated active learning and regularized Cox models to predict survival. It uses a small size of training data to build a Cox model to begin with, then for each iteration, the most discriminative observation is selected using a model discriminative gradient based sampling strategy, labelled by the domain expert, and added to the training set.

A comparison was made among Cox models, RSF and proposed integrations between active learning and Cox models using C-index. It was concluded that active regularized cox regression with kernelized elastic net has the highest discriminative ability on most datasets.

Drawbacks of this algorithm are, the selection of the most discriminative observation is computationally expensive, because the expected change is calculated over all possible unique time-to-event labels to approximate the true change and determine the selection. Moreover, active regularized cox algorithm is not practical for the purpose of this project. The common scenario of active learning is where there are few labelled observations and a lot of unlabelled observations, and the purpose is to build a good model without querying domain expert too many times [14]. However, in this project, datasets are very different from the setting of active learning - all observations in the dataset are labelled with either exact event time or censored time. A possible alternative is to treat censored observations as unlabelled observations with constraints, but the proposed method still adopted Cox models to handle censored data. As a result, this approach is not further explored.

4.8 Multi-Task Learning for Survival Analysis

More recently, Li et al. in 'A Multi-Task Learning Formulation for Survival Analysis' [30] adopted the idea of MTL and predicted survival probabilities for each individual at all discrete time points. Here a task means predicting the survival probability at a discrete time point.

To handle censored data, they encoded the original dataset to output matrix Y and binary indicator matrix W as seen in Figure 4.3. Entries in Y are 1 when an individual is still alive, 0 after the event of interest occurs, or unknown after censoring. Entries after censoring are indicated as 0 in W. They formulated an optimization problem with regularization terms as seen in Equation 4.6, where errors at entries after censoring are not included. Alternating Direction Method of Multipliers algorithm were employed to solve for B. The non-negative max-heap projection guaranteed survival probabilities of each individual at all discrete time points are non-negative and non-increasing.

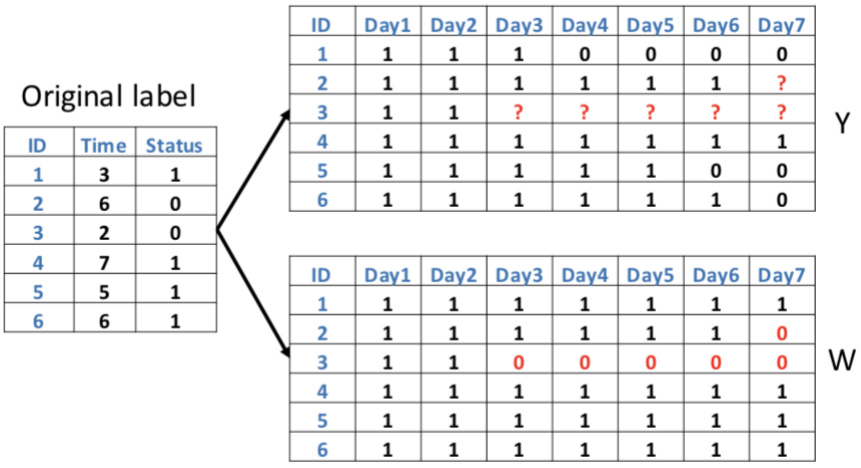

Figure 4.3: MTLSA formulation [30]

$$\min_{XB \in P} \frac{1}{2} \|\Pi_W(Y - XB)\|_F^2 + R(B) \tag{4.6}$$

where

- n denotes the number of observations, q denotes the number of predictor variables, and k denotes the maximum survival time.
- $X \in \mathbb{R}^{n \times q}$ are predictor variables for each individual.
- $B \in \mathbb{R}^{q \times k}$ is the estimated coefficient matrix.
- $Y \in \mathbb{R}^{n \times k}$ is the target matrix.
- $W \in \mathbb{R}^{n \times k}$ is the binary indicator matrix.
- The function Π_W can be interpreted as an indicator function. When the survival status of an individual at a particular time is known, the optimization problem takes the difference between Y and XB into account; otherwise, the optimization problem ignores the difference. It is an asymmetric minimization problem and the idea is similar to IBS. Mathematically,

$$\Pi_W(U)_{ij} = \begin{cases} U_{ij} & \text{if} \quad W_{ij} = 1 \\ 0 & \text{if} \quad W_{ij} = 0 \end{cases} \tag{4.7}$$

- XB satisfies the non-negative non-increasing list structure, where

$$P = \{Y \geq 0, Y_{ij} \geq Y_{il} | j \leq l, \forall j = 1, \cdots, k, \forall l = 1, \cdots, k\} \tag{4.8}$$

Source code suggests the area under the survival curve is used as predicted outcome for comparison in C-index. The authors concluded that MTLSA outperformed state-of-the-art methods.

However, all datasets used in the experiment have small sample sizes (the largest sample size is around 300 observations), high number of features (the number of predictor variables is in thousands) and low number of tasks (the largest event time is no more than 200). On the contrary, our datasets, as displayed in Table 3.2, all have large sample sizes, low numbers of predictor variables and high numbers of tasks. It was hard to say whether this algorithm could work well on very different datasets.

Moreover, as seen from Equation 4.8, the algorithm only guarantees the predicted survival probabilities to be non-negative and non-increasing, but there is no guarantee that they are less than or equal to one.

Despite of these two weaknesses, We believed that MTLSA was an approach which was worth trying on our datasets, because it was proposed

recently, yielded good C-index, and most importantly, it took a completely non-statistical approach to predict survival probabilities for each individual, which is uncommon in survival analysis literature.

The implementation of MTLSA algorithm is provided by the authors in MATLAB. Two user inputs are required for the program - number of iterations and the smallest lambda rate, both are used to compute regularization parameters. Experiments showed that MTLSA tended to predict the same survival probability for each individual at all discrete time points, and increasing number of iterations did not improve predicted outcomes. The best C-index on the nature dataset is just around 0.6.

Therefore it was concluded that MTLSA only worked well on high-dimensional datasets with small sample sizes. It did not work on low-dimensional datasets, because regularization terms still performed feature extraction on low-dimensional datasets and attempted to select the most important features among a handful of predictor variables. Consequently, most covariate coefficients were predicted as zero and survival probabilities for each individual were flat across all discrete time points. Unnecessary regularization parameters should be eliminated for the algorithm to work.

Chapter 5

Proposed Machine Learning Methods for Survival Analysis

5.1 Linear Multi-Task Learning for Survival Analysis

To solve the problem that MTLSA does not work on high-dimensional datasets, I rewrote the optimization problem in Equation 4.6 as below to eliminate the regularization terms.

$$\min_{XB \in P} \frac{1}{2} \|\Pi_W (Y - XB)\|_F^2 \tag{5.1}$$

where all variables are still the same as the original MTLSA optimization problem. Following, I attempted to solve this minimization problem in Equation 5.1 algebraically. The definition of Frobenius inner product is given below.

Definition 6. Frobenius inner product of two matrices $A, B \in \mathbb{R}^{m \times n}$ is

$$\langle A, B \rangle_F = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} B_{ij} \tag{5.2}$$

with following properties -

$$\langle A, B \rangle_F = \langle B, A \rangle_F \tag{5.3}$$

$$\langle \langle A, B \rangle_F, C \rangle_F = \langle A, \langle B, C \rangle_F \rangle_F \tag{5.4}$$

$$\langle aA, bB \rangle_F = ab \langle A, B \rangle_F \qquad a, b \in \mathbb{R} \tag{5.5}$$

$$\langle A + C, B + D \rangle_F = \langle A, B \rangle_F + \langle A, D \rangle_F + \langle C, B \rangle_F + \langle C, D \rangle_F \tag{5.6}$$

According to the definition of Frobenius inner product and the definition of the indicator function Π_W , the optimization problem can be rewritten as

$$\min_{XB \in P} \frac{1}{2} \| \langle W,\ Y - XB \rangle_F \|_F^2 \tag{5.7}$$

Let $M = \langle W, Y - XB \rangle_F$ and $L = \frac{1}{2}\|M\|_F^2$. Notice that W has the special property that $\langle W, W \rangle_F = W$ because all entries of W are binaries. To minimize L, take the first derivative of L with respect to XB. Recall the definition of Frobenius norm,

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij}^2 \right)^{1/2} \tag{5.8}$$

with property

$$\frac{\partial}{\partial A} \|A\|_F^2 = 2A \tag{5.9}$$

$$\begin{aligned}
\frac{dL}{dM} &= \langle W, Y - XB \rangle_F \\
dL &= \langle \langle W, Y - XB \rangle_F,\ d(\langle W, Y - XB \rangle_F) \rangle_F \\
dL &= \langle \langle W, Y \rangle_F - \langle W, XB \rangle_F,\ -\langle W, d(XB) \rangle_F \rangle_F && \text{by Eq. } 5.5, 5.6 \\
dL &= \langle -\langle \langle W, Y \rangle_F, W \rangle_F + \langle \langle W, XB \rangle_F, W \rangle_F,\ d(XB) \rangle_F && \text{by Eq. } 5.4 \\
\frac{\partial L}{\partial (XB)} &= \langle W, XB \rangle_F - \langle W, Y \rangle_F && \text{by Eq. } 5.3, 5.4
\end{aligned}$$

The second derivative of L with respect to XB is greater than 0, hence we can obtain the minima. To minimize L, set

$$\frac{\partial L}{\partial (XB)} = \langle W, XB \rangle_F - \langle W, Y \rangle_F = 0 \quad \Rightarrow \quad \langle W, XB \rangle_F = \langle W, Y \rangle_F \tag{5.10}$$

Because W is a binary indicator matrix, $XB = Y$ guarantees $\langle W, XB \rangle_F = \langle W, Y \rangle_F$. This multivariate linear regression of solving $XB = Y$ using LU factorization is referred to as linear MTLSA. Unknown entries in Y were assumed as 0.5. It was a reasonable assumption because after censored time, one could say that the individual has 50% probability of survival and 50% probability that the event of interest occurs at each discrete time point. The non-negative max-heap projection used in MTLSA was still employed to ensure that $X_{\text{test}}\hat{B}$ follows the non-negative non-increasing list structure.

I have simplified the optimization problem and now low-dimensional dataset can work on this variant of MTLSA. Although linear MTLSA does not outperform the Cox PH model and RSF when measuring by C-index, it performs so far the best when measuring by IBS. But the drawbacks are - firstly, predicted survival probabilities $X_{\text{test}}\hat{B}$ is only guaranteed to be non-negative and non-increasing as stated in Equation 4.8. There is still no guarantee that all entries of $X_{\text{test}}\hat{B}$ are less than or equal to one. Consequently, estimated survival probabilities were slightly over one at beginning time points. Secondly, this simple model is not flexible for future extensions. The robustness of LU factorization largely depends on whether known matrices are sparse and the size of the matrices. In all, this model is not desirable. Nevertheless, the proof shows that the asymmetric modification of the MSE like optimization problem is differentiable and feasible.

It was believed that the model could be improved if I further explored down this approach and introduced non-linear representation. ANN is popular for its ability to learn non-linear relationship between inputs and outputs and it can be structured in MTL form. A sigmoid activation function can easily guarantee predicted outcomes to be probabilities. The proposed MTL-ANN is introduced in the next section.

5.2 Multi-Task Learning - Artificial Neural Networks

For various motivations stated in both the end of ANN literature review and the end of linear MTLSA section, I integrated the ideas of MTLSA in [30], the concept of MTL in [12], ANN and IBS and proposed MTL-ANN. It was noticed that the architecture of MTL-ANN is similar to the ANN approach proposed by Brown et al. [26]. But instead of predicting discrete hazard at each time, each output neuron in MTL-ANN predicts the survival probability at each time. Non-negative max-heap projection adopted by MTLSA guaranteed monotonically deceasing survival probabilities. The area under the survival curve was still used for comparison in C-index. IBS expressed in the matrix form was used as the loss function as well as the performance evaluation metric. These variables define the general structure of MTL-ANN.

MTL-ANN was built in Keras [36]. Keras is a neural networks API written in Python and allows quick implementation of complicated ANN architectures. It was connected to MTLSA MATLAB code through MATLAB Engine API for Python [37] to evaluate C-index and make non-negative max-heap projection.

Unlike RSF, it is time consuming to define a broad range for each hyperparameter and perform grid or random search. Alternatively, reasonable values were selected for most hyperparameters, this left hyperparameters like the number of hidden layers and learning rate to be determined by experiments. Table 5.1 summarizes important hyperparameters which were considered in order to build a good MTL-ANN model.

To overcome the drawback in linear MTLSA, sigmoid activation function should be used in the output layer to ensure predicted outcomes are within $(0, 1)$. Observations propagated through the network all in once to obtain a smooth loss curve. `Adam` was used as the optimizer to update weights and it always works well in practice [38].

Table 5.1: Hyperparameters in MTL-ANN

Parameter	Value
Loss function	IBS
Number of hidden layers	1
Number of neurons in each hidden layer	200
Batch size	All training data
Optimizer	`Adam`
Activation function	[`Relu`, `Sigmoid`]
Learning rate	0.001 on nature and colon datasets, 0.0001 on breast and PBC datasets
Maximum number of epochs	500
Prevent overfitting	Early stopping

5.2.1 Non-Linear Form

The maximum number of hidden layers normally does not exceed two so it does not overfit the training dataset. In this case, as the performance of linear MTLSA is reasonable well when measuring by IBS, one hidden layer with 200 neurons should be sufficient for the non-linear representation.

In terms of choosing the learning rate for `Adam`, Figure 5.1 demonstrates how tuning on learning rate can affect the loss during the training process. High learning rate allows the loss function to quickly converge but may miss the minima point, consequently, ANN may fail to converge or even start to diverge at the end. Whereas low learning rate slows down the training process but may not end up in a satisfactory small error at the end of the training process.

Figure 5.2 to Figure 5.6 show loss curves on five datasets on both training and validation sets during the training process. According to Figure 5.1, learning rates can be considered as high on all datasets except PBC. However, lower learning rates largely compromised the computational efficiency and there was no improvement in model performance within 500 epochs. Decay in the learning rate further slowed down the training process and it did not improve the performance. There should not be a huge trade-off between a good learning rate, model performance and computational cost. The ultimate goal of ANN is to find the model with the least error. I tried to reduce the learning rate, add a decay in learning rate, extend the maximum number of epochs, adjust the number of neurons in the hidden layer, to name a few, but none of these techniques significantly improved the performance on all datasets. So the learning rate was not further tuned. The default `Adam` learning rate 0.001 was good enough on nature, colon recurrence and colon survival datasets to learn the training data, and learning rates were 0.0001 on breast and PBC datasets.

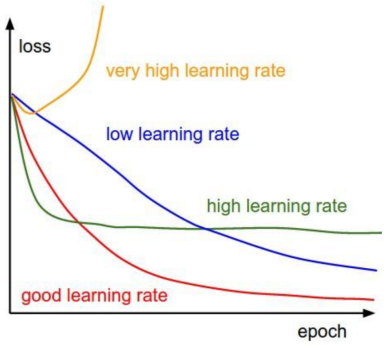

Figure 5.1: Effect of learning rate on loss during the training process [38]

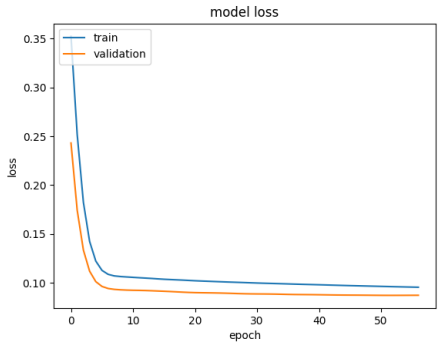

Figure 5.2: Nature dataset

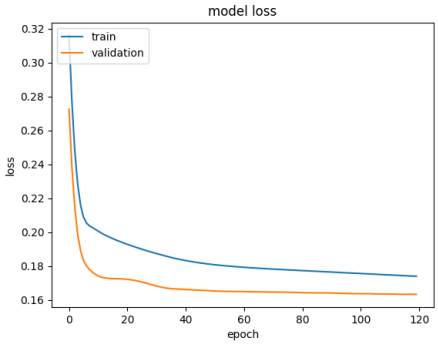

Figure 5.3: Colon recurrence dataset

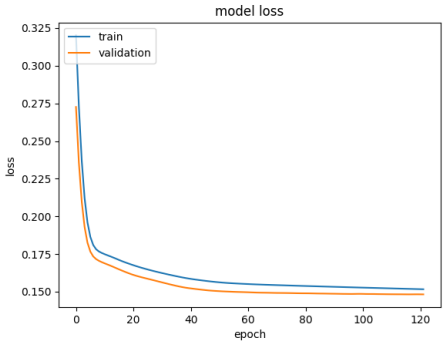

Figure 5.4: Colon survival dataset

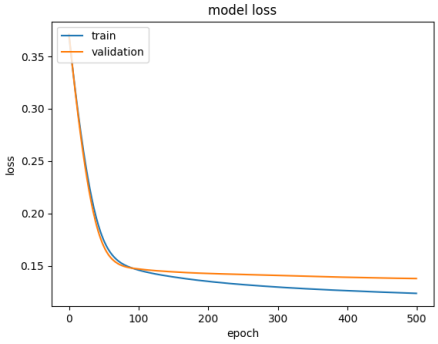

Figure 5.5: Breast dataset

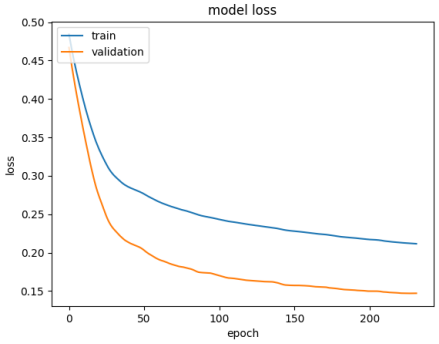

Figure 5.6: PBC dataset

To avoid overfitting, I used early stopping to stop the training process if the loss on validation set increases for 5 consecutive epochs. Comparing with early stopping, regularization has shown its weakness in MTLSA on low-dimensional datasets, and dropout resulted in oscillated loss curves in later epochs. As shown in Figure 5.2 to Figure 5.4 and Figure 5.6, early stopping occurred around a hundred epochs on nature, colon recurrence and colon survival datasets and two hundred epoches on the PBC dataset. This number varied on each dataset, and even in each fold of the same dataset.

Early stopping suggested that the model was fairly easy to train and the best model could be achieved without much complication. Current MTL-ANN does better than previous models when measuring by IBS, and the performance in C-index is acceptable except on the nature dataset. The C-index on the nature dataset is approximately 16% lower than C-indices of Cox PH model and RSF as seen in Table 6.1 later. But there should not be a huge trade-off between C-index and IBS, because both are measured based on predicted survival probabilities. It is possible that the architecture of current MTL-ANN is too complicated so it learnt the training set quickly. It may also be the case that, complicated architecture makes the model overfit on IBS, so it cannot perform well on any other performance evaluation metric. To test the hypothesis, the only hidden layer was removed in the network and I implemented multivariate linear regression in the form of ANN.

5.2.2 Linear Form

The concept of MTL is no longer precise in this case, because there is no more shared representation in the architecture. But the modification served the purpose for testing the hypothesis, I refer it as linear MTL-ANN to echo with previous non-linear MTL-ANN.

Hyperparameters were easy to determine because the model structure is straightforward. Predictor variables were directly connected with survival probabilities. Weights were trained by `Adam` with 0.001 on all datasets except 0.0001 on PBC. As seen from Figure 5.7 to Figure 5.11, training processes lasted 500 epochs without being early stopped. Moreover, loss curves on colon datasets are smoother than Figure 5.3 and Figure 5.4 previously, and it is more desirable because it means the model is more robust to noise. The gap between training and validation loss is still large on the PBC dataset in Figure 5.11, but due to the small sample size of the PBC dataset, the behaviour was acceptable. Nevertheless, results in the next chapter shows that, with almost identical learning rates, the linear MTL-ANN can be as good as non-linear MTL-ANN when measuring by IBS, and better when measuring by C-index.

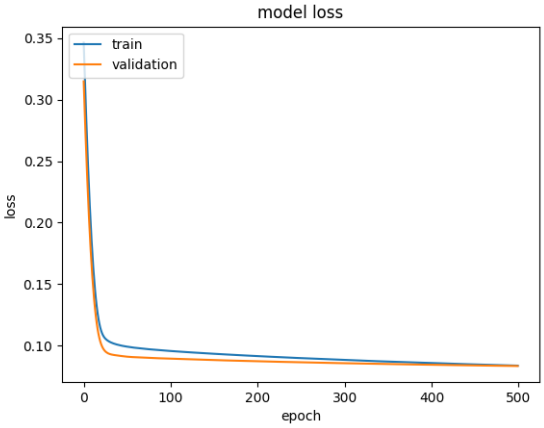

Figure 5.7: Nature dataset

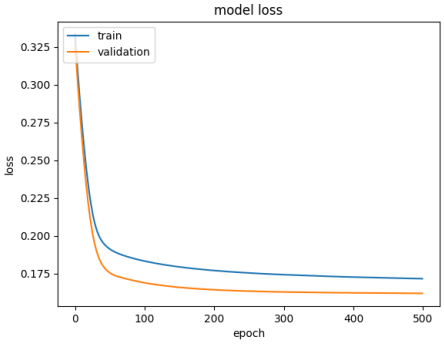

Figure 5.8: Colon recurrence dataset

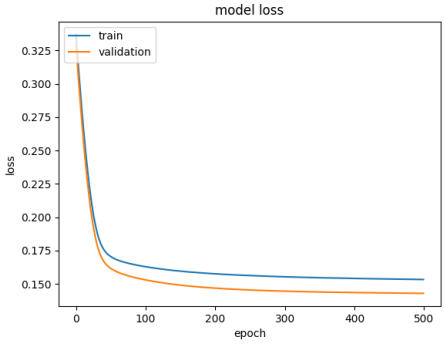

Figure 5.9: Colon survival dataset

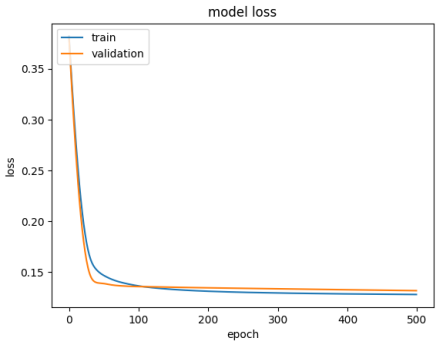

Figure 5.10: Breast dataset

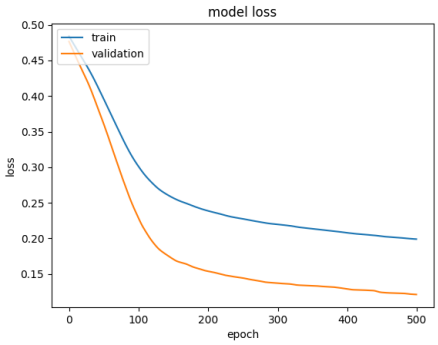

Figure 5.11: PBC dataset

Chapter 6

Results and Discussion

6.1 Results

Experimental results on five datasets across six survival models measured by C-index and IBS are summarized in Table 6.1 and Table 6.2 respectively. The best survival model for each dataset is highlighted in bold.

Table 6.1: Results measured by C-index

	Cox PH Model	RSF	MTLSA	Linear MTLSA	Non-linear MTL-ANN	Linear MTL-ANN
Nature	**0.795**	**0.795**	0.604	0.721	0.634	0.717
Colon recurrence	0.659	**0.666**	-	0.616	0.653	0.656
Colon survival	0.661	**0.667**	-	0.587	0.635	0.639
Breast	0.668	**0.683**	-	0.658	0.667	0.679
PBC	**0.849**	0.829	-	0.837	0.787	0.800

C-index as 1 means perfect agreement among comparable pairs and C-index as 0.5 is no better than random guessing. In survival analysis, C-index

is normally between 0.6 and 0.7 [16]. Table 6.1 shows that

- When using C-index as the performance evaluation metric, RSF outperformed other survival models.
- C-index of the Cox PH model is close to the C-index of RSF in most cases.
- MTLSA does not work on low-dimensional datasets as tested on the nature dataset, thus it was meaningless to test it on other datasets.
- Linear MTLSA was capable of yielding reasonable C-index. For example, C-index of linear MTLSA is 1% less than C-index of the Cox PH model on breast and PBC datasets.
- C-index of non-linear MTL-ANN are reasonable on most datasets except on the nature dataset, where the performance is 16% lower. It was a major motivation to move from non-linear MTL-ANN to linear MTL-ANN.
- C-index of linear MTL-ANN is reasonably good on all datasets and is better than the C-index of non-linear MTL-ANN, but it does not outperform state-of-the-art methods.

Table 6.2: Results measured by IBS

	Cox PH Model	RSF	Linear MTLSA	Non-linear MTL-ANN	Linear MTL-ANN
Nature	0.0699	0.0701	0.0513	0.0397	**0.0362**
Colon recurrence	0.158	0.154	0.159	**0.147**	0.148
Colon survival	0.152	0.143	0.148	**0.130**	0.133
Breast	0.125	0.122	0.110	0.102	**0.0993**
PBC	0.0690	**0.0620**	0.0680	0.0786	0.0772

Recall that IBS is always within $[0, 1]$ and smaller IBS means better probability calibration. Table 6.2 shows that

- IBS of both non-linear and linear MTL-ANN are better than IBS of other models on most datasets.
- IBS of the Cox PH model and RSF are still close to each other.
- IBS of RSF is the best on the PBC dataset, which is possibly caused by the fact that the PBC dataset has the smallest sample size. Other methods are not robust in learning from small samples, whereas bootstrapping in RSF can provide better discriminative power on small samples as mentioned previously.

6.2 Discussion

6.2.1 Similarity between Cox Proportional Hazard Model and Random Survival Forests

RSF performs as good as the Cox PH model when measuring both by C-index and IBS. This result is similar to the conclusion drawn by authors of RSF. Following, I give a brief interpretation for the reason behind this empirical result.

Recall in Cox PH model, survival probability estimation in Equation 2.13 is formed by two parts, the exponential term to deal with multivariate regression and cumulative baseline hazard rate by Efron approximation to deal with censored data. In RSF, random forests deals with multivariate regression and Nelson-Aalen estimate in Equation 2.15 deals with censored data. Efron and Nelson-Aalen estimates normally yield similar results, only Efron estimate is more computationally efficient [16]. In short, both models handle censored data in a similar way. Cumulative hazard estimates are similar, and it leads to similar survival probability estimates by Equation 2.6.

6.2.2 A Linear Model in Survival Analysis

MTL-ANN in both forms achieved the best results when measuring by IBS. It was surprising to find out that linear MTL-ANN does better than non-linear MTL-ANN when measuring by C-index, because I always thought the problem must be more complicated than a linear regression. However, looking back at different examples, a linear model is likely to be sufficient in survival

analysis.

Linear MTLSA and linear MTL-ANN have demonstrated their predictive power in experiments. Recall from literature review, SVCR and SVRc both used linear kernel for experiments and stated to outperform the state-of-the-art method [28, 35], though the reason behind the choice of linear kernel was unclear.The most convincing evidence would be the linear predictor in the Cox PH model. Recall from literature review, [27] replaced linear predictor in Cox PH model with non-linear output from ANN, but experimental results in [34] showed that it did not outperform the original Cox PH model. All evidence suggests a non-linear relationship is too complicated for survival analysis and a linear relationship should be sufficient for survival predictions.

6.2.3 Computation Time

Finally, in terms of computation time, Cox PH model and linear MTLSA only took a few seconds to train. The training time of RSF was within half an hour, depending on the hyperparameter space of the grid search. MTL-ANN took around two hours to do 10-fold cross-validation, with two CPUs being allocated to the job. Although MTL-ANN took the longest time to train, it was trained on modest hardware and the processing time can certainly improve when more resources are allocated to the job.

Furthermore, in MTL-ANN, MATLAB Engine API for Python contributes significant overhead cost, i.e. there is additional time to connect to MATLAB Engine at the beginning and to evaluate C-index and perform non-negative max-heap projection at the end of the training process. So the computation time can be improved if we reimplemented non-negative max-heap projection as well as C-index metric in Python.

Chapter 7

Conclusion and Future Work

It this project, I assessed the performance of six survival models on five real-world low-dimensional datasets. Six survival models include the Cox PH model and two existing machine learning methods for survival analysis - RSF and MTLSA. Upon identifying weaknesses in MTLSA, I proposed linear MTLSA and MTL-ANN in both non-linear and linear forms to overcome weaknesses in MTLSA.

Experimental results show that MTL-ANN results in better IBS than state-of-the-art methods, and linear MTL-ANN performs well in both C-index and IBS. Based on the available data, a linear model is likely to be sufficient for survival analysis.

Although linear MTL-ANN has shown some surprising yet promising results, experiments on more datasets are more desirable. This project only used five publicly available real-world datasets for experiments because medical data is highly private to patients and the process of obtaining permissions to get access to the data is time consuming. Despite of the small number of datasets, to the best of our knowledge, the nature dataset is one of the largest real-world dataset which was used to evaluate survival models. Whereas the largest real-world datasets used in other survival analysis literatures are under 1000 observations, and most are about the size of the PBC dataset.

High-dimensional survival model was considered with the nature dataset, but could not be carried out because there are so many missing values in gene expressions in the original nature dataset. Time-dependent predictive model was also considered when building the Cox PH model, but was given up because the lack of sufficient samples made testing impossible. More data should allow us to extent MTL-ANN to a predictive model which is robust

to high dimensional datasets and time-dependent predictor variables.

In general, it is believed that MTL-ANN has the potential to outperform state-of-the-art methods because it has demonstrated its predictive power when measuring by IBS, it is independent of any statistical methods, its structure is simple and has great flexibility for future extensions.

For future work, an accurate point estimation is more desirable. A point estimation can be meaningful for both doctors and patients, because instead of getting a survival probability curve, an accurate survival time in days can help with decision makings by both parties. we have considered the use of median survival time as a point estimation for survival, however, it lacks the ability to represent the general predictive ability of a model, and can easily lead to a biased evaluation when asymmetric MSE is used as the performance evaluation metric. Further research should be carried on in finding a standard predicted outcome in survival analysis and correspondingly, a good performance evaluation metric, before moving onto developing more machine learning methods for survival analysis.

A cancer survival prediction tool powered by machine learning methods for survival analysis can also provide more guidance to both patients and doctors. There are some cancer survival prediction tools online which were published in recent years. They allow users to input some information such as age, gender, cancer type and so on, and tools return survival predictions. However, these tools are all based on statistical methods. For example, the University of Leicester published InterPreT [39] in July 2017 and the prediction is based on flexible parametric relative survival models. PREDCIT [40] provides survival prediction on breast cancer based on the Cox PH model. QCancer [41] published by University of Nottingham in 2017 provides prediction on colorectal cancer survival, but their algorithm is unstated - only the term 'equation' is mentioned. Therefore, to the best of my knowledge, a cancer survival prediction tool powered by machine learning based survival models is novel and it can put machine learning approaches for survival analysis into practice.

Appendix A

Raw Data

A.1 The Nature Dataset

Listing A.1: Original Survival Data

	Sample.ID	Years.to.birth	Days.to.death	Days.to.last.followup	Vital.status	Gender
1	TCGA-BL-A0C8-01A-11D-A10S-08	73	NA	389	0	male
2	TCGA-BL-A13I-01A-11D-A13W-08	57	223	NA	1	female
3	TCGA-BL-A13J-01A-11D-A10S-08	65	81	NA	1	male
4	TCGA-BL-A3JM-01A-12D-A21A-08	62	205	NA	1	male
5	TCGA-BT-A0S7-01A-11D-A10S-08	75	200	NA	1	male
6	TCGA-BT-A0YX-01A-11D-A10S-08	70	NA	363	0	female

	Date.of.initial.pathologic.diagnosis	TCGA.tumor.type	Somatic.mutations	Nonsilent.somatic.mutations	Tumor.stage
1	2009	blca	184	137	1
2	2010	blca	124	99	3
3	2010	blca	153	116	4
4	2010	blca	258	186	3
5	2007	blca	68	52	4
6	2009	blca	776	583	3

	Tumor.grade	ACVR1B	ACVR2A	AJUBA	AKT1	APC	AR	ARHGAP35	ARID1A	ARID5B	ASXL1	ATM	ATR	ATRX	AXIN2	B4GALT3	BAP1	BRAF

1		2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
2		2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3		2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4		2	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
5		2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6		2	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
	BRCA1	BRCA2	CBFB	CCND1	CDH1	CDK12	CDKN1A	CDKN1B	CDKN2A	CDKN2C	CEBPA	CHEK2	CRIPAK	CTCF	CTNNB1	DNMT3A	EGFR	EGR3				
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0				
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0				
	EIF4A2	ELF3	EP300	EPHA3	EPHB6	EPPK1	ERBB4	ERCC2	EZH2	FBXW7	FGFR2	FGFR3	FLT3	FOXA1	FOXA2	GATA3	H3F3C	HGF				
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0				
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0				
5	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0				
6	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0				
	HIST1H1C	HIST1H2BD	IDH1	IDH2	KDM5C	KDM6A	KEAP1	KIT	KRAS	LIFR	LRRK2	MALAT1	MAP2K4	MAP3K1	MAPK8IP1	MECOM	MIR142					
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0					
6	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	1	0					
	MLL2	MLL3	MLL4	MTOR	NAV3	NCOR1	NF1	NFE2L2	NFE2L3	NOTCH1	NPM1	NRAS	NSD1	PBRM1	PCBP1	PDGFRA	PHF6	PIK3CA	PIK3CG			
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
2	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0			
3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
6	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0			
	PIK3R1	POLQ	PPP2R1A	PRX	PTEN	PTPN11	RAD21	RB1	RPL22	RPL5	RUNX1	SETBP1	SETD2	SF3B1	SIN3A	SMAD2	SMAD4	SMC1A	SMC3			
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0			
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	

	SOX17	SOX9	SPOP	STAG2	STK11	TAF1	TBL1XR1	TBX3	TET2	TGFBR2	TLR4	TP53	TSHZ2	TSHZ3	U2AF1	USP9X	VEZF1	VHL	WT1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

A.2 The Colon Dataset

Listing A.2: The Colon Dataset

	id	study	rx	sex	age	obstruct	perfor	adhere	nodes	status	differ	extent	surg	node4	time	etype
1	1	1	Lev+5FU	1	43	0	0	0	5	1	2	3	0	1	1521	2
2	1	1	Lev+5FU	1	43	0	0	0	5	1	2	3	0	1	968	1
3	2	1	Lev+5FU	1	63	0	0	0	1	0	2	3	0	0	3087	2
4	2	1	Lev+5FU	1	63	0	0	0	1	0	2	3	0	0	3087	1
5	3	1	Obs	0	71	0	0	1	7	1	2	2	0	1	963	2
6	3	1	Obs	0	71	0	0	1	7	1	2	2	0	1	542	1

A.3 The Breast Dataset

Listing A.3: The Breast Dataset

	time	event	hormone	age	menostat	size	grade	pnodes	progrec	estrec
1	1814	1	0	70	1	21	2	3	48	66
2	2018	1	1	56	1	12	2	7	61	77
3	712	1	1	58	1	35	2	9	52	271
4	1807	1	1	59	1	17	2	4	60	29
5	772	1	0	73	1	35	2	1	26	65
6	448	1	0	32	0	57	3	24	0	13

A.4 The PBC Dataset

Listing A.4: The PBC Dataset

	time	event	treatment	age	sex	ascites	hepatom	spiders	edema	bili	chol	albumin	copper	alk	sgot	trig	platelet	prothrombin	stage
1	400	1	1	58.7652	1	1	1	1	1.0	14.5	261	2.60	156	1718.0	137.95	172	190.0000	12.2	4
2	4500	0	1	56.4463	1	0	1	1	0.0	1.1	302	4.14	54	7394.8	113.52	88	221.0000	10.6	3
3	1012	1	1	70.0726	0	0	0	0	0.5	1.4	176	3.48	210	516.0	96.10	55	151.0000	12.0	4
4	1925	1	1	54.7406	1	0	1	1	0.5	1.8	244	2.54	64	6121.8	60.63	92	183.0000	10.3	4
5	1504	0	2	38.1054	1	0	1	1	0.0	3.4	279	3.53	143	671.0	113.15	72	136.0000	10.9	3
6	2503	1	2	66.2587	1	0	1	0	0.0	0.8	248	3.98	50	944.0	93.00	63	261.9351	11.0	3

Appendix B

Exploratory Data Analysis

B.1 Common Imputation Methods in R `mice` Package

Table B.1: Common imputation methods [18]

Imputation method	Type of variables	Description
pmm	any	Predictive mean matching.
sample	any	Random sample from observed values.
cart	any	Classification and regression trees.
rf	any	Random forest imputations.
mean	numeric	Unconditional mean imputation.
norm	numeric	Bayesian linear regression.
logreg	binary	Logistic regression.

B.2 The Colon Dataset

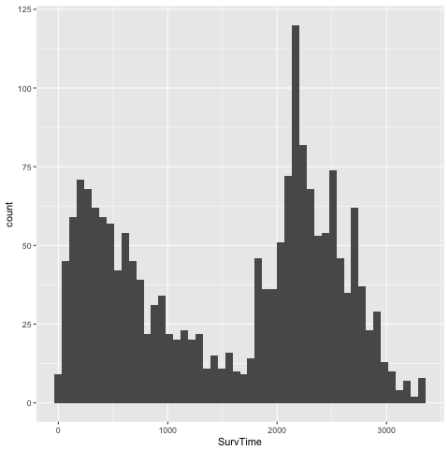

Figure B.1: SurvTime

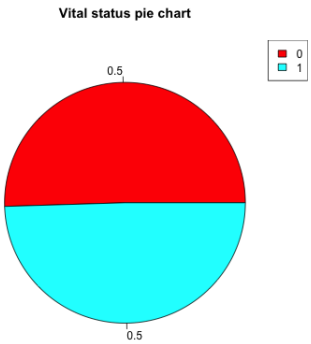

Figure B.2: Vital.status

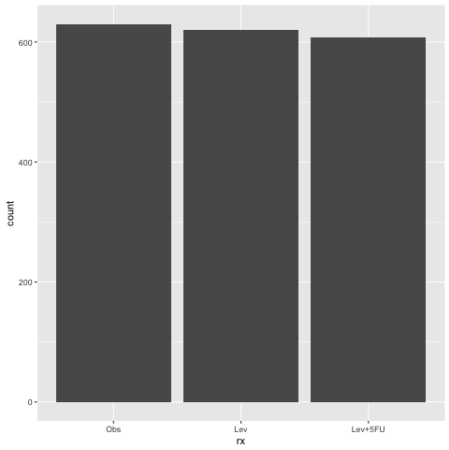

Figure B.3: rx

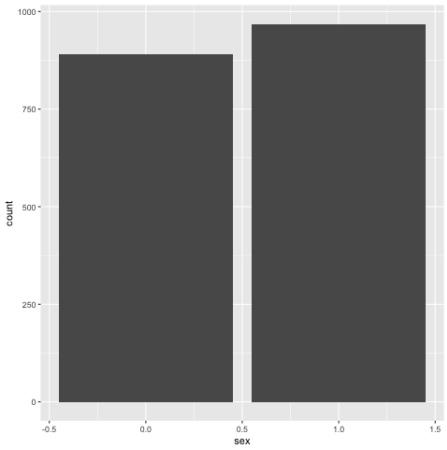

Figure B.4: sex

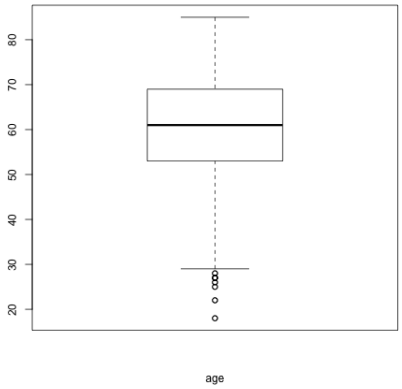

Figure B.5: age

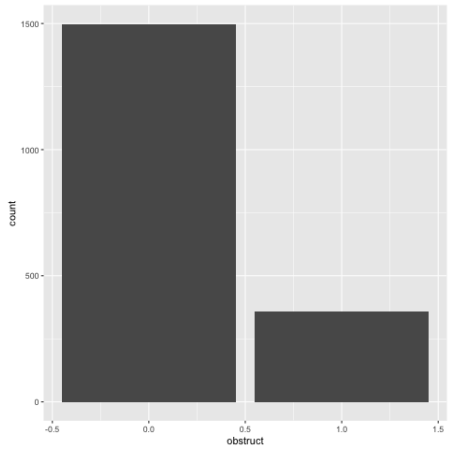

Figure B.6: obstruct

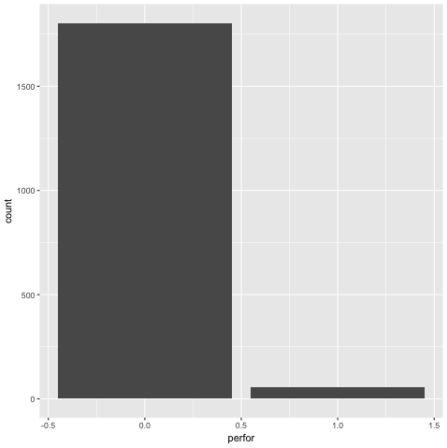

Figure B.7: perfor

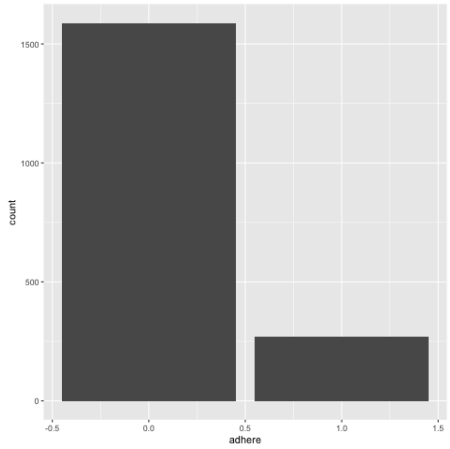

Figure B.8: adhere

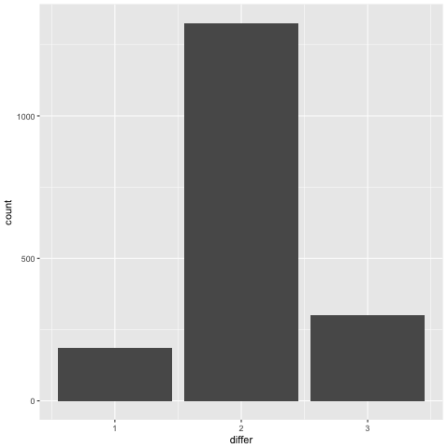

Figure B.9: differ

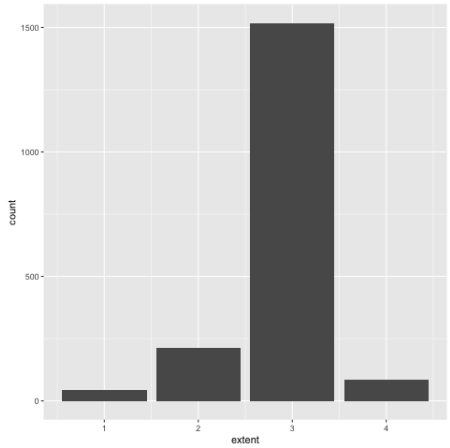

Figure B.10: extent

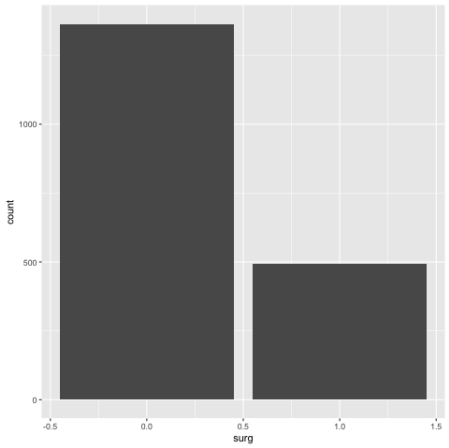

Figure B.11: surg

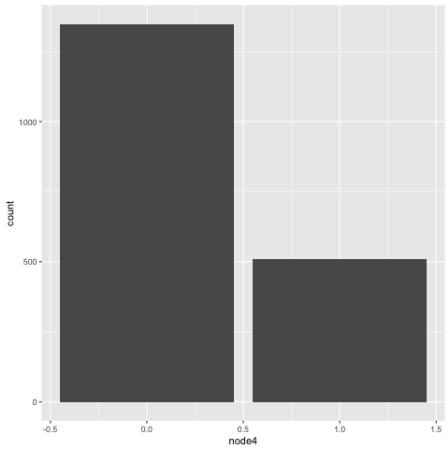

Figure B.12: node4

B.3 The Breast Dataset

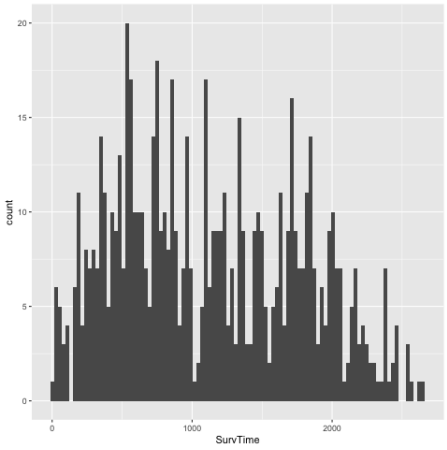

Figure B.13: SurvTime

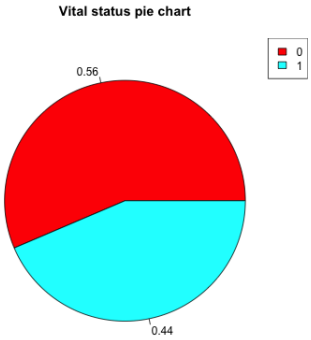

Figure B.14: Vital.status

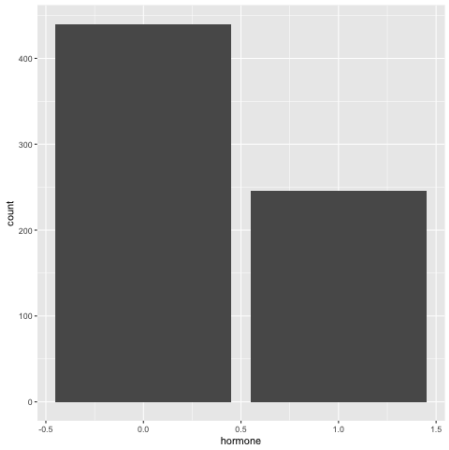

Figure B.15: hormone

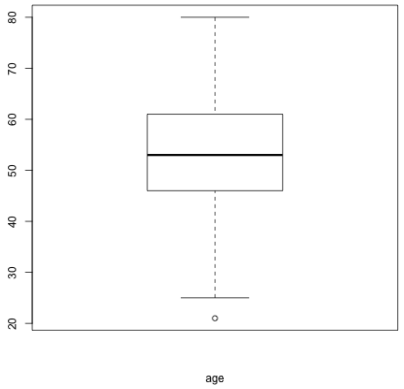

Figure B.16: age

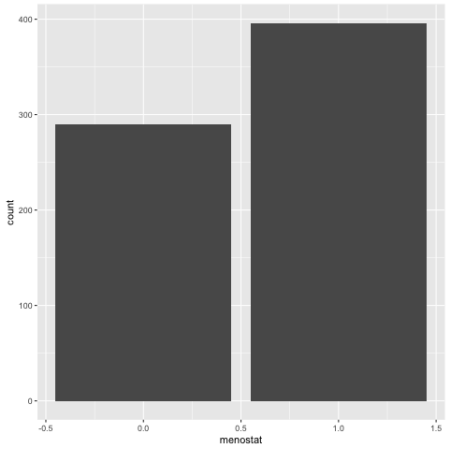

Figure B.17: menostat

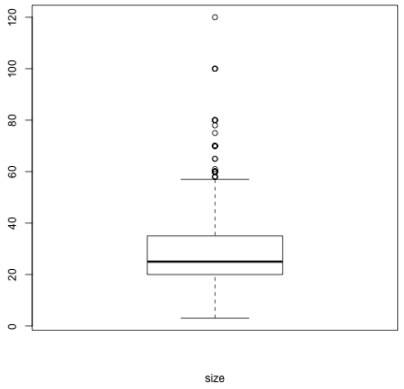

Figure B.18: size

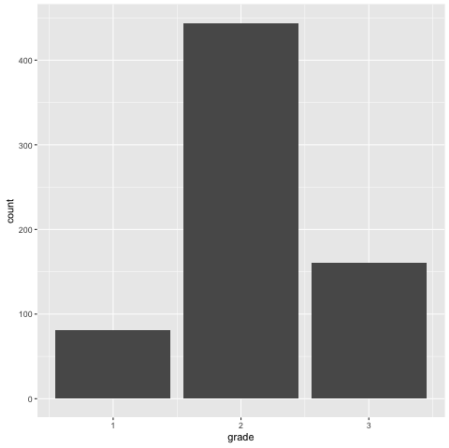

Figure B.19: grade

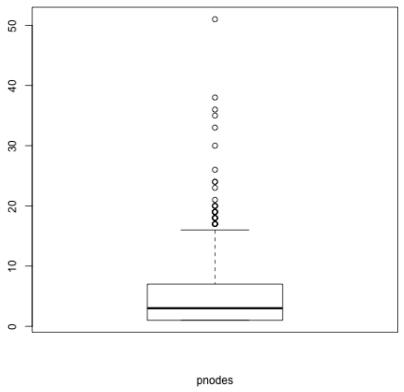

Figure B.20: pnodes

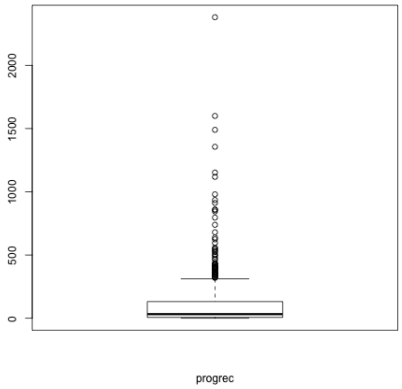

Figure B.21: progrec

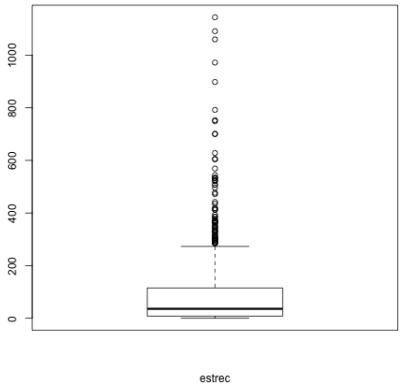

Figure B.22: estrec

B.4 The PBC Dataset

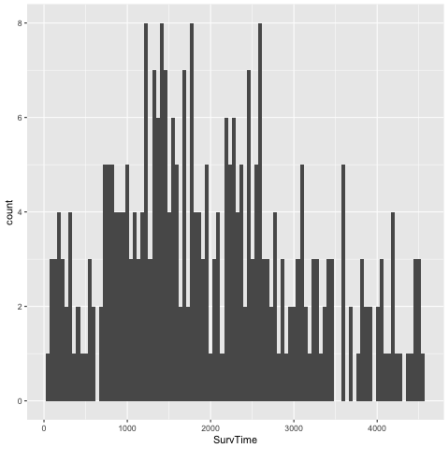

Figure B.23: SurvTime

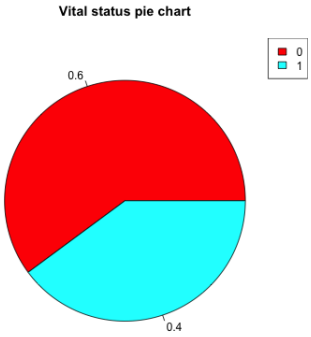

Figure B.24: Vital.status

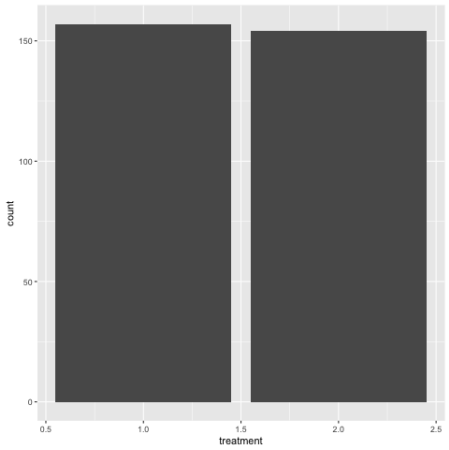

Figure B.25: treatment

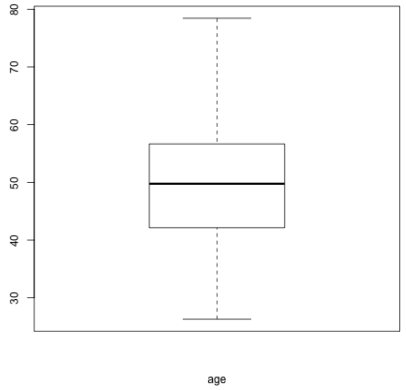

Figure B.26: age

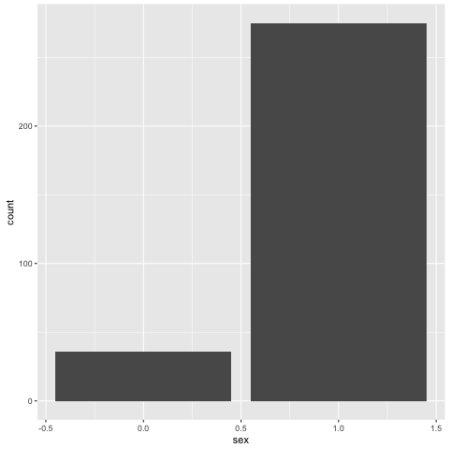

Figure B.27: sex

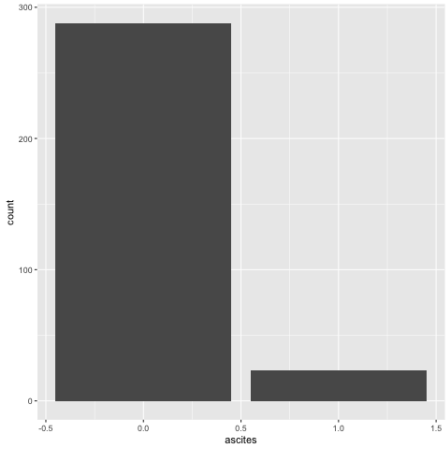

Figure B.28: ascites

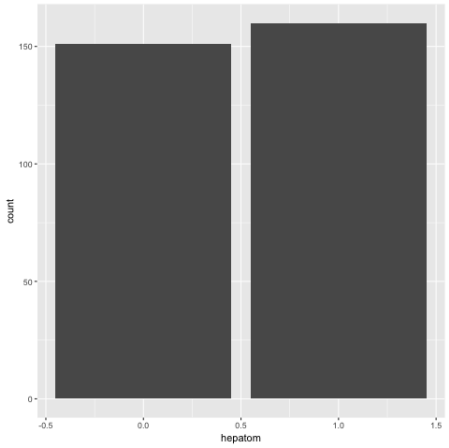

Figure B.29: hepatom

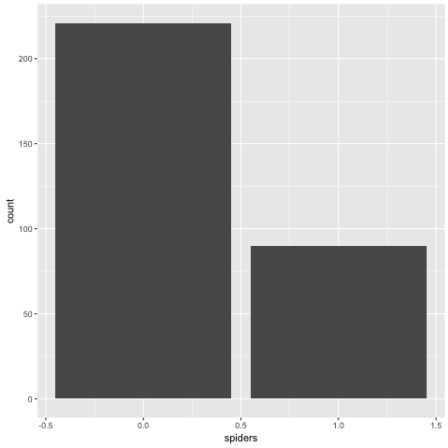

Figure B.30: spiders

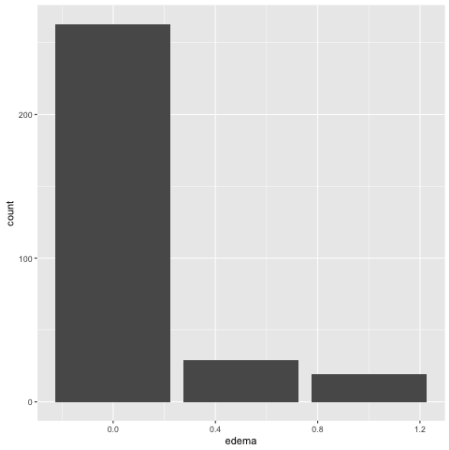

Figure B.31: edema

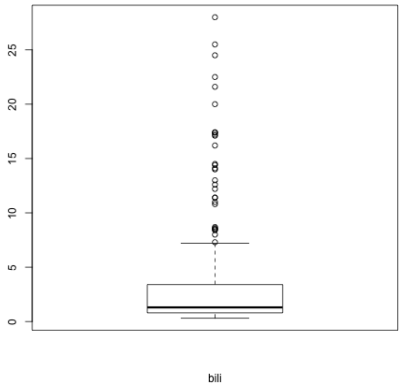

Figure B.32: bili

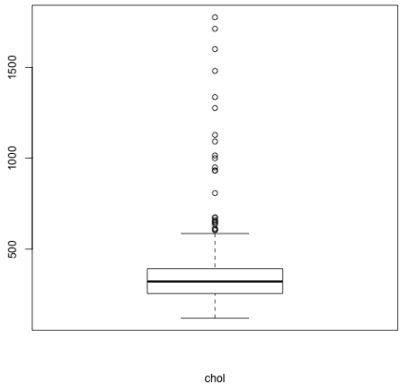

Figure B.33: chol

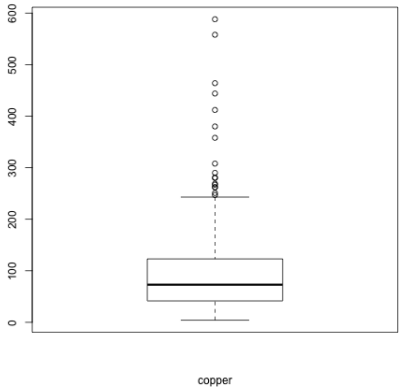

Figure B.34: copper

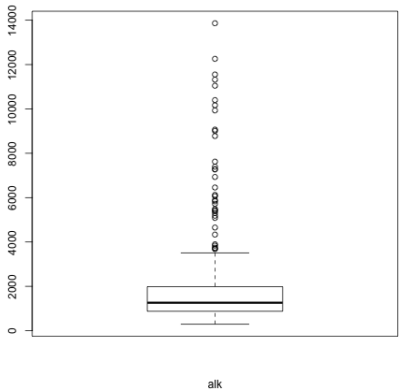

Figure B.35: alk

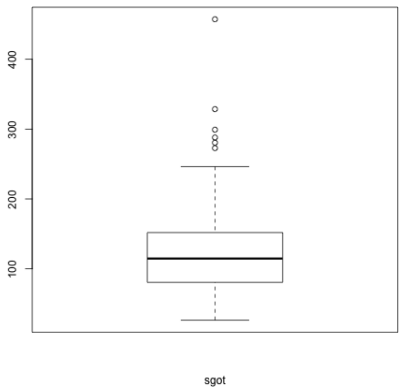

Figure B.36: sgot

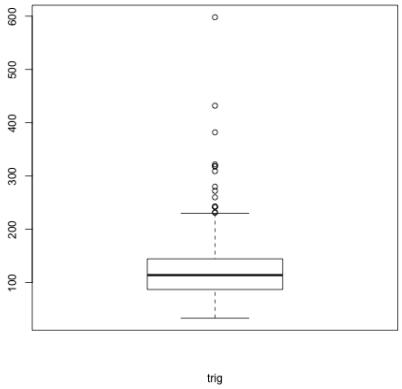

Figure B.37: trig

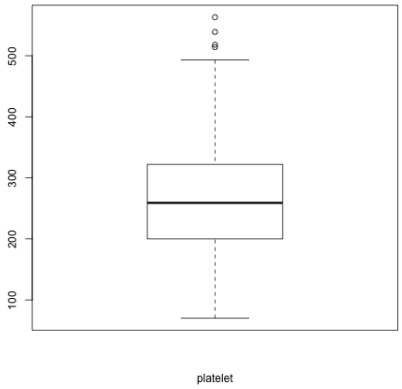

Figure B.38: platelet

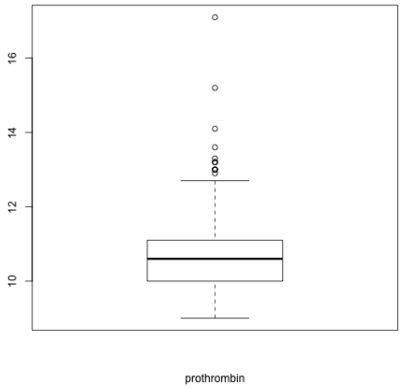

Figure B.39: prothrombin

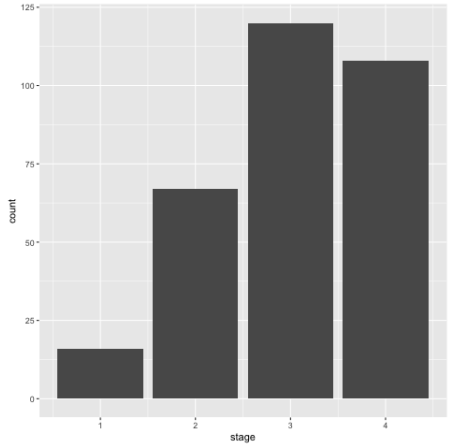

Figure B.40: stage

Appendix C

Summary of Cox PH Models

Listing C.1: Unstratified Cox PH Model on the Entire Dataset

```
Call:
coxph(formula = Surv(SurvTime, Vital.status) ~ ., data = survival_data,
    ties = "efron")

  n= 2572, number of events= 894

                                        coef exp(coef)  se(coef)      z Pr(>|z|)
Years.to.birth                      0.029726  1.030172  0.002945 10.095  < 2e-16 ***
Date.of.initial.pathologic.diagnosis -0.022728  0.977528  0.009401 -2.418  0.01562 *
Somatic.mutations                   -0.009777  0.990271  0.006976 -1.401  0.16107
Nonsilent.somatic.mutations          0.011542  1.011609  0.009002  1.282  0.19977
Tumor.stage                          0.406096  1.500947  0.036035 11.269  < 2e-16 ***
Gender.female                        0.036125  1.036785  0.083900  0.431  0.66678
Tumor.type.blca                      2.487657 12.033050  0.294129  8.458  < 2e-16 ***
Tumor.type.coadread                  0.872201  2.392171  0.291719  2.990  0.00279 **
Tumor.type.gbm                       2.677302 14.545798  0.150104 17.836  < 2e-16 ***
Tumor.type.hnsc                      1.004082  2.729400  0.184272  5.449 5.07e-08 ***
Tumor.type.kirc                      0.744909  2.106249  0.150761  4.941 7.77e-07 ***
Tumor.type.laml                      2.404333 11.071039  0.156375 15.375  < 2e-16 ***
Tumor.type.luad                      1.708586  5.521149  0.259247  6.591 4.38e-11 ***
Tumor.type.lusc                      1.613158  5.018636  0.276454  5.835 5.37e-09 ***
Tumor.type.ov                        0.926785  2.526374  0.135425  6.844 7.73e-12 ***
Tumor.type.ucec                     -0.294820  0.744666  0.300827 -0.980  0.32707
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

                                    exp(coef) exp(-coef) lower .95 upper .95
Years.to.birth                         1.0302    0.97071    1.0242    1.0361
Date.of.initial.pathologic.diagnosis   0.9775    1.02299    0.9597    0.9957
Somatic.mutations                      0.9903    1.00983    0.9768    1.0039
Nonsilent.somatic.mutations            1.0116    0.98852    0.9939    1.0296
Tumor.stage                            1.5009    0.66625    1.3986    1.6108
Gender.female                          1.0368    0.96452    0.8796    1.2221
Tumor.type.blca                       12.0330    0.08310    6.7610   21.4160
Tumor.type.coadread                    2.3922    0.41803    1.3505    4.2374
Tumor.type.gbm                        14.5458    0.06875   10.8385   19.5212
Tumor.type.hnsc                        2.7294    0.36638    1.9020    3.9167
Tumor.type.kirc                        2.1062    0.47478    1.5674    2.8303
Tumor.type.laml                       11.0710    0.09033    8.1486   15.0416
Tumor.type.luad                        5.5211    0.18112    3.3217    9.1770
```

```
Tumor.type.lusc                          5.0186    0.19926    2.9192    8.6279
Tumor.type.ov                            2.5264    0.39582    1.9374    3.2944
Tumor.type.ucec                          0.7447    1.34288    0.4129    1.3428

Concordance= 0.798  (se = 0.011 )
Rsquare= 0.292   (max possible= 0.992 )
Likelihood ratio test= 887.8  on 16 df,   p=0
Wald test            = 835.7  on 16 df,   p=0
Score (logrank) test = 1047  on 16 df,   p=0
```

Listing C.2: Stratified Cox PH Model on the Entire Dataset

```
Call:
coxph(formula = coxph_formula, data = survival_data, ties = "efron")

  n= 2572, number of events= 894

                                  coef exp(coef)  se(coef)      z Pr(>|z|)
Somatic.mutations           -0.033146  0.967397  0.011606 -2.856 0.004291 **
Nonsilent.somatic.mutations  0.042531  1.043448  0.014995  2.836 0.004564 **
Tumor.stage                  0.563536  1.756873  0.059879  9.411  < 2e-16 ***
Gender.female               -0.006926  0.993098  0.136808 -0.051 0.959623
Tumor.type.blca              2.202905  9.051271  0.503667  4.374 1.22e-05 ***
Tumor.type.coadread          1.552621  4.723837  0.444796  3.491 0.000482 ***
Tumor.type.gbm               3.242389 25.594806  0.286616 11.313  < 2e-16 ***
Tumor.type.hnsc              1.199258  3.317654  0.340768  3.519 0.000433 ***
Tumor.type.kirc              0.997983  2.712803  0.261986  3.809 0.000139 ***
Tumor.type.laml              3.162524 23.630152  0.293598 10.772  < 2e-16 ***
Tumor.type.luad              2.149706  8.582331  0.427209  5.032 4.85e-07 ***
Tumor.type.lusc              2.228316  9.284218  0.455674  4.890 1.01e-06 ***
Tumor.type.ov                1.097793  2.997542  0.264701  4.147 3.36e-05 ***
Tumor.type.ucec              0.558313  1.747721  0.376605  1.482 0.138211
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

                            exp(coef) exp(-coef) lower .95 upper .95
Somatic.mutations              0.9674    1.03370    0.9456    0.9897
Nonsilent.somatic.mutations    1.0434    0.95836    1.0132    1.0746
Tumor.stage                    1.7569    0.56919    1.5623    1.9756
Gender.female                  0.9931    1.00695    0.7595    1.2985
Tumor.type.blca                9.0513    0.11048    3.3728   24.2902
Tumor.type.coadread            4.7238    0.21169    1.9755   11.2955
Tumor.type.gbm                25.5948    0.03907   14.5943   44.8869
Tumor.type.hnsc                3.3177    0.30142    1.7013    6.4698
Tumor.type.kirc                2.7128    0.36862    1.6234    4.5334
Tumor.type.laml               23.6302    0.04232   13.2909   42.0124
Tumor.type.luad                8.5823    0.11652    3.7150   19.8265
Tumor.type.lusc                9.2842    0.10771    3.8008   22.6786
Tumor.type.ov                  2.9975    0.33361    1.7842    5.0359
Tumor.type.ucec                1.7477    0.57217    0.8354    3.6563

Concordance= 0.806  (se = 0.248 )
Rsquare= 0.166   (max possible= 0.478 )
Likelihood ratio test= 467.7  on 14 df,   p=0
Wald test            = 297.4  on 14 df,   p=0
Score (logrank) test = 430.4  on 14 df,   p=0
```

Appendix D

Hyperparameters in Random Survival Forest

Table D.1: Hyperparameters in RSF

Hyperparameter	Description [32]	Value
`bootstrap`	Bootstrap protocol - `by.root` bootstraps the data by sampling with replacement at the root node before growing the tree, `by.node` bootstraps the data at each node during the growing process, `none` the data is not bootstrapped at all.	`by.root`
`samptype`	Type of bootstrap when `by.root` is in effect. Choices are `swr` (sampling with replacement, the default action) and `swor` (sampling without replacement).	`swr`
`sampsize`	Requested size of bootstrap when `by.root` is in effect (if missing the default action is the usual bootstrap).	default
`nodedepth`	Maximum depth to which a tree should be grown. The default behaviour is that this parameter is ignored.	default

Table D.1: Hyperparameters in RSF

Hyperparameter	Description [32]	Value
`ntree`	Number of trees in the forest.	500
`splitrule`	Splitting rule used to grow trees - `logrank` which implements log-rank splitting or `logrankscore` which implements log-rank score splitting.	`logrank`
`ntime`	Integer value used for survival families to constrain ensemble calculations to a grid of time values of no more than `ntime` time points. If no value is specified, the default action is to use all observed event times.	default
`nsplit`	Non-negative integer value. When zero, deterministic splitting for an x-variable is in effect. When non-zero, a maximum of nsplit split points are randomly chosen among the possible split points for the x-variable.	1
`mtry`	Number of variables randomly selected as candidates for splitting a node.	Grid search
`nodesize`	Forest average number of unique cases (data points) in a terminal node.	Grid search

Bibliography

[1] *People fear cancer more than other serious illness*, Cancer Research UK, Aug. 2011. [Online]. Available: http://www.cancerresearchuk.org/about-us/cancer-news/press-release/2011-08-15-people-fear-cancer-more-than-other-serious-illness (visited on 11/14/2017).

[2] *Cancer statistics for the UK*, Cancer Research UK. [Online]. Available: http://www.cancerresearchuk.org/health-professional/cancer-statistics-for-the-uk (visited on 06/16/2018).

[3] R. Henderson, M. Jones, and J. Stare, "Accuracy of point predictions in survival analysis", eng, *Statistics in medicine*, vol. 20, no. 20, Oct. 2001, ISSN: 0277-6715.

[4] J. A. Cruz and D. S. Wishart, "Applications of machine learning in cancer prediction and prognosis", *Cancer informatics*, vol. 2, p. 59, 2006.

[5] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction", *Computational and structural biotechnology journal*, vol. 13, pp. 8–17, 2015.

[6] E. T. Lee and J. Wang, *Statistical methods for survival data analysis*. John Wiley & Sons, 2003, vol. 476.

[7] A. Gandy, *Lecture Notes: Survival Models and Actuarial Applications*, Jan. 2017.

[8] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997, ISBN: 978-0-07-042807-2.

[9] C. McDonald. (Dec. 2017). Machine learning fundamentals (II): Neural networks, [Online]. Available: https://towardsdatascience.com/machine-learning-fundamentals-ii-neural-networks-f1e7b2cb3eef (visited on 04/03/2018).

[10] A. Karpathy, A. Simpelo, O. Moindrot, P. Kreitmann, M. Bosnjak, R. Ring, J. Hu, G. Ambwani, Ethdragon, and I. Korsunov. (Nov. 2017). Convolutional Neural Networks for Visual Recognition, [Online]. Available: http://cs231n.github.io/neural-networks-1/ (visited on 04/12/2018).

[11] Dake∼commonswiki. (Aug. 2005). Overfitting, [Online]. Available: https://commons.wikimedia.org/wiki/File:Overfitting.png (visited on 05/31/2018).

[12] R. Caruana, "Multitask Learning", *Machine learning*, vol. 28, no. 1, pp. 41–75, Jul. 1997, ISSN: 1573-0565. DOI: 10.1023/A:1007379606734. [Online]. Available: https://doi.org/10.1023/A:1007379606734.

[13] R. M. S. d. Oliveira, R. C. F. Araujo, F. J. B. Barros, A. P. Segundo, R. F. Zampolo, W. Fonseca, V. Dmitriev, and F. S. Brasil, "A System Based on Artificial Neural Networks for Automatic Classification of Hydro-generator Stator Windings Partial Discharges", en, *Journal of Microwaves, Optoelectronics and Electromagnetic Applications*, vol. 16, pp. 628–645, Sep. 2017, ISSN: 2179-1074. [Online]. Available: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S2179-10742017000300628&nrm=iso (visited on 04/12/2018).

[14] Z. Zhou, *Machine Learning*. Beijing: Tsinghua Unversity Press, 2016.

[15] C. Kandoth, M. D. McLellan, F. Vandin, K. Ye, B. Niu, C. Lu, M. Xie, Q. Zhang, J. F. McMichael, M. A. Wyczalkowski, *et al.*, "Mutational landscape and significance across 12 major cancer types", *Nature*, vol. 502, no. 7471, p. 333, 2013.

[16] T. M. Therneau and T. Lumley, *survival: Survival Analysis*, comp. software, version 2.41–3, CRAN, Apr. 2017. [Online]. Available: https://github.com/therneau/survival.

[17] B. Vinzamuri, Y. Li, and C. K. Reddy, "Active learning based survival regression for censored data", in *Proceedings of the 23rd acm international conference on conference on information and knowledge management*, ACM, 2014, pp. 241–250.

[18] S. v. BBuuren and K. Croothuis-Oudshoorn, *mice: Multivariate Imputation by Chained Equations*, comp. software, version 2.46.0, CRAN, Oct. 2017. [Online]. Available: https://cloud.r-project.org/src/contrib/Archive/mice/mice_2.46.0.tar.gz.

[19] T. Bellotti, *Lecture Notes: Chapter 9: Data Preparation*, Nov. 2017.

[20] *PBC in Depth*, PBC Foundation, 2015. [Online]. Available: https://www.pbcfoundation.org.uk/health-professionals/pbc-in-depth (visited on 06/07/2018).

[21] P. Wang, Y. Li, and C. K. Reddy, "Machine learning for survival analysis: A survey", *Arxiv preprint arxiv:1708.04649*, 2017.

[22] L.-P. Kronek and A. Reddy, "Logical analysis of survival data: Prognostic survival models by detecting high-degree interactions in right-censored data", *Bioinformatics*, vol. 24, no. 16, pp. 248–253, 2008. DOI: 10.1093/bioinformatics/btn265. eprint: /oup/backfile/content_public/journal/bioinformatics/24/16/10.1093/bioinformatics/btn265/2/btn265.pdf. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btn265.

[23] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer, "Random survival forests", *The annals of applied statistics*, pp. 841–860, 2008.

[24] P. M. Ravdin and G. M. Clark, "A practical application of neural network analysis for predicting outcome of individual breast cancer patients", *Breast cancer research and treatment*, vol. 22, no. 3, pp. 285–293, 1992.

[25] E. Biganzoli, P. Boracchi, L. Mariani, and E. Marubini, "Feed forward neural networks for the analysis of censored survival data: A partial logistic regression approach", *Statistics in medicine*, vol. 17, no. 10, pp. 1169–1186, 1998.

[26] S. F. Brown, A. J. Branford, and W. Moran, "On the use of artificial neural networks for the analysis of survival data", *Ieee transactions on neural networks*, vol. 8, no. 5, pp. 1071–1077, 1997.

[27] D. Faraggi and R. Simon, "A neural network model for survival data", *Statistics in medicine*, vol. 14, no. 1, pp. 73–82, 1995.

[28] F. M. Khan and V. B. Zubek, "Support vector regression for censored data (SVRc): A novel tool for survival analysis", in *Data mining, 2008. icdm'08. eighth ieee international conference on*, IEEE, 2008, pp. 863–868.

[29] P. K. Shivaswamy, W. Chu, and M. Jansche, "A support vector approach to censored targets", in *Data mining, 2007. icdm 2007. seventh ieee international conference on*, IEEE, 2007, pp. 655–660.

[30] Y. Li, J. Wang, J. Ye, and C. K. Reddy, "A multi-task learning formulation for survival analysis", in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, ACM, 2016, pp. 1715–1724.

[31] I. K. Omurlu, M. Ture, and F. Tokatli, "The comparisons of random survival forests and Cox regression analysis with simulation and an application related to breast cancer", *Expert systems with applications*, vol. 36, no. 4, pp. 8582–8588, 2009.

[32] H. Ishwaran and U. B. Kogalur, *randomForestSRC: Random Forests for Survival, Regression, and Classificaiton (RF-SRC)*, comp. software, version 2.5.1, CRAN, Oct. 2017. [Online]. Available: `https://github.com/kogalur/randomForestSRC`.

[33] B. Baesens, T. Van Gestel, M. Stepanova, D. Van den Poel, and J. Vanthienen, "Neural network survival analysis for personal loan data", *Journal of the operational research society*, vol. 56, no. 9, pp. 1089–1098, 2005.

[34] L. Mariani, D. Coradini, E. Biganzoli, P. Boracchi, E. Marubini, S. Pilotti, B. Salvadori, R. Silvestrini, U. Veronesi, R. Zucali, *et al.*, "Prognostic factors for metachronous contralateral breast cancer: A comparison of the linear cox regression model and its artificial neural network extension", *Breast cancer research and treatment*, vol. 44, no. 2, pp. 167–178, 1997.

[35] V. Van Belle, K. Pelckmans, S. Van Huffel, and J. A. Suykens, "Support vector methods for survival analysis: A comparison between ranking and regression approaches", *Artificial intelligence in medicine*, vol. 53, no. 2, pp. 107–118, 2011.

[36] F. Chollet *et al.*, *Keras*, comp. software, GitHub, 2015. [Online]. Available: `https://keras.io`.

[37] MathWorks, *MATLAB Engine API for Python*, comp. software, MathWorks. [Online]. Available: `https://uk.mathworks.com/help/matlab/matlab-engine-for-python.html`.

[38] S. Petridis, *Lecture Notes: Artificial Neural Networks II*, Mar. 2017. [Online]. Available: `https://ibug.doc.ic.ac.uk/courses`.

[39] S. Islam, *InterPreT*, comp. software, Jul. 2017. [Online]. Available: `https://interpret.le.ac.uk/index.php`.

[40] D. R. F. J. Candido, G. C. Wishart, E. M. Dicks, *et al.*, *PREDICT*, comp. software, version 2.0, Public Health England and Cambridge University, May 2017. [Online]. Available: `http://www.predict.nhs.uk/predict_v2.0.html`.

[41] J. Hippisley-Cox and C. Coupland, *QCancer*, comp. software, ClinRisk Ltd., Jun. 2017. [Online]. Available: `https://qcancer.org/colorectal-survival/index.php`.